

MATLAB Marina: Conditional Statements, if-else

Student Learning Objectives

After completing this module, one should:

1. Be able to use if-else statements to selectively execute blocks of statements.
2. Be able to use if-else statements for inline error handling.

Terms

logical true, logical false, logic expression, test, condition, inline error handling

MATLAB Functions, Keywords, and Operators

if, elseif, else, end

If-Else Statements

Conditional program structures (*if-else*, *switch*) allow programs to selectively execute blocks of statements depending upon the result of a logic expression. By default, all program statements are executed sequentially. An `if` statement (no else block) operates according to the flowchart of Figure 1a. The general syntax of an `if` statement is shown in Figure 1b.

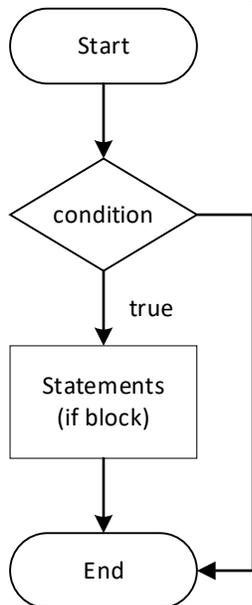


Figure 1a. if Statement Flowchart

```

if (logic expression)
    statements to be executed if logic expression true
end
  
```

Figure 1b. General Form of if-else Statement (no else block)

An *if-else* statement operates according to the flowchart of Figure 2a. The general syntax of an *if-else* statement is shown in Figure 2b.

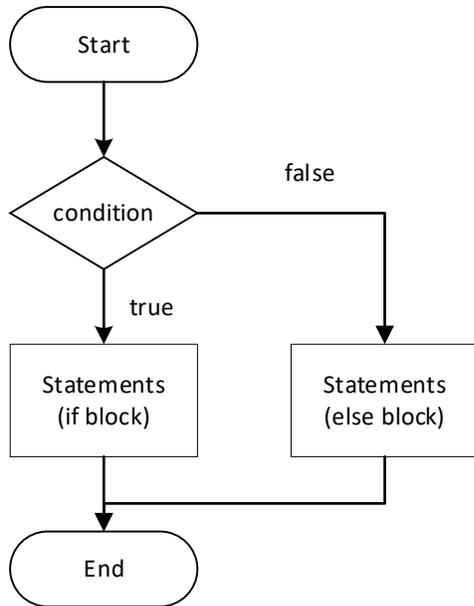


Figure 2a. if-else Statement Flowchart

```

if (logic expression)
    statements to be executed if logic expression true
else
    statements to be executed if logic expression false
end
  
```

Figure 2b. General Form of if-else Statement

A compound if-else statement (if-elseif-else) has operation and general form shown in Figures 3a and 3b.

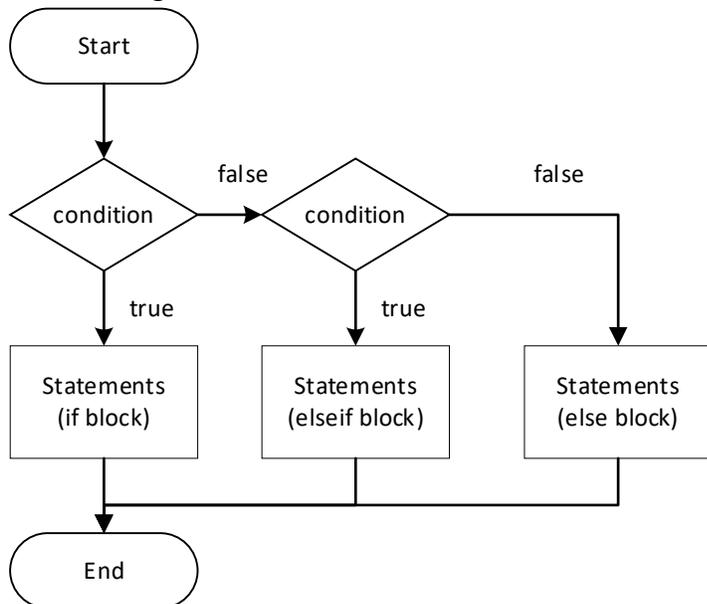


Figure 3a. Compound if-else Statement Flowchart

```
if (first logic expression)
    statements to be executed if first logic expression true
elseif (second logic expression)
    statements to be executed if second logic expression true
else
    statements to be executed if both logic expressions false
end
```

Figure 3b. General Form of Compound if-else Statement

The `if-else` statement is characterized by a single logic expression and one or two blocks of statements to selectively execute. The compound `if-else` is characterized by two or more logic expressions and generally three or more blocks of statements to selectively execute. There can be an unlimited number of `elseif` blocks in the compound `if-else` statement. The `else` block is optional for `if-else` statements and does not have a separate logic expression. Note that `if-else` statements must be terminated with the `end` keyword.

