E7: Introduction to Computer Programming for Scientists and Engineers

University of California at Berkeley, Spring 2017

Instructor: Lucas A. J. Bastien

Diary for lecture 06: Branching

Version: release

```
% This document presents and illustrates concepts related to:
%
% - Branching
% - Comparing character strings
% - Using the function fprintf to print information to screen

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BRANCHING (IF-STATEMENTS) %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% To illustrate the concept of branching and the syntax of if-statements,
% we first work on a series of functions that calculate the cosine value of
% an angle, where the angle can be given in radians or degrees. Below are
% three different implementations of the same function

% -> Implementation 1
>> type my_cos_1

function [result] = my_cos_1(theta, is_degrees)
% Calculate cosine of theta. If is_degrees is true, assume that theta is in
% degrees. If is_degrees is false, assume that theta is in radians.
%
% In this implementation of the function, we use an if-statement with an
% else clause

if is_degrees
    result = cosd(theta);
else
    result = cos(theta);
end

end

% -> Implementation 2
>> type my_cos_2

function [result] = my_cos_2(theta, is_degrees)
% Calculate cosine of theta. If is_degrees is true, assume that theta is in
% degrees. If is_degrees is false, assume that theta is in radians.
%
% In this implementation of the function, we use an if-statement without an
% else clause. The variable theta keeps its original value if is_degree is
% false

if is_degrees
    % Convert to radians
    theta = theta * pi / 180;
end

result = cos(theta);

end

% -> Implementation 3
>> type my_cos_3

function [result] = my_cos_3(theta, is_degrees)
% Calculate cosine of theta. If is_degrees is true, assume that theta is in
% degrees. If is_degrees is false, assume that theta is in radians.
%
% In this implementation of the function, we use an if-statement with an
% else clause. The if-statement is used to create a handle to the
% appropriate function to use, given the units of theta

if is_degrees
    cos_function = @cosd;
else
    cos_function = @cos;
end

result = cos_function(theta);

end

% Let us test these three implementations
>> my_cos_1(30, true)

ans =

    0.8660

>> my_cos_1(pi/6, false)

ans =

    0.8660

>> my_cos_2(30, true)

ans =

    0.8660
```

```
>> my_cos_2(pi/6, false)

ans =

    0.8660

>> my_cos_3(30, true)

ans =

    0.8660

>> my_cos_3(pi/6, false)

ans =

    0.8660

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% COMPARING CHARACTER STRINGS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The synopsis of this section is that you should not use the == operator
% when trying to determine whether two character strings are equal. Rather,
% you should use one of the two functions: "strcmp" or "strcmpi". See below
% for more detail and examples

% Character strings are arrays of class "char"
>> my_char = 'Hello E7';
>> class(my_char)

ans =

char

>> size(my_char)

ans =

     1     8

% Using the relational operator == between two character strings will
% result in an element-wise comparison
>> 'abcde' == 'aaaee'

ans =

  1x5 logical array

   1   0   0   0   1

% and will result in an error if the strings have different lengths
```

```
>> 'abcde' == 'abc'

Matrix dimensions must agree

% In conclusion, DO NOT use the relational operator == to check whether two
% character strings are the same! Instead, use the functions strcmp (case
% sensitive) or strcmpi (case insensitive). For example:
>> strcmp('abcde', 'abc')

  logical

   0

>> strcmp('abcde', 'abcde')

  logical

   1

>> strcmp('abcde', 'AbCdE')

  logical

   0

>> strcmpi('abcde', 'AbCdE')

  logical

   1

% The functions "lower" and "upper" may be useful
% -> Use the function "lower" to convert a character string to lower case
>> lower('AbCdE')

ans =

abcde

% -> Use the function "upper" to convert a character string to upper case
>> upper('AbCdE')

ans =

ABCDE

%%%%%%%%%%%%%%%%%%%%%%%
% FPRINTF AND SPRINTF %
%%%%%%%%%%%%%%%%%%%%%%%

% Below are examples of the use of the function fprintf to print
% information to screen
```

```
>> i = 10;
>> j = 20;

% -> Use the conversion specification %d to format integers
>> fprintf('One integer: %d\n', i)

One integer: 10

% -> One call to fprintf can feature more than one conversion specification
>> fprintf('Two integers: %d and %d\n', i, j)

Two integers: 10 and 20

% -> Use the conversion specification %f to format numbers using decimal
%    notation
>> fprintf('The value of pi is %f\n', pi)

The value of pi is 3.141593

% -> You can control the number of decimal places to show when using the
%    conversion specification %f to format numbers using decimal notation
>> fprintf('The value of pi is %.15f\n', pi)

The value of pi is 3.141592653589793

% -> Use the conversion specification %e to format numbers using scientific
%    notation
>> fprintf('The value of pi is %e\n', pi)

The value of pi is 3.141593e+00

% -> You can control the number of decimal places to show when using the
%    conversion specification %e to format numbers using scientific
%    notation
>> fprintf('The value of pi is %.15e\n', pi)

The value of pi is 3.141592653589793e+00

% Use the conversion specification %s to include character strings
>> audience = 'E7';
>> fprintf('Hello %s!\n', audience)

Hello E7!

% The syntax of using sprintf is similar to the syntax of using fprintf to
% print information to screen. The difference between these two functions
% is that fprintf prints information to screen or to a file, whereas
% sprintf returns a character string. For example
>> my_string = sprintf('The value of pi is %.15f', pi)

my_string =
The value of pi is 3.141592653589793

%%%%%%%%%%%%%%%%%%%%
% ONE LAST EXAMPLE %
%%%%%%%%%%%%%%%%%%%%

% The function below prints to screen a comment that qualitatively
% describes the input temperature

>> type my_temperature_comment

function [] = my_temperature_comment(temperature)
% Print (to screen) a comment that qualitatively describes the input
% temperature (which is given in degrees farenheit)

if temperature > 90
    fprintf('It''s hot\n');
elseif temperature > 70
    fprintf('It''s warm\n');
elseif temperature > 50
    fprintf('It''s not so warm\n');
elseif temperature > 32
    fprintf('It''s rather cold\n');
else
    fprintf('It''s really cold\n');
end

end

% Let us try a few examples
>> my_temperature_comment(95)

It's hot

>> my_temperature_comment(-10)

It's really cold

>> my_temperature_comment(70)

It's not so warm

>> my_temperature_comment(71)

It's warm

>> my_temperature_comment(40)

It's rather cold
```