Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе №1 по курсу «Защита информации»

Тема Реализация электронного аналога шифровальной машины «Энигма»

Студент Волков Г.В.

Группа ИУ7-71Б

Оценка (баллы)

Преподаватели Чиж И. С.

Москва — 2023 г.

Введение

Шифровальная машина «Энигма» — одна из самых известрых шифровальнных машин, использовавшихся для шифрования и расшифровывания секретных сообщений.

Цель — Реализация электронного аналога шифровальной машины «Энигма».

Для достижения поставленной цели необходимо выполнить следующие задачи:

1) изучить алгоритм работы шифровальной машины «Энигма»;

2) реализовать в виде программы электронный аналог шифровальной машины «Энигма»;

3) обеспечить шифрование и расшифровку произвольного файла с использованием разработанной программы;

4) предусмотреть работу программы с пустым, однобайтовым файлом и с файлами архива (rar, zip или др.).

1 Аналитическая часть

В этом разделе будет рассмотрено устройство шифровальной машины «Энигма» и ей комплектующих и приведён пример её работы.

1.1 Основные детали

Шифровальная машина «Энигма» состоит из следующих деталей: роторы, рефлектор, а также коммутационная панель.

1.1.1 Роторы

Ротор — диск с 26 зубьями, для регулировки, и 26 контактами с обоих сторон (26 как букв в алфавите, количество может быть любым). Ротор производил подстановку входного символа. В результате прохождение через ротор символ менялся на другой и поступал дале по цепочке в следующий ротор или рефлектор. В конфигурации, используемой во Второй мировой, три ротора с 26 зубьями крепились на шпиндель и вставлялись в машинку. Схема работы роторов изображена на рисунке 1.1.

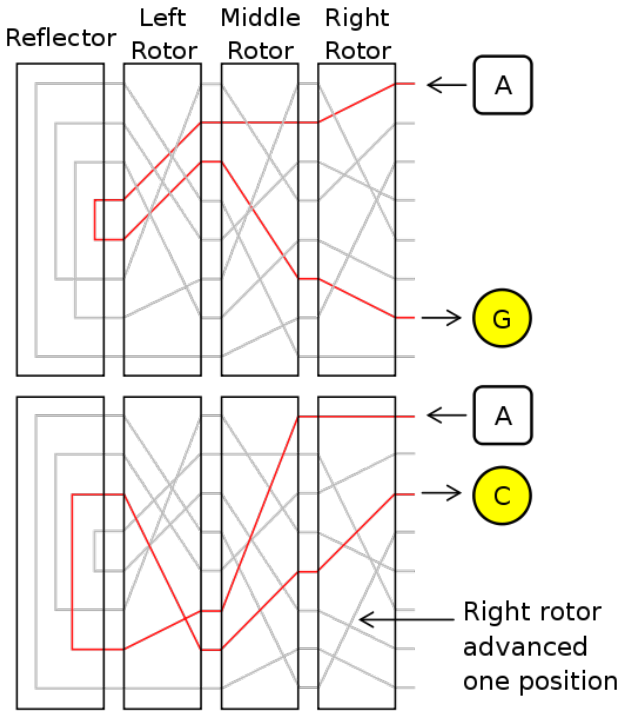

Рисунок 1.1 – Схема работы роторов с рефлектором

1.1.2 Рефлектор

Рефлектор — элемент, попарно соединяющий контакты последнего ротора, тем самым направляя ток обратно на последний ротор. Так, после этого электрический сигнал пойдёт в обратном направлении, пройдя через все роторы повторно.

1.1.3 Коммутационная панель

Коммутационная панель позволяет оператору шифровальной машины варьировать содержимое проводов, попарно соединяющих буквы английского алфавита. Эффект состоял в том, чтобы усложнить работу машины, не увеличивая число роторов. Так, если на коммутационной панели соединены буквы 'A' и 'Z', то каждая буква 'A', проходящая через коммутационную панель, будет заменена на 'Z' и наоборот. Сигналы попадали на коммутационную панель 2 раза: в начале и в конце обработки отдельного символа.

