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 работа №1

по курсу «Защита информации»

на тему: «Шифровальная машина Энигма»

Студент ИУ7-73Б (Группа) _____ (Подпись, дата) Лысцев Н. Д. (И. О. Фамилия)

Преподаватель _____ (Подпись, дата) Чиж И. С. (И. О. Фамилия)

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной лабораторной работы является программная реализация аналога шифровальной машины «Энигма».

Для достижения поставленной цели необходимо решить следующие задачи:

1) описать алгоритм работы шифровальной машины «Энигма»;
2) выбрать средства программной реализации;
3) реализовать данный алгоритм.

1 Аналитический раздел

В данном разделе будет формально описано устройство шифровальной машины «Энигма», а также кратко будет описан алгоритм ее работы.

1.1 Шифровальная машина «Энигма»

Энигма – криптографическая машина, созданная немецкими военными для обеспечения своих коммуникаций [1]. В ее основе лежали 3 основных механизма:

- *роторы* – наборы вращающихся дисков. На каждом диске было 26 граней, на каждой из которых была нанесена буква английского алфавита. Число роторов варьируется от трех до восьми [1; 2]. Они используются для преобразования одной буквы в другую;
- *рефлектор* – статичный механизм, позволяющий не вводить дополнительную операцию расщифрования;
- *коммутатор* – механизм, позволяющий оператору шифровальной машины соединять попарно одни буквы с другими.

1.2 Алгоритм работы шифровальной машины «Энигма»

Пусть Энигма состоит из трех роторов и одного рефлектора, а также 26-ти соединительных проводов для коммутационной панели. Алгоритм работы Энигмы состоит из следующих шагов:

1) на вход поступает текстовое сообщение;
2) каждый символ поступает в коммутационную панель;
3) определяется символ, парный данному;
4) символ проходит через каждый ротор, где осуществляется преобразование в новый символ;
5) после 3 роторов символ поступает в рефлектор и определяется символ, парный данному;

6) полученный на предыдущем шаге символ в обратном направлении проходит через все роторы;

7) новый символ поступает в коммутатор и определятся символ, парный данному;

8) определенный на предыдущем шаге символ есть шифр исходного символа;

9) первый ротор поворачивается на одну позицию. Если первый ротор совершил полный оборот, то на одну позицию поворачивается соседний ротор. Если второй ротор совершил полный оборот, то третий поворачивается на одну позицию.

Вывод

В данном разделе было формально описано устройство шифровальной машины «Энигма», а также был кратко описан алгоритм ее работы.

2 Конструкторский раздел

На рисунке 2.1 представлена схема алгоритма работы шифровальной машины «Энигма».

