PS1 Linear Feedback Shift Register and PhotoMagic

PS1a and PS1b were the two halves of this project. PS1a required us to create a Fibonacci linear feedback shift register (LFSR) that would generate pseudo-random bits for use in PS1b picture encryption. After shifting each bit once to the left, the LFSR had to take a 16-bit binary seed and XOR the designated tap bits from its current state to create the new rightmost bit of its next state. PS1b would then have a main function that would accept an image file as input, encrypt its pixel data using the LFSR, and output the result.

Key Concepts

The FibLFSR class is a container for a dynamic array with a size equal to the length of the binary seed provided to it during creation; however, if the string is not precisely 16 bits long, an error is thrown. Any exceptions thrown by FibLFSR will be handled by the PhotoMagic class. The required copy/move/destructor functions are provided in the class since FibLFSR has dynamically allocated memory.

What I accomplished

PS1a: FibLFSR stores a series of shifted bits to the left and acquires the rightmost bit using tap bits. To store the provided seed, I utilized a dynamic array of integers (input bits). The class provides a generate() function that steps through the LFSR using step() as a helper function. I utilized the Boost C++ libraries to execute test cases on the FibLFSR class for this section of the assignment, which is specified in test.cpp. I double-checked that both the step() and create() methods returned the appropriate results, and I looked for an error if the given seed was shorter or longer than the register's needed 16 bits.

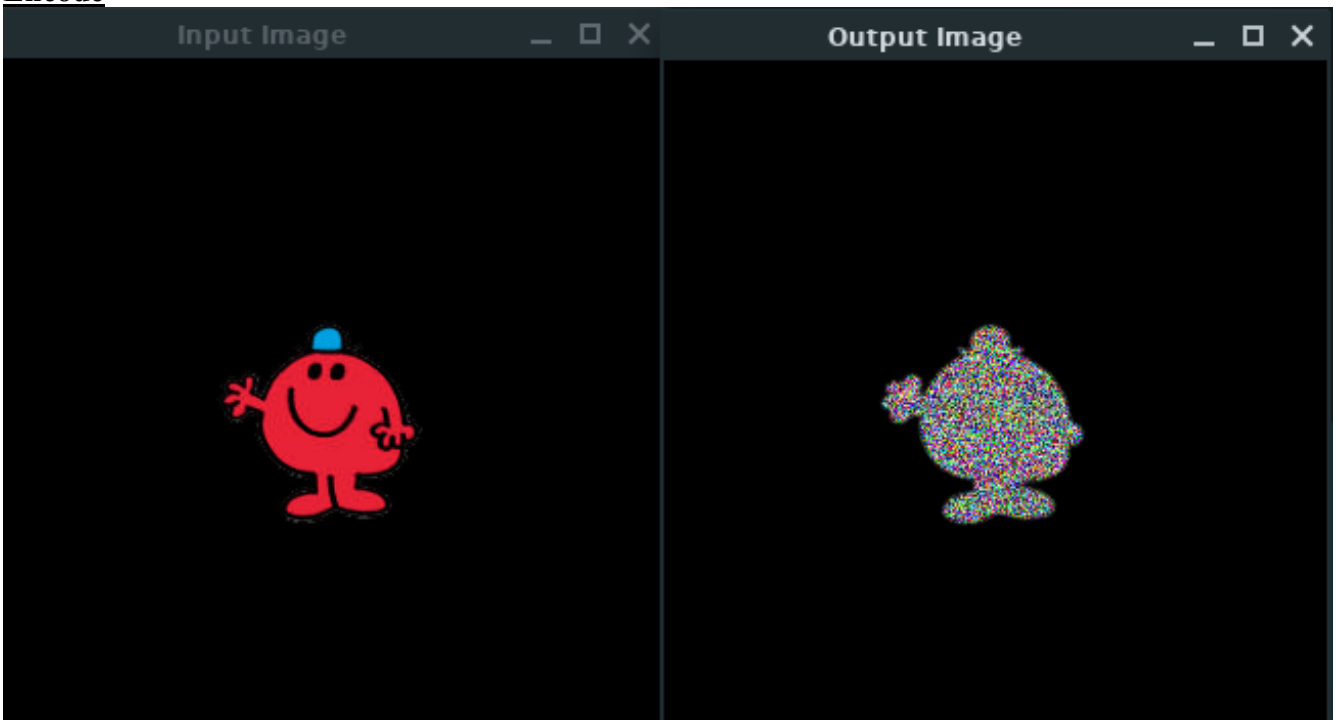
PS1b: The input filename, output filename, and a 16-bit binary seed are all required command line parameters for PhotoMagic. After entering the command, a FibLFSR object is formed, and the transform() function on the picture is invoked, which encrypts or decrypts images by XORing pixel data with register bits. The software additionally double-checks that all command-line options are accurate. Two new windows will appear in the SFML display loop, displaying both the original and the encrypted/decrypted picture.

What I Learned

This project provided a fantastic opportunity for me to revisit some of the C++ programming methods I had previously studied, such as the rule of 5, operator overloading, exception handling, and SFML. In addition, while my FibLFSR class was still in development, I learnt how to leverage the Boost libraries to quickly perform unit tests on it. Unit test are my biggest take away from this assignment, as it also helped me at work developing them to test validation annotations on some of the DTOs I've been reconstructing.

Output

Encode



Decode