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

Студент Романов Семен Константинович

Группа ИУ7-75Б

Предмет Защита информации

Студент _____ **Романов С. К.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Чиж И. С.**
подпись, дата *фамилия, и.о.*

2023 г.

ВВЕДЕНИЕ

Цель лабораторной работы — разработать программу шифровальной машины «Энигма» [1].

Задачи лабораторной работы:

1) провести анализ работы шифровальной машина «Энигма»;
2) описать алгоритм шифрования;
3) релизовать описанный алгоритм.

1 Аналитическая часть

Шифровальная машина «Энигма» состоит из трех основных частей:

1) роторы — диски обладающие 26 гранями, где каждая грань представляла собой нумерацию английского алфавита;
2) рефлектор — статический механизм, позволящий машине также расшифровать текст;
3) коммутатор — набор парных шифров.

1.1 Алгоритм работы машины

На вход «Энигме» подается строка, которая разбивается на символы. Далее символ проходит через коммутационную панель, который меняет символ в соотвествии с настройкой. После прохождения панели, символ проходит черз три диска и попадает на рефлектор. После работы рефлектора, символ отправляется обратно на диске и оканчательно шифруется через коммутатор. Затем один ротор совершает оборот, если ротор обернулся 26 раз, то поворачивается следующий.

2 Конструкторская часть

2.1 Разработка алгоритма

На рисунке 1 приведена схема работы шифровальной машины Энигма.

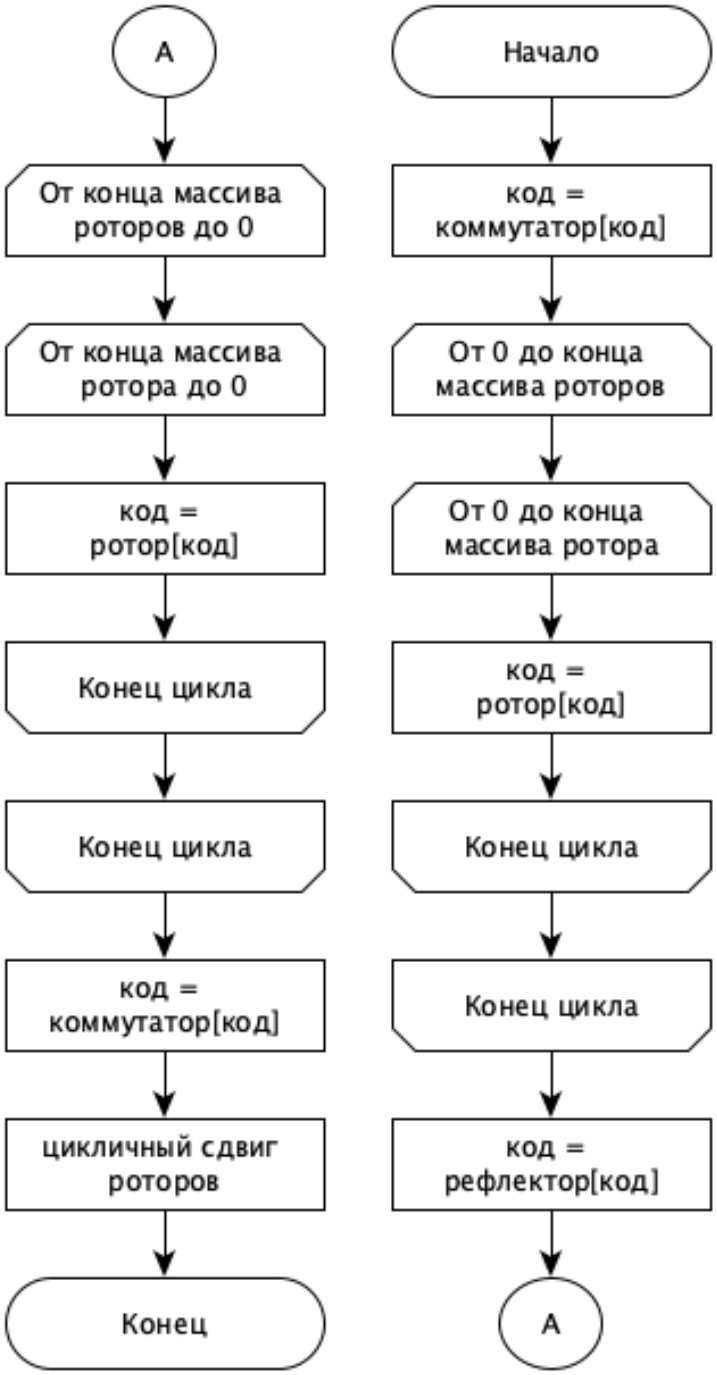

Рисунок 1 – Схема работы шифровальной машина Энигма

3 Технологическая часть

3.1 Средства реализации

Для реализации ПО был выбран язык C++ [2]. В данном языке есть все требующиеся инструменты для данной лабораторной работы. В качестве среды разработки была выбрана среда NeoVim[3].

