RILEY POOLE
ME 477 – LAB REPORT 01
WINTER 2019

DESCRIPTION

The overall purpose of this program is to prompt the user on the LCD display attached to the MyRio to enter a floating-point number on the keypad. This string is then parsed and returned to the console of the PC. The source file is split into three primary functions. The main() function is called to test the operation of the other two functions by calling double_in() twice which in turn uses putchar_lcd().

THE MAJOR TASKS PERFORMED BY THIS PROGRAM WERE

- Prompt the user for input
  - The prompt is easily customizable and is used as a parameter of the input function double_in().
- Receive and parse that input
  - The input from the LCD should be checked for errors alerting the operator if they are detected.
- Return the input to the console of the PC
  - As a simple test return the input from the MyRio to the PC for verification.

THE LIMITATIONS OF THIS PROGRAMS CAPABILITY ARE

- Not all possible error inputs are included in the error tree.
  - for example, there is an error for “..” dots but not “...”
  - the error for dots out of order is undefined (e.g. 1.1.)
- a fundamental flaw in the organization of the program is that it only saves the input string to the buffer after checking for possible errors which means that all possible errors must be defined. It would be better to structure the program such that it only prints if the input is of a form (e.g. floating point two sig figs 01.00.) This however is awkward and constrains the user.

USER CREATED FUNCTIONS OF THE PROGRAM

- **MAIN()**
  - Prototype:
    - int main(void)
  - Purpose:
    - To use the function double_in() and test its operation.
  - Rationale for creation:
    - Need a procedural section to layout testing environment.
  - Inputs and parameters:
    - No inputs or parameters.
  - Return:
    - int 0 for success
  - Calls:
    - double_in()

- **DOUBLE_IN()**
  - Prototype:
    - double double_in(char* prompt)
  - Purpose:
    - Prompt the operator for input on the LCD display. Retrieve the inputted string and parse it as a floating-point value. Then return this floating-point value.
  - Rationale for creation:
    - Need a compact, customizable, and fast way to collect floating point values entered to the MyRio.
  - Inputs and parameters:
    - char* prompt; The customizable input prompt string
  - Return:
    - double; The floating-point value entered to the MyRio
    - error; if an erroneous input is detected then alert the operator on the LCD display.
  - Calls:
    - printf_lcd()
    - fgets_keypad()
    - strstr()
    - strpbrk()
    - sscanf()
    - printf_lcd()

- **PRINTF_LCD()**
  - Prototype:
    - int printf_lcd(const char* format, ...)
  - Purpose:
    - Print a formatted input string to the LCD while making sure that there is no text overflow.
  - Rationale for creation:
    - Need a procedural section to layout testing environment.
  - Inputs and parameters:
    - No inputs or parameters.
  - Return:
    - Int 0 for success
  - Calls:
    - double_in()

HIERARCHICAL STRUCTURE

The overall structure of the program is to break up the calling and outputting tasks into separate functions and then call the two procedurally in main().

Functions Previously found to be implementation ready are marked with an x_

```
main()
      double_in()
double_in()
      my_printf_lcd()
      x_fgets_keypad()
      x_strstr()
      x_strpbrk()
      x_sscanf()
printf_lcd()
      double_in()
```

PSEUDOCODE

```
main ()
{
      initialize variables
      save the results of double_in() to two different variables
      print those variables to the console
}

double_in ()
{
      initialize variables
      print a prompt to the lcd screen
      check the input for errors
      if there are no errors return the value as floating point
}

my_printf_lcd ()
{
      initialize variables
      create a buffer
      print that buffer character by character to the screen
}
```

TESTING

Essential functionality testing should consist of at least:

1. Double in returns an accurate value of the input value to console
    a. run the program
    b. input floating point value with 0 to 6 decimal places on keypad hit <ENTR>
    c. check that #1 your number is displayed on PC console
    d. repeat b and c for #2
2. Double in uses a custom prompt
    a. change the source code at the top for #define PROMPT to a different prompt
    b. re-compile and run the program
    c. check that the prompt is now the new prompt
3. Double in checks for four types of errors

    #1 NO DIGITS
    a. run the program
    b. enter no digits
    c. press <ENTR> on keypad
    d. check that "Short. Try Again" is displayed on keypad
    e. press <ENTR> on keypad to advance

    #2 UP IS BAD KEY
    f. enter <UP> on keypad
    g. press <ENTR>
    h. check that "Up&Down Not Valid" is displayed on keypad
    i. press <ENTR> on keypad to advance

    DOWN IS BAD KEY
    j. enter <DWN> on keypad
    k. check that " Up&Down Not Valid " is displayed on keypad
    l. press <ENTR> on keypad to advance

    #3 - IN WRONG PLACE
    m. enter <1><-><2><3> on keypad
    n. check that "Incorrect – Position" is displayed on keypad
    o. press <ENTR> on keypad to advance

    TOO MANY -
    p. enter <-><-><1><2><3> on keypad
    q. check that "Incorrect – Position " is displayed on keypad
    r. press <ENTR> on keypad to advance

#4 .. IS NOT ALLOWED INPUT
s. enter <1><.><.><2><3> on keypad
t. check that "Too Many “.” " is displayed on keypad
u. press <ENTR> on keypad to advance