



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 дисциплине «Защита информации»

Тема Разработка шифровальной машины «Энигма»

Студент Лукьяненко В.А.

Группа ИУ7-71Б

Преподаватель Руденкова Ю.С.

2025 г.

# 1 Задание

## 1.1 Цель работы

**Цель работы:** разработка электронного аналога машины «Энигма».

## 1.2 Содержание работы

Для выполнения данной лабораторной работы необходимо решить следующие задачи:

1. реализовать в виде программы электронный аналог машины «Энигма»;
2. обеспечить шифрование произвольного файла;
3. обеспечить расшифровку произвольного файла;
4. предусмотреть работу программы с пустым 1-байтовым файлом.

## 2 Теоретическая часть

### 1. Определение информации, защиты информации, актив, информационная сфера, угроза, шифровальная машина «Энигма».

- **Информация** — это данные, которые могут быть восприняты, обработаны или использованы человеком или техническими средствами.
- **Защита информации** — это комплекс организационных, технических и программных мер, направленных на предотвращение несанкционированного доступа, искажения, утраты или уничтожения информации.
- **Актив** — это любой объект, обладающий ценностью для владельца (данные, оборудование, программное обеспечение и т.д.).
- **Информационная сфера** — это область деятельности, связанная с формированием, хранением, обработкой и использованием информации, а также воздействием информации на общество и человека.
- **Угроза** — это потенциальное событие или действие, которое может нанести ущерб информационным активам (кража данных, модификация, уничтожение).
- **Шифровальная машина «Энигма»** — электромеханическое устройство для шифрования текста, применявшееся Германией во время Второй мировой войны, основанное на многоалфавитной подстановке с использованием роторов.

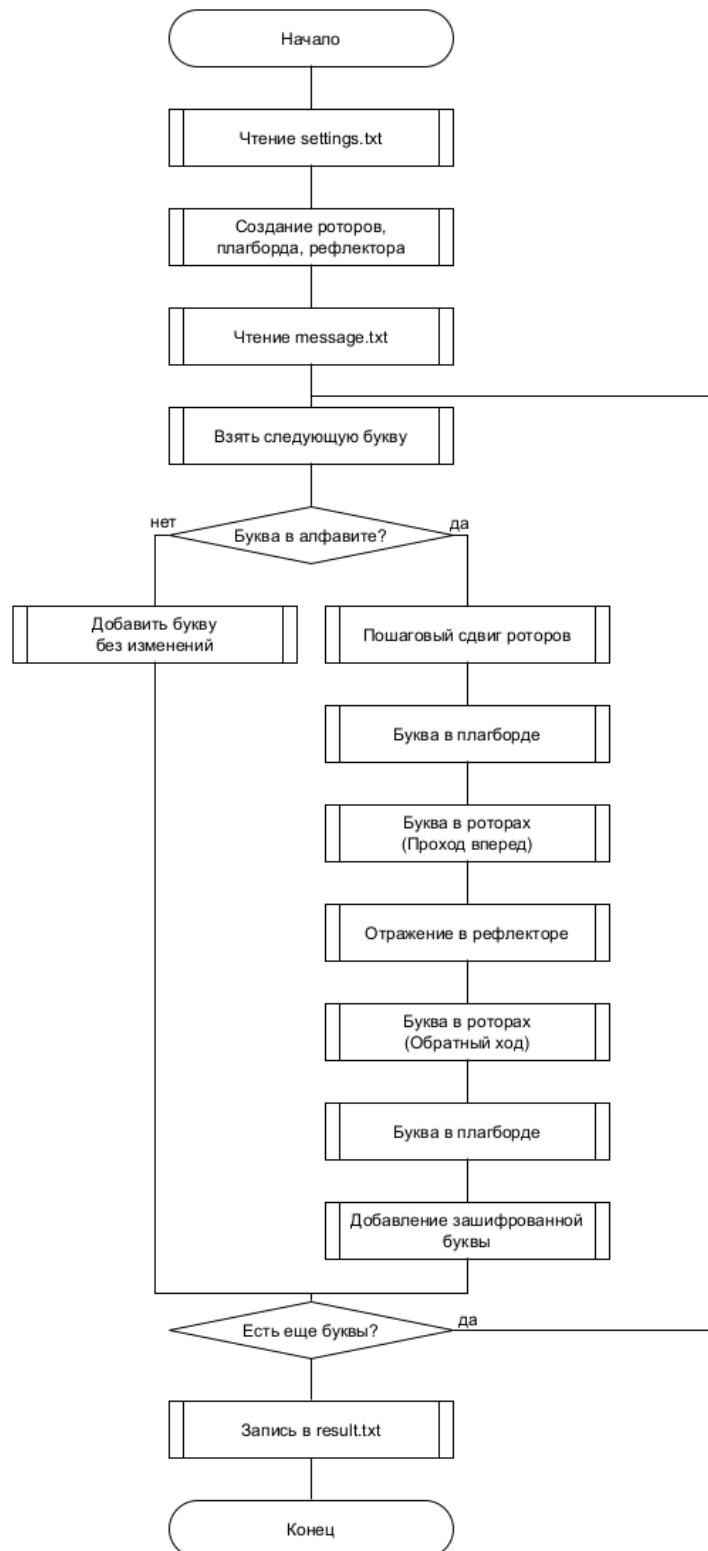
### 2. Дать определение одно- и многоалфавитной подстановки.

