МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

Высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В.И. ВЕРНАДСКОГО»

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

**РАЗРАБОТКА ЭМУЛЯТОРА ШИФРОВАЛЬНОЙ МАШИНЫ «ЭНИГМА»**

Курсовая работа

По дисциплине программирование

Студента 1 курса группы ИВТ-б-о-191(2)

Тройслит Андрея Александровича

Направления подготовки 09.03.01 «Информатика и вычислительная техника»

|  |  |  |
| --- | --- | --- |
| Научный руководитель  Старший преподаватель кафедры  Компьютерной инженерии и моделирования | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись,дата) | Чабанов В.В. |

Симферополь 2020

# **Реферат**

Разработка симулятора шифровальной машины «Энигма» .- Симферополь: ФТИ КФУ им В.И.ВЕРНАДСКОГО, 2019, стр.,

*Объект разработки* – компьютерная программа создающая симуляцию шифровальной машины Энигма.

*Цель работы* – создать программу, дающую возможность шифрования сообщений по принципу шифрования Энигмы ,использующей принцип шифрования Джефферсона и шифра Цезаря.

Рассмотрены основные принципы шифрования Энигмы, изучены возможности для написания кода и его визуализации.

ПРОГРАММИРОВАНИЕ, ШИФРОВАНИЕ, C++, ЭНИГМА, VISUALSTUDIO, ШИФРАТОР ДЖЕФФЕРСОНА, ШИФР ЦЕЗАРЯ

# **Оглавление**

[Реферат 2](#_Toc72969466)

[Оглавление 3](#_Toc72969467)

[Введение 4](#_Toc72969468)

[Глава 1 Постановка задачи 5](#_Toc72969469)

[1.1 Цель проекта: 5](#_Toc72969470)

[1.2 Аналоги такого проекта: 5](#_Toc72969471)

[1.3 Техническая часть 5](#_Toc72969472)

[1.3.1 Требования к данному симулятору: 5](#_Toc72969473)

[1.3.2 Требования к алгоритмам шифрования: 5](#_Toc72969474)

[1.3.3 Требования к параметрам запуска данного симулятора: 6](#_Toc72969475)

[Глава 2 Программная реализация симулятора 7](#_Toc72969476)

[2.1 Анализ используемых инструментов 7](#_Toc72969477)

[2.2 Использованные фреймворки (платформы для написания кода): 7](#_Toc72969478)

[2.3 Описание алгоритма шифрования: 7](#_Toc72969479)

[2.4 Программная реализация шифрования 9](#_Toc72969480)

[2.4.1 «Голова» программы 9](#_Toc72969481)

[2.4.2 «Тело» программы 11](#_Toc72969482)

[2.5 Работа программы 16](#_Toc72969483)

[Глава 3 Тестирование проекта 18](#_Toc72969484)

[3.1 Первое тестирование. 18](#_Toc72969485)

[3.2 Второе тестирование 19](#_Toc72969486)

[3.3 Третье тестирование 19](#_Toc72969487)

[3.4 Четвертое тестирование 20](#_Toc72969488)

[Глава 4 Дальнейшее развитие проекта 22](#_Toc72969489)

[4.1 Перспективы развития в техническом направлении: 22](#_Toc72969490)

[4.2 Перспективы развития, как надежного шифрования. 22](#_Toc72969491)

[Заключение 24](#_Toc72969492)

[Литература 26](#_Toc72969493)

# **Введение**

Целью данной работы, как было уже сказано выше является закрепление навыков работы с языком программирования С++, получение навыки по работе с методами шифрования и дешифрования информации, изучение методов визуализации с помощью WindowsForms в среде разработки VisualStudio.

В настоящее время секретность информации играет большую роль. Для скрытия информации от посторонних глаз, люди часто прибегают к каким- либо методам шифрования.

Зашифрованная информация позволяет людям более безопасно общаться между собой, а также сохраняет данные пользователей. Проект симулятора шифровальной машины позволит людям взглянуть на один из способов шифрования информации, который был особенно популярен примерно половину века назад.

Задачи которые необходимо решить для разработки данного приложения:

* Изучение WindowsForms для создания интерфейса программы, и последующей визуализации.
* Изучение принципа шифратора Джефферсона и шифра Цезаря для создания шифратора Энигмы.
* Подробное изучения языка программирования C++.

# **Глава 1 Постановка задачи**

## **1.1 Цель проекта:**

Конечная версия данной должны будет мочь зашифровать текст, предложенный пользователем, достаточно сильным и серьезным алгоритмом, для взлома которого, без известного ключа шифрования, понадобиться достаточно большое время, прежде чем можно будет прочитать изначально зашифрованное сообщение.