For `if-else` statements, the statements in the `if` and `elseif` blocks are only executed if the corresponding logic expression for the block is true. The block of statements for the `else` block is executed only if all logic expressions in the `if` and `elseif` portions evaluate to false. If the `else` block is omitted, no statements are executed when the logic expressions in the `if` and `elseif` portions evaluate to false.

The logic expression(s) (tests, comparisons) for `if-else` statements should result in scalar Boolean results (true or false) and are typically a combination of relational and logic operators although in some cases mathematical operations are also part of the comparison. Some examples of legal logic expressions for `if-else` statements are: a Boolean constant, a variable containing a Boolean value, a logical or relational operation on two scalars, a compound logical expression resulting from the logical AND or OR of two or more logic expressions, or a MATLAB function that returns a Boolean value.

In a compound `if-else` statement, only one of the logic expressions should evaluate to true. The MATLAB `if-else` statement is generally used when one has four or less outcomes to choose among. More than four outcomes can often be better handled using a `switch` statement.

The MATLAB programs of Figure 4a and 4b use `if-else` statements to determine the larger of two numbers.

```
num1 = input('Enter number 1: ');
num2 = input('Enter number 2: ');
if (num1 > num2)
    fprintf('num1 is greater than num2\n');
end
if (num1 < num2)
    fprintf('num1 is less than num2\n');
end
if (num1 == num2)
    fprintf('num1 is equal to num2\n');
end
```

Figure 4a. MATLAB Program to Determine Larger of Two Numbers

The program of Figure 4a uses three `if-else` statements with no `else` blocks. Each of the `if-else` statements tests for one of three possible cases and executes the block of code contained if the test is true. The program of Figure 4b uses one compound `if-else` statement to perform the same tests. The `if` and `elseif` blocks handle two of the cases and if these are both false the `else` block is executed.

```
num1 = input('Enter number 1: ');
num2 = input('Enter number 2: ');
if (num1 > num2)
    fprintf('num1 is greater than num2\n');
elseif (num1 < num2)
    fprintf('num1 is less than num2\n');
else
    fprintf('num1 is equal to num2\n');
end
```

Figure 4b. MATLAB Program to Determine Larger of Two Numbers

The program of Figure 4b can be interpreted as follows:

- Two numbers are read from the user and saved in the variables `num1` and `num2`.
- The condition `(num1 > num2)` is checked. If this condition is true the statements in the `if` block are executed.
- If the first condition is false, the next condition `(num1 < num2)` is checked. If this condition is true the statements in the `elseif` block are executed.
- If all previous conditions are false, the statements in the `else` block are executed.

The operation of the program of Figure 4b is also shown by the flow chart of Figure 5.

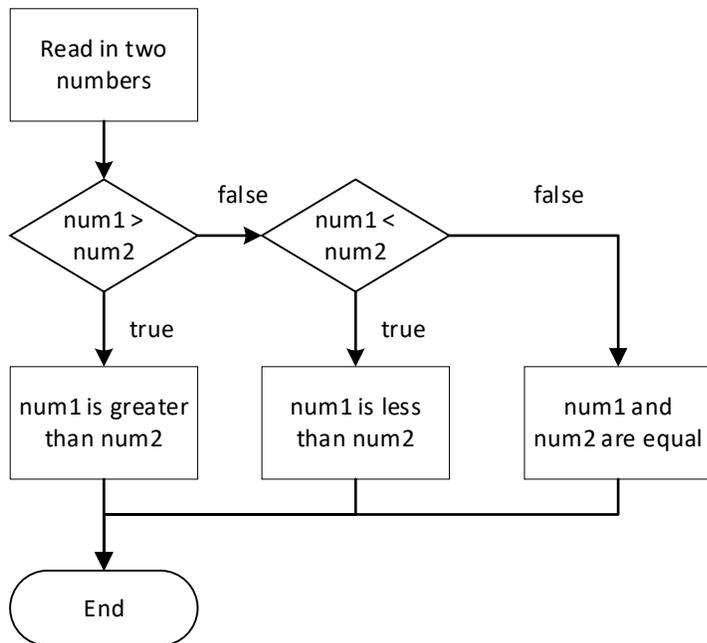


Figure 5. Flowchart for MATLAB Program to Determine Larger of Two Numbers

Last modified Monday, September 28, 2020

 [MATLAB Marina](https://creativecommons.org/licenses/by-nc-sa/4.0/) is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.