1.2 Алгоритм работы

При нажатии клавиши электрический сигнал проходит через коммутационную панель, роторы, рефлектор и идёт в обратном направлении чтоб включить лампочку на панели. Подсвеченная буква и будет являться зашифрованным символом. После каждой обработанной буквы роторы сдвигаются следующим образом: самый первый ротор проворачивается на одно деление после каждого нажатия клавиши, второй после полного оборота первого и тд. Для дешифрации зашифрованного сообщения, необходимо применить алгоритм шифрования энигмы к зашифрованному сообщению с настройками, которые имели место при шифровании исходного сообщения. Пример шифрования сообщения «AAA» для алфавита «ABCDEFGH» показан на рисунке 1.2.

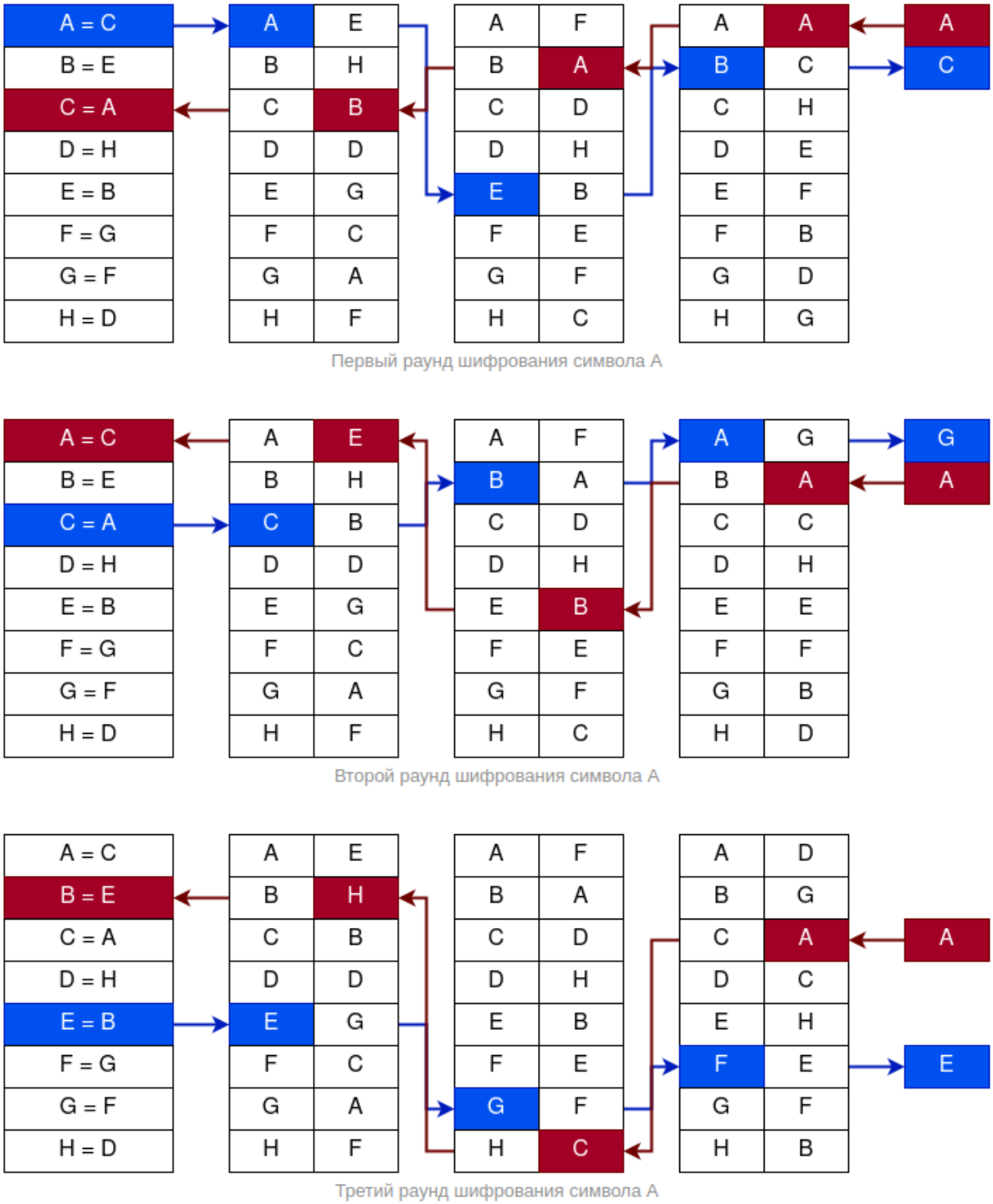

Рисунок 1.2 – Пример шифрования

Вывод

В данном разделе были рассмотрены алгоритм работы шифровальной машины «Энигма», использовавшейся во время Второй мировой войны.

2 Конструкторская часть

В этом разделе будут представлены описания используемых типов данных, а также схема алгоритма разрабатываемой программы.

2.1 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие типы данных для соответствующих значений:

— набор роторов — матрица;

— рефлектор — массив;

— коммутатор — массив;

— сообщение — последовательность байт входного файла.

2.2 Сведения о модулях программы

Программа состоит из двух модулей:

1) *main* — файл, содержащий точку входа;

2) *enigma* — файл, содержащий реализацию «энигмы».

2.3 Разработка алгоритмов

На рисунке 2.1 представлена схема работы программы, реализующей шифровальную машину «Энигма».

Рисунок 2.1 – Схема алгоритма шифрования, используемого в программе

Алгоритм шифрования реализован для алфавита размером 256, что необходимо для работы с данными файлов любых форматов.

Вывод

В данном разделе были представлены описания используемых типов данных, а также схема алгоритма разрабатываемой программы.

3 Технологическая часть

В данном разделе будут рассмотрены средства реализации, а также представлены листинги реализаций алгоритма шифрования машины «Энигма».

3.1 Средства реализации

В данной работе для реализации был выбран язык программирования C. Данный язык удоволетворяет поставленным критериям по средствам реализации.

3.2 Реализация алгоритма

В листингах 3.1 представлена реализация алгоритма шифрования машины «Энигма».

