CS 201 Homework 07

Bryan Beus

November 25, 2019

Source Code Link: `https://github.com/siddhartha-crypto/cs201/tree/master/hw7`

1 Design

1.1 Ascii Art

My intention with this project is to learn from the overall structure laid out by Dr. Metzgar. Were I attempt to do this work on my own, I would likely not know how to format the code as professionally.

1.2 Mandelbrot

This project intrigues me because I have seen fractal mathematics for years, and would like to understand more about how it works. The software application comes with premade pseudocode, and therefore I expect to follow the outline provided.

1.3 Caesar Cypher

This project is similar to many of the projects previously completed in class, and therefore I expect to reuse (and therefore improve my mastery over) many concepts regarding input and output taught in class. The cypher itself I intend to achieve by duplicating the Alphabet three times, in both upper and lower case,

and setting the (modulo/remainder) amount of integers to move the letter equal to the middle set of the alphabet (for both upper and lower case), and then move the final letter according to the user input.

2 Post Mortem

2.1 Ascii Art

This was an extremely challenging assignment. I could not have finished the work without help from Izac, our fellow classmate. He was kind enough to explain to me the expectations regarding image formatting and many other concepts required for this assignment.

The first challenge was to accomplish the operator customization. Dr. Metzgar assisted here, and once I understood the conceptual side of this task, it was fairly easy to repeat throughout the many iterations in the provided template.

The usage of classes and the mathematics behind `rgb` manipulation may seem simple to a professional or a more experienced user, but for me, I needed a lot of help to understand concepts such as accessing members within functions and more.

2.2 Mandelbrot

The provided outline made most of this assignment relatively straightforward. However, I found the aspect of scaling the initial `x0, y0` values conceptually challenging, for some reason (other students did not), and I also was confused until the end regarding the nature of using the `iterator` value to obtain rgb data. Both Henryk and Izac helped me to understand the concepts here, and I will need to continue developing my understanding.

2.3 Caesar Cypher

This project was the easiest, and the only project where I did not need help from anyone else, and was even able to offer suggestions and ideas to other students. I came up with the simple solution of the three-times-repeated alphabet and using `modulo` to swiftly obtain the encrypted letter.

3 Answers to Questions

- A sequence of characters that is given a name and stored in a mass storage device. Opening a file is the act of connecting to the location in memory, retrieving the data, and making it available at least for reading, if not also manipulation and writing.
- A program should always ensure that no errors occured in the act of opening a file.
- The program will automatically close the file when the program ceases to run. We can, if needed, use `fout.close()` on an `fout` stream.
- When inputting a file, many things can go wrong. The file may not be formatted as expected, there may be corruption or data that was improperly stored, and any other number of inconsistencies with the expected data. Assuming the data loads as expected, we have greater control over how to store the data in the end.
- Valid data is data that meets our program expectations.
- Because when we're at the last item, it flags `eof` as true. To make sure we process the final item, we make sure that there is an error. This will ensure that it is truly the last item.
- A pointer is a variable that points directly to an address, as opposed to the value that the address represents. The address is the location where the value is stored in memory.

A null pointer is a pointer that points to nothing, and we should initialize pointers to point to a null pointer to ensure that the pointer is not pointing to an unexpected value (which would result in undefined behavior).

- Initialize: `int *ptr = nullptr;` Dereference: `*ptr;`
- Because a vector is a highly manipulatable container that allows us to easily create iterators and access many other useful tools associated with the standard library.
-
  - `intarr[100];`
  - `int * ip1 = &(arr[8]`
  - `auto ip2 = ip1 + 10`
- The distance between two pointers is simply the distance between their indices. The distances between two pointers can easily be calculated if they are in the same container. If not in the same container, generally we cannot user tools like iterators to calculate distance between them.