## **1.2 Аналоги такого проекта:**

* Сайт <https://cryptii.com/pipes/enigma-decoder> полностью реализующий шифровальную машину Энигма в формате веб-сайта.
* Физические аналоги шифровальной машины Энигмы.

## **1.3 Техническая часть**

В данной части описаны поставленные требования, которые конечная программа должны будет выполнять.

### **1.3.1 Требования к данному симулятору:**

* Возможность зашифровать сообщения с известным ключом шифрования.
* Возможность зашифровать сообщения без известного ключа шифрования.

### **1.3.2 Требования к алгоритмам шифрования:**

* Алгоритм шифрования программы должен быть аналогичным тому, который используется в физическом аналоге шифровальной машины Энигма.
* Возможность менять ключ для шифровки, не прибегая к перезапуску программы

### **1.3.3 Требования к параметрам запуска данного симулятора:**

* Операционная система Windows, с версией начиная от Windows 7.
* Оперативная память 512 Мб
* Физическая память – 10 Мб

# **Глава 2 Программная реализация симулятора**

## **2.1 Анализ используемых инструментов**

Языки программирования, которые были использованы для создания данного эмулятора:

* С++, основной язык программирования, на котором написана большая часть работы. Он позволяет удобно реализовывать алгоритмы обработки данных, что является первостепенной задачей в текущем проекте.
* WindowsForms, дополнительное программное обеспечение, позволяющее без особых усилий нарисовать интерфейс программы, который будет легко понятен пользователю.

## **2.2 Использованные фреймворки (платформы для написания кода):**

* VisualStudio 2019 –в ходе выбора платформы был выбор между VisualStudio и QT, однако в конечном итоге была выбрана именно VisualStudio из-за своего понятного интерфейса, возможности работы с разными языками, а также опыта работы с данной программой, что при написании большой и сложной работы играет немаловажную роль.

## **2.3 Описание алгоритма шифрования:**

Для шифрования текста, в шифровальной машине «Энигма», используется два принципа шифрования, один из них - шифратор Джефферсона, а другой называется шифром Цезаря. О принципах их шифрования, будет рассказано далее.

Для шифрования текста используется устройство шифратора Джефферсона(Рис.2.3.1) - это устройство представляло собой несколько дисков, на которых нанесены буквы алфавита, идущие в случайном порядке. Для проведения шифровки сообщения, изначальный текст разбивается на части по 36 букв в каждой, после этого, на первом диске выбирается первая буква из набора, которая не стоит прямо на цилиндрах, после на втором диске ищется вторая буква по той-же линии, и по такой аналогии выстраивались оставшиеся 34 буквы. После выставления, таким образом, набора букв сообщения, собеседнику передавали код, состоящий из набора букв, который ему необходимо было выставить на цилиндрах, и после этого искать в остальных линиях осмысленное сообщение.

Кроме шифратора Джефферсона, в шифровальной машине «Энигма», используется шифр Цезаря (Рис.2.3.2), смысл которого заключается в смещении каждой буквы. Например, вместо буквы А**,** будет идти буква **В**, таким образом, полученное смещение будет составлять 2 буквы, т.е. буква **Б** будет заменена на букву **Г,** и по такой аналогии, будет подвергаться шифровке весь оставшийся текст.

