Date: 19/08/2024

Experiment 3
Programs on input and output of values

Aim

i) **User Input and Output**

Write a MATLAB script that:

- Prompts the user to enter two numbers.
- Calculates the sum, difference, product, and quotient of the two numbers.
- Displays the results in a formatted manner
- Prompts the user to enter strings and try out the string functions for Concatenation, String Comparison, Substring Operations, Case Conversion, Padding and Trimming,

ii) **Matrix Input and Output**

Write a MATLAB script that: (don't use loop to enter the values)

- Prompts the user to enter the elements of a 2x2 matrix.
- Displays the entered matrix and its transpose.

Theoretical Background

Reading User Input

- *Reading Numeric Values*
  Use the input function to read numeric values from the user. By default, MATLAB treats input as a numeric value unless specified otherwise.
  ```
  >> % Read a scalar numeric value
      userNum = input('Enter a number: ');
  ```
- *Reading Matrices or Arrays*
  To read a matrix or array from the user, you can use the input function and ask the user to input data in MATLAB's matrix format.
  ```
  >> % Read a matrix from the user
     userMatrix = input('Enter a matrix (e.g., [1 2; 3 4]): ');
  ```
  The user needs to provide the matrix in the correct MATLAB syntax.
- *Displaying Numeric Values*
  Use disp or fprintf to display numeric values.
  ```
  >> % Display a number
     disp(['The number you entered is: ', num2str(userNum)]);
  >> % Using fprintf for formatted output
     fprintf('The number you entered is: %.2f\n', userNum);
  ```

- *Displaying Matrices or Arrays*
  Use disp to show matrices and arrays. You can also use fprintf for formatted output, though it's more common for matrices.
  >> % Display a matrix
  disp(userMatrix);
- *Reading User Input as Strings*
  In MATLAB, the input function is used to get input from the user. When reading strings specifically, the input function can take a second argument, 's', to indicate that the input should be treated as a string.
  >> userStr = input('Enter a string: ', 's');
  This line prompts the user to enter a string, and userStr will store the entered string.

Creating and Manipulating Strings

String and Character Arrays

MATLAB provides two primary ways to handle strings:

- **String Arrays**: These are part of MATLAB's string data type introduced in R2016b. They are created using the string function.
  >> str = string(123); % Converts the number 123 to the string "123"

- **Character Arrays**: These are the traditional way of handling strings in MATLAB and are created using the char function.
  >> chr = char("Hello, World!"); % Converts the string to a character array

Concatenation

- **strcat**: This function concatenates strings horizontally. Trailing whitespace is removed in the process.
  >> result = strcat('Hello', ' ', 'World'); % Result: 'Hello World'
- **strjoin**: This function joins elements of a string array into a single string with a specified delimiter.
  >> result = strjoin(["This", "is", "MATLAB"], " "); % Result: "This is MATLAB"

String Comparison

- **strcmp**: This function compares two strings for equality in a case-sensitive manner.
  >> isEqual = strcmp('Hello', 'hello'); % Result: false

- **strcmpi**: This function compares two strings for equality in a case-insensitive manner.
  >> isEqual = strcmpi('Hello', 'hello'); % Result: true

Substring Operations

- **strfind**: This function finds all occurrences of a substring within a string and returns their starting indices.
  >> indices = strfind('This is a test', 'is'); % Result: [3, 6]

- **extractBetween**: This function extracts a substring between two specified substrings.
  >> subStr = extractBetween('This is a test', 'This', 'test'); % Result: " is a "

- **strrep**: This function replaces all occurrences of a substring with another substring.
  >> newStr = strrep('This is a test', 'test', 'demo'); % Result: 'This is a demo'

Case Conversion

- **lower**: Converts all characters in a string to lowercase.
  >> lowerStr = lower('MATLAB'); % Result: 'matlab'

- **upper**: Converts all characters in a string to uppercase.
  >> upperStr = upper('matlab'); % Result: 'MATLAB'

String Length and Splitting

- **strlength**: Returns the number of characters in a string.
  >> len = strlength('MATLAB'); % Result: 6

- **split**: Splits a string into parts using a specified delimiter.
  >> parts = split('one,two,three', ','); % Result: ["one", "two", "three"]

Padding and Trimming

- **strtrim**: Removes leading and trailing whitespace from a string.
  >> trimmedStr = strtrim(' Hello, World! '); % Result: 'Hello, World!'

