UMBC

ENME 303 LAB

Week 2: Control Flow I

Nameless Lab

UMBC

Week 2: Control Flow I

I. Refresher
II. Relational/Logical Operators
III. Control Structure
IV. Branch Statements: If-statements
V. Branch Statements: Switch-statements

I. Refreshers: Administrative

1. Assignments are ONE .m file with various exercises within
    a. You can use %% [SPACE] [EXERCISE NAME] to create **sections**
        i. If doing so, do not start each section with clear, clc all. **Just once at the top.**
        ii. You can run the code BY SECTION this way to check each exercise
2. Assignments are titled [LAST NAME]_[FIRST NAME]_LABHW_[NUMBER]
3. fprintf is used to display text or a variable
    a. Use %f or %d to display a variable
        i. Can control the number of digits before and after the period. (e.g. %4.2f)
    b. Use \n to create a new line in the command window when using fprintf or input functions

II. Relational & Logical Operators

- Relational Operators
  - Produce a single logical values and are:

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to (equivalent)
~=	not equal

  - Know difference between the assignment operator (=) and the equality operator (==)
    - = assigns a value to a variable, == checks if two values/variables are equal
- Try examples
  - Relational operators in arrays are applied element by element

```
[1 2 3] < [3 2 1]
'abcd' < 'cbaq'
[false false true] == [true false true]
[1 2; 3 4] > [5 6; 1 2]
[1 2 3 4] > [1 2]   % error -- both operands must have the same dimensions
```

II. Relational & Logical Operators

- Logical Operators
  - Has logical values as operands and returns single logical value

`&`	logical AND	`true` only if **both** values are `true`
`&&`	logical AND with shortcut evaluation	`true` only if **both** values are `true`
`|`	logical OR	`true` if **either** value is `true`
`||`	logical OR with shortcut evaluation	`true` if **either** value is `true`
`~`	logical NOT, unary	if `true`, then `false`; if `false`, then `true`
`xor`	logical EXCLUSIVE OR (actually a *logical function*)	`true` only if **both** values are different

- Example

```
x = input('Enter a value for x: ');
(x >= 0) && (x <= 10)
(x < 0) || (x > 10)
```

What do each mean?
What's returned?

II. Relational & Logical Operators

- The shortcut operations (&& or ||) operations will stop the eval of an expression as soon as the results of expression are known
  - **Reads left → right**. If operand on the left is false, whole expression is too
    i. x = (b ~= 0) && (a/b > 18.5)
- When to use which?
  - Use shortcut operators (&&,||) when comparing single logical values (scalars)
  - Use non-shortcut operators (&,|) when comparing arrays of logical values

III. Control Structure

A ***control structure*** is a programming language mechanism for changing the order in which statements in that program are executed

Branches (selection statements): Structure that causes execution to jump *forwards* in program
Skipping over some code

Branch statements: **if** and **switch**

Loops (repetition statements): Structure that causes execution to jump *backwards* in program
Same code executed more than once

Loop statements: **for** and **while**

IV. Branch Statements: If-Statements

MATLAB'S selection of if-statements:

1. One-alternative **if**
2. Two-alternative **if**
3. Multiple-alternative **if** with else
4. Multiple-alternative **if** without else

You always pick the simplest form that will accomplish the task of your code

IV. Branch Statements: One-alternative if

One-alternative **if** has the syntax:

```
if  <logical expression>
    <one or more statements>
end
```

For example,

```
if Body_temp > 98.6
     fprintf('The patient has a fever.\n')
end
```

- If the logical expression evaluates to **true**, statement(s) until **end** are executed.
- Otherwise (if logical expression **false),** statements before **end** not executed

Tip// Indenting statements doesn't matter in MATLAB, but will make your life easier in terms of readability.
Use MATLAB's smart indent feature (Text -> Smart Indent or Ctrl+I) to automatically indent

IV. Branch Statements: Two-alternative if

Two-alternative **if** has the syntax:

```
if  <logical expression>
    <one or more statements>
else
    <one or more statements>
end
```

For example,

```
if Body_temp > 98.6
     fprintf('The patient has a fever.\n')
else
     fprintf('The patient does not have a fever.\n');
end
```

- If the logical expression evaluates to **true**, statement(s) before the **else** are execute and execution skips to **end**
- Otherwise (if logical expression **false),** statements before the else are skipped and statements after the **else** and before **end** are executed
- Statements before and after the **else** will never both be executed