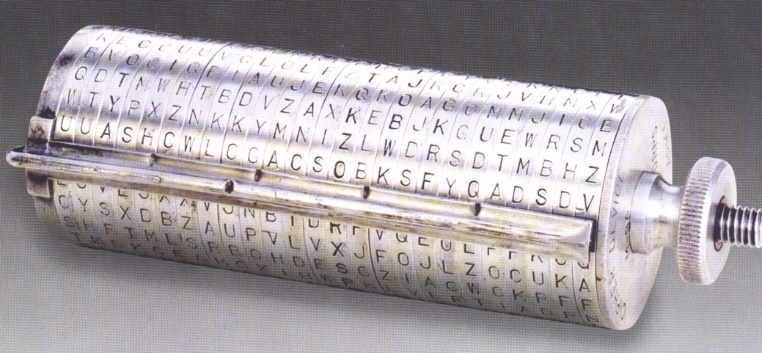


Рисунок 2.1 Шифратор Джефферсона

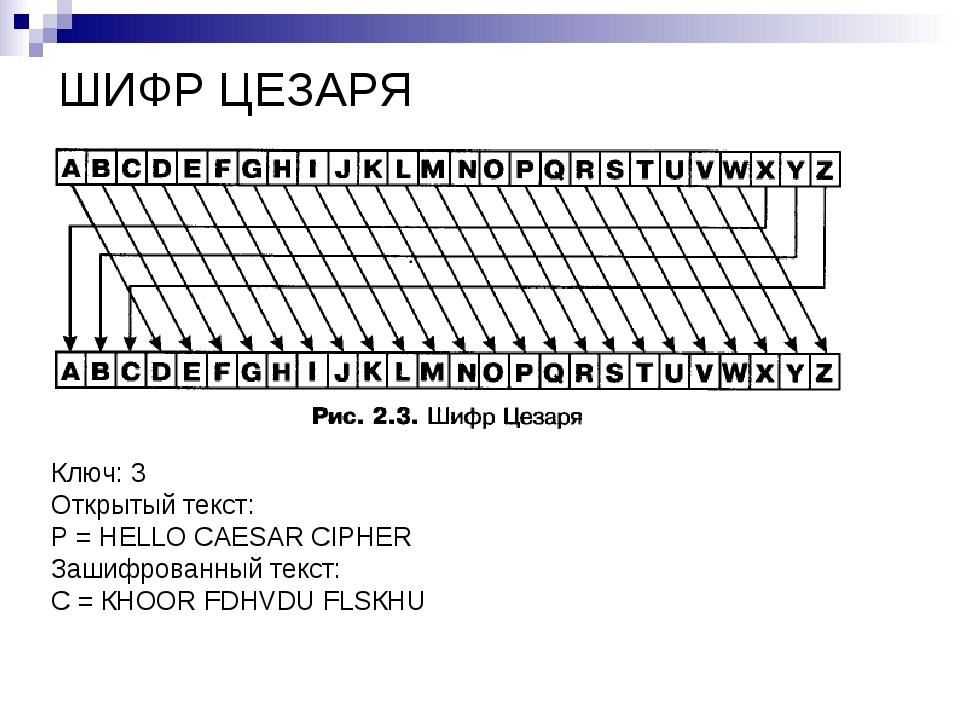


Рисунок 2.2 Шифрование Цезаря

## **2.4 Программная реализация шифрования**

### **2.4.1 «Голова» программы**

В голове программы содержится перечисление необходимых библиотек, несколько массивов с данными, а также инициализация переменных. Подробнее о них пойдет речь далее.

#### 2.4.1.1. Использованные библиотеки

В данной программе используется 4 библиотеки и 1 команда для облегчения процесса написания кода

Библиотеки:

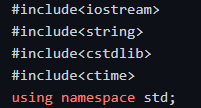


Рис.2.3 использованные библиотеки

Подробнее о каждой использованной библиотеке и ее конкретной цели в данном проекте:

* «iostream» - помогает выводить на печать символы
* «string» - организация массивов данных
* «cstdlib» - организация сложных вычислений
* «ctime» - организация рандомизации генерируемого ключа

Теперь о команде, которая используется для облегчения написания кода программы:

using namespace std;

Данная команда используется для того, чтобы каждый раз не писать следующие приставки к командам:

std :: cout << ”Text” << std :: endl ;

Если мы введем команду «using namespace std;» то пример команды сверху, сократиться до следующего, что экономит очень много времени при написании кода.

cout << “Text” << endl ;

#### 2.4.1.2 Использованные массивы данных

В данной программе используется 5 заполненных и 4 незаполненных массивов с данными.

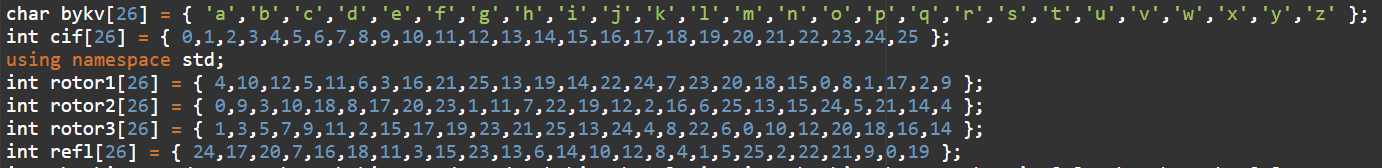


Рис.2.4 скриншот заполненных массивов

Заполненные

* char bykv[26] – хранит буквы английского алфавита для того, чтобы можно было преобразовывать буквы в цифры и обратно
* int rotor1[26] – хранит буквы английского алфавита в формате цифр, расположенные в том же порядке, как и на реальном первом роторе «Энигмы»
* int rotor2[26] – хранит буквы английского алфавита в формате цифр, расположенные в том же порядке, как и на реальном втором роторе «Энигмы»
* int rotor3[26] – хранит буквы английского алфавита в формате цифр, расположенные в том же порядке, как и на реальном третьем роторе «Энигмы»
* int refl[26] – хранит буквы английского алфавита в формате цифр, расположенные в том же порядке, как и на реальном рефлекторе «Энигмы»