3.2 Реализация алгоритма

Листинг 1 – Реализация алгоритма Энигмы.

```
uint8_t Enigma::encrypt(Encoder& encoder, uint8_t code)
{
    code = encoder.encode(code);
    uint64_t rotor_queue = 1;

    if(code > size_rotor)
    {
        throw std::out_of_range("Code bigger than size of rotor");
    }
    code = commutator.inner[code];

    for(auto& rotor : rotors)
    {
        code = rotor.inner[code];
    }
    code = reflector.inner[code];

    for(int i = num_rotors - 1; i >= 0; --i)
    {
        try
        {
            code = find_rotor(i, code);
        }
        catch(const std::overflow_error& e)
        {
            std::cout << e.what() << std::endl;
        }
    }
```

Листинг 2 – Реализация алгоритма Энигмы, продолжение.

```
1
2       counter++;
3       for(int i = 0; i < num_rotors; ++i)
4       {
5           if(counter % rotor_queue == 0)
6           {
7               rotor_shift(i);
8           }
9           rotor_queue *= size_rotor;
10      }
11      code = commutator.inner[code];
12
13      return code;
14  }
15
16  std::vector<uint8_t> Enigma::encrypt(Encoder& encoder, std::vector<uint8_t>&
        codes)
17  {
18      std::vector<uint8_t> res;
19      for(auto& symbol : codes)
20      {
21          try
22          {
23              uint8_t encoded_ch = encrypt(encoder, symbol);
24              uint8_t decoded_ch = encoder.decode(encoded_ch);
25              res.push_back(decoded_ch);
26          }
27          catch(std::overflow_error& e)
28          {
29              std::cout << e.what() << std::endl;
30          }
31      }
32
33      return res;
34  }
```

3.3 Тестовые данные

В таблице 1 приведены тесты, описанные в листинге 4 для алгоритма шифрования Энигмы. Применена методология черного ящика. Тесты пройдены *успешно*.

Листинг 3 – Реализация функциональных тестов.

```
TEST(Enigma, Encode)
{
    Encoder encoder = setup_encoder();
    Enigma enigma = setup_enigma();
    std::string new_message{"test"};
    std::vector<uint8_t> message(new_message.begin(), new_message.end());

    ASSERT_FALSE(enigma.encrypt(encoder, message) == message);
}

TEST(Enigma, Decode)
{
    Encoder encoder = setup_encoder();
    Enigma enigma_encrypt = setup_enigma();
    Enigma enigma_decrypt = setup_enigma();
    std::string new_message{"test"};
    std::vector<uint8_t> message(new_message.begin(), new_message.end());
    std::vector<uint8_t> enc_msg = enigma_encrypt.encrypt(encoder, message);

    ASSERT_TRUE(enigma_decrypt.encrypt(encoder, enc_msg) == message);
}

TEST(Enigma, Encode_Empty)
{
    Encoder encoder = setup_encoder();
    Enigma enigma = setup_enigma();
    std::string new_message{""};
    std::vector<uint8_t> message(new_message.begin(), new_message.end());

    ASSERT_TRUE(enigma.encrypt(encoder, message) == message);
}
```

Листинг 4 – Реализация функциональных тестов, продолжение.

```
TEST(Enigma, Encode_One)
{
    Encoder encoder = setup_encoder();
    Enigma enigma = setup_enigma();
    std::string new_message{"test"};
    std::vector<uint8_t> message(new_message.begin(), new_message.end());

    ASSERT_FALSE(enigma.encrypt(encoder, message) == message);
}

TEST(Enigma, Decode_One)
{
    Encoder encoder = setup_encoder();
    Enigma enigma_encrypt = setup_enigma();
    Enigma enigma_decrypt = setup_enigma();
    std::string new_message{"F"};
    std::vector<uint8_t> message(new_message.begin(), new_message.end());
    std::vector<uint8_t> enc_msg = enigma_encrypt.encrypt(encoder, message);

    ASSERT_TRUE(enigma_decrypt.encrypt(encoder, enc_msg) == message);
}
```

Таблица 1 – Функциональные тесты

Входная строка	Выходная строка
$TEST$	$FGxKNR$
$FGxKNR$	$TEST$
F	R
R	F

ЗАКЛЮЧЕНИЕ

В данной лабораторной работе:

1) проведен анализ работы шифровальной машина «Энигма»;
2) описан алгоритм шифрования;
3) реализован описанный алгоритм;

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Список литературы

1. Enigma german code device. https://www.britannica.com/topic/Enigma-German-code-device. дата обращения: 17.09.2023.
2. Язык программирования C++. https://learn.microsoft.com/en-us/cpp/cpp/cpp-language-reference?view=msvc-170. дата обращения: 17.09.2023.
3. Neovim. https://neovim.io/. дата обращения: 17.09.2023.