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.

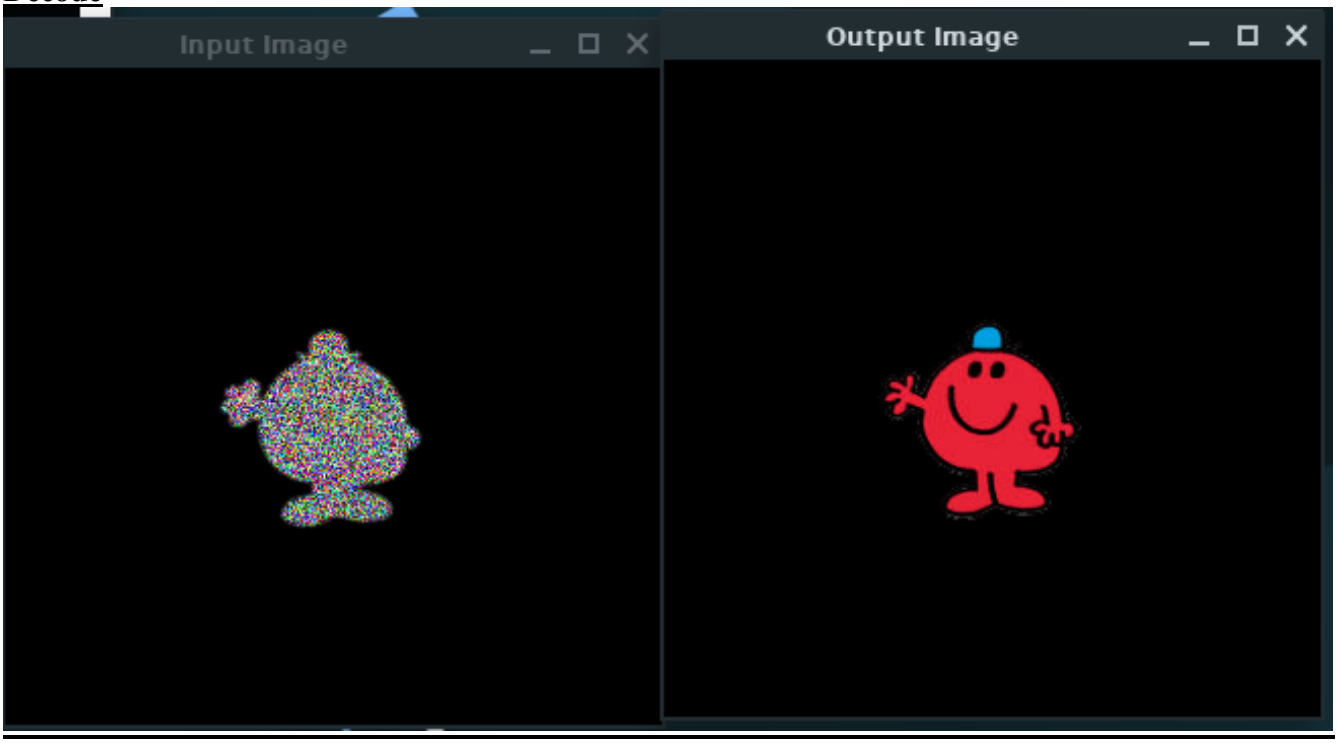
What I Learned

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



```
 1: CC= g++
 2: CFLAGS= -Wall -Werror -ansi -pedantic
 3: SFMLFLAGS = -lsfml-graphics -lsfml-window -lsfml-system
 4:
 5: all:
 6:         make PhotoMagic
 7:
 8: PhotoMagic: PhotoMagic.o FibLFSR.o
 9:         $(CC) PhotoMagic.o FibLFSR.o -o PhotoMagic $(SFMLFLAGS)
10:
11: PhotoMagic.o: PhotoMagic.cpp FibLFSR.h
12:         $(CC) $(CFLAGS) -c PhotoMagic.cpp FibLFSR.h
13:
14: FibLFSR.o: FibLFSR.cpp FibLFSR.h
15:         $(CC) -c FibLFSR.cpp
16:
17: clean:
18:         $(RM) *.o
19:         $(RM) PhotoMagic
```

```
    1: /*
    2: Computing IV - Assignment - PS1a + b
    3: Instructor: Prof. Yelena Rykalova
    4: Due Date: 02/07/22
    5: Author: Matthew Lorette Anaya
    6: Description: This program is an implementation of a Fibonacci Linear Feed
back                Shift Register
    7:              This is the implementation of the PhotoMagic.class which tak
es thre             e arguments an input image an output image and a seed. The p
rogram              uses the the seed to encode the input image and display it a
s the o             utput image.