Незаполненные массивы:

* int key\_int[3] – хранение ключа в виде цифр
* char key\_char[3] – хранение ключа в виде букв
* char primal\_text\_char[200] = {'@'} – хранение входного текста в виде букв, относится к незаполненным массивам, поскольку заполнен бесполезными знаками, которые необходимы только для работы программы.
* int primal\_text\_int[200] – хранение входного текста в виде цифр

Инициализированные переменные:

* int temp,temp\_1,temp\_2,temp\_3,temp\_4,temp\_5,temp\_6,temp\_7,temp\_8 – хранят букву в виде цифры в процессе путешествия с ротора на ротор.
* int temp\_key – сумма либо разность цифр роторов в процессе шифрования машины
* int checking\_ready = 1; char cheking\_en\_de\_code,cheking\_key\_logic; int checking\_key = 1 – переменные использованные в теле программы, для общения с пользователем
* int rotate\_1, rotate\_2, rotate\_3,key\_int\_temp\_1,key\_int\_temp\_2, key\_int\_temp\_3,temp\_rotate – переменные использующиеся при вычислении прокрутки роторов

### **2.4.2 «Тело» программы**

Тело программы поделено на основную часть, он же int main(){}, и часть с функцией. Поговорим о них поподробнее.

#### 2.4.2.1 Основная часть кода

Для начала рассмотрим основную часть программы «int main(){}». В основной части находится: привязка ко времени, установление английской локализации, приветствие программы, один большой цикл, где происходит общение с пользователем. Рассмотрим все вышеперечисленные элементы поподробнее:

**1)** Привязка ко времени – привязка ко времени задается следующей командой:

srand(time(NULL));

Данная команда необходима для того, чтобы при каждом запуске программы, при выборе для генерации ключа автоматически, программа не выдавала одинаковые ключи, что невероятно сильно повысит шансы на надежность зашифрованной информации.

**2)** Установление английской локализации – данная строка кода необходима для того, чтобы программа на выводе показывала английские буквы, а не набор непонятных символов. Данное действие задается следующей командой:

setlocale(LC\_ALL, "English");

**3)** Приветствие программы, данная часть кода состоит из одной строки, которая выводит для пользователя приветственное сообщение на экран, при каждом запуске программы:

cout << "Hello, it's simulator Enigma,let's get to work!";

**4)** Цикл, где происходит общение с пользователем. Данный цикл занимает большую часть кода основной части. Для начала пользователю на экран выводиться сообщение, с вопросом о том, что он хочет сделать: начать шифрование информации, либо выход из программы. Этот вопрос задается следующей строкой кода:

cout <<endl<< "You want to encode?,Type 'e' to encode,or type 'x' to close the programm" << endl;

После ввода ответа пользователя, программа обрабатывает его , после чего программа либо закрывается, либо начинается процесс шифрования, о котором будет сказано чуть ниже, либо программа не может распознать ответ пользователя, и предлагает вновь ввести ответ на вопрос о цели открытия программы. Данная часть состоит из команды switch(), в которой записаны варианты обработки ввода пользователя:

switch (cheking\_en\_de\_code) {

case 'e': {checking\_key=1; encode(); break; }

default:cout << endl << "I cannot read your answer,please try again" << endl;

#### 2.4.2.2 Часть с функцией

Данная часть программы представляет собой самую большую часть кода программы, поскольку именно тут находится механизм шифрования. Данная часть подразделяется на несколько частей: часть с ключом шифрования, часть с шифрованием текста, часть с прокруткой ротора. О каждой из частей пойдет речь далее.

1)Часть с ключом шифрования.

Данная часть подразумевает с собой общение с пользователем, где пользователь вводит свой ответ на вопрос о наличии ключа шифрования:

cout << "You have a key?Type 'y' if you got,or 'n', and i generate him" << endl;

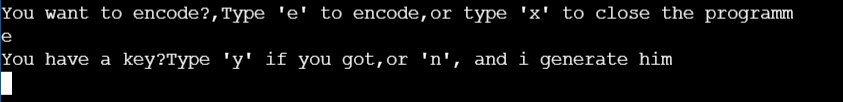


Рис.2.5 вопрос о наличии ключа

Существует три обработчика вводимых данных: есть ключ, нет ключа, не распознано. Данные обработчики написаны при помощи конструкции if()else(). При ответе пользователя о том, что ключ есть, программа запрашивает его, после программа переходит к запросу текста и непосредственно шифрованию, о чем будет рассказано далее. При ответе пользователя, что ключа нет, программа сама сгенерирует ключ, при помощи простой команды rand() ,после чего при помощи заполненного массива с буквами и при помощи цикла, ключ из версии из цифр, будет переведен в версию из букв и выведен на экран пользователя:

