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»

ОТЧЕТ

по лабораторной работе №1

по курсу «Защита информации»

на тему: «Программная реализация шифровальной машины Энигма»

Студент ИУ7-73Б _____ Лагутин Д. В.

(Группа) (Подпись, дата) (Фамилия И. О.)

Преподаватель _____ Чиж И. С.

(Подпись, дата) (Фамилия И. О.)

Москва, 2023 г.

Содержание

Введение

Целью данной работы является разработка программы-аналога шифровальной машины Энигма.

Задачи:

1) изучить шифровальную машину и определить ее особенности;
2) разработать алгоритм работы машины;
3) реализовать программу-аналог.

1 Аналитическая часть

1.1 Описание машины

Энигма — переносная шифровальная машина, использовавшаяся для шифрования и расшифрования секретных сообщений.

Первая версия машины была разработана в 1918 году немецким инженером Артуром Шербиусом. Данное устройство активно использовалась в военной сфере, чтобы солдаты и командиры могли обмениваться конфиденциальной информацией.

1.2 Составные части

Энигма состояла из комбинации механических и электрических систем. Механическая часть включала в себя клавиатуру, набор вращающихся дисков — роторов, которые были расположены вдоль вала и прилегали к нему, ступенчатого механизма, двигающего один или несколько роторов при каждом нажатии на клавишу, и рефлектора (набор попарно соединенных контактов, отражающих сигнал в обратном направлении).

Электрическая часть, в свою очередь, состояла из электрической схемы, соединяющей между собой клавиатуру, коммутационную панель, лампочки и роторы (для соединения роторов, первичного вала и отразателя использовались скользящие контакты).

1.3 Принцип работы

Текст, который нужно было зашифровать, печатался на машине с использованием клавиатуры. Перед началом использования оператор открывал

крышку аппарата и устанавливал начальную позицию — три номера, последовательность которых передавалась заранее.

С каждым нажатием на клавишу производится смещение первого ротора на одну ступень, при полном обороте текущего ротора также происходит сдвиг следующего и так далее.

Электронный сигнал поступает от клавиатуры через коммутационную панель, производящую первичное перемешивание пар сигналов. Далее сигнал поступает на первичный ротор после чего проходит по текущей конфигурации роторов. Дойдя до отражателя сигнал «разворачивается» и проходит через систему роторов еще раз. Полученный сигнал отображается на индикационной панели, после обработки на коммутационной панели.

Использование отражателя в системе позволяет кодировать и расшифровывать сообщения на одной машине. Все что необходимо знать: первоначальную настройку роторов и соединений на коммутационной панели.