4 Sample Output

4.1 Ascii Art

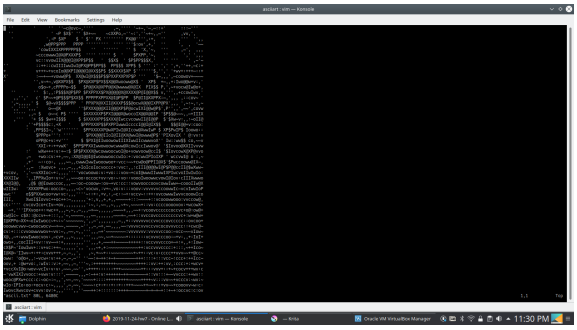

Figure 1: Ascii Art

4.2 Mandelbrot

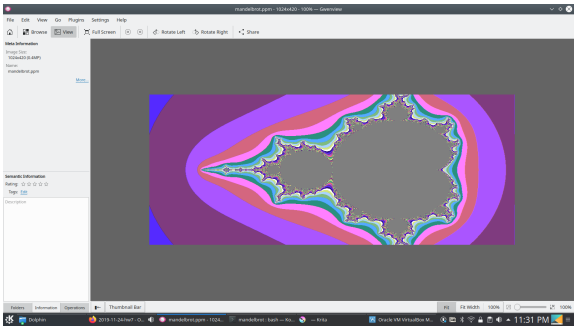

Figure 2: Mandelbrot

4.3 Caeser Cypher

```
Enter a phrase to encrypt (blank to quit): Hello, everyone!

By how many digits would you like to encrypt the phrase? 3


The new message is: Khoor, hyhubrqh!

Enter a phrase to encrypt (blank to quit): Khoor, hyhubrqh!

By how many digits would you like to encrypt the phrase? -3


The new message is: Hello, everyone!

Enter a phrase to encrypt (blank to quit):


Program complete
```

5 My Programs

5.1 Ascii Art

```
/*
 * main.cpp
```

```
 * Bryan Beus
 * November 24, 2019
 * The main file for the ascii art project
 */

#include <iostream>
#include <vector>
#include <fstream>
#include <sstream>

#include "Color3.hpp"
#include "Image3.hpp"

using std::cout;
using std::cin;
using std::endl;

using std::ifstream;
using std::ofstream;

using std::vector;
using std::string;

int main() {

  // Declare file paths
  string input_path = "parrot.ppm";
  string output_path = "test.ppm";
  string ascii_output_path = "ascii.txt";

  // Create empty image object for manipulation
  Image3 image(0, 0);

  // Create input stream from input path
  ifstream fin2(input_path);

  if (!fin2)
    cout << "no fin" << endl;

  // Import the file into the image object
  fin2 >> image;

  // Ensure fin2 is closed
  fin2.close();

  // Create output path for the test PPM
  // This allows us to ensure that the file is loaded and writing
   ↪  correctly
  ofstream fout2(output_path);

  if (!fout2)
    cout << "no fout" << endl;

  // Write the image object to the file
```

```
  fout2 << image;

  // Ensure output is closed
  fout2.close();

  // Create output stream for ascii art
  ofstream fout3(ascii_output_path);

  if (!fout3)
    cout << "no fout3" << endl;


  // Print the ascii art to the file
  image.printASCII(fout3);

  return 0;
}
```

5.2 Mandelbrot

```
/*
 * main.cpp
 * Bryan Beus
 * November 24, 2019
 * The main file for the mandelbrot project in hw7
 */

#include <cmath>
#include <iostream>
#include <vector>
#include <fstream>
#include <sstream>

#include "Color3.hpp"
#include "Image3.hpp"

using std::cout;
using std::cin;
using std::endl;

using std::ifstream;
using std::ofstream;

using std::vector;
using std::string;

// Process the mandelbrot image
void process_mandelbrot(Image3& image);

int main() {