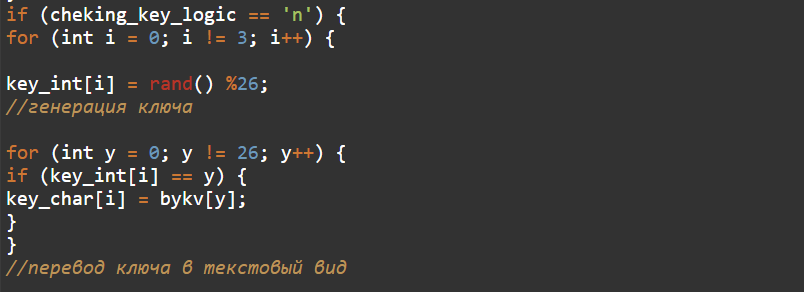


Рис.2.6 часть кода с генерацией ключа

При невозможности прочитать ответ пользователя, программа выдаст соответствующее сообщение о том, что не может прочитать отчет, и предложит заново ввести ответ.

2) Часть с шифрованием текста. Данная функция является самой важной частью программы, где происходит сам процесс шифрования. Подробнее технология шифрования была описана выше, поэтому в данной части отчета я продемонстрирую код программы, отвечающий за сам процесс шифрования:

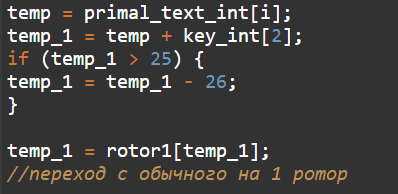


Рис.2.7 обработка буквы на 1 роторе

На предоставленном рисунке показан участок программы, отвечающий за обработку буквы на 1 роторе. Аналогичные блоки кода идут далее, но на каждом разные вычисления, для того, чтобы правильно зашифровать сообщение.

**3)** Часть с прокруткой ротора. Данная часть кода предназначена, для того, чтобы увеличить надежность шифрования, за счет прокрутки роторов. За это отвечает следующий участок кода:

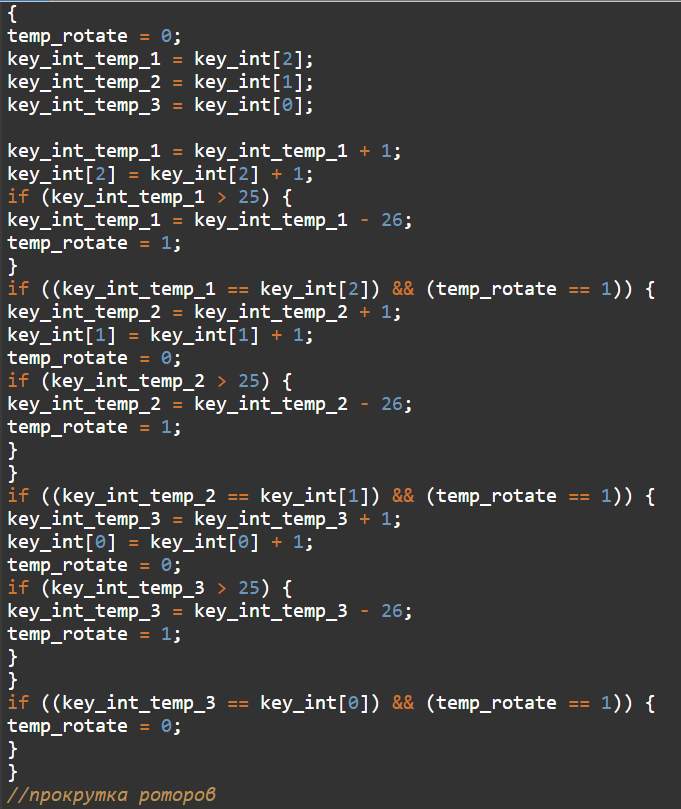


Рис.2.8 часть кода с прокруткой ротора

Прокрутка +роторов действует по следующему принципу: первый ротор прокручивается на следующую букву каждый раз, когда на вход поступает буква, второй ротор прокручивается каждый раз, когда первый ротор совершает один полный оборот, а третий ротор прокручивается только тогда, когда второй ротор совершит один полный оборот. Таким образом, перед тем, как следующая буква зашифруется по тому же ключу шифрования, что и первая буква, пройдет 25\*26\*26=16900 букв, что достаточно даже для очень больших сообщений, поэтому шифровальная машина «Энигма» может гарантировать большую надежность данных без использования специальных программ для взлома шифра и последующей расшифровки сообщений. Именно эта часть шифровальной машины наиболее уязвима, поэтому на реальных моделях, которые обеспечивали наибольшую сохранность данных, третий ротор мог прокручиваться не один раз, что очень сильно повышало надежность шифрования.

