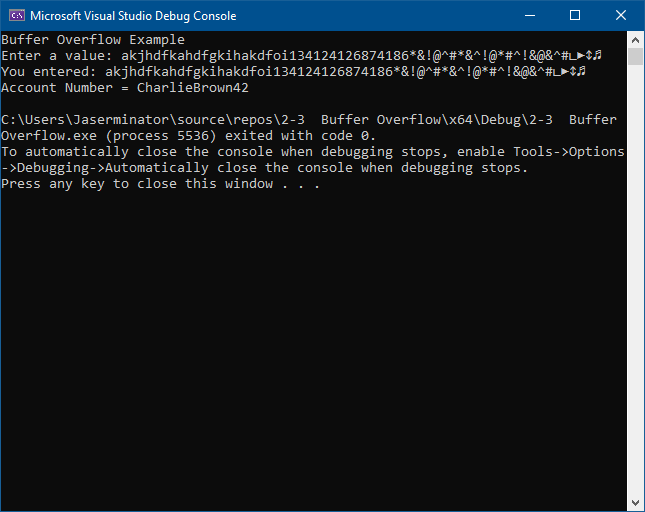
**Code Summary**

In the provided code, I was tasked with ensuring that user input would not overflow the array that would contain the user input. An overflow would compromise the security of the application, possibly altering the account number. To prevent the overflow, I used the C++ standard library’s std::string type which is a specialization of C++ std::basic\_string template (Seacord, 2013). The std::string type uses dynamic memory which automatically resizes the array to allow enough room for user input, thereby preventing a buffer overflow and making the code secure. The assignment did not have any further special requirements, such as restricting input to 20 characters in length or disallowing the use of dynamic memory (as would be needed for critical-safety systems) so std::string was sufficient (Seacord, 2013). If the input needed to be restricted to a certain length, there are a number of solutions. One might be a while loop that checks that the input length is within 20 characters, and if not, clear the std input stream, set it to a valid state, let the user know what went wrong, and ask for input again. If dynamic memory was disallowed, a good alternative is the use of fgets() which acts much like a file input stream and each character can be read one at a time until the maximum read input has been reached. In this way, you can also detect if the user attempted to put more than the maximum allowed by checking for a newline character in the array. If there is no newline, than the input was truncated (Seacord, 2013).

*No overflow with long string*



**References**

Seacord, R. C. (20130323). Secure Coding in C and C, 2nd Edition. VitalSource Bookshelf version 10.2.26.0. Retrieved from vbk://9780132981972