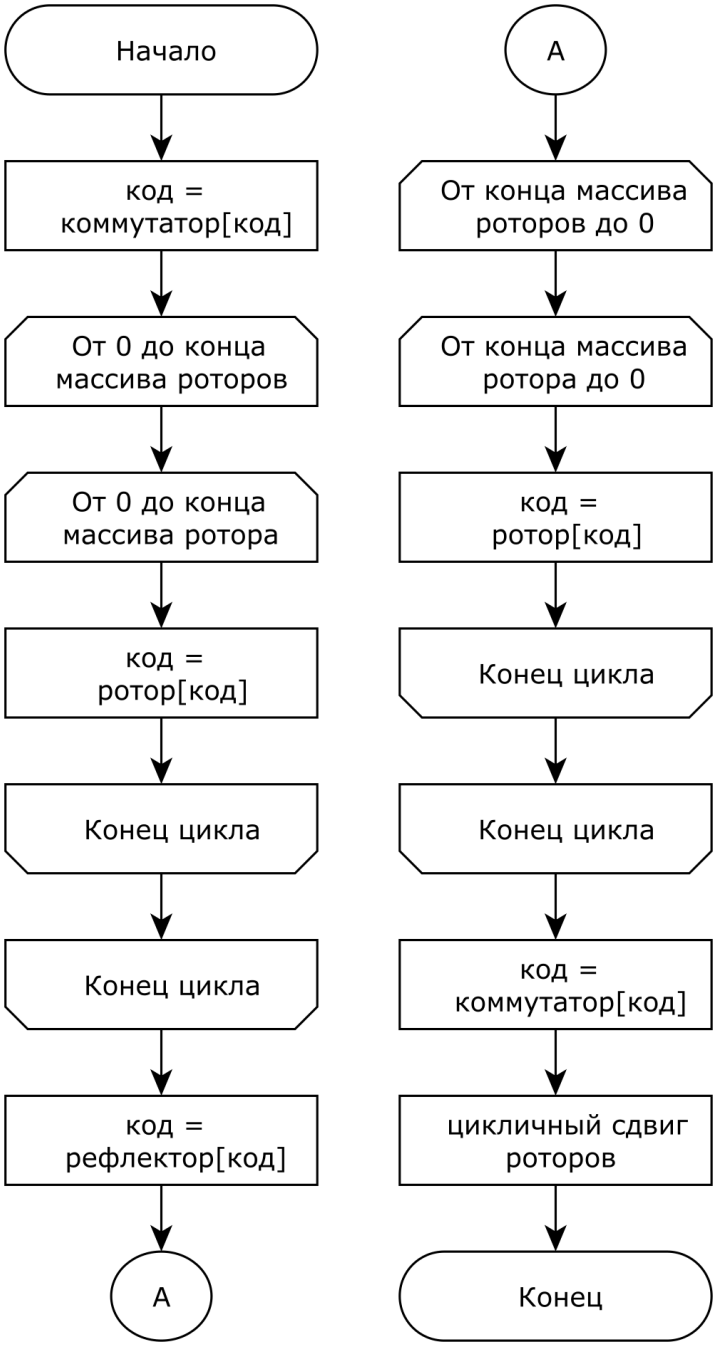

Рисунок 2.1 – Схема алгоритма работы шифровальной машины «Энигма»

Вывод

В данном разделе была представлена схема алгоритма работы шифровальной машины «Энигма».

3 Технологический раздел

В данном разделе будут перечислены требования к программному обеспечению, средства реализации и листинги кода.

3.1 Требования к программному обеспечению

К программе предъявляется ряд требований:

– мощность алфавита не должна превышать 64 символа;

– программа должна предоставлять возможность шифрования и расшифровки произвольного файла, а также текстового сообщения;

– программа должна корректно работать с пустым однобайтовым файлом;

– программа должна уметь обрабатывать файл архива.

3.2 Средства реализации

В качестве языка программирования для этой лабораторной работы был выбран $C++$ [3].

3.3 Сведения о модулях программы

Программа состоит из трех модулей:

– `main.cpp` — файл, содержащий точку входа в программу;

– `enigma.cpp` — модуль, реализующий шифровальную машину «Энигма»;

– `encoder.cpp` — модуль, реализующий -кодировщик.

3.4 Тестирование

Таблица 3.1 – Функциональные тесты

Входная строка	**Шифрованная строка**
HELLO	XFPRG
XFPRG	HELLO
HEL LO	XFP RG
XFP RG	HEL LO

Все тесты были успешно пройдены. Тесты с файлами были также успешно пройдены.

3.5 Реализации алгоритмов

В листинге 3.1 представлено объявление класса, реализующего шифровальную машину «Энигма».

Листинг 3.1 – Объявление класса, реализующего шифровальную машину «Энигма»

```
class Enigma {
private:
    int counter = 0;
    int sizeRotor;
    int numRotors;
    vector<uint8_t> reflector;
    vector<uint8_t>
    vector<vector<uint8_t>> rotors;
    Encoder encoder;

    uint8_t getIndexByValueInRotor(int numRotor, uint8_t code,
       bool &isFind);

    void rotorRotate(int numRotor);

    uint8_t encrypt(uint8_t code, bool &isValid);

public:
    Enigma(const vector<uint8_t> &alph, int numRotors);

    void setReflector(vector<uint8_t> &newReflector);

    void setCommutator(vector<uint8_t> &newCommutator);

    void setRotors(vector<vector<uint8_t>> &newRotors);

    uint8_t encryptCharByCode(uint8_t code, bool &isEncrypt);

    string encryptString(const string &str);

    void encryptFile(const string &filepath, const string
       &outputFilepath);
};
```

В листинге 3.2 представлена реализация алгоритма шифрования символа.

Листинг 3.2 – Реализация шифрования символа

```
uint8_t Enigma::encryptCharByCode(uint8_t code, bool &isEncrypt)
  {
    if (code > this->sizeRotor) {
        isEncrypt = false;
        return 0;
    }

    int encryptCode = code;

    encryptCode = this->commutator[encryptCode];

    for (int i = 0; i < this->numRotors; ++i)
        encryptCode = this->rotors[i][encryptCode];

    encryptCode = this->reflector[encryptCode];

    for (int i = this->numRotors - 1; i >= 0; --i) {
        encryptCode = this->getIndexByValueInRotor(i,
           encryptCode, isEncrypt);
        if (!isEncrypt)
            return 0;
    }

    int rotorQueue = 1;
    this->counter += 1;

    for (int i = 0; i < this->numRotors; ++i) {
        if (this->counter % rotorQueue == 0)
            this->rotorRotate(i);

        rotorQueue *= this->sizeRotor;
    }

    encryptCode = this->commutator[encryptCode];
    isEncrypt = true;
    return encryptCode;
}
```

В листинге 3.3 представлена реализация алгоритма шифрования символа с использованием кодировщика.

