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.

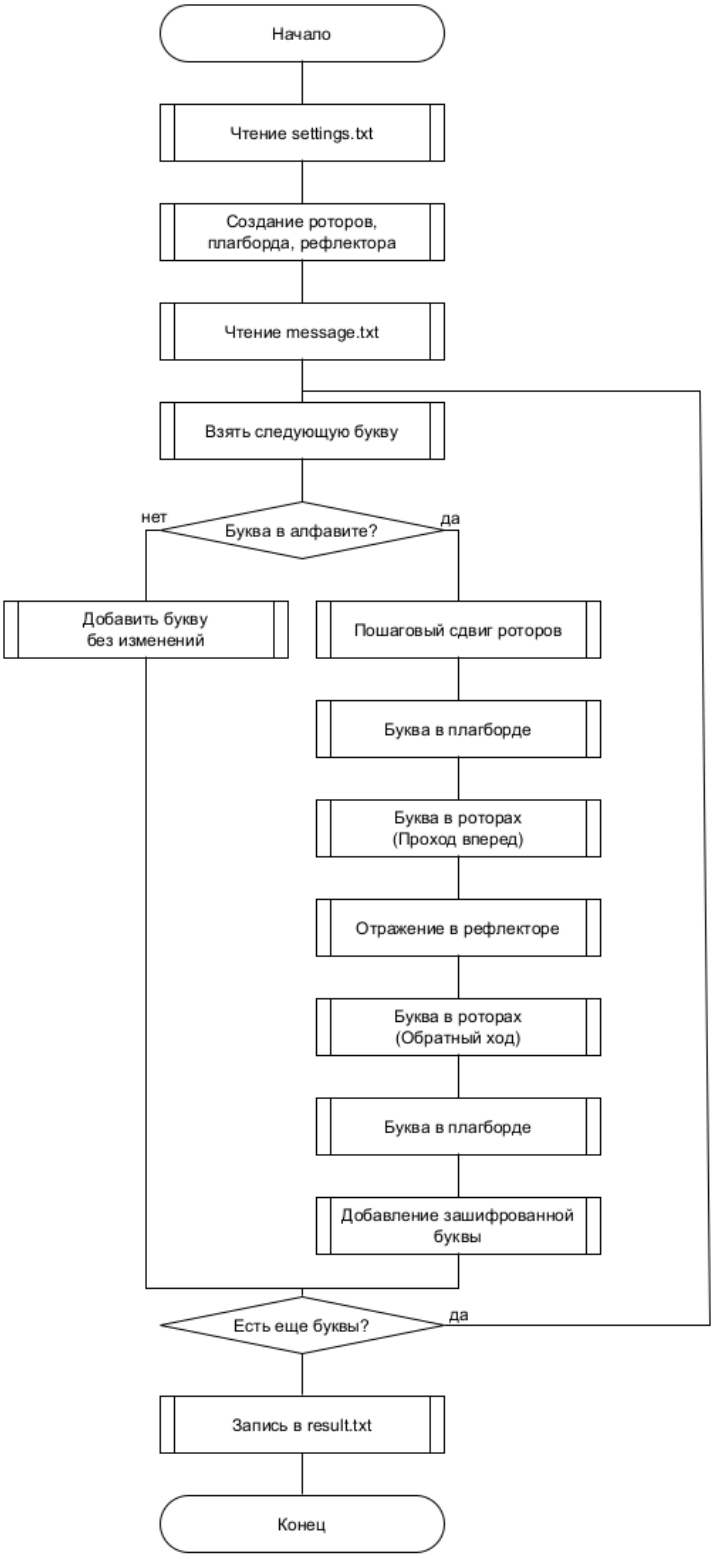
— **Однаалфавитная подстановка** — это криптографический метод, при котором каждой букве исходного алфавита соответствует только одна буква шифрованного алфавита на протяжении всего текста (шифр Цезаря).