- **pad**: Adds leading or trailing spaces to a string to achieve a specified length.
  >> paddedStr = pad('Hello', 10); % Result: 'Hello '

String Conversion

- **num2str**: Converts numerical values to a character array (string).
  >> str = num2str(123.45); % Result: '123.45'

- **int2str**: Converts integer values to a character array.
  >> str = int2str(123); % Result: '123'

Pattern Matching and Replacement

- **regexp**: Performs regular expression matching, allowing complex pattern searches within strings.
  >> match = regexp('This is a test', '\w+', 'match'); % Result: {'This', 'is', 'a', 'test'}

- **regexprep**: Replaces substrings that match a regular expression with a specified replacement.
  >> result = regexprep('This is a test', '\s', '_'); % Result: 'This_is_a_test'

Code

i)

```
format("compact")

num1 = input('Enter first number: ');
Enter first number: 1
num2 = input('Enter the second number:
');
Enter the second number: 9

sum = num1 + num2;
diff = num2 - num1;
prod = num1 * num2;
quot = num2/num1;

fprintf('Sum: %f\nDifference:
%f\nProduct: %f\nQuotient: %f\n', sum,
diff, prod, quot);
Sum: 10.000000
Difference: 8.000000
Product: 9.000000
Quotient: 9.000000

str1 = input('Enter string 1: ', 's');
Enter string 1: hello
str2 = input('Enter string 2: ', 's');
Enter string 2: world

concat = strcat(str1, str2);
compare = strcmp(str1, str2);
substr = extractBetween(concat, 'l', 'r');
substra = extractAfter(concat, 'o');
substrb = extractBefore(concat, 'w');
internal.matlab.datatoolsservices.Variabl
eUtils.saveWorkspace
ustr = upper(concat);
lstr = lower(ustr);
padded = pad(concat, 20);
trimmed = strtrim(padded);

fprintf('Concatenated string:
"%s"\nString comparison:
%s\nSubstring: %s\nSubstring after:
%s\nSubstring before: %s\nUppercase:
%s\nLowercase: %s\n', concat,
mat2str(compare), substr{1}, substra,
substrb, ustr, lstr);
Concatenated string: "helloworld"
String comparison: false
```

```
Substring: lowo
Substring after: world
Substring before: hello
Uppercase: HELLOWORLD
Lowercase: helloworld

fprintf('Size of padded string: %d\nSize
of trimmed string: %d',
strlength(padded), strlength(trimmed));
Size of padded string: 20
Size of trimmed string: 10
```

ii)

```
a11 = input('Enter element(1,1): ');
Enter element(1,1): 12
a12 = input('Enter element(1,2): ');
Enter element(1,2): 24
a21 = input('Enter element(2,1): ');
Enter element(2,1): 36
a22 = input('Enter element(2,2): ');
Enter element(2,2): 48

matrix = [a11 a12; a21 a22];
matrixt = matrix';

fprintf('Matrix inputted: ');
Matrix inputted: disp(matrix)
   12    24
   36    48
fprintf('Matrix transpose: ');
Matrix transpose: disp(matrixt)
   12    36
   24    48
```

Conclusion

In MATLAB, user input is handled flexibly using the input function for numeric values, strings, and matrices. Outputs are displayed using disp or fprintf for formatted results. This capability enables interactive and dynamic data handling in MATLAB scripts and functions.