Листинг 3.3 – Реализация шифрования символа с использованием кодировщика

```
uint8_t Enigma::encrypt(uint8_t symbol, bool &isValid) {
    uint8_t symbolCode = this->encoder.encode(symbol, isValid);
    if (!isValid) {
        cout << "no encode" << endl;
        return 0;
    }
    isValid = false;
    uint8_t encryptSymbolCode =
       this->encryptCharByCode(symbolCode, isValid);
    if (!isValid) {
        cout << "no encrypt" << endl;
        return 0;
    }
    isValid = false;
    uint8_t encryptSymbol =
       this->encoder.decode(encryptSymbolCode, isValid);
    if (!isValid) {
        cout << "no decode" << endl;
        return 0;
    }

    isValid = true;
    return encryptSymbol;
}
```

В листингах 3.4 и 3.5 представлена реализация шифрования строки.

Листинг 3.4 – Реализация шифрования строки (начало)

```
string Enigma::encryptString(const string &str) {
    std::string encryptStr;

    for (uint8_t symbol: str) {
        bool isValid = false;

        char encryptSymbol =
           static_cast<char>(this->encrypt(symbol, isValid));
        if (!isValid) {
            encryptStr += static_cast<char>(symbol);
```

Листинг 3.5 – Реализация шифрования строки (конец)

```
            continue;
        }

        encryptStr += static_cast<char>(encryptSymbol);
    }

    return encryptStr;
}
```

В листингах 3.6 и 3.7 представлена реализация шифрования произвольного файла.

Листинг 3.6 – Реализация шифрования произвольного файла (начало)

```
void Enigma::encryptFile(const string &filepath, const string
   &outputFilepath) {
    ifstream file(filepath, ios::binary);
    if (!file.is_open()) {
        cout << "file dont open" << endl;
        return;
    }

    ofstream outputFile(outputFilepath, ios::binary);
    if (!outputFile.is_open()) {
        cout << "out file dont open" << endl;
        return;
    }

    size_t filesize = filesystem::file_size(filepath);
    size_t currByte = 0;

    while (currByte != filesize) {
        currByte++;
        char byte;
        file.read(&byte, sizeof(byte));

        unsigned char firstHalf = (byte >> 4) & 0x0F;
        unsigned char secondHalf = byte & 0x0F;

        bool isValid = false;
        char encryptFirstHalf =
           static_cast<char>(this->encrypt(firstHalf, isValid));
```

Листинг 3.7 – Реализация шифрования произовльного файла (конец)

```
        if (!isValid) {
            outputFile.write(&byte, sizeof(byte));
            continue;
        }

        char encryptSecondHalf =
           static_cast<char>(this->encrypt(secondHalf, isValid));
        if (!isValid) {
            outputFile.write(&byte, sizeof(byte));
            continue;
        }

        char encryptByte = static_cast<char>((encryptFirstHalf
           << 4) | encryptSecondHalf);

        outputFile.write(&encryptByte, sizeof(encryptByte));
    }

    file.close();
    outputFile.close();
}
```

Вывод

В данном разделе были перечислены требования к программному обеспечению, средства реализации и листинги кода.

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были решены следующие задачи:

1) описан алгоритм работы шифровальной машины «Энигма»;

2) выбраны средства программной реализации;

3) реализован данный алгоритм.

Цель работы, а именно программная реализация аналога шифровальной машины «Энигма», также была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алгоритм переносной шифровальной машины Энигма [Электронный ресурс]. — Режим доступа: https://cyberleninka.ru/article/n/algoritm-perenosnoy-shifrovalnoy-mashiny-enigma/viewer (дата обращения: 04.10.2024).
2. Программная реализация шифровальной машины «Энигма» на языке Си [Электронный ресурс]. — Режим доступа: https://habr.com/ru/articles/721790/ (дата обращения: 04.10.2024).
3. Справочник по языку C++ [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=msvc-170 (дата обращения: 28.09.2022).