Lab 2 - getchar_keypad()

1. Description

In this laboratory, we created getchar_keypad() subroutine that obtains characters from the keypad from the user. The main takeaway from this lab is pointer manipulation, and the idea of a buffer. The subroutine getchar_keypad() reserves a static buffer containing the characters from the user input. Through incrementing the buffer pointer and using keeping track of the buffer size, we can implement a control algorithm that obtains the keypad value, puts it on the LCD screen, and updates the buffer for every keypad press.

The subroutine will stop prompting for input once the ENT key has been pressed. The buffer is editable by pressing the DEL key. Implementing the DEL and ENT button involves using if/else algorithm to catch the key presses. While loop is used to keep on prompting for characters as long as ENT button is not pressed. If the subroutine is called the second time and the buffer is not empty, it will print the next character in the buffer.

The subroutine getchar_keypad() will be tested through printing the result of fgets_keypad() in main. The subroutine fgets_keypad() uses getchar_keypad() to prompt for one keypad presses at a time and returning the character. The subroutine is successful if fgets_keypad() returns the user input precisely.

The limitation of this program is the length of the buffer, which is set to 40. Therefore, the character input is limited to 40 characters for this subroutine.

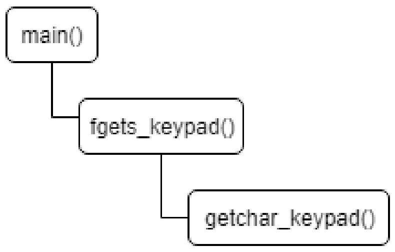

Figure 1: Hierarchical Diagram

2. Testing

1. Run main.c, (this will call fgets_keypad() twice)
2. The screen should show: Lab2, then prompts the user input in the next line
3. Try pressing the keypad and see if the numbers are stored and shown in the LCD
4. Try deleting the characters to edit the buffer
5. Delete the buffer until it is empty, check if extra deletion causes the deletion of the previous characters (it should not)
6. Press enter after some character has been inputted. The screen should print exactly the same characters as the one inputted previously
7. Repeat the same thing from step 3 to ensure that fgets_keypad() works twice
8. If nothing is entered when getchar_keypad() is called, null is returned

3. Results

The main() routine passes all the tests specified above. We were able to input an array of characters, and we were able to delete any character inputs. Pressing the DEL key will modify the buffer and deletes the current character in the LCD screen. In addition, pressing DEL when there is no user input will not cause the deletion of the previous characters.

Calling fgets_keypad() also works normally like how it should be tested in the previous test, which means that getchar_keypad() integrates properly to higher level subroutines.

The limitation of 40 characters in the buffer still holds, such that after the buffer is full, the user is not able to input any more characters.

When the DEL key is pressed when n = 0, a while loop ensures that the cursor stays in its position, and re-prompts another key while unchanging the buffer.

One way to improve this code is to increase the buffer size. We can also add a class constant to specify the buffer length so that the code can be modified more easily and this will improve the code's readability.