2 Конструкторская часть

2.1 Схема алгоритма работы машины

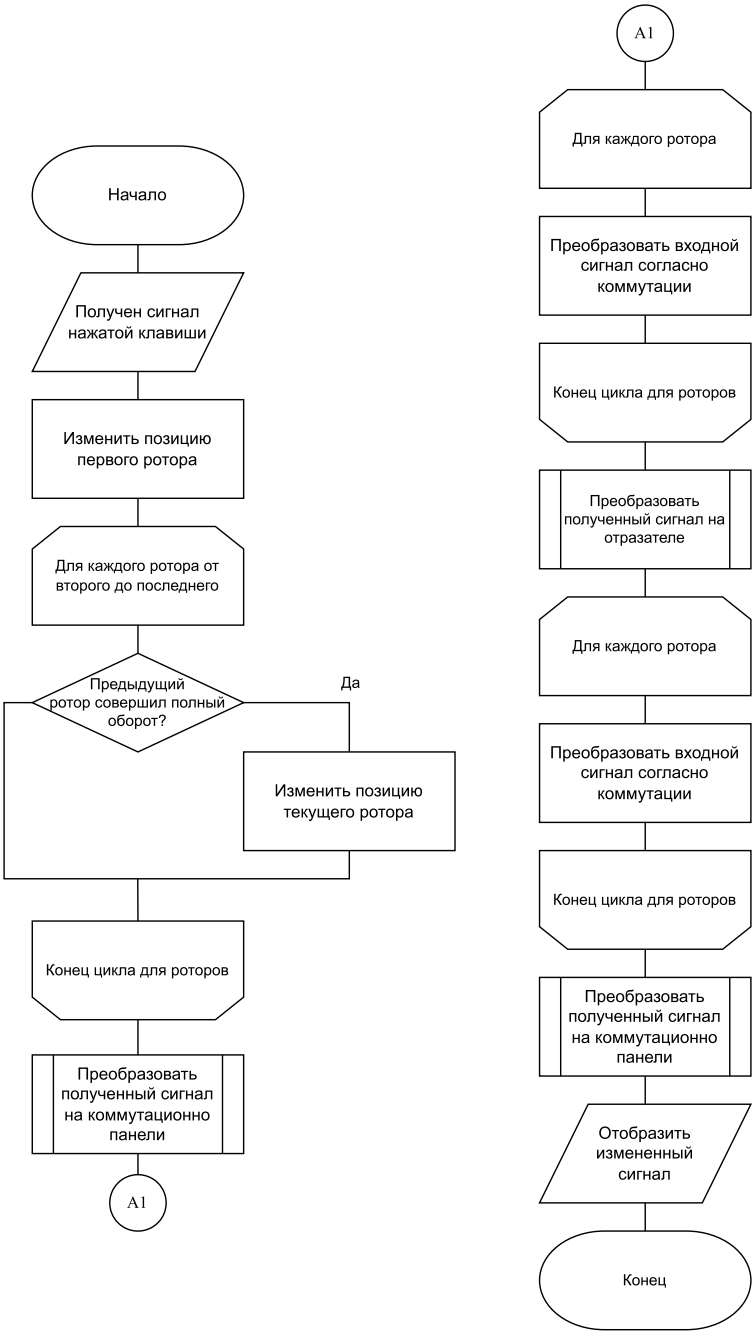

Рисунок 2.1 – Алгоритм работы шифровальной машины

3 Технологическая часть

Для реализации машины использовался язык C++. Конфигурация осуществляется размещением соответствующих файлов в каталоге `config`.

```
config/
├── rotors
│   ├── 1 -> ../../maps/disk1
│   ├── 2 -> ../../maps/disk2
│   └── 3 -> ../../maps/disk3
├── plugboard -> ../maps/plugboard1
└── reflector -> ../maps/reflector1
```

Рисунок 3.1 – Пример конфигурации системы

```
$ hexdump maps/disk1 -C
00000000  00 9c 01 45 02 e6 03 0a  04 87 05 44 06 da 07 ce
00000010  08 ec 09 42 0a dd 0b 7c  0c 89 0d 23 0e 46 0f 7b
00000020  10 f0 11 36 12 e4 13 ad  14 79 15 1a 16 e1 17 09
00000030  18 ca 19 75 1a 37 1b 80  1c 9d 1d 18 1e cf 1f 39
00000040  20 5e 21 b4 22 43 23 e5  24 f8 25 1d 26 b3 27 e7
00000050  28 60 29 90 2a 61 2b e9  2c b5 2d a5 2e 64 2f a3
00000060  30 dc 31 48 32 51 33 55  34 62 35 32 36 5f 37 2d
00000070  38 a8 39 94 3a ae 3b 47  3c af 3d 7d 3e 7e 3f 0b
00000080  40 2f 41 c1 42 f1 43 27  44 de 45 a4 46 0c 47 3e
00000090  48 33 49 6c 4a 28 4b e8  4c 11 4d 8c 4e 8a 4f ed
000000a0  50 d4 51 db 52 49 53 38  54 0d 55 a1 56 65 57 b6
000000b0  58 35 59 12 5a fa 5b e2  5c 8d 5d 77 5e ee 5f bb
000000c0  60 3a 61 df 62 e3 63 17  64 81 65 ef 66 56 67 b7
000000d0  68 59 69 7f 6a 9a 6b 6b  6c 0e 6d 24 6e 58 6f e0
000000e0  70 ff 71 9b 72 14 73 0f  74 3c 75 78 76 c2 77 71
000000f0  78 8b 79 bc 7a 54 7b 15  7c 34 7d 41 7e d0 7f 6d
00000100  80 1f 81 b2 82 83 83 9f  84 a0 85 d8 86 53 87 f9
00000110  88 57 89 f2 8a 66 8b 63  8c 13 8d bd 8e 3d 8f 10
00000120  90 5a 91 50 92 1c 93 95  94 c9 95 ea 96 05 97 52
00000130  98 9e 99 5b 9a 68 9b d1  9c a2 9d 3b 9e 3f 9f ba
00000140  a0 eb a1 be a2 5c a3 8e  a4 96 a5 ac a6 84 a7 f3
00000150  a8 99 a9 f4 aa 4a ab aa  ac a6 ad 85 ae bf af fe
00000160  b0 d6 b1 d7 b2 92 b3 a7  b4 b8 b5 97 b6 f5 b7 5d
00000170  b8 f6 b9 67 ba 1e bb 8f  bc 93 bd 69 be 4b bf 82
00000180  c0 16 c1 a9 c2 08 c3 ab  c4 4c c5 91 c6 98 c7 f7
00000190  c8 76 c9 fb ca b0 cb 1b  cc 6a cd 4d ce 19 cf 40
000001a0  d0 21 d1 b1 d2 d9 d3 fc  d4 4e d5 cb d6 29 d7 4f
000001b0  d8 25 d9 6e da c0 db b9  dc c3 dd 06 de 6f df c4
000001c0  e0 c5 e1 70 e2 72 e3 fd  e4 c8 e5 00 e6 01 e7 73
000001d0  e8 d2 e9 74 ea 7a eb 86  ec c6 ed 88 ee c7 ef d3
000001e0  f0 20 f1 cc f2 cd f3 d5  f4 22 f5 02 f6 03 f7 04
000001f0  f8 26 f9 07 fa 2a fb 2b  fc 2c fd 2e fe 30 ff 31
00000200
```

Рисунок 3.2 – Пример файла конфигурации

Концигурационный файл состоит из пар байт, ключ и значение соответственно.