- **Однаалфавитная подстановка** — это криптографический метод, при котором каждой букве исходного алфавита соответствует только одна буква шифрованного алфавита на протяжении всего текста (шифр Цезаря).
- **Многоалфавитная подстановка** — это криптографический метод, при котором для разных символов текста могут использоваться разные алфавиты подстановки. Последовательность алфавитов задаётся ключом или механизмом (машина «Энигма»).

### 3. К какому виду относится алгоритм «Энигма»?

Алгоритм работы шифровальной машины «Энигма» относится к многоалфавитным подстановкам, так как каждый символ может шифроваться с использованием различных алфавитов в зависимости от текущего положения роторов.

### 4. Приведите схему алгоритма «Энигма».



### 3 Практическая часть.

Листинг 3.1 – Файл rotor.py, описывающий поведение роторов

```
1 import string
2
3 ALPHABET = string.ascii_uppercase
4
5 class Rotor:
6     def __init__(self, wiring, notch, ring_setting=0, position=0):
7         self.wiring = wiring
8         self.notch = notch
9         self.ring_setting = ring_setting
10        self.position = position
11
12    def step(self):
13        self.position = (self.position + 1) % 26
14        return self.position == ALPHABET.index(self.notch)
15
16    def encode_forward(self, c):
17        idx = ALPHABET.index(c)
18        shifted_idx = (idx + self.position - self.ring_setting) % 26
19        encoded_char = self.wiring[shifted_idx]
20        out_idx = (ALPHABET.index(encoded_char) - self.position +
21                  self.ring_setting) % 26
22        return ALPHABET[out_idx]
23
24    def encode_backward(self, c):
25        idx = ALPHABET.index(c)
26        shifted_idx = (idx + self.position - self.ring_setting) % 26
27        encoded_idx = self.wiring.index(ALPHABET[shifted_idx])
28        out_idx = (encoded_idx - self.position + self.ring_setting) %
29                  26
30        return ALPHABET[out_idx]
```

Листинг 3.2 – Файл reflector.py, описывающий поведение рефлектора

```
1 import string
2 ALPHABET = string.ascii_uppercase
3 class Reflector:
4     def __init__(self, pairs=None):
5         self.mapping = {}
6         for ch in ALPHABET:
7             self.mapping[ch] = ch
8         if pairs:
9             for pair in pairs:
10                 if len(pair) == 2:
11                     a, b = pair[0].upper(), pair[1].upper()
12                     if a in ALPHABET and b in ALPHABET:
13                         self.mapping[a] = b
14                         self.mapping[b] = a
15     def reflect(self, c):
16         return self.mapping.get(c.upper(), c)
```

Листинг 3.3 – Файл plugboard.py, описывающий поведение плагборда

```
1 import string
2 ALPHABET = string.ascii_uppercase
3 class Plugboard:
4     def __init__(self, pairs=None):
5         self.mapping = {}
6         for ch in ALPHABET:
7             self.mapping[ch] = ch
8         if pairs:
9             for pair in pairs:
10                 if len(pair) == 2:
11                     a, b = pair[0].upper(), pair[1].upper()
12                     if a in ALPHABET and b in ALPHABET:
13                         self.mapping[a] = b
14                         self.mapping[b] = a
15     def encode(self, c):
16         return self.mapping.get(c.upper(), c)
```

### Листинг 3.4 – Файл main.py,

```
1 from rotor import Rotor
2 from reflector import Reflector
3 from plugboard import Plugboard
4 import string
5
6 ALPHABET = string.ascii_uppercase
7
8 ROTOR_WIRINGS = {
9     "1": ("EKMFLGDQVZNTOWYHXUSPAIBRCJ", "Q"),
10    "2": ("AJDKSIRUXBLHWTMCQGZNPYFVOE", "E"),
11    "3": ("BDFHJLCPRTXVZNYEIWGAKMUSQO", "V"),
12    "4": ("ESOVZPJAYQUIRXLNFTGKDCMWB", "J"),
13    "5": ("VZBRGITYUPSDNHLXAWMJQOFECK", "Z")
14 }
15
16 def read_settings(filename):
17     with open(filename, "r") as f:
18         lines = [line.strip() for line in f if line.strip()]
19         rotor_types = lines[0].split()
20         positions = [int(x) for x in lines[1].split()]
21         ring_settings = [int(x) for x in lines[2].split()]
22         plug_idx = next(i for i, line in enumerate(lines) if
23             line.startswith("Plugboard"))
24         reflector_idx = next(i for i, line in enumerate(lines) if
25             line.startswith("Reflector"))
26         plugboard_pairs = lines[plug_idx+1:reflector_idx]
27         reflector_pairs = lines[reflector_idx+1:]
28         return rotor_types, positions, ring_settings, plugboard_pairs,
29             reflector_pairs
30
31 def build_machine(rotor_types, positions, ring_settings,
32     plugboard_pairs, reflector_pairs):
33     rotors = []
34     for i, rtype in enumerate(rotor_types):
35         wiring, notch = ROTOR_WIRINGS[rtype]
36         rotors.append(Rotor(wiring, notch,
37             ring_setting=ring_settings[i], position=positions[i]))
38     plugboard = Plugboard(plugboard_pairs)
39     reflector = Reflector(reflector_pairs)
40     return rotors, plugboard, reflector
```

