Introduction to Computing for Engineers EE050IU

Strings, Cells and Files

Dr. Nguyen Ngoc Truong Minh

Strings

- A string is an array of characters
  - s = 'abc'

    is equivalent to s = ['a' 'b' 'c']
- All operations that apply to vectors and arrays can be used together with strings as well
  - s(1) $\rightarrow$ 'a'
  - s([1 2]) = 'XX' $\rightarrow$ s = 'XXc'
  - s(end) $\rightarrow$ 'c'

String Conversion

- Conversion of strings to numerical arrays
  - double('abc xyz')

```
ans =
   97   98   99   32  120  121  122
```

  - double('ABC XYZ')

```
ans =
   65   66   67   32   88   89   90
```

- Conversion of numerical arrays to strings
  - char([72 101 108 108 111 33])

```
ans =
Hello!
```

String Tests

- ischar() : returns 1 for a character array
  - ischar ('CS 111')

```
ans =
     1
```

- isletter() : returns 1 for letters of the alphabet
  - isletter('CS 111')

```
ans =
     1     1     0     0     0     0
```

- isspace() : returns 1 for whitespace (blank, tab, new line)
  - isspace('CS 111')

```
ans =
     0     0     1     0     0     0
```

String Comparison

- Comparing two characters
  - 'a' < 'e'

```
ans =
     1
```

- Comparing two strings character by character
  - 'fate' == 'cake'

```
ans =
     0     1     0     1
```

  - 'fate' > 'cake'

```
ans =
     1     0     1     0
```

String Comparison

- strcmp() : returns 1 if two strings are identical
  - a = 'Bilkent';
  - strcmp(a, 'Bilkent')
```
ans =
     1
```
  - strcmp('Hello', 'hello')
```
ans =
     0
```
- strcmpi() : returns 1 if two strings are identical ignoring case
  - strcmpi('Hello', 'hello')
```
ans =
     1
```

Searching in Strings

- findstr() : finds one string within another one
  - test = 'This is a test!';
  - pos = findstr(test, 'is')

```
pos =
     3     6
```

  - pos = findstr(test, ' ')

```
pos =
     5     8    10
```

Replacing in Strings

- strrep() : replaces one string with another
  - s1 = 'This is a good example';
  - s2 = strrep(s1, 'good', 'great')

    s2 =

    This is a great example

String Conversion

- Recall num2str() for numeric-to-string conversion
  - str = ['Plot for x = ' num2str(10.3)]

    ```
    str =
    Plot for x = 10.3
    ```
- str2num(): converts strings containing numbers to numeric form
  - x = str2num('3.1415')

    ```
    x =
        3.1415
    ```

Formatting Data (#’s and/or strings)

- The fprintf(*format*, *data*) function

  Write text to instrument

- The sprintf(*format*, *data*) function

  Format data into string

  – %d integer

  – %f floating point format

  – %e exponential format

  – \n new line character

  – \t tab character

MATLAB Basics: Displaying Data

- fprintf('Result is %d', 3);
  Result is 3
- fprintf('Area of a circle with radius %d is %f', 3, pi*3^2);
  Area of a circle with radius 3 is 28.274334
- x = 5;
- fprintf('x = %3d', x);
  ```
  x =   5
  ```
- x = pi;
- fprintf('x = %0.2f', x);
  x = 3.14
- fprintf('x = %6.2f', x);
  ```
  x =   3.14
  ```
- fprintf('x = %d\ny = %d\n', 3, 13);
  ```
  x = 3
  y = 13
  ```

Why does this not work?

- >> A = 'blue';
- >> B = 'red';
- >> C = 'green';
- >> colors = [A; B; C];

??? Error using ==> vertcat

All rows in the bracketed expression must have the same number of columns.

Why does this not work?

- >> A = 'blue';
- >> B = 'red';
- >> C = 'green';
- >> whos

```
Name        Size        Bytes  Class
A           1x4         8  char array
B           1x3         6  char array
C           1x5         10  char array
```

How do we solve this problem?
I want a list of words and I want to save them as a single variable (i.e., colors, names, …)

SOLUTION: CELL ARRAYS

- >> A = 'blue';
- >> B = 'red';
- >> C = 'green';
- >> colors = {A; B; C};
- >> whos

Curly bracket indicates a cell array

```
Name        Size        Bytes  Class
 A          1x4         8  char array
 B          1x3         6  char array
 C          1x5         10  char array
 color      3x1         336 cell array
```

So, what is a Cell Array?

not the same data types

- A cell is a collection of arrays of various sizes and types.
- >> A = {'string array', [1:.1:100], ones(100,100)}
- >> A{1} = 'string array'
- >> A{2} = [1:.1:100];
- >> A{3} = ones(100,100);

Example: Cell Array

- Use the textread function to read a column of names in the text file *names.txt*

```
Jones
Smith
Collins
Portis
James
```

Variable names is a cell array

```
>> [names] = textread( 'names.txt', '%s' );
>> whos
  Name          Size          Bytes  Class
  names         5x1           576  cell array
Grand total is 33 elements using 576 bytes
>> names{1}
ans = 'Jones'
>> names{2}
ans = 'Smith'
```