    8: */
    9: #include <iostream>
   10: #include <string>
   11: #include <sstream>
   12: #include <SFML/System.hpp>
   13: #include <SFML/Window.hpp>
   14: #include <SFML/Graphics.hpp>
   15: #include "FibLFSR.h"
   16:
   17: void transform( sf::Image& img, FibLFSR* bit_generator) {
   18:   // randomize the bits in the image
   19:   sf::Vector2u imgsize = img.getSize();
   20:   // initialize an SFML pixel
   21:   sf::Color p;
   22:
   23:   for(int x = 0; x < (signed)imgsize.x; x++) {
   24:     for(int y = 0; y < (signed)imgsize.y; y++) {
   25:       // get the current pixel from the input image
   26:       p = img.getPixel(x, y);
   27:
   28:       // generate encoded pixels
   29:       p.r = p.r ^ bit_generator -> generate(8);
   30:       p.g = p.g ^ bit_generator -> generate(8);
   31:       p.b = p.b ^ bit_generator -> generate(8);
   32:
   33:     // edit the image in-place with new encoded pixels
   34:       img.setPixel(x, y, p);
   35:     }
   36:   }
   37: }
   38: int main(int argc, char* argv[]) {
   39:   if(argc != 4) {
   40:     std::cout << "Incorrect Input Format" << std::endl
   41:               << "Input should be as follows: ./PhotoMagic <inputfilename
> <outputfilename> <seed>\n";
   42:     return -1;
   43:   }
   44:
   45:   // store input in variables
   46:   std::string input_fname(argv[1]);
   47:   std::string output_fname(argv[2]);
   48:   std::string seed = argv[3];
   49:
   50:   // create an LSFR object
   51:   FibLFSR bit_generator(seed);
   52:
   53:   // load images
   54:   sf::Image input_image;
   55:   if (!input_image.loadFromFile(input_fname)) {
   56:       return -1;
   57:   }
   58:
   59:   sf::Image output_image;
   60:   if (!output_image.loadFromFile(input_fname)) {
```

```
   61:       return -1;
   62:   }
   63:
   64:   // display 2 windows
   65:   sf::Vector2u imgsize = input_image.getSize();
   66:   sf::RenderWindow input_window(sf::VideoMode(imgsize.x, imgsize.y), "Inp
ut Image");
   67:   sf::RenderWindow output_window(sf::VideoMode(imgsize.x, imgsize.y), "Ou
tput Image");
   68:
   69:   // load the images into textures
   70:   sf::Texture in_texture, out_texture;
   71:   in_texture.loadFromImage(input_image);
   72:
   73:   transform(input_image, &bit_generator);
   74:
   75:   out_texture.loadFromImage(input_image);
   76:
   77:   // load textures -> sprites
   78:   sf::Sprite in_sprite, out_sprite;
   79:   in_sprite.setTexture(in_texture);
   80:   out_sprite.setTexture(out_texture);
   81:
   82:   // main loop
   83:   while (input_window.isOpen() && output_window.isOpen()) {
   84:       sf::Event event;
   85:
   86:       while (input_window.pollEvent(event)) {
   87:           if (event.type == sf::Event::Closed) {
   88:               input_window.close();
   89:             }
   90:         }
   91:
   92:       while (output_window.pollEvent(event)) {
   93:           if (event.type == sf::Event::Closed) {
   94:               output_window.close();
   95:             }
   96:         }
   97:
   98:       input_window.clear();
   99:       input_window.draw(in_sprite);    // Input image
  100:       input_window.display();
  101:
  102:       output_window.clear();
  103:       output_window.draw(out_sprite);    // Output image
  104:       output_window.display();
  105:     }
  106:
  107:   // save the image
  108:   if (!input_image.saveToFile(output_fname)) {
  109:       return -1;
  110:     }
  111:
  112:   return 0;
  113: }
```

```
    1: /*
    2: Computing IV - Assignment - PS1a + b
    3: Instructor: Prof. Yelena Rykalova
    4: Due Date: 02/07/22
    5: Author: Matthew Lorette Anaya
    6: Description: This program is an implementation of a Fibonacci Linear Feed
back                Shift Register
    7:              This is a header file with the FibLFSR class definition
    8: */
    9:
   10: #include <iostream>
   11:
   12: class FibLFSR {
   13:
   14:  public:
   15:   FibLFSR(std::string seed);
   16:
   17:   int step();
   18:
   19:   int generate(int k);
   20:
   21:   friend std::ostream& operator<<(std::ostream& os, FibLFSR &lfsr);
   22:
   23:  private:
   24:   std::string reg;
   25:
   26:   int getBit(char a);
   27:
   28:   int xOr(int a, int b);
   29:
   30: };
   31:
   32:
```

```
    1: /*