```

```
32    // Set image dimesions
33    double scale = 0.2;
34    unsigned width = (unsigned)(std::round(5120 * scale));
35    unsigned height = (unsigned)(std::round(2098 * scale));
36
37    // Create default image
38    Image3 image(width, height);
39
40    // Process the image with mandelbrot calculations
41    process_mandelbrot(image);
42
43    // Create output stream and save file
44    string output_file("mandelbrot.ppm");
45    ofstream fout(output_file);
46
47    if (!fout) {
48      cout << "no fout" << endl;
49      return 0;
50    }
51
52    fout << image;
53
54    return 0;
55  }
56
57  // Process the mandelbrot image
58  void process_mandelbrot(Image3& image) {
59
60    // Allow for w, h variables in int form
61    int w = (int)image.w;
62    int h = (int)image.h;
63
64    // For each pixel(Px, Py) on the screen, do:
65    for (int i = 0; i < w; i++) {
66      for (int j = 0; j < h; j++) {
67
68        // Set x = i iteration
69        // Set y = j iteration
70        int x = i;
71        int y = j;
72
73        // x0 = scaled x coordinate of pixel (scaled to lie in the
          ↪  Mandelbrot X scale (-2.5, 1))
74        double x0 = ( (double)x / (double)w) * 3.5 - 2.5;
75
76        // y0 = scaled y coordinate of pixel (scaled to lie in the
          ↪  Mandelbrot Y scale (-1, 1))
77        double y0 = ( (double)y / (double)h) * 2 - 1;
78
79        // x = 0.0
80        double x_val = 0.0;
81
82        // y = 0.0
83        double y_val = 0.0;
```

```
84
85        // iteration = 0
86        int iteration = 0;
87
88        // max_iteration = 1000
89        int max_iteration = 1000;
90
91        // while (x*x + y*y <= 2*2 AND iteration < max+iteration
92        while (x_val*x_val + y_val * y_val <= 2*2*2*2 && iteration
          ↪  < max_iteration) {
93
94          // xtemp = x*x - y*y + x0
95          double x_temp = (double) x_val*x_val - y_val*y_val + x0;
96
97          // y = 2 * x * y + y0
98          double y_temp = (double) 2 * x_val * y_val + y0;
99          y_val = y_temp;
100
101         // x = xtemp
102         x_val = x_temp;
103
104         // iteration = iteration + 1
105         iteration++;
106       }
107
108
109       // color = palette[iteration]
110       int r, g, b;
111       r = iteration * 15;
112       g = r / 3;
113       b = g / 2;
114
115       // plot(Px, Py, color)
116       image.setPixel(x, y, Color3(r, g, b));
117     }
118   }
119 }
```

5.3 Caesar Cypher

```
1  /*
2   * main.cpp
3   * CS 201
4   * November 15, 2019
5   * Bryan Beus
6   * Main file for caesar cypher in hw7
7   */
8
9  #include <cstdlib>
10 #include <iostream>
```

```cpp
#include <iomanip>
#include <algorithm>
#include <iterator>
#include <utility>
#include <vector>
#include <string>
#include <map>

#include "caesar_cypher.hpp"

using std::vector;
using std::pair;
using std::string;
using std::cout;
using std::cin;
using std::endl;
using std::make_pair;
using std::for_each;
using std::getline;
using std::istringstream;
using std::ostringstream;
using std::to_string;
using std::find;
using std::map;
using std::setw;
using std::left;
using std::getline;

int main() {
  bool isFinished = false;
  clearConsole();

  while (!isFinished) {

    string user_input;
    int shift_amount;

    capture_user_input_string(user_input, isFinished);

    if (isFinished) {
      break;
    }

    capture_user_input_int(shift_amount);

    string res_str;

    shift_message(user_input, shift_amount, res_str);

    cout << endl << "The new message is: " << res_str << endl <<
    ↪  endl;

  }

```

```
64    // Print goodbye message
65    cout << endl << "Program complete" << endl;
66
67    return 0;
68 }
```