W1005
Intro to CS and Programming in MATLAB

Data Structures

Fall 2014

Instructor: Ilia Vovsha

http://www.cs.columbia.edu/~vovsha/w1005

Outline

- Cell arrays
- Functions with variable arguments
- Structure arrays (Structs)

Cell Arrays (purpose)

- Group different data types into a single container
  - Mathematical operations are not defined on the container
  - Must address content of each cell
  - Must identify number of cell
  - Each element (cell) of a cell array can hold any data type
  - Manipulation of cell arrays is similar to arrays

Cell Arrays (initialization)

- Initialization:
  - C = cell(N); % N by N array of empty cells
  - C = cell(N,M); % N by M array of empty cells
  - Use brackets {} to access content of cells, () to index
  - C = {[1] [1,2,3]; rand(3), rand(3)}; % Create directly
  - A(1,1) = { [1,2,3] }; % add element
  - A{1,1} = [1,2,3]; % add element
  - celldisp(A); % display all contents

Cell Arrays (manipulation)

- Similar to arrays:
  - A = cell(3); B = cell(3); % 3 by 3 arrays of empty cells
  - C(3,:) = []; % access
  - D = C([1,2],:) % access
  - [r,c] = size(A); % size
  - iscell(B) % check
  - [a,b,c] = deal(A{1,:}); % deal contents to variables

num2cell()

- Problem 1: create a vector ‘V’ of length N, where each element, V(k) = 2^k
- Problem 2: create a cell array ‘C’ of length N, where each element, C{k} = 2^k
- Solution (1):

```
1. idx = 1:N;
2. V = 2.^idx;          % The dot is required here
```

- Solution (2):
  - Cell array, solution (1) doesn’t work!
  - Instead of using a for-loop could convert numeric array to cell

num2cell()

- Problem 2: create a cell array ‘C’ of length N, where each element, C{k} = 2^k
- Solution (2):

```
1. idx = 1:N;
2. V = 2.^idx;          % V is a numeric array
3. C = num2cell(V);     % Converted V to a cell array C
```

cellfun()

- Apply certain functions to all cells:
- Example:

1. cellfun('<function>', < cell array>); % generic form
2. cellfun('isempty', A); % returns logical matrix
3. cellfun('length', A); % returns length

Functions (variable arguments)

- Function which can have any number of input args
- Rules:
  - Keyword: varargin
  - 'varargin' is a cell array containing the optional arguments ONLY
  - Must be declared as the last input argument
  - Must be lowercase
  - Example: "function my_fun(x,y,z,varargin)"
  - Similarly, use varargout to create a function with any number of output args. The same rules apply (last, lowercase)
  - function [<out_var1>, varargout] = <name> (<var1>,<var2>, varargin)
  - varargin, varargout are cell arrays, ith cell is the ith argument
  - nargin, nargout return the actual number of arguments used

Exercise (in class)

- Variable input arguments
- Write a function with *at least* 3 input parameters {A, B, fh1, fh2, fh3…}, which applies the functions stored in the strings 'fh1' (fh2, fh3…if present) to the variables {A,B} which can be either cell arrays or matrices (you need to account for both options).

Structure Arrays (initialization)

- Structs are similar in purpose to cell arrays
  - Multiple *fields*, different data structures for each field
  - Use 'dot' to access fields
  - Must identify name (not number) of element
- Initialization:
  - S = struct([]) % Empty struct, no fields
  - S = struct ('f1', v1, 'f2', v2) % Struct with two fields
  - S.f1 = 2.5; % Create directly

Structs (get content)

- Add element:
  - S.f1 = 2.5; S.f2 = ‘hi’; % Create directly
  - S(2).f1 = 3; % Add element
  - S(1).f1 = 1.5; % Access changed!
  - S(3).f3 = rand(3); % Add field
  - S(:).f1 % get field values
  - S(2:3).f3 % get field values

Structs (functionality)

- Useful functions:
  - isstruct(S); % Check if S is a struct
  - S = setfield(S, 'field1', 5); % Set a field to a value
  - isfield(S, 'field1'); % Check if field exists
  - fieldnames(S) % return cell array of strings
- Example:

1. S = struct('vec', [1,2,3], 'mat', rand(3));
2. my_field = 'mat';
3. isfield(S, my_field);

Cell to Struct Conversion

- Commands: cell2struct(), struct2cell()
  - S = cell2struct(C, my_fields, my_dim)
  - 'my_fields' is a cell array of strings
- Example:

  1. my_fd = {'num', 'name', 'nation'};
  2. my_arr = {8, 'Iniesta', 'Spain'};
  3. my_st = cell2struct(my_arr, my_fd, 2);

  % You are 'folding' the dimension, size must match

Strings (representation)

- Strings are a sequence of characters, hence numerical arrays of ASCII values
- Functions for manipulating arrays apply:
  - t = 'Hello World!';
  - size(t), double(t), char(double(t))
- Similar to arrays, must have equal # of columns, hence blanks must be padded:
  - v = char('a', 'ab', 'abc'); v = strvcat('a', '', 'ab', 'abc');
  - strcat(), deblank()

Strings (functions)

- MATLAB is not recommended as a tool for manipulating strings. However, the functionality is available
- Check string:
  - isletter(str), isspace(str)
- Convert string:
  - lower(str), upper(str)
  - str2num(str), num2str(num)

Strings (examples)

- Operations on strings:
  - strtok(str, delim), strcmp(str1,str2), strfind(str1,pattern)
- Regular expressions (help regexp): concise and flexible means for matching strings
- Example:

1. str1 = 'one!, no two';
2. [bef_delim, aft_delim] = strtok(str1, '!');
3. idx = strfind(str1, 'n');
4. res1 = strcmp('hi', 'HI');
5. res2 = strcmp('hi', lower('HI');