— **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

```
import string

ALPHABET = string.ascii_uppercase

class Rotor:
    def __init__(self, wiring, notch, ring_setting=0, position=0):
        self.wiring = wiring
        self.notch = notch
        self.ring_setting = ring_setting
        self.position = position

    def step(self):
        self.position = (self.position + 1) % 26
        return self.position == ALPHABET.index(self.notch)

    def encode_forward(self, c):
        idx = ALPHABET.index(c)
        shifted_idx = (idx + self.position - self.ring_setting) % 26
        encoded_char = self.wiring[shifted_idx]
        out_idx = (ALPHABET.index(encoded_char) - self.position +
            self.ring_setting) % 26
        return ALPHABET[out_idx]

    def encode_backward(self, c):
        idx = ALPHABET.index(c)
        shifted_idx = (idx + self.position - self.ring_setting) % 26
        encoded_idx = self.wiring.index(ALPHABET[shifted_idx])
        out_idx = (encoded_idx - self.position + self.ring_setting) %
            26
        return ALPHABET[out_idx]
```

Листинг 3.2 – Файл `reflector.py`, описывающий поведение рефлектора

```
import string
ALPHABET = string.ascii_uppercase
class Reflector:
    def __init__(self, pairs=None):
        self.mapping = {}
        for ch in ALPHABET:
            self.mapping[ch] = ch
        if pairs:
            for pair in pairs:
                if len(pair) == 2:
                    a, b = pair[0].upper(), pair[1].upper()
                    if a in ALPHABET and b in ALPHABET:
                        self.mapping[a] = b
                        self.mapping[b] = a
    def reflect(self, c):
        return self.mapping.get(c.upper(), c)
```

Листинг 3.3 – Файл `plugboard.py`, описывающий поведение плагборда

```
import string
ALPHABET = string.ascii_uppercase
class Plugboard:
    def __init__(self, pairs=None):
        self.mapping = {}
        for ch in ALPHABET:
            self.mapping[ch] = ch
        if pairs:
            for pair in pairs:
                if len(pair) == 2:
                    a, b = pair[0].upper(), pair[1].upper()
                    if a in ALPHABET and b in ALPHABET:
                        self.mapping[a] = b
                        self.mapping[b] = a
    def encode(self, c):
        return self.mapping.get(c.upper(), c)
```

Листинг 3.4 – Файл `main.py`,

```
from rotor import Rotor
from reflector import Reflector
from plugboard import Plugboard
import string

ALPHABET = string.ascii_uppercase

ROTOR_WIRINGS = {
    "1": ("EKMFLGDQVZNTOWYHXUSPAIBRCJ", "Q"),
    "2": ("AJDKSIRUXBLHWTMCQGZNPYFVOE", "E"),
    "3": ("BDFHJLCPRTXVZNYEIWGAKMUSQO", "V"),
    "4": ("ESOVPZJAYQUIRHXLNFTGKDCMWB", "J"),
    "5": ("VZBRGITYUPSDNHLXAWMJQOFECK", "Z")
}

def read_settings(filename):
    with open(filename, "r") as f:
        lines = [line.strip() for line in f if line.strip()]
    rotor_types = lines[0].split()
    positions = [int(x) for x in lines[1].split()]
    ring_settings = [int(x) for x in lines[2].split()]
    plug_idx = next(i for i, line in enumerate(lines) if
       line.startswith("Plugboard"))
    reflector_idx = next(i for i, line in enumerate(lines) if
       line.startswith("Reflector"))
    plugboard_pairs = lines[plug_idx+1:reflector_idx]
    reflector_pairs = lines[reflector_idx+1:]
    return rotor_types, positions, ring_settings, plugboard_pairs,
       reflector_pairs

def build_machine(rotor_types, positions, ring_settings,
   plugboard_pairs, reflector_pairs):
    rotors = []
    for i, rtype in enumerate(rotor_types):
        wiring, notch = ROTOR_WIRINGS[rtype]
        rotors.append(Rotor(wiring, notch,
           ring_setting=ring_settings[i], position=positions[i]))
    plugboard = Plugboard(plugboard_pairs)
    reflector = Reflector(reflector_pairs)
    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,
          reflector_pairs = read_settings("settings.txt")
65     rotors, plugboard, reflector = build_machine(rotor_types,
          positions, ring_settings, plugboard_pairs, reflector_pairs)
66     with open("message.txt", "r") as f:
67         message = f.read()
68     result = encode_message(message, rotors, plugboard, reflector)
69     with open("result.txt", "w") as f:
70         f.write(result)
71     print("Сообщение␣зашифровано.")
```