Листинг 3.1 – Класс, описывающий машину

```
1  #include "Enigma.h"
2
3  #include <ctype.h>
4
5  Enigma::Enigma(std::shared_ptr<Alphabet<char>> alpahbet,
6                 std::shared_ptr<EnigmaIO> io,
7                 std::list<std::shared_ptr<EnigmaRotor>>
8                     rotors,
9                 std::shared_ptr<EnigmaReflector> reflector)
10     : alpahbet(alpahbet), io(io), rotors(rotors),
11       reflector(reflector)
12 {
13     if (io->size() != alpahbet->size()
14         || reflector->size() != alpahbet->size())
15         throw;
16
17     for (auto ptr : rotors)
18         if (alpahbet->size() != ptr->size())
19             throw;
20 }
21
22 std::string Enigma::encode(const std::string &origin)
23 {
24     return this->process(origin);
25 }
26
27 std::string Enigma::decode(const std::string &origin)
28 {
29     return this->process(origin);
30 }
31
32 std::string Enigma::process(const std::string &origin)
33 {
34     std::string out;
35
36     for (char c : origin)
37     {
38         size_t i = this->alpahbet->order(c);
39         i = this->io->input(i);
40         auto iter = this->rotors.begin();
```

```
41
42          for (; this->rotors.end() != iter; ++iter)
43              i = (*iter)->direct(i);
44
45          i = this->reflector->reflect(i);
46
47          while (this->rotors.begin() != --iter)
48              i = (*iter)->invert(i);
49
50          i = (*iter)->invert(i);
51
52          out += this->alpahbet->letter(this->io->output(i));
53      }
54
55      return out;
56  }
```

Листинг 3.2 – Реализация ротора

```
1  #include "PlainRotor.h"
2
3  PlainRotor::PlainRotor(std::shared_ptr<CharMap> map)
4      : map(map)
5  {}
6
7  size_t PlainRotor::size(void) const
8  {
9      return this->map->size();
10 }
11
12 size_t PlainRotor::direct(size_t c)
13 {
14     return this->map->encode(c);
15 }
16
17 size_t PlainRotor::invert(size_t c)
18 {
19     return this->map->decode(c);
20 }
```

Листинг 3.3 – Реализация ротора (декоратор вращения)

```
#include "RotateRotorDecorator.h"

RotateRotorDecorator::\
RotateRotorDecorator(std::shared_ptr<EnigmaRotor> rotor)
    : rotor(rotor)
{}

size_t RotateRotorDecorator::size(void) const
{
    return this->rotor->size();
}

size_t RotateRotorDecorator::direct(size_t c)
{
    if (!this->isPrevious)
        this->adjust();

    size_t tmp = this->apply_correction(c);
    tmp = this->rotor->direct(tmp);

    return this->undo_correction(tmp);
}

size_t RotateRotorDecorator::invert(size_t c)
{
    size_t tmp = this->apply_correction(c);
    tmp = this->rotor->invert(tmp);

    return this->undo_correction(tmp);
}

void
RotateRotorDecorator::\
stack(std::shared_ptr<RotateRotorDecorator> previous)
{
    previous->registerFullTurn(this);
    this->isPrevious = true;
}

void
```

```
41 RotateRotorDecorator::\
42 registerFullTurn(RotateRotorDecorator *next)
43 {
44     this->next = next;
45 }
46
47 void RotateRotorDecorator::adjust(void)
48 {
49     if (this->rotor->size() <= ++this->correction)
50     {
51         this->correction = 0;
52
53         if (this->next)
54             this->next->adjust();
55     }
56 }
57
58 size_t RotateRotorDecorator::apply_correction(size_t c) const
59 {
60     c += this->correction;
61
62     if (c >= this->rotor->size())
63         c -= this->rotor->size();
64
65     return c;
66 }
67
68 size_t RotateRotorDecorator::undo_correction(size_t c) const
69 {
70     if (c >= this->correction)
71         return  c - this->correction;
72
73     return this->rotor->size() - this->correction + c;
74 }
```

Реализации отражателя и коммутационной панели аналогичны ротору и не будут приведены в отчете.

Заключение

Была разработана программы-аналог шифровальной машины Энигма.

Были решены следующие задачи:

1) изучена шифровальная машина и определены ее особенности;
2) разработан алгоритм работы машины;
3) реализована программа-аналог.