В реальных аналогах шифровальной машины «Энигма» часто применялся трюк с заменой букв на выходе. Предположим на обычной шифровальной машине, при вводе буквы А мы могли получить Л, на усовершенствованной версии можно было сделать замену выходящей буквы Л например на Е, что невероятно сильно повышало защищенность информации. В данном симуляторе отсутствует данная функция, однако, возможно она появится в будущих версиях программы.

## **2.5 Работа программы**

В данном разделе представлена работа программы.

Для того, чтобы зашифровать текст, после открытия программы нам необходимо ввести «у», для того чтобы перейти к выбору ключа. После этого выбираем тип ключа, либо мы введем его самостоятельно, либо программа автоматически сгенерирует его и выведет на экран. После вводим текст и получаем результат:

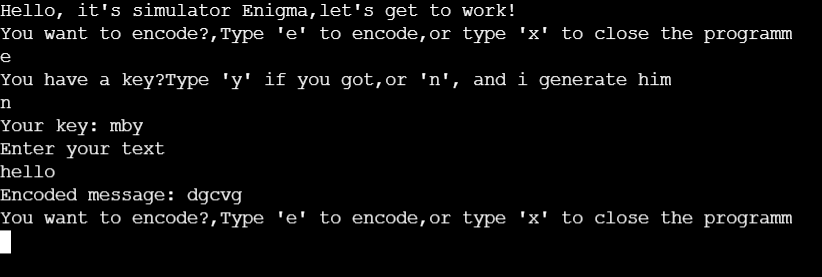


Рис.2.9 работа программы на зашифровку текста

Для того, чтобы расшифровать текст, нам необходимо перейти к выбору ключа, и ввести известный нам ключ, а после ввести зашифрованный текст и программа выдаст изначальный текст:

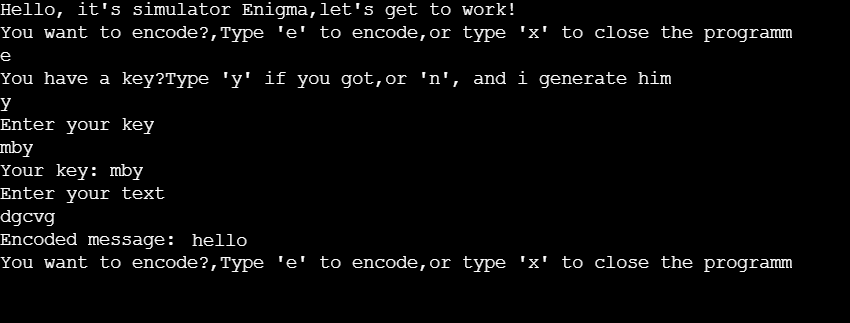


Рис.2.10 работа программы на расшифровку текста

# **Глава 3 Тестирование проекта**

## **3.1 Первое тестирование.**

Тестирование первой версии программы было проведено 12.04.2020. Тогда программа была отправлена на тестирование, и 13.04.2020 был получен следующий отчет:

Тест №1 /Test 1

|  |  |
| --- | --- |
| Среда тестирования  /Environment | Windows 10 |
| Предварительные действия  /Pre requisites | - |
| Комментарии  /Comments | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Шаг  /Step | Действие  /Action | Ожидаемый результат  /Expected results | Результат | Пройден,  Не пройден, Недоступен | Комментарии  /Notes |
| 1 | Двойной ЛКМ по ярлыку программы | Открытие приложения | Приложение открылось | Пройден | - |
| 2 | Ввести в командную строку «e» для дальнейшего взаимодействия с программой | Программа спросит вас про ключ шифрования | Программа запросила ключ | Пройден | - |
| 3 | Ввести в командную строку «у» для дальнейшего взаимодействия с программой | Программа предложит ввести ключ для шифрования | Предложение поступило | Пройден | Вместо букв непонятные символы, работать можно только «наугад» |
| 4 | Ввести тестовый ключ шифрования «abc» | Программа примет ключ и предложит ввести текст для шифрования | Программа приняла ключ и предложила ввести текст | Пройден | Вместо букв непонятные символы, работать можно только «наугад» |

Исходя из результатов тестирования, программный код был доработан, и отправлен на второе тестирование. В частности, был исправлена недоработка с отображением символом, выводимых программой на экран.

