Alex Surprenant

CS-405

7/12/2025

For my approach to this activity, my goal was to prevent the user from being able to enter input that would exceed the limit of 20 characters. This limit includes the null terminator that occurs in the C style declaration of user\_input, so the true limit of the user input is 19 characters. I decided to use an input flag Boolean to allow the program to jump into and out of a while loop when the correct conditions are met.

Initially, input\_flag is true, and the program jumps into the loop. From there, the flag is set to false, and the user is prompted for input. I use the .getline() method, which takes the input, and an input limit as parameters. This method allows the input to be read up to the point when a buffer overflow would occur, and then stop reading so that the buffer does not overflow.

When an overflow would occur, std::cin.fail() is triggered, so I made an if statement that will clean the error state. I then use std::cin.ignore() with std::numeric\_limits<std::streamsize>::max() as the first parameter, which allows the input stream to ignore all characters up to a limit of how many can possibly be put in the stream, and the ‘\n’ as the second parameter, telling the stream to stop searching once it hits the new line.

Since we have gone into what would be a potential overflow, input\_flag is set back to true, and the while loop will trigger again to ask the user for input. If the input is the correct length, the flag will stay false and the loop will exit, outputting the final to outputs and prompting the user to end the program.

Using the while loop with a nested if loop allows the program to execute based on specific and controlled conditions, and makes the program easier to maintain and understand.

A screenshot of a computer

AI-generated content may be incorrect.