# MAT 1206 – Introduction to MATLAB

CHAPTER 03: Basic programming structures

Lesson 1: Decision Making

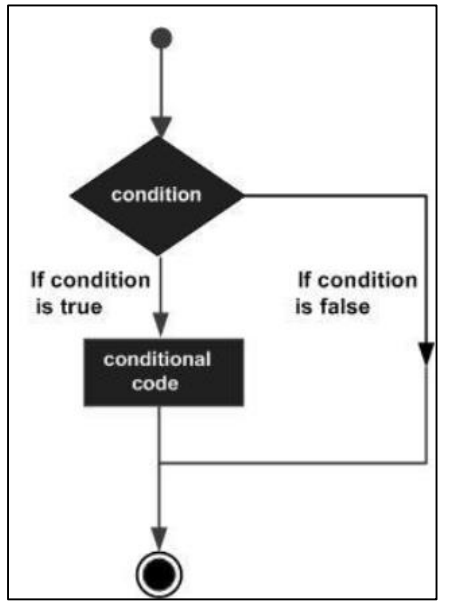
# Content

- if ... end statement
- if...else...end statement
- If... elseif...elseif...else...end statements
- nested if statements
- switch statement

# Decision Making

Decision making structures require that the programmer should specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

Following is the general form of a typical decision making structure found in most of the programming languages:



# Cont.

MATLAB provides following types of decision making statements.

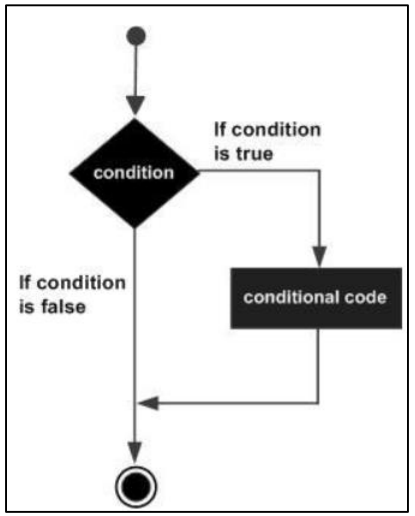
| Statement | Description |
|---|---|
| if ... end statement | An if ... end statement consists of a boolean expression followed by one or more statements. |
| if...else...end statement | An if statement can be followed by an optional else statement, which executes when the boolean expression is false. |
| If... elseif...elseif...else...end statements | An if statement can be followed by one (or more) optional elseif... and an else statement, which is very useful to test various conditions. |
| nested if statements | You can use one if or elseif statement inside another if or elseif statement(s). |
| switch statement | A switch statement allows a variable to be tested for equality against a list of values. |

# if... end Statement

An if ... end statement consists of an if statement and a boolean expression followed by one or more statements. It is delimited by the end statement.

**Syntax:**

```
if <expression>

% statement(s) will execute if the boolean expression is true

<statements>

end
```




If the expression evaluates to true, then the block of code inside the if statement will be executed.

# Cont.

## Example:

```
a = 10;
% check the condition using if statement
if a < 20
      % if condition is true then print the following
      disp('a is less than 20');
end
disp(['value of a is: ', num2str(a)]);
```

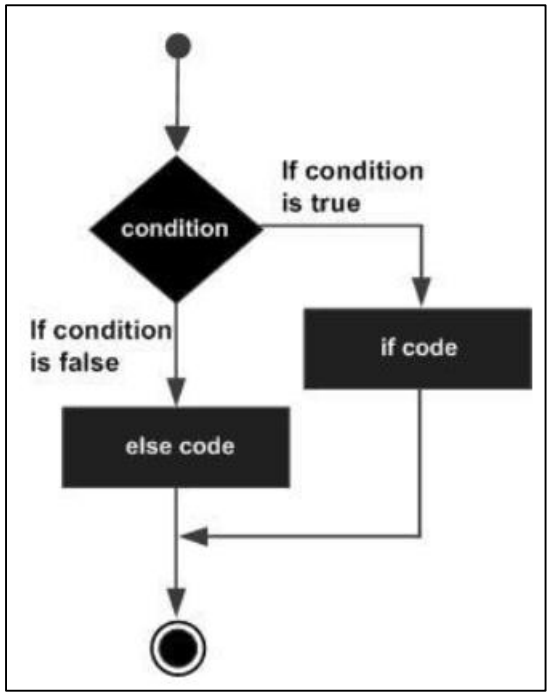
```
a is less than 20
value of a is: 10
```

# if...else...end Statement

An if statement can be followed by an optional else statement, which executes when the expression is false.

