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 Шифровальная машина «Энигма»

Студент Фам Минь Хиеу.

Группа ИУ7И-72Б

Оценка (баллы)

Преподаватели Чиж И. С.

Москва — 2024 г.

Введение

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

Целью данной работы является реализация в виде программы на языке программирования C++ аналога шифровальной машины «Энигма», обеспечеие шифрования и расшифровки файла.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1) изучить алгоритм работы шифровальной машины «Энигма»;
2) реализовать алгоритм работы шифровальной машины «Энигма» в виде программы;
3) протестировать разработанную программу;
4) описать и обосновать полученные результаты в отчёте о выполненной лабораторной работе.

1 Аналитическая часть

В этом разделе будут рассмотрены классический алгоритм работы шифровальной машины «Энигма», а также её вариант, использованный во время Второй мировой войны.

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

1.1 Роторы

«Энигма» предназначена для шифрации сообщений, написанных на английском языке. Ротор — прикреплённый к шестерёнке с 26 зубцами (по одному на каждую букву алфавита) элемент, предназначенный для преобразования одной буквы в другую.

В разное время в разных реализациях «Энигмы» использовалось разное количество роторов. Во время Второй мировой войны использовались 3 ротора, причём всего было 10 роторов, преобразовывающих буквы в соответствии с таблицей 1.1.

Таблица 1.1 – Преобразования роторов «Энигмы»

Ротор	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
I	E	K	M	F	L	G	D	Q	V	Z	N	T	O	W	Y	H	X	U	S	P	A	I	B	R	C	J
II	A	J	D	K	S	I	R	U	X	B	L	H	W	T	M	C	Q	G	Z	N	P	Y	F	V	O	E
III	B	D	F	H	J	L	C	P	R	T	X	V	Z	N	Y	E	I	W	G	A	K	M	U	S	Q	O
IV	E	S	O	V	P	Z	J	A	Y	Q	U	I	R	H	X	L	N	F	T	G	K	D	C	M	W	B
V	V	Z	B	R	G	I	T	Y	U	P	S	D	N	H	L	X	A	W	M	J	Q	O	F	E	C	K
VI	J	P	G	V	O	U	M	F	Y	Q	B	E	N	H	Z	R	D	K	A	S	X	L	I	C	T	W
VII	N	Z	J	H	G	R	C	X	M	Y	S	W	B	O	U	F	A	I	V	L	P	E	K	Q	D	T
VIII	F	K	Q	H	T	L	X	O	C	B	J	S	P	D	Z	R	A	M	E	W	N	I	U	Y	G	V
IX	L	E	Y	J	V	C	N	I	X	W	P	B	Q	M	D	R	T	A	K	Z	G	F	U	H	O	S
X	F	S	O	K	A	N	U	E	R	H	M	B	T	I	Y	C	W	L	Q	P	Z	X	V	G	J	D

1.2 Входное колесо

Входное колесо — элемент, позволяющий выставить роторы в необходимые значения. В физической машине было 3 отверстия, позволяющих про-

сматривать, в каком состоянии находится каждый ротор. Положения роторов является ключевым для процесса шифрования, поскольку в зависимости от них одно и то же сообщение будет зашифровано по-разному и будет требовать сответствующих начальных значений роторов для дешифрации.

1.3 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. Во время Второй мировой войны было создано 2 рефлектора, представленных в таблице

Таблица 1.2 – Преобразования роторов «Энигмы»

Рефлектор	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
I	F	V	P	J	I	A	O	Y	E	D	R	Z	X	W	G	C	T	K	U	Q	S	B	N	M	H	L
II	Y	R	U	H	Q	S	L	D	P	X	N	G	O	K	M	I	E	B	F	Z	C	W	V	J	A	T

1.4 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.

2 Конструкторская часть

В этом разделе будут представлены описания используемых типов данных, а также требования к программе.

В этом разделе представлена схема алгоритма шифровальной машины «Энигма».

2.1 Разработка алгоритмов

На рисунке 2.1 приведена схема работы шифровальной машины Энигма.

