ME477, Lab 2

Ali Jones

01/28/2022

Description:

This program tests the function `getchar-keypad()`, which prompts a user for a keypad input, displays that input on an LCD screen, and returns it as a string to the computer. This is achieved with the low-level functions `getkey()` and `putchar_lcd()`, which communicate with the keypad and LCD screen. This program also includes the function `printf_lcd()`, which prints strings to the LCD screen. The overall hierarchical structure of the program is shown below.

```
main
|_ fgets_keypad()       - user keypad entry (acquire input string)
|   |_ getchar_keypad()    - character input(function to test)
|       |_ putchar_lcd() - one char. to LCD output
|       |_ getkey()   - get 1 char from keypad
|       |_ sscanf()   - ascii to decimal conversion
|
|_ printf_lcd()         - print string to LCD display
    |_ putchar_lcd()    - one char. to LCD output
    |_ vsprintf()    - print to string
```

The main function calls `getchar-keypad()` twice, so only allows for two user inputs. It also can’t accept character strings longer than the keypad input, 60 characters. Otherwise, it allows the user full control over inputting numeric values and deleting unwanted inputs.

Testing:

1. Run the program
2. The LCD screen will display the prompt “Enter First Value”.
   a. Hit delete before entering any value. Nothing will happen.
   b. Type any value from the keypad. Special characters ok.
   c. Test the delete functionality by deleting a character.
   d. Hit enter.
   e. The console will print the string you entered.
3. The LCD screen will display the prompt “Enter Second Value”.
   a. Type another value from the keypad. Special characters ok.
   b. Delete the string fully. The cursor will stop after deleting the first character entered.
   c. Re-type your value.
   d. Hit enter.
   e. The console will print the string you entered.

Results:

The `getchar_keypad()` function successfully stores user keypad input in a buffer string until ‘Enter’ is pressed, at which point it returns the input to the main program. This buffer allows the user to delete any keypress before submitting their string – with no previous input, pressing delete does nothing to the keypad; with a previous input, pressing delete erases one character at a time until the cursor returns to the start. The pseudocode for a buffered `getchar_keypad()` with delete functionality is shown below.

```
buffer     // character array of length buf_len
n              // number of characters in buffer
pointer    // points to position in buffer for next character

if n is zero{
    -  set the pointer to the start of the buffer
        while input is not ENTR
            if n < buf_len{
                -  put character in buffer at pointer
                -  increment pointer for next character
                -  increment n
                if character is ‘delete’ key{
                    -  decrement n
                    -  decrement pointer
                    if n is nonzero{
                        -  decrement n
                        -  decrement pointer
                        -  move the cursor back one space
                        -  send a space
                        -  move the cursor back another space
                    }
                }
                if character is not ‘delete’ key{
                    -  put character on lcd screen
                }
            }
        }
    - increment n
    - set pointer to the start of the buffer
}

if n is greater than 1{
    -   decrement n
    -   return the character pointed to
    -   increment the pointer
}

else{
    -   decrement n
    -   return EOF
}
```

This function could be improved by adding a feature that allows the user to clear the screen (e.g by pressing ‘delete’ twice, or by pressing ‘delete’ and ‘minus’ simultaneously) or restart the input process entirely.