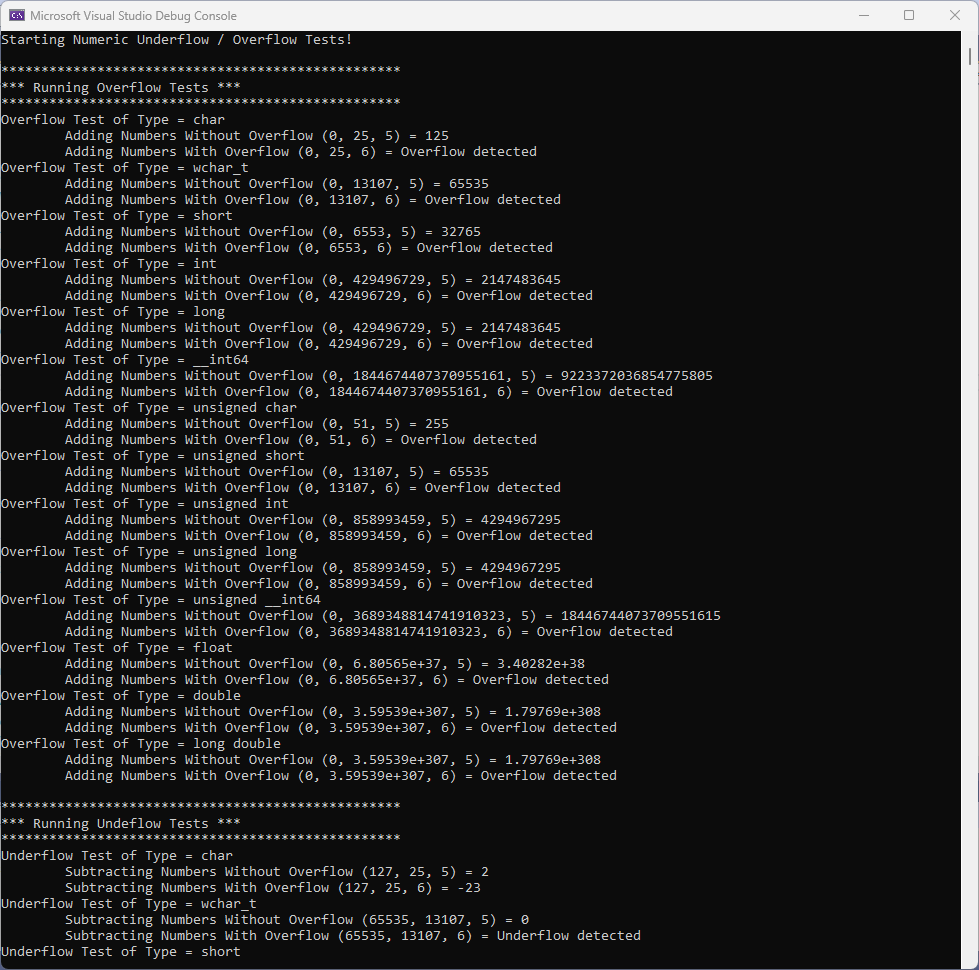
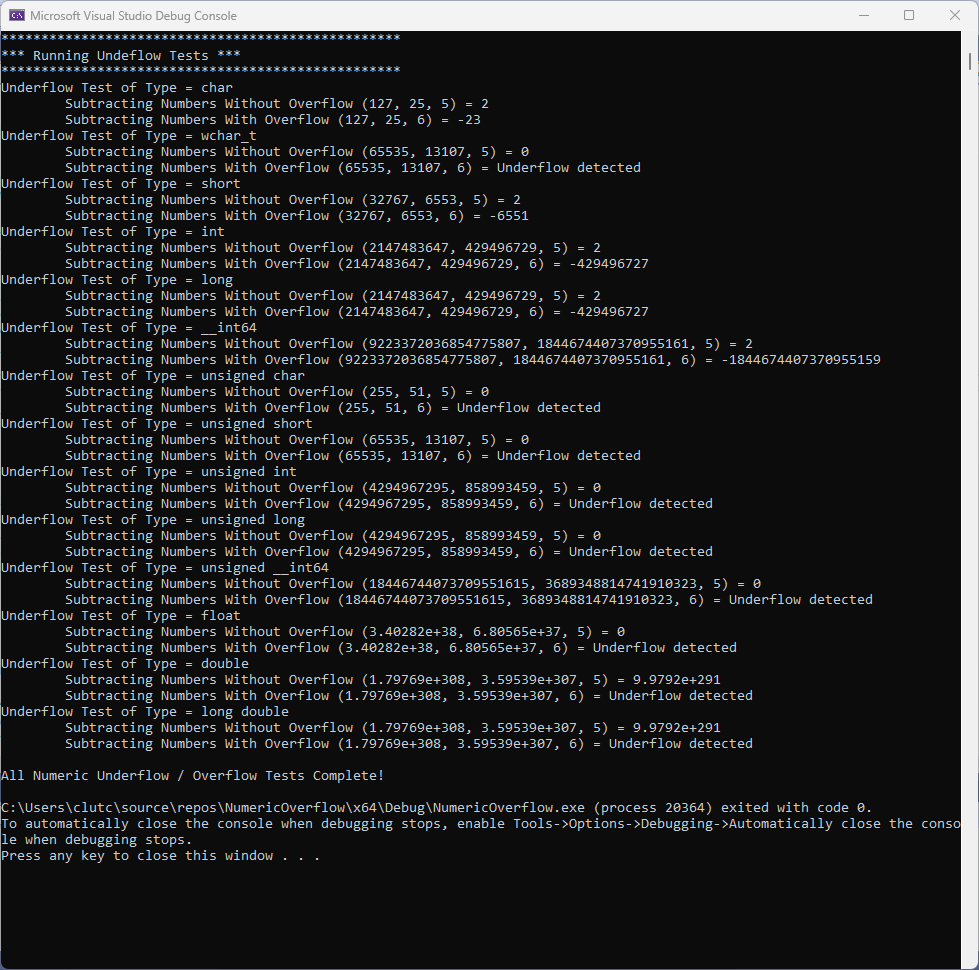
Brian Engel

Module 1 Numeric Overflow





In order to detect overflow I created a variable for the maximum value for the type passed in. My initial thought was to compare the result + increment to the maxValue but that would have been an overflow in itself, so I rearranged the inequality to result > maxValue – increment so everything is under the maximum value for the type. If an overflow is detected by the result being greater than maxValue – increment it returns the original starting value, otherwise there is no overflow, and the result is returned. I then created a function to use in test\_overflow called displayOverflowResults that takes the result and start as arguments. This function basically just compares the two numbers and if they are the same an overflow occurred and “Overflow detected” is displayed and if they are different no overflow occurred and the result is displayed.

I handled the underflow almost exactly the same as the overflow, except switched out min for max, < for >, and added decrement instead of subtract increment. There is a logic error in the actual testing for the underflow though in the amount that decrement is set for.

It works fine for unsigned number types, but doesn’t take negative numbers into consideration. It should be something like ”const T decrement = (std::numeric\_limits<T>::max() - std::numeric\_limits<T>::min()) / steps;” if it is intended to work like the test\_overflow function.



Unfortunately this is in the do not change section, so it has not been fixed.