## **3.2 Второе тестирование**

На второе тестирование программа была отправлена 17.04.2020, а 20.04.2020 пришел отчет по работе программы:

Тест №2 / Test 2

|  |  |
| --- | --- |
| Среда тестирования  /Environment | Windows 10 |
| Предварительные действия  /Pre requisites | Открытие приложения, ввод ключа шифрования |
| Комментарии  /Comments | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Шаг  /Step | Действие  /Action | Ожидаемый результат  /Expected results | Результат | Пройден,  Не пройден, Недоступен | Комментарии  /Notes |
| 1 | Ввода текста для шифрования | Программа примет текст | Программа приняла текст | Пройден | - |
| 2 | Получение результата | Программа выдаст результат | Программа выдала результат | Пройден | - |
| 3 | Повторение действий 1-2 | Программа должна выдавать одинаковый текст при одинаковом ключе и исходном тексте | Программа выдает разные результаты | Не пройден | - |

В ходе изучения отчета, была выполнена работа по улучшению программного кода, а также исправлению некоторых недочетов и сбоев.

## **3.3 Третье тестирование**

На третье тестирование программа была отправлена 24.04.2020, и 25.04.2020 был получен следующий отчет:

Тест №2 / Test 2

|  |  |
| --- | --- |
| Среда тестирования  /Environment | Windows 10 |
| Предварительные действия  /Pre requisites | Открытие приложения, подготовка к вводу ключа |
| Комментарии  /Comments | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Шаг  /Step | Действие  /Action | Ожидаемый результат  /Expected results | Результат | Пройден,  Не пройден, Недоступен | Комментарии  /Notes |
| 1 | При запросе ключа шифрования ответить «n» | Программа сгенерирует ключ самостоятельно | Программа сгенерировала ключ и вывела его на экран | Пройден | - |
| 2 | Ввод текста для шифрования | Программа примет текст | Программа приняла текст | Пройден | - |
| 3 | Получение результата | Программа выдаст зашифрованный текст | Программа выдала зашифрованный текст | Пройден | - |
| 4 | Ввести «y» при запроса ключа | Программа запросит ключ | Программа запросила ключ | Пройден | - |
| 5 | Вести ключ который программа сгенерировала ранее | Программа примет ключ | Программа приняла ключ | Пройден | - |
| 6 | Ввести такой-же текст, который вы использовали в пункте 2 | Программа примет текст | Программа приняла текст | Пройден | - |
| 7 | Получение результата | Программа должна выдать такой-же зашифрованный текст как и в пункте 3 | Программа выдала разные тексты | Не пройден | - |
| 8 | Повторение | Повторить шаги 1-7, для удостоверения работоспособности программы | Программа выдает разные зашифрованные тексты | Не пройден | - |

После получения отчета, была продолжена работа над главной проблемой программы – невозможность получать одинаковые зашифрованные тексты в разное время, что приводит к невозможности использования программы .

## **3.4 Четвертое тестирование**

На четвертое тестирование, которое оказалось финальным, программа была отправлена 1.05.2020, а 5.05.2020 был получен следующий отчет:

|  |  |
| --- | --- |
| Среда тестирования  /Environment | Windows 10 |
| Предварительные действия  /Pre requisites | Открытие приложения, подготовка к вводу ключа |
| Комментарии  /Comments | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Шаг  /Step | Действие  /Action | Ожидаемый результат  /Expected results | Результат | Пройден,  Не пройден, Недоступен | Комментарии  /Notes |
| 1 | При запросе ключа шифрования ответить «n» | Программа сгенерирует ключ самостоятельно | Программа сгенерировала ключ и вывела его на экран | Пройден | - |
| 2 | Ввод текста для шифрования | Программа примет текст | Программа приняла текст | Пройден | - |
| 3 | Получение результата | Программа выдаст зашифрованный текст | Программа выдала зашифрованный текст | Пройден | - |
| 4 | Ввести «y» при запроса ключа | Программа запросит ключ | Программа запросила ключ | Пройден | - |
| 5 | Вести ключ который программа сгенерировала ранее | Программа примет ключ | Программа приняла ключ | Пройден | - |
| 6 | Ввести такой-же текст, который вы использовали в пункте 2 | Программа примет текст | Программа приняла текст | Пройден | - |
| 7 | Получение результата | Программа должна выдать такой-же зашифрованный текст как и в пункте 3 | Программа выдала разные тексты | Не пройден | - |
| 8 | Повторение | Повторить шаги 1-7, для удостоверения работоспособности программы | Программа выдает разные зашифрованные тексты | Не пройден | - |

Как следует из отчета, первая версия программы, которая выполняет шифрование текста с и без известного ключа готова.

