Name: Alex Maass

NetID: am838

CS 2024: C++

Bonus Assignment: Binary Stream Manipulator

CS 2024: Bonus Assignment Report:

For this assignment, we were asked to build a custom “sticky” stream manipulator called binary. If the stream manipulator binary were called, any subsequent int we wanted to be printed out would be converted to a binary string representing that int. This stream manipulator should be sticky so that every time after we called binary just once, it would still print out all ints in binary format. We also had to write another custom “sticky” stream manipulator called nobinary, which would essentially turn off binary and print out ints normally.

Building a custom stream manipulator that is “sticky” is a very challenging thing to do. After looking around, it seems that the most common way to do it is by setting a state in the output streams. Effectively what I would do is catch the output stream operator when an int is called and then check the stream’s state to see if the binary stream manipulator was turned on or not. If it is turned on, change the output of the operator to be the binary format of the initial number. If the flag is not set, the int should not be modified.

Looking deep into the IOStream classes, it seems that these states can be stored using the function ios\_base::iword() and an index given by ios\_base::xalloc(). Whenever the binary or the nobinary stream manipulator is called, the stream manipulator goes into the stream’s iword field and sets the state depending on whether binary is being turned on or off. By default the iword() function returns 0 to specify that the state is off. Now that there is a way to set a state in the stream by using iword() and xalloc(), all that is left is to actually modify the number when the flag is set. Since we are dealing with ints, we can customize the way integers get formatted through the std::do\_put function. This is done by creating a num\_put class that inherits from the original num\_put class and then overriding the do\_put function. In the do\_put function, we check to see if the state in the stream is on or not, and if it is on, change the return to be the string equivalent of the binary format of the input int. If the state is not on, the normal do\_put function of the superclass should be called and everything should happen as if it were a normal print out. Once the overriding do\_put function is in place, a locale object must be created. Finally, ios\_base::imbue() must be called to imbue the stream with the new locale.

This assignment gave me a deeper look into how core functions such as streams work in C++. Stream manipulators are very complicated and there is a lot of code behind each stream call that has to be manipulated for a custom stream manipulator. Also, since the assignment was only to convert ints (not longs, decimals, etc) to binary, I was able to take some simple shortcuts, such as dealing with iword() instead of pword(). Essentially, I was able to take deeper look into streams and see how deep the rabbit hole goes for even such a simple thing as printing using cout.