**Syntax**:

```
if <expression>

% statement(s) will execute if the boolean expression is true

<statement(s)>

else

<statement(s)>

% statement(s) will execute if the boolean expression is false

end
```



# Cont.

## Example:

```
a = 10;
% check the condition using if statement
if a < 20
      % if condition is true then print the following
      disp('a is less than 20');
else
      % if condition is false then print the following
      disp('a is not less than 20');
end
disp(['value of a is: ', num2str(a)]);
```

```
a is less than 20
value of a is: 10
```

```
a is not less than 20
value of a is: 30
```

# if...elseif...elseif...else...end Statements

An if statement can be followed by one (or more) optional elseif... and an else statement, which is very useful to test various conditions.

When using if... elseif...else statements, there are few points to keep in mind:

- An if can have zero or one else's and it must come after any elseif's.
- An if can have zero to many elseif's and they must come before the else.
- Once an else if succeeds, none of the remaining elseif's or else's will be tested.

# Cont.

Syntax:

```
if <expression 1>

% Executes when the expression 1 is true

<statement(s)>

elseif <expression 2>

% Executes when the boolean expression 2 is true

<statement(s)>

Elseif <expression 3>

% Executes when the boolean expression 3 is true

<statement(s)>

else

%  executes when the none of the above condition is true

<statement(s)>

end
```

# Cont.

Example:

```
a = 100;
%check the boolean condition
if a == 10
      % if condition is true then print the following
      disp('Value of a is 10' );
elseif( a == 20 )
      % if else if condition is true
      disp('Value of a is 20' );
elseif a == 30
      % if else if condition is true
      disp('Value of a is 30' );
else
      % if none of the conditions is true '
      disp('None of the values are matching');
      disp(['Exact value of a is: ', num2str(a)] );
end
```

```
None of the values are matching
Exact value of a is: 100
```

# Cont.

## Example:

```
TestScore = input('Enter a score from the keyboard ')
if(TestScore > 85)
        disp('Your Letter grade is A+')
elseif(TestScore > 75)
        disp('Your Letter grade is A')
elseif(TestScore > 70)
        disp('Your Letter grade is A-')
elseif(TestScore > 65)
        disp('Your Letter grade is B+')
elseif(TestScore > 60)
        disp('Your Letter grade is B')
elseif(TestScore > 55)
        disp('Your Letter grade is B-')
elseif(TestScore > 50)
        disp('Your Letter grade is C+')
elseif(TestScore > 40)
        disp('Your Letter grade is C')
elseif(TestScore > 35)
        disp('Your Letter grade is C-')
elseif(TestScore > 30)
        disp('Your Letter grade is D+')
elseif(TestScore > 25)
        disp('Your Letter grade is D')
else
        disp('Your Letter grade is E')
end
```

```
Enter a score from the keyboard 59
TestScore =
    59
Your Letter grade is B-
```

# The Nested if Statements

It is always legal in MATLAB to nest if-else statements which means you can use one if or elseif statement inside another if or elseif statement(s).

**Syntax**:

```
if <expression 1>
% Executes when the boolean expression 1 is true
   if <expression 2>
      % Executes when the boolean expression 2 is true
  end
end
```

# Cont.

## Example:

```
a = 100;
b = 200;
% check the boolean condition
if( a == 100 )
      % if condition is true then check the following
      if( b == 200 )
            % if condition is true then print the following
            disp('Value of a is 100 and b is 200' );
      end
end
disp(['Exact value of a is: ', num2str(a)] );
disp(['Exact value of b is: ', num2str(b)] );
```

```
Value of a is 100 and b is 200
Exact value of a is: 100
Exact value of b is: 200
```

# The switch Statement

A **switch** block conditionally executes one set of statements from several choices. Each choice is covered by a case statement.

An evaluated case_expression is a scalar, a string or a cell array of scalars or strings.

The switch block tests each case until one of the cases is true. When a case is true, MATLAB executes the corresponding statements and then exits the switch block.

The otherwise block is optional and executes only when no case is true.

# Cont.

**Syntax** :

```
switch <switch_expression>
   case <case_expression>
     <statements>
   case <case_expression>
     <statements>
     ...
     ...
   otherwise
       <statements>
end
```

# Cont.

Example:

```
grade = 'B';
switch grade
        case 'A'
                disp('Excellent!' );
        case 'B'
                disp('Well done' );
        case 'C'
                disp('Well done' );
        case 'D'
                disp('You passed' );
        case 'F'
                disp('Better try again' );
        otherwise
                disp('Invalid grade' );
end
```

```
Well done
```

# Cont.

Example:

```
n = input('Enter a number -1, 0, or 1: ');

switch n
case -1
      disp('negative one')
case 0
      disp('zero')
case 1
      disp('positive one')
otherwise
      disp('other value')
end
```

```
Enter a number -1, 0, or 1: 1
positive one
```

# Questions/queries?