# **Глава 4 Дальнейшее развитие проекта**

## **4.1 Перспективы развития в техническом направлении:**

* Добавить дополнительные роторы для большего качества шифрования информации.
* Добавить другие языки, на которых может осуществляться шифрование.
* Реализовать приложение в виде web-сайта, для возможности использования шифратора без необходимости скачивания программы на свой ПК.
* Добавить возможность шифрования в реальном времени, т.е. при каждая буква будет сразу шифроваться без необходимости ввода полноценного предложения.
* Создать клиент-серверную систему для использования данного алгоритма в виде шифрования сообщений, для развлекательных целей, либо для передачи какой-либо важной информации.
* Усовершенствовать графический интерфейс.

При добавлении вышеперечисленных пунктов, эмулятор сможет составить полноценную конкуренцию остальным электронным приложениям в Web-среде. Возможно, больше людей заинтересуется способами шифрования и защиты информации, и смогут расширить свои знания о криптографии – науки о шифровании сообщений.

## **4.2 Перспективы развития, как надежного шифрования.**

На данный момент, механизм шифрования «Энигмы» устарел, как морально, так и физически, поскольку не может обеспечить защищенность информации. Поэтому в современном мире необходимы все более и более надежные методы шифрования информации через создание новых способов шифрования, таких как симметричное, ассиметричное, хеширование информации, подробнее о которых можно почитать в источнике[6]. При более детальном изучении данных способов шифрования, становится понятно, что алгоритм шифрования «Энигмы», на их фоне кажется просто детской игрушкой, нежели действенным способом скрыть данные от посторонних глаз.

# **Заключение**

В ходе выполнения курсовой работы были получены навыки работы с C++, визуализации программ с помощью WindowsForms и их кастомизации в произвольной форме.

Был изучен принцип шифрования реальных аналогов Энигмы, и возможность воссоздать симулятор такой машины с аналогичным принципом шифрования.

Был разработан симулятор Энигмы, который позволяет зашифровывать сообщения с достаточным уровнем защиты, и предусматривает реальный прототип, т.е. получатель не сможет быстро расшифровать сообщения, не имея специального ключа.

При создании данного симулятора я обращал внимание на недостатки реального аналога, например, возможность заклинивания механизма шифрования. На практике могла бы получиться неловкая ситуация, когда необходимо срочно зашифровать сообщение, но устройство вышло из строя. В случае программы, можно просто перезапустить приложение, что займет гораздо меньше времени, чем починка шифровальной машины. Также программа не подвержена износу, что также является большим преимуществом перед физической копией.

Однако программа не может быть идеальна, данный случай не исключение. Например, для замены роторов необходимо дописывать дополнительный код, в то время как в механизме можно потратить гораздо меньше времени, и данная проблема должна быть решена быстрее всего.

Разработанный симулятор выполняет базовые функции реального механизма, а именно – шифрование и дешифрование информации. Однако данной программе еще далеко до полноценного эмулятора, который бы выполнял все функции, которые может предложить реальный механизм.

В ближайшем будущем, планируется добавить возможность шифрования на разных языках, дополнительные роторы для надежности шифрования.

В конечном виде программа должна будет принять вид небольшого окна, в котором можно будет с легкостью зашифровать сообщение на разных языках,

# **Литература**

1.Уроки по программирования с помощью WindowsForms с YouTube канала «FPS FirePlanetStudio» [Электронный ресурс <https://www.youtube.com/watch?v=WHF8mtnJnl0>]

2.Статья о принципе шифрования Энигмы [Электронный ресурс <https://habr.com/ru/post/217331/>]

3.Курс уроков по C++ [Электронный ресурс <https://www.youtube.com/watch?v=kRcbYLK3OnQ&list=PLQOaTSbfxUtCrKs0nicOg2npJQYSPGO9r>]

4.Объяснение принципа работы шифратора Джефферсона [Электронный ресурс <http://hijos.ru/2014/01/15/dzhefferson-shifrovalshhik/>]

5.Обзор других методов шифрования информации [Электронный ресурс <https://scienceforum.ru/2016/article/2016019504>]

6. Обзор современных методов шифрования [Электронный ресурс https://proglib.io/p/methods-of-encryption/]