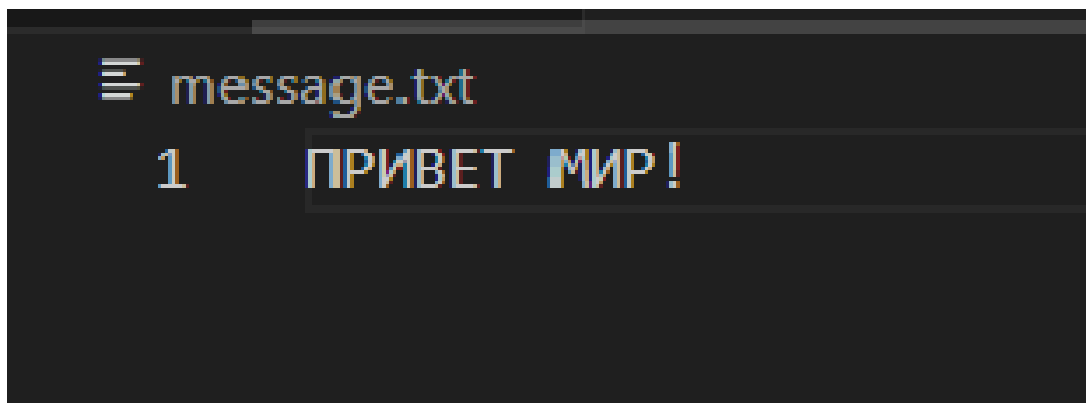
```

36
37 def step_rotors(rotors):
38     right, middle, left = rotors
39     middle_on_notch = ALPHABET[middle.position] == middle.notch
40     right.step()
41     if ALPHABET[right.position] == right.notch or middle_on_notch:
42         middle.step()
43         if ALPHABET[middle.position] == middle.notch:
44             left.step()
45
46 def encode_message(message, rotors, plugboard, reflector):
47     result = ""
48     for letter in message.upper():
49         if letter not in ALPHABET:
50             result += letter
51             continue
52         step_rotors(rotors)
53         c = plugboard.encode(letter)
54         for rotor in reversed(rotors):
55             c = rotor.encode_forward(c)
56         c = reflector.reflect(c)
57         for rotor in rotors:
58             c = rotor.encode_backward(c)
59         c = plugboard.encode(c)
60         result += c
61     return result
62
63 if __name__ == "__main__":
64     rotor_types, positions, ring_settings, plugboard_pairs,
65     reflector_pairs = read_settings("settings.txt")
66     rotors, plugboard, reflector = build_machine(rotor_types,
67     positions, ring_settings, plugboard_pairs, reflector_pairs)
68     with open("message.txt", "r") as f:
69         message = f.read()
70     result = encode_message(message, rotors, plugboard, reflector)
71     with open("result.txt", "w") as f:
72         f.write(result)
73     print("Сообщение зашифровано.")

```

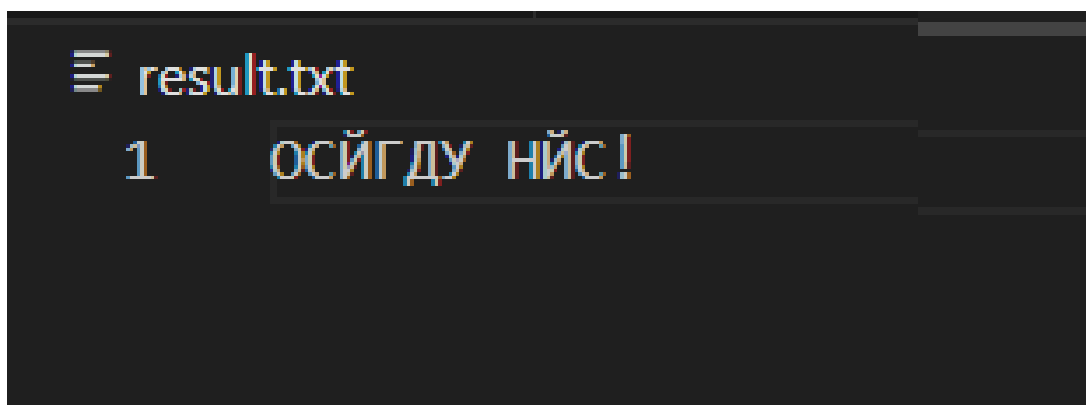


## 4 Пример работы программы



```
message.txt
1 ПРИВЕТ МИР!
```

Рисунок 4.1 – Текстовый файл до шифрования



```
result.txt
1 ОСЙГДУ НЙС!
```

Рисунок 4.2 – Текстовый файл после шифрования