CS 105

Introduction to Scientific Computing

Topic #14 – String Manipulation

Matt Burlick
Stevens Institute of Technology

ASSIGNMENT 9

- Process text typed by a user
  - Sum only the numeric entries of comma separated text, i.e
    - Input: ‘4, 5, 6, 0’
      - Output: 15
    - Input ‘4, dkf, 8, fff.0’
      - Output 12

TOPICS

- String Manipulation Functions
- Comparing Strings
- Searching Within Strings

READING

- Section 6.2
- Table 6.3

MANIPULATING STRINGS

- One thing we may want to do to strings is *manipulate* or *alter* them.
- Types of manipulation include:
  - Type Conversion
  - Cleaning
  - Changing cases
  - String Replacement
- While we can (hopefully) write our own algorithms to do these things, Matlab has many nice built-in functions to help us

MANIPULATING STRINGS

- Conversion (we already have been using this this!)
  - xnum=str2num(xstr);
  - xstr=num2str(xnum);
- Cleaning
  - xclean = strtrim('   hello   ');
  - "Trims" the string to remove all extra whitespace
- Changing cases
  - xcap = upper(x);
  - Xlower = lower(x);
- String Replacement
  - Result = strrep(str,srch,repl);
  - Replaces all instances of the string *srch* within *str* with the string *repl*
  - Ex: result = strrep('This is a test of tests', 'test', 'pest');

What we're replacing

Original String

What we're replacing it with

GROUPING STRINGS

- Recall that strings are just *vectors of characters*
  - *‘hello’ == [‘h’, ‘e’, ‘l’, ‘l’, ‘o’]*
- Therefore we can get *substrings* just like we got sub-vectors
  - X = ‘Hello’
  - X2 = X(2:end);
- We can also *concatenate* or *join* vectors by putting them next to each other within brackets
  - X = ‘Hello’;
  - Y = ‘You’;
  - Msg = [X ‘ ‘ Y];

GROUPING STRINGS

- What if we want to create a list strings
  - So each string is on its own row
- This is essentially a *matrix of characters*
  - *str1 = 'Hello'*
  - *str2 = 'You'*
  - Xmat = [str1; str2]
  - But each row must be the same length!
- Matlab has a function *strvcat* (string vertical concatenation) that *pads* strings with spaces so that each row has the same length
  - strings = strvcat('Hello', 'You');

COMPARISON FUNCTIONS

- Let X=‘Hello’ and Y=‘Yecko’
- What would X==Y mean?
  - It would try to compare each character
- What would happen if Y=‘You’?
- For strings we want to compare them *alphabetically*
- Matlab has a built in function that does this
  - res = strcmp(str1, str2);
- You can also ignore the cases
  - strcmpi

SEARCHING FUNCTIONS

- We should be able to write our own function to find a string within a string
- Matlab has this built-in as well
- locs = findstr(str, srchstr)
  - The function *findstr* tells us the locations of **all** instances of srchstr within str
  - It returns an empty vector if there are no instances
- Ex:
  - Position = findstr('This is a test', 'is');

SEARCHING FUNCTIONS

- Ex: Process a string by printing each word on its own line (words are separated by the space character)

SEARCHING FUNCTIONS

- Very often we want to use the locations returned by findstr to create a bunch of substrings
- To help make this somewhat easier, Matlab has a function called *strtok* which stands for "string tokenizing"
- The strtok function returns the characters before the first occurrence of a delimiting character and the rest of the string after the delimiting character
  - [token, remainder] = strtok('This is a test', ' ');
- We can keep calling this **while** the remainder is not empty