MATLAB Basics: Data Files

- File types:
  - Binary files
    - Data is stored in program readable format
    - Processing is fast
  - Text (ASCII) files
    - Data is stored in human readable format
    - Processing is slower
    - Can be used to export/import data that can be used in programs other than MATLAB

MATLAB Basics: Data Files

- save *filename var1 var2* …
  - save homework.mat x y → binary
  - save x.dat .txt, .docx,... → ascii
- load *filename*
  - load filename.mat → binary
  - load x.dat –ascii → ascii

Files-MATLAB

- Opening file
- Reading/writing file
- Closing file

Opening Files

- fid = fopen(filename, permission)
  opens the file *filename* in the mode specified by permission
  - fid is the file id (a positive integer) that is assigned to the file by MATLAB
  - fid is used for all reading, writing and control operations on that file
  - file id 1 is the standard output device and file id 2 is the standard error device
  - fid will contain -1 if the file could not be opened

Opening Files

- Permission can be:
  - ‘r’: open file for reading (default)
  - ‘w’: open file, or create a new file, for writing; discard existing contents, if any
  - ‘a’: open file, or create a new file, for writing; append data to the end of the file
  - ‘r+’: open file for reading and writing
  - ‘w+’: open file, or create a new file, for reading and writing; discard existing contents, if any
  - ‘a+’: open file, or create a new file, for reading and writing; append data to the end of the file
- Add ‘t’ to the permission string for a text file

Opening Files

- Examples: (t, is for text)
  - fid = fopen(‘example.dat’, ‘r’)
    opens a binary file for input
  - fid = fopen(‘example.dat’, ‘wt’)
    opens a text file for output (if example.dat already exists, it will be deleted)
  - fid = fopen(‘example.dat’, ‘at’)
    opens a text file for output (if example.dat already exists, new data will be appended to the end)

Closing Files

- status = fclose(fid)
  closes the file with file id fid
  - If the closing operation is successful, status will be 0
  - If the closing operation is unsuccessful, status will be -1
- status = fclose(‘all’)
  closes all open files (except for standard output and standard error)

Writing Formatted Text Data

- count = fprintf(fid,format,val1,val2,…) writes formatted text data in a user-specified format
  - fid: file id (if fid is missing, data is written to the standard output device (command window)
  - format: same as what we have been using (combination of format specifiers that start with %)
  - count: number of characters written

Writing Formatted Text Data

- Make sure there is a one-to-one correspondence between format specifiers and types of data in variables
- Format strings are scanned from left to right
- Program goes back to the beginning of the format string if there are still values to write (format string is recycled)
- If you want to print the actual % character, you can use %% in the format string

Reading Formatted Text Data

- [array,count] = fscanf(fid,format,size) reads formatted text data in a user-specified format
  - fid: file id
  - format: same as format in fprintf
  - size: same as size in fread
  - array: array that receives the data
  - count: number of elements read

Reading Formatted Text Data

- line = fgetl(fid)
  reads the next line excluding the end-of-line characters from a file as a character string
  - line: character array that receives the data
  - line is set to -1 if fgetl encounters the end of a file

Reading Formatted Text Data

- line = fgets(fid)
  reads the next line including the end-of-line characters from a file as a character string
  - line: character array that receives the data
  - line is set to -1 if fgets encounters the end of a file

The textread Function

- It is designed to read ASCII files that are formatted into columns of data
- Each column can be of a different type
- It is useful for importing tables of data printed out by other applications

The textread Function

- [a,b,c,…] = textread(filename,format,n)
  - filename: a string that is the name of the file to be read
  - format: a string containing the format primitives (just like in fprintf)
  - n: number of lines to read (if not specified, the file is read until the end)

The textread Function

- Example: Assume that you have a file called phones.txt

```
Varol Akman Prof 1538
Selim Aksoy AsstProf 3405
Erol Arkun Prof 2249
Cevdet Aykanat Prof 1625
Mehmet Baray Prof 1208
Cengiz Çelik Instructor 2613
Ilyas Çiçekli AsstProf 1589
David Davenport AsstProf 1248
...
```

The textread Function

- [fname,lname,rank,phone] = textread('phones.txt', '%s %s %s %d’);
  - fname =
    'Varol'
    'Selim'
    'Erol'
    'Cevdet'
    'Mehmet'
    'Cengiz'
    ...
  - phone =
    1537
    3405
    2249
    1625
    1208
    2613
    ...

cell array

double array

The textread Function

- The textread function skips the columns that have an asterisk (*) in the format descriptor
  - [fname, phone] = textread('phones.txt', '%s %*s %*s %d')
- The load command (with ASCII option) assumes all the data is of a single type but textread is more flexible

The textread Function

- Example: Searching for telephone numbers

```
name = 'Selim';
for ii = 1:length(fname),
    if ( strcmp( fname(ii), name ) ),
        disp( phone(ii) );
    end
end
```

END OF LECTURE