Topics: 1-dimensional array, more iteration!

Reading (ML): Sec 2.1–2.4, 2.8 for discussion on 1-d array, make sure you've done all the required reading!

1-Dimensional Array: Vector

An array is a *named* collection of data values organized into rows and/or columns. A 1-d array is a row or a column, also known as a *vector*. An *index* is a positive integer that identifies the position of a value in the vector.

Suppose vector **v** is a collection of 4 values, i.e., vector **v** has 4 cells.

The ith value can be accessed as **v(i)**.

Assign a value of 9 to into the 4th cell of vector **v**: **v(4) = 9**.

Copy the value in the 4th cell to the 2nd cell of vector **v**: **v(2) = v(4)**.

Copy the value in the current cell to the next cell of vector **v**: **v(i+1) = v(i)**.

Array Initialization

MATLAB function **zeros**: **vecA = zeros(1,5)**
MATLAB function **ones**: **vecB = ones(5,1)**
MATLAB short-cut expression for consecutive numbers: **1:6** or **1:1:6**
"Manual": **vecC(5) = 10**

Can you write a program for calculating the average of 10 numbers (Example 1 from 2/5 lecture) that stores all the data entered by the user? Below is the original program that doesn't store all user input.

```
% Average 10 numbers from user input




n = 10;     % number of data values
total = 0;  % current sum (initialized to zero)
i = 1;      % initialize counter
while (i<=n)
  % read and process input value
    num   = input('Enter a number: ');
    total = total + num;
  % update
    i = i + 1;
end
ave = total/n  % average of n numbers
```

What are some useful MATLAB built-in functions for the above problem?

Example 1

Write a program segment that calculates the *cumulative sums* of a given vector **v**. The cumulative sums should be stored in a vector of the same length as **v**. E.g., the cumulative sums for the sequence 1,3,5,0 is 1,4,9,9. Do not use MATLAB predefined functions other than **length**.

Example 2

Write a program segment that determines whether a given integer **n** is prime. Assume n>2. (Hint: MATLAB function **mod(x,y)** returns the value of the remainder of x divided by y assuming integer values of x, y.)

Example 3

Sketch a program that will list all the prime numbers in the range of [2,**n**] given an integer **n**>1.

Example 4

Develop an algorithm for calculating the *mode* of a sequence. The mode is the number in the sequence that occurs with maximum frequency. Assume that the sequence is (a) non-negative, (b) entered one by one and terminated by a negative number, and (c) entered in non-decreasing order. E.g., the mode of the sequence 87,92,92,98,98,98,100 is 98. Assume that only scalar variables are allowed.

Programming Rules of Thumb

- *Learn program patterns* of general utility and *use relevant pattern* for the problem at hand.
- *Seek inspiration* by systematically working test data by hand. Be introspective; ask yourself: "what am I doing?"
- *Declare variables* for each piece of information you maintain when working problem by hand. *Write comments* that precisely describe the contents of each variable.
- *Decompose* problem into manageable tasks.
- *Remember* the problem's boundary conditions.
- *Validate* your program by tracing it on simple test data.