

Joseph Morgan
Homework 10

CISP440

1 Source code for inverse table generator program

inverse_table.h

```
1 #ifndef INVERSE_TABLE_H_
2 #define INVERSE_TABLE_H_
3
4 static unsigned char fi[4][256];
5
6 void fill_fi();
7 #endif
```

inverse_table.cpp

```
1 #include "encryptor.h"
2 #include "inverse_table.h"
3
4 void fill_fi()
5 {
6     unsigned char temp;
7     for (int i = 0; i < 4; ++i) {
8         for (int j = 0; j < 256; ++j) {
9             temp = f[i][j];
10            fi[i][temp] = j;
11        }
12    }
13 }
```

2 Source code for decryption program

decrytor.h

```
1 #ifndef DECRYPTOR_H_
2 #define DECRYPTOR_H_
3
4 #include <fstream>
5
6 unsigned char decrypt (unsigned char w, unsigned char key);
7 void decrypt_file (std::ifstream &infile, std::ofstream &outfile, unsigned
    char key);
8 void decrypt_without_key (std::ifstream &infile, std::ofstream &outfile);
9
10 #endif
```

decrytor.cpp

```
1 #include <climits>
2 #include <iostream>
3 #include "inverse_table.h"
4 #include "decrytor.h"
5 #include "encryptor.h"
6
```

```

7  unsigned char decrypt (unsigned char w, unsigned char key)
8  {
9      unsigned char o_char = 0;
10     unsigned char x0, y0, z0 = 0;
11     unsigned char x1, y1, z1 = 0;
12     unsigned char x2, y2, z2 = 0;
13     unsigned char x3, y3 = 0;
14     unsigned char p, q, r, s = 0;
15
16     p = (key & 0x03);
17     q = (key & 0x0C) >> 2;
18     r = (key & 0x30) >> 4;
19     s = (key & 0xC0) >> 6;
20
21     y3 = w ^ key;
22     x3 = swapbytes (y3);
23     z2 = fi [p][x3];
24
25     y2 = z2 ^ key;
26     x2 = swapbytes (y2);
27     z1 = fi [q][x2];
28
29     y1 = z1 ^ key;
30     x1 = swapbytes (y1);
31     z0 = fi [r][x1];
32
33     y0 = z0 ^ key;
34     x0 = swapbytes (y0);
35     o_char = fi [s][x0];
36
37     return o_char;
38 }
39
40 void decrypt_file (std::ifstream &infile , std::ofstream &outfile , unsigned
    char key)
41 {
42     char c;
43     while (!infile.eof())
44     {
45         infile.read(&c, 1);
46         if (!infile.eof())
47         {
48             c = decrypt(c, key);
49             outfile.write(&c, 1);
50         }
51     }
52 }
53
54 void decrypt_without_key (std::ifstream &infile , std::ofstream &outfile)
55 {
56     short key;
57     char *c = (char*)malloc(2 * sizeof(char));
58     char *d = (char*)malloc(2 * sizeof(char));
59     infile.read (c, 2);

```

```

60     for (key = 0; key <= UCHAR_MAX; ++key) {
61         d[0] = decrypt (c[0], key);
62         d[1] = decrypt (c[1], key);
63         if (d[0] == 'B' && d[1] == 'M') {
64             std::cout << "Key_found_as:" << (char)key << std::endl;
65             break;
66         }
67     }
68     if (key > UCHAR_MAX) {
69         std::cout << "Key_could_not_be_found\n";
70         exit(1);
71     } else {
72         infile.seekg(0, infile.beg);
73         decrypt_file (infile, outfile, key);
74     }
75 }

```

main.cpp

```

1  #include <string.h>
2  #include <iostream>
3  #include "inverse_table.h"
4  #include "encryptor.h"
5  #include "decryptor.h"
6
7  int main (int argc, char* argv[])
8  {
9      fill_fi();
10     char selection;
11     std::ifstream infile;
12     std::ofstream outfile;
13     std::string filename;
14     char key = 0;
15
16     if (argc >= 3 && strcmp(argv[1], "-f") == 0) {
17         filename = argv[2];
18         infile.open(filename, std::ios_base::binary);
19         if (!infile)
20             std::cout << "Invalid_filename!\n", exit(1);
21         outfile.open("cracked.bmp", std::ios_base::binary);
22         decrypt_without_key (infile, outfile);
23         if (argc == 4 && strcmp(argv[3], "-o") == 0) {
24             for (int i = 0; i < 4; ++i) {
25                 std::cout << "{";
26                 for (int j = 0; j < 256; ++j) {
27                     std::cout << " " << fi[i][j];
28                 }
29                 std::cout << "}" << std::endl;
30             }
31         }
32     } else {
33
34         std::cout
35             << "***_miniDES_encryption_***\n"

```

```

36     << "e)\Encrypt_file\n"
37     << "d)\Decrypt_file\n"
38     << "u)\Decrypt_file_with_unknown_key"
39     << "\nPlease_make_a_selection: ";
40 std::cin >> selection;
41 std::cout << std::endl;
42
43 switch (selection) {
44     case ('e') :
45         std::cout
46             << "You've_selected_'encrypt_file'\n"
47             << "Please_input_the_filename: ";
48         std::cin >> filename;
49         std::cout << std::endl << "Please_input_the_encryption_key: ";
50         std::cin >> key;
51         std::cout << std::endl;
52         infile.open (filename, std::ios_base::binary);
53         if (!infile)
54             std::cerr << "Input_could_not_be_opened\n";
55         outfile.open ("encrypted.bmp", std::ios_base::binary);
56         encrypt_file (infile, outfile, key);
57         infile.close();
58         outfile.close();
59         break;
60
61     case ('d') :
62         std::cout
63             << "You've_selected_'decrypt_file'\n"
64             << "Please_input_the_filename: ";
65         std::cin >> filename;
66         std::cout << std::endl << "Please_input_the_encryption_key: ";
67         std::cin >> key;
68         std::cout << std::endl;
69         infile.open (filename, std::ios_base::binary);
70         if (!infile)
71             std::cerr << "Input_could_not_be_opened\n";
72         outfile.open ("decrypted.bmp", std::ios_base::binary);
73         decrypt_file (infile, outfile, key);
74         infile.close();
75         outfile.close();
76         break;
77
78     case ('u') :
79         std::cout
80             << "You've_selected_'decrypt_file_with_unknown_key'\n"
81             << "Please_input_the_filename: ";
82         std::cin >> filename;
83         infile.open (filename, std::ios_base::binary);
84         if (!infile)
85             std::cerr << "Input_could_not_be_opened\n";
86         outfile.open ("cracked.bmp", std::ios_base::binary);
87         decrypt_without_key (infile, outfile);
88         infile.close();
89         outfile.close();

```

```

90         break;
91
92     default:
93         std::cout << "Invalid entry\n", exit(1);
94     }
95 }
96 }

```

3 Inverse function tables printout

4 Key value

5 Inverse function composition order

6 Decrypted Output