UMBC

ENME 303 LAB

Week 3: While-Loops

Nameless Lab

Week 3: Control Flow II

I. Arrays
II. Refresher: Control Structure
III. Purpose of Loops
IV. The `while` Loop
V. Input Validation
VI. Counting and `while` Loops

UPDATE ON SUBMITTING ASSIGNMENTS

```
%Name your .m file : Lastname_firstname_Lab#
%Ie: Negrete_Karla_Lab1.m
clc, clear all
%% Section 1
    fprintf('Hello ')

%% Section 2
    fprintf('Whats up ')

%% Section 3
    fprintf('Cool ')
```

MAKE SURE YOUR CLC, CLEAR ALL IS OUTSIDE OF YOUR SECTIONS

Make sure you always suppress unnecessary outputs !!!

I. Data Stored in Arrays

- Data structure is an organized way of storing data
- MATLAB's primary data structures is the **double array**
  - A double array has a 2D structure; 1 or more rows and 1 or more columns
  - At the intersection of a col and row is an **element**
- Arrays have different names depending on their dimensions
  - A ***scalar*** is a single-element array (1x1).
  - A row ***vector*** is a single-row array (1xn).
  - A column ***vector*** is a single-column array (nx1)
  - The term ***matrix*** generally refers to an array with 2 or more non-singular dimensions (not a vector)
  - The term ***array*** is a general term for all of the above

I. Arrays Creation

- Arrays can be constructed using:
  - Square brackets

MATLAB command	Description
`[1 2 3 4 5]`	% A row vector with 1 row and 5 columns
`[1; 2; 3]`	% A column vector with 3 rows and 1 column
`[1 2; 3 4]`	% A matrix with 2 rows and 2 columns

  - Built-in functions

MATLAB command	Description
`zeros(4)`	% Creates a 4x4 matrix (4 rows, 4 columns) with each element equal to 0
`ones(3,4)`	% Creates a 3x4 matrix (3 rows, 4 columns) with each element equal to 1
`ones(3,4)*2`	% Creates a 3x4 matrix (3 rows, 4 columns) with each element equal to 2

  - Or a combination of the two

MATLAB command	Description
`[1:3; 4:6]`	% Creates the 2x3 matrix `[1 2 3; 4 5 6]`
`[zeros(3) ones(3)]`	% Creates the 3x6 matrix `[0 0 0 1 1 1; 0 0 0 1 1 1; 0 0 0 1 1 1]`

I. Arrays Size and Indexing

- The dimensions of an array can be found using:
  - The function size(A). This function returns the rows and columns of the array in a row vector.
    - The rows and columns can be separated into two variables
  - The function length(A). This function returns the largest array dimension.
- Indexing an array
  - Array indexing is used to access an element of an array based on its location in the array.
  - Example:

```
A = 4×4

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

MATLAB command	Description
A(3,2)	Returns the element in the 3rd row and 2nd column
A(2, [1 3])	Returns the elements in the 2rd row and the 1st and 3rd column
A(1:3, 2:4)	Returns the elements from the 1st to 3rd row and 2nd to 4th column
A(2, [1 3])	Returns the elements in the 2rd row and the 1st and 3rd column
A(:,3)	Returns the entire 3rd column

II. 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**

III. Purpose of Loops

- The purpose of loops is to **repeatedly** execute a series of statements
  - Note// Executing the same series of statements does not mean performing the same actions-- often times variables inside these loops will change each iteration
- Two classifications of loops:
  - **Condition-controlled statements** aka a `while` loop
    - Repeats one or more statements while some condition is true
    - In general, once loop commences, **you don't know** how long it will iterate
  - **Count-controlled statements** aka a `for` loop
    - Repeats one or more statements for a fixed number of times
    - In general, once loop commences, **you know** how long it will iterate

IV. The `while` Loop

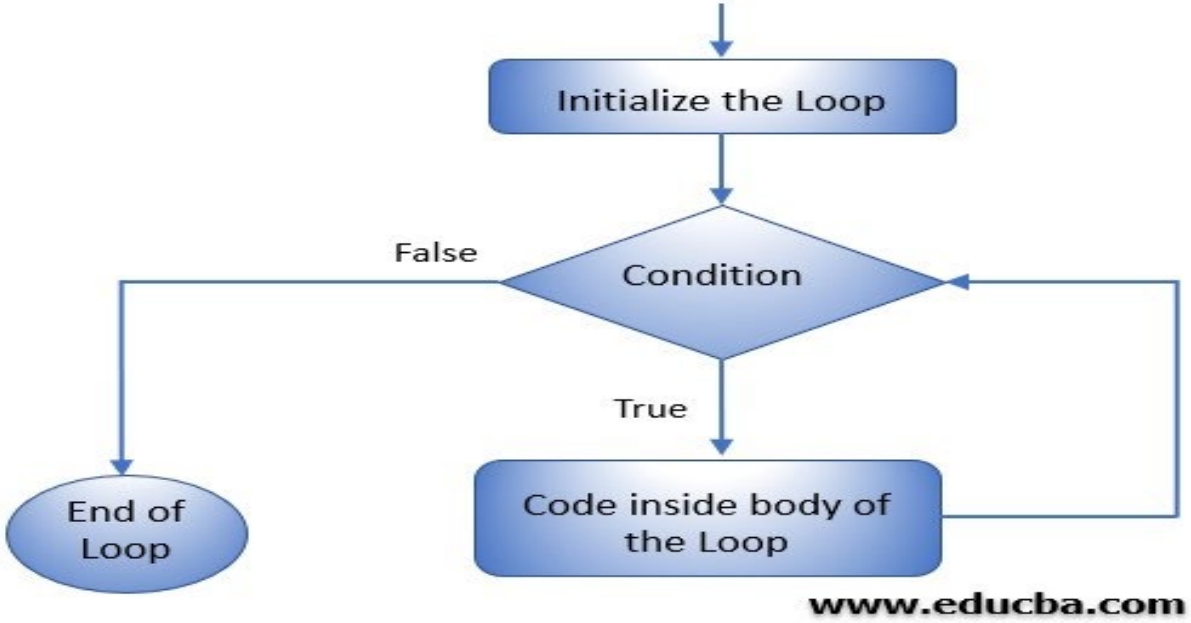
The `while` loop has the following syntax:

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

The `while` loop behavior:

- When execution reaches the `while`, the logical expression is evaluated
  - If logical expression = `true`, statement(s) executed
    - After loop body executed, executions returns to beginning of while
- Body of loop executes repeatedly until logical expression= `false`

While Loop Diagram



IV. The `while` Loop

Example:

```
Y = [];                 % initialize empty array to be filled in the
loop
X = 0;                  % initialize loop control variable (X)

while X <= 100          % test loop control variable
      Y(end+1) = X^2;       % specifies index to execute
operations
      X = X + 1;            % modify loop control variable
end

disp(Y)
```

What does this `while` loop do?

IV. The `while` Loop

More about the `while` loop:

1. The body of the loop may execute zero, one, or MANY times
   a. A loop that never stops repeating is called an **infinite loop**
      i. **Not good**
      ii. Use Ctrl + c to end the loop
2. The logical expression for `while` loops includes a variable called the **loop control variable**
   a. Most loops have a single control variable, but it's possible to have multiple

IV. The `while` Loop

Non-infinite *`while`* loops must satisfy the following:

1. The loop control variable must be ***initialized*** before the loop begins
   a. Yes that means before the `while` line
2. The loop control variable must be ***tested*** each iteration of the loop
   a. That means in the `while` loop's logical expression
3. The loop control variable must be ***modified*** in the body of the loop
   a. So that the loop will eventually stop

Always ***ITM*** (initialize, test, and modify) your `while` loop

IV. The `while` Loop

Stuck in an infinite loop or malfunctioning program?

Terminate the program by typing CTRL+C into the command line

V. Input Validation

- A common use for `while` loops is a **input validation loop**
  - It repeatedly prompts for and gets a value from user until a valid value is entered
- For example, this script validates that the input age is a non-negative age:

```
Age = input('Enter patient age in years: '); % initialize Age
while Age < 0       % test Age
    disp('INVALID INPUT - the age cannot be negative!')
    Age = input('Re-enter patients age in years: '); % modify
Age
end
```

V. Input Validation

- A modified version of the last example that does not duplicate the input statement, but instead requires (Age<0) be double checked

```
Age = -1;              % initialize Age
while Age < 0      % test Age
    Age = input('Enter patient age in years: ');   % modify Age
     if (Age < 0)
    disp('INVALID INPUT - the age cannot be negative!')
     end
end
```

VI. Counting and `while` Loops

- Count-controlled loops *can* be implemented with `while` loops too
  - For example, this loop displays integers from 1-10--thus loop body always executes 10 times

```
Count = 1;                  % initialize Count
while Count <= 10           % test Count
        disp(Count)
        Count = Count + 1;  % modify Count
End
```

Note// For loops are your best bet for counting..we cover next week.

Example 1: Watch X

```
x = 10;

fprintf('At start of loop: x = %d \n' ,x);

while x > 0

x = x - 3;

fprintf('x= %d \n',x);

end

fprintf('At end of loop: x= %d \n',x);
```

What's the x-value at the end of the program?

Can you identify the ITM components of the loop?

Loop #	Value of X
Before loop	10
1	7
2	
..	

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- Mahamoudou Bah, Matt Moeller and Michael Mullaney