Листинг 3.1 – Реализация алгоритма шифрования машины «Энигма»

```
1  typedef struct enigma_t {
2      unsigned int rotor_size;
3      unsigned int rotor_num;
4      unsigned int *commutator;
5      unsigned int *reflector;
6      unsigned int **rotors;
7      unsigned int *indexes;
8  } enigma_t;
9
10 enigma_t *new_enigma(unsigned int rotor_num, unsigned int
       rotor_size) {
11     enigma_t *enigma = malloc(sizeof(enigma_t));
12     if (enigma == NULL) return NULL;
13
14
15     enigma->commutator = malloc(sizeof(unsigned int) * rotor_size);
16     if (enigma->commutator == NULL) {
17         free(enigma);
18         return NULL;
```

```
19      }
20      enigma->reflector = malloc(sizeof(unsigned int) * rotor_size);
21      if (enigma->reflector == NULL) {
22          free(enigma->commutator);
23          free(enigma);
24          return NULL;
25      }
26
27      enigma->rotors = malloc(sizeof(unsigned int *) * rotor_num);
28      for (int i = 0; i < rotor_num; ++i) enigma->rotors[i] =
            malloc(sizeof(unsigned int) * rotor_size);
29
30      enigma->indexes = calloc(rotor_num, sizeof(unsigned int));
31
32      enigma->rotor_size = rotor_size;
33      enigma->rotor_num = rotor_num;
34
35      return enigma;
36  }
37
38  void free_enigma(enigma_t *enigma) {
39      for (int i = 0; i < enigma->rotor_num; ++i)
            free(enigma->rotors[i]);
40      free(enigma->rotors);
41      free(enigma->reflector);
42      free(enigma->commutator);
43      free(enigma->indexes);
44      free(enigma);
45  }
46
47  void set_commutator(enigma_t *enigma, const unsigned int
        *commutator) {
48      for (int i = 0; i < enigma->rotor_size; ++i)
            enigma->commutator[i] = commutator[i];
49  }
50
51  void set_rotors(enigma_t *enigma, unsigned int **rotors) {
52      for (int i = 0; i < enigma->rotor_num; ++i)
53      for (int j = 0; j < enigma->rotor_size; ++j)
54      enigma->rotors[i][j] = rotors[i][j];
55
```

```
56 }
57
58 void set_reflector(enigma_t *enigma, const unsigned int *reflector)
       {
59     for (int i = 0; i < enigma->rotor_size; ++i)
           enigma->reflector[i] = reflector[i];
60 }
61
62 unsigned int encrypt(enigma_t *enigma, unsigned int symbol, int
       *status) {
63     *status = 1;
64     if (enigma == NULL) {
65         *status = 0;
66         return 0;
67     }
68     if (symbol >= enigma->rotor_size) {
69         *status = 0;
70         return 0;
71     }
72
73     symbol = enigma->commutator[symbol];
74     for (int i = 0; i < enigma->rotor_num; ++i)
75     symbol = enigma->rotors[i][(symbol + enigma->indexes[i]) %
           enigma->rotor_size];
76
77     symbol = enigma->reflector[symbol];
78
