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

PROJECT ENIGMA

Курсовая работа
по дисциплине «Программирование»
студента 1 курса группы ПИ-б-о-11
Круглекова Дмитрия Вячеславовича

направления подготовки 09.03.04 «Программная инженерия»

Научный руководитель _____
(оценка)
старший преподаватель кафедры
компьютерной инженерии и моделирования _____ Чабанов В.В.
(подпись, дата)

Симферополь, 2020

РЕФЕРАТ

23 февраля 1918 года немецкий инженер Артур Шербиус получил патент на шифровальную машину, принцип работы которой заключался в роторах. Именно эта машина и была первой из семейства Энигм. Позже, совместно с Рихардом Риттером, Артур Шербиус создал фирму Шербиус и Риттер. В 20-х годах 20 века была создана “Энигма” (в переводе с немецкого:” Загадка”) такая, какой ее знают многие люди, увлекающиеся криптографией и не только.

Идея проекта была предложена преподавателем по дисциплине “Программирование” Чабановым Владимиром Викторовичем. Она мне очень понравилась. Была поставлена цель: разработать шифровальную машину “Энигма” классического образца (с тремя роторами, без коммутационной панели), именно эта модификация являлась коммерческой.

Нужно-было добиться удобного графического интерфейса и архитектуры клиент-сервер. “Project Enigma” простимулирует интерес пользователя к изучению криптографии и рассмотрению прочих шифровальных систем и технологий. Программа позволит общаться с помощью шифрования, что будет препятствовать похищению важной информации злоумышленниками. Есть возможность внедрения данного алгоритма шифрования в чат, который будет обеспечивать довольно-таки высокий уровень защиты переписки.

Создать симулятор немецкой шифровальной машины “Энигма”. Должен