User-Defined Function

- **Procedural abstraction**—can easily reuse code
- Functions can be independently tested
- Upon invocation, each function has its own memory space that is inaccessible by other functions or the command window space—variables in a function can be “seen” only inside the function
- Arguments are “passed by value”
- Values stored in variables are not preserved between function calls.

Subfunctions

There can be more than one function in an M-file. The top function is normal while the remaining functions are subfunctions, accessible only by top function. The file name should be the name of the top function.

Global Memory

Global memory can be accessed from any workspace. A global variable must be declared to be global before it is used for the first time in a function.

```
global variable1 variable2 ...
```

*Note:* Do *not* use global variables excessively! In general, we want a function to have local variables that are visible only within the function—protected from other functions.

Persistent Memory

Persistent memory can be accessed from within the function only and is preserved unchanged between calls to the function.

```
persistent variable1 variable2 ...
```

Random number generator `rand`

MATLAB’s built-in function `rand` generates a number in the range of 0 to 1 randomly. In other words, function `rand` generates a number from the standard uniform distribution: any number in the range of 0 to 1 is *equally likely to occur*. Note that the range is the open interval (0,1).

```
x = rand(1,1); % one random number in (0,1)
x = 6*rand(1,1); % one random number in (0,6)
```

Which expression below gives an integer in (1..6) with *equal* likelihood?

```
x = round(rand(1,1)*6)  % x = ceil(rand(1,1)*6)
```

A random example. I mean a random walk example!

Write a program that performs a “random walk.” In a random walk, possible moves are left, right, up, or down (in a Cartesian plane). Prompt the user for the number of steps and the starting point. Display the final location.
% Perform n steps of random walk starting from (x0, y0). Display the final location

disp('Do a random walk!')
n = input('How many steps? ');
x = input('From what x-coordinate? ');
y = input('From what y-coordinate? ');

% Perform walk, each step is based on a random integer

fprintf('End up at (%d,%d)!
', x, y)

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. MATLAB array index starts at one, not zero.

Review: developing algorithms

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.

The savvy programmer...

- 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 the problem into manageable tasks.
- Refine the algorithm iteratively: solve a simpler problem first
- Remember the problem’s boundary conditions.
- Validate your program by tracing it on simple test data.