79     for (int i = enigma->rotor_num - 1; i >= 0; --i) {
80         symbol = backtrack(enigma, symbol, i, status);
81         if (*status == 0) return 0;
82     }
83     symbol = enigma->commutator[symbol];
84
85     int shift = 1;
86     for (int i = 0; i < enigma->rotor_num && shift == 1; ++i) {
87         shift = 0;
88         if (enigma->indexes[i] >= enigma->rotor_size - 1) shift = 1;
89         enigma->indexes[i] = (enigma->indexes[i] + 1) %
               enigma->rotor_size;
90     }
91
```

```
92      return symbol;
93  }
94
95  unsigned int backtrack(enigma_t *enigma, unsigned int symbol,
        unsigned int rotor_index, int *status) {
96      *status = 1;
97
98      for (int i = 0; i < enigma->rotor_size; ++i)
99      if (enigma->rotors[rotor_index][i] == symbol)
100     return i < enigma->indexes[rotor_index] ? enigma->rotor_size -
            (enigma->indexes[rotor_index] - i) : i -
            enigma->indexes[rotor_index];
101
102     *status = 0;
103     return 0;
104 }
```

3.3 Тестирование

Для тестирование написанной программы можно воспользоваться тем, что для декодирования информации нужно прогнать зашифрованный текст через машину ещё раз с теми же настройками, что использовались для кодирования. Поэтому для тестирования входной файл кодируется и результат записывается во временный файл, это файл кодируется ещё раз с теми же настройка и записывается в выходной файл. В результате входной и выходной файлы должны совпадать.

Таблица 3.1 – Функциональные тесты

Входной файл	Ожидаемый результат	Результат
1_in.txt (9 bytes) «encode me»	1_out.txt (9 bytes) «encode me»	1_out.txt (9 bytes) «encode me»
2_in.txt (1 bytes) «a»	2_out.txt (1 bytes) «a»	2_out.txt (1 bytes) «a»
3_in.txt (0 bytes) «»	3_out.txt (0 bytes) «»	3_out.txt (0 bytes) «»
4_in.tar.gz (150 bytes) tst.txt (32 bytes) «encode me»	4_out.tar.gz (150 bytes) tst.txt (32 bytes «encode me»	4_out.tar.gz (150 bytes) tst.txt (32 bytes «encode me»

Вывод

Были представлен листинг реализации алгоритма работы энигмы. Также в данном разделе была приведена информация о выбранных средствах для разработки алгоритмов.

Заключение

В результате лабораторной работы были изучены принципы работы шифровальной машины «Энигма», была реализована программа, способная шифровать и дешифровать текстовый файл, позволять настраивать роторы, рефлектор и коммутационную панель.

Были решены следующие задачи:

1) изучен алгоритм работы шифровальной машины «Энигма»;

2) реализован в виде программы электронный аналог шифровальной машины «Энигма»;

3) обеспечено шифрование и расшифровка произвольного файла с использованием разработанной программы;

4) предусмотрена работа программы с пустым, однобайтовым файлом и с файлами архива (rar, zip или др.).