EECS 2110

Professor
Brent Nowlin

Computer Architecture
and Organization

# Project 1: Silly Strings

Using Visual Studio with 32-bit x86 assembly language to write a string-processing program. Your code will obtain a string (max 50 characters) from the keyboard, using either a loop with readchar calls (with screen echo) or by using a call to readstring. The <CR> will terminate the string, and is not a valid character to use inside the string.

Once the string is input, output the user menu to the screen, prompting the user for which function to use. The menu will simply give the function numbers and a brief description for its function. The function definitions, as well as sample outputs, are below. Each function group is to be written as its own programing module, or set of instructions. You may place each function, or group of related functions, in its own procedure or macro if you wish. Regardless of programming choice, ensure that each function group is discernable in your source code by comments or whitespace (for debugging purposes).

Note that each modified function acts on the last modified version of the string. For example, if a character is replaced, then replace the characters in the string, saving the new string as the string to operate on for future functions. Future string modifications will then use this newly-modified string for changes.

**Function 1: Determine the position (0-indexed) of the 1st occurrence of a user-input character (prompt for character input) is in the string. Spaces count as a character.**

Enter a character to determine the first occurrence of: *a*
The first ‘a’ is at position 14 in the string “Independence Day, 2017!”

**Function 2: Find the number of occurrences of a certain letter in a string**

Enter a character to determine the number of occurrences: *e*
The letter ‘e’ occurs 4 times in the string “Independence Day, 2017!”

**Function 3: Find the length of the input string (including spaces).**

There are 23 total characters in the string “Independence Day, 2017!”

**Function 4: Find the number of alphanumeric characters of the input string.**

There are 19 alphanumeric characters in the string “Independence Day, 2017!”

**Function 5: Replace every occurrence of a certain letter with another symbol (case-sensitive; D remains unchanged in the example, while d is replaced)**

Enter a character to replace: *d*
Enter replacement character: *h*
Replacing all of the d’s in the string “Independence Day, 2017!”
With h yields “Inhepenhence Day”

**Function 6: Capitalize letters in the string (non-alphabetic characters unchanged)**

Capitalizing each letter in the string “Independence Day, 2017!”
yields “INDEPENDENCE DAY, 2017!”

**Function 7: Make each letter lower case (non-alphabetic characters unchanged)**
Making each letter lowercase in the string "Independence Day, 2017!"
yields "independence day, 2017!"

**Function 8: Toggle the case of each letter (non-alphabetic characters unchanged)**
Toggling of the letter in the string "Independence Day, 2017!"
yields "iNDEPENDENCE dAY, 2017!"

**Function 9: Undo the last modification to the string, i.e. functions 5-8 and 10. The following example assumes function 8 was the last function to modify the string. (note: only 1-level deep undo's are required, i.e. you only have to undo the last modification, not anything previous to the most recent modification)**
iNDEPENDENCE dAY, 2017!
Reverts back to
Independence Day, 2017!

**Function 10: input a new string**

**Function 0: output the menu again, prompt for new function**

**Function 999: exit program normally**

Key points:

1. Display prompts for any user keyboard input needed.
2. The program must loop back for additional user input (new strings or functions), and provide an exit function, which is function 999.
3. If an invalid function number is entered…
   a. notify the user that the input is invalid,
   b. output the menu again (if necessary),
   c. prompt for function number re-entry,
   d. and resume your function number checks.
4. You may work in teams of 3 students maximum.
5. Deliverables consist of one zip file, submitted by one group member to Blackboard, containing:
   a. your project and source code
   b. Screen shot documentation showing successful build/execution, and successful execution of each function
   c. 1-page write-up describing partner contributions, obstacles and their solutions, cool implementation ideas, and what you learned

Sample user interaction (user entries in *underlined italics*, unprinted notes are in **underlined bold**):

(beginning of interaction)
Please enter a string of at most 50 characters.  Type <enter> to terminate your string.
*I LOVE Assembly Language 2021!*<*enter*>

Please enter a function from the following menu:
….(output function numbers and brief descriptions, including numbers 0, 10, and 999…)
Enter function #:  *2*
Enter the character of which to find the number of occurrences:  *a*
The character a occurs 2 times in the string.  **(note: the capital A doesn't count!)**

Please enter a function from the following menu:
….(output function numbers and brief descriptions, including numbers 0, 10, and 999…)
Enter function #:  *7*
Making all alphabetic characters lower-case in the string yields
i love assembly language 2021!  **(note: numberic/special chars are unchanged)**

Please enter a function from the following menu:
….(output function numbers and brief descriptions, including numbers 0, 10, and 999…)
Enter function #:  *9*
The string reverts back to:
I LOVE Assembly Language 2021!  **(note: the capital E doesn't get replaced)**

Please enter a function from the following menu:
….(output function numbers and brief descriptions, including numbers 0, 10, and 999…)
Enter function #:  *5*
Enter a character to replace: *e*
Enter replacement character: *Z*
Replacing all of the e's in the string with Z yields
I LOVE AssZmbly LanguagZ 2021!  **(note: the capital E doesn't get replaced)**

Please enter a function from the following menu:
….(output function numbers and brief descriptions, including numbers 0, 10, and 999…)
Enter function #:  *51*
The function number entered is invalid.  Enter a valid function number for the menu
(re-output function menu if it's no longer visible from previous menu output)
Enter function #: *10*
Enter new string:
*We TOTALLY LOVE Assembly Language 3/23/2021!*<*enter*>

**… (and so on until user function number = 999)**