IV. Branch Statements: Multiple-alt if with else

Multiple-alt **if** with else has the syntax:

```
if  <logical expression>
    <one or more statements>
elseif <logical expression>
    <one or more statements>
<0 or more additional elseif blocks>
else
     <one or more statements>
end
```

For example:

```
if Body_temp >= 102
      fprintf('The patient has a high fever.\n')
elseif Body_temp >= 99
      fprintf('The patient has a low grade fever.\n');
elseif Body_temp >= 92
      fprintf('The patient has a low body
temperature.\n');
else
      fprintf('The patient is hypothermic.\n');
end
```

Use a multiple-alternative if with else statement when your code should do something for all possible cases.

IV. Branch Statements: Multiple-alt if without else

Multiple-alt **if** without else has the syntax:

```
if  <logical expression>
    <one or more statements>
elseif <logical expression>
    <one or more statements>
<0 or more additional elseif blocks>
end
```

For example:

```
if Body_temp >= 102
     fprintf('The patient has a high fever.\n')
elseif Body_temp > 98.6
     fprintf('The patient has a low grade fever.\n');
elseif Body_temp < 95
     fprintf('The patient has a low body temperature.\
end
```

Use a multiple-alternative if without else statement when the code should take no action in at least one case

IV. Branch Statements: Nested if

You can have nested if statements, such as:

```
if Body_temp >= 102
    fprintf('The patient has a high fever.\n')
    if Body_weight > 200 && Body_height < 64
      fprintf('Send the patient to the hospital\n');
    end
elseif Body_temp > 98.6
    fprintf('The patient has a low grade fever.\n');
elseif Body_temp < 95
    fprintf('The patient has a low body temperature.\n');
    if Temperature_time > 90
      fprintf('Send the patient to the hospital\n');
    end
end
```

IV. Branch Statements: Reminder

1. Design your **if** statements to minimize logical tests
2. Don't be redundant, you'll risk having code that is:
   a. Difficult to understand
   b. Harder to maintain
   c. Slower to execute

If statements execute in order!

Not great:

```
if Score >= 90
   disp('A')
elseif Score >= 80 & Score < 90
   disp('B')
elseif Score >= 70 & Score < 80
   disp('C')
elseif Score >= 60 & Score < 70
   disp('D')
elseif Score < 60
   disp('F')
end
```

Great:

```
if Score >= 90
   disp('A')
elseif Score >= 80
   disp('B')
elseif Score >= 70
   disp('C')
elseif Score >= 60
   disp('D')
else
   disp('F')
end
```

IV. Branch Statements: If Statement

Exercise: If-Statement

a. Write a script that asks the user for two numeric values, one for x and one for y. Then displays a message saying whether x is greater than y, equal to y, or less than y. The output should be formatted using `fprintf` to display answers.

IV. Branch Statements: If Statement

```
%% Exercise If-statement 1

   x = input('Enter x: ');
   y = input('Enter y: ');

   if x > y
       fprintf('x is greater than y \n')
   elseif x < y
       fprintf('x is less than y \n')
   else
       fprintf('x is equal to y \n')
   end
```

V. Branch Statements: Switch

Switch statements are useful for comparing an input to a list of specific cases (relational operators such < or > cannot be used with a switch statement. To test an inequality use if/else instead). It often uses less code than an if statement because the variable does not have to be repeated on every line.

```
Soup_choice = input('Choose a soup: input M for Mushroom, T for Tomato, C for
Chicken', 's');
```

(The 's' tells MATLAB that the input should be a string)

```
switch Soup_choice
    case 'M'
        disp('The mushroom soup will be $5.99')
    case 'T'
        disp('The tomato soup will be $4.50')
    case 'C'
        disp('The chicken soup will be $6.99')
    otherwise
        disp('That soup is not on our menu')
end
```

Acknowledgement

The lab slides you see are not made by one person. All the TA/TFs served for this course have contributed their effort and time to the slides. Below are the leading TFs for each semester:

- 2021 FA - Karla Negrete (GTA)
- 2022 SP - Justin Grahovac
- 2022 FA - Kelli Boyer and Yisrael Wealcatch
- 2023 SP - Matt Moeller and Mahamoudou Bah