    2: Computing IV - Assignment - PS1a + b
    3: Instructor: Prof. Yelena Rykalova
    4: Due Date: 02/07/22
    5: Author: Matthew Lorette Anaya
    6: Description: This program is an implementation of a Fibonacci Linear Feed
back                Shift Register
    7:              Takes in a seed and generates bits with seed() and numbers w
ith g               enerate(int)
    8: */
    9: #include <string>
   10: #include <sstream>
   11: #include <math.h>
   12: #include "FibLFSR.h"
   13:
   14: FibLFSR::FibLFSR(std::string seed) {
   15:   int size = seed.length();
   16:   // No try-catchblock for BOOST test
   17:   if(size != 16)
   18:     throw std::invalid_argument("Incorect seed bit length, must be 16.");
   19:   reg = seed;
   20: }
   21:
   22: int FibLFSR::getBit(char a) {
   23:   if (a == '1') return 1;
   24:   else if (a == '0') return 0;
   25:   else return 1;
   26: }
   27:
   28: int FibLFSR::xOr(int a, int b) {
   29:   return a != b;
   30: }
   31:
   32: std::ostream& operator<<(std::ostream& os, FibLFSR &lfsr) {
   33:   os << lfsr.reg;
   34:
   35:   return os;
   36: }
   37:
   38: int FibLFSR::step() {
   39:
   40:   //new register after shifting
   41:   std::string new_reg = reg.substr(1);
   42:
   43:   //Taps(10, 12, and 13)
   44:   //{Equal = 0}{Not Equal = 1}
   45:   int tap = xOr(reg[0], reg[2]);
   46:   tap = xOr(tap, getBit(reg[3]));
   47:   tap = xOr(tap, getBit(reg[5]));
   48:
   49:   FibLFSR::reg = new_reg;
   50:   FibLFSR::reg += std::to_string(tap);
   51:
   52:   return tap;
   53: }
   54:
   55: int FibLFSR::generate(int k) {
   56:   int result = 0;
   57:   for(int i = 0; i < k; i++){
   58:     int z = step();
   59:     result = (result * 2) + z;
   60:   }
   61:
   62:   return result;
   63: }
```

```
64:
65:
66:
```

```
    1: // Dr. Rykalova
    2: // test.cpp for PS1a
    3: // updated 1/31/2020
    4: /*
    5: *  Computing IV - Assignment - PS1a
    6: *  Instructor: Prof. Yelena Rykalova
    7: *
    8: *  Due Date: 01/31/22
    9: *
   10: *  Author: Matthew Lorette Anaya
   11: *
   12: *  Description: This program is an implementation of a Fibonacci Linear F
eedback Shift Register
   13:                 Takes in a seed and generates bits with seed() and number
s with generate(int)
   14:                 This is the test file with BOOST unit tests.
   15: */
   16:
   17:
   18: #include <iostream>
   19: #include <string>
   20:
   21: #include "FibLFSR.h"
   22:
   23: #define BOOST_TEST_DYN_LINK
   24: #define BOOST_TEST_MODULE Main
   25: #include <boost/test/unit_test.hpp>
   26:
   27: BOOST_AUTO_TEST_CASE(sixteenBitsThreeTaps) {
   28:
   29:   FibLFSR l("1011011000110110");
   30:   BOOST_REQUIRE(l.step() == 0);
   31:   BOOST_REQUIRE(l.step() == 0);
   32:   BOOST_REQUIRE(l.step() == 0);
   33:   BOOST_REQUIRE(l.step() == 1);
   34:   BOOST_REQUIRE(l.step() == 1);
   35:   BOOST_REQUIRE(l.step() == 0);
   36:   BOOST_REQUIRE(l.step() == 0);
   37:   BOOST_REQUIRE(l.step() == 1);
   38:
   39:   FibLFSR l2("1011011000110110");
   40:   BOOST_REQUIRE(l2.generate(9) == 51);
   41: }
   42:
   43:   // Test case that prints out the starting and the resulting bit
   44:   // patterns whilst checking to make sure the correct result is printed
   45: BOOST_AUTO_TEST_CASE(customTestCase1) {
   46:   std::cout << "\n------Custom Test Case 1------" << std::endl;
   47:   FibLFSR l("1011011000110110");
   48:   std::cout << "\tOriginal seed: " << l << std::endl;
   49:
   50:   int res = l.generate(5);
   51:   BOOST_REQUIRE(res == 3);
   52:
   53:   std::cout << "Results of generate(5): " << l << " " << res << std::endl
;
   54:   std::cout << std::endl;
   55: }
   56:
   57: BOOST_AUTO_TEST_CASE(customTestCase2) {
   58:   std::cout << "\n------Custom Test Case 2------" << std::endl;
   59:
   60:   std::string tooShort = "10010110";
   61:   std::string tooLong = "100110010010101101101";
   62:
```

```
   63:   std::cout << "Test exception thrown for too short seed: 10010110" << st
d::endl;
   64:   BOOST_REQUIRE_THROW(FibLFSR("10010110"), std::invalid_argument);
   65:
   66:  std::cout << "Test exception thrown for too long seed: 10011001001010101
101" << std::endl;
   67:  BOOST_REQUIRE_THROW(FibLFSR("100110010010101101"), std::invalid_argume
nt);
   68: }
   69:
```