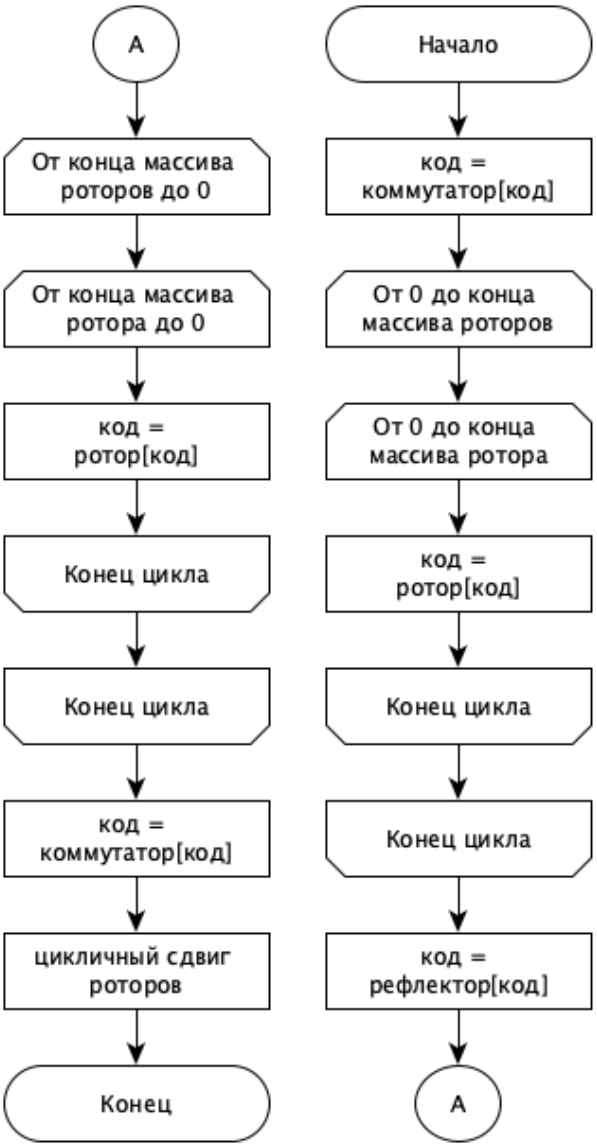

Рисунок 2.1 – Схема работы шифровальной машина Энигма

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  #include <iostream>
2
3  #include <iostream>
4  #include <vector>
5  #include <fstream>
6  #include <string>
7  #include <unordered_map>
8  using namespace std;
9
10 const int ALPHABET_SIZE = 64;
11
12 const string ALPHABET =
       "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789.,";
13
14 class Rotor {
15     public:
16     Rotor(const string wiring, int position = 0)
17     {
18         string part1 = wiring.substr(0, position);
```

```
19          string part2 = wiring.substr(position, ALPHABET_SIZE -
                position);
20          wiring_ = part2 + part1;
21          //wiring_ = wiring;
22      }
23
24      char encryptForward(char c) {
25          int index = wiring_.find(c);
26          return ALPHABET[index];
27      }
28
29      char encryptBackward(char c) {
30          size_t index = ALPHABET.find(c);
31          return wiring_[index];
32      }
33
34      void rotate() {
35          char tmp = wiring_[0];
36          for (int i = 0; i < ALPHABET_SIZE - 1; i++)
37          {
38              wiring_[i] = wiring_[i + 1];
39          }
40          wiring_[ALPHABET_SIZE - 1] = tmp;
41      }
42
43      private:
44      string wiring_;
45  };
46
47  class Reflector {
48      public:
49      Reflector(const string wiring)
50      {
51          wiring_ = wiring;
52          for (int i = 0; i < wiring.size(); i += 1)
53          {
54              if (i % 2 == 0)
55              {
56                  wiring_[i] = wiring[i + 1];
57              }
58              else
```

```
59              {
60                  wiring_[i] = wiring[i - 1];
61              }
62          }
63      }
64
65      char reflect(char c) {
66          return wiring_[ALPHABET.find(c)];
67      }
68
69      private:
70      string wiring_;
71  };
72
73  class Enigma {
74      public:
75      Enigma(const vector<Rotor>& rotors, const Reflector& reflector)
76      : rotors_(rotors), reflector_(reflector) {}
77
78      string encryptMessage(const string& message) {
79          string encryptedMessage;
80          for (char c : message) {
81              if (isInAlphabet(c)) {
82                  encryptedMessage += encryptChar(c);
83                  rotateRotors();
84              }
85              else {
86                  encryptedMessage += c;
87              }
88          }
89          return encryptedMessage;
90      }
91
92      void encryptFile(const string& inputFile, const string&
           outputFile) {
93          ifstream inFile(inputFile, ios::binary);
94          ofstream outFile(outputFile, ios::binary);
95
96          if (!inFile.is_open() || !outFile.is_open()) {
97              cerr << "Error" << endl;
98              return;
```

```
99          }
100
101         char buffer;
102         while (inFile.get(buffer)) {
103             if (isInAlphabet(buffer)) {
104                 buffer = encryptChar(buffer);
105                 rotateRotors();
106             }
107             outFile.put(buffer);
108         }
109
110         inFile.close();
111         outFile.close();
112     }
113
114     private:
115     vector<Rotor> rotors_;
116     Reflector reflector_;
117     int cnt_ = 0;
118
119
120     char encryptChar(char c) {
121         for (Rotor& rotor : rotors_) {
122             c = rotor.encryptForward(c);
123         }
124
125         c = reflector_.reflect(c);
126
127         for (auto it = rotors_.rbegin(); it != rotors_.rend();
              ++it) {
128             c = it->encryptBackward(c);
129         }
130         return c;
131     }
132
133     void rotateRotors() {
134         cnt_++;
135         rotors_[0].rotate();
136         if (cnt_ % ALPHABET_SIZE == 0)
137         rotors_[1].rotate();
138         if (cnt_ % (ALPHABET_SIZE * ALPHABET_SIZE) == 0)
```

```
139             rotors_[2].rotate();
140
141         }
142
143         bool isInAlphabet(char c) {
144             return ALPHABET.find(c) != string::npos;
145         }
146 };
```

3.3 Тестирование

Таблица 3.1 – Функциональные тесты

Входная строка	Выходная строка
HelloEnigma123	>hfeinduehfo2m
hello	1ma8r
A	k
«»	«»

Вывод

Были представлены листинги реализаций алгоритма шифрования в машине «Энигма» согласно алгоритму, представленному в первой части, а также проведено тестирование разработанной программы.

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 алгоритм работы шифровальной машины «Энигма» в виде программы, обеспечив возможности шифрования и расшифровки текстового файла;

3) полученная программа протестирована, произведена демонстрация того, что во всех случаях сообщение удаётся дешифровать и получить исходное;

4) полученные результаты описаны в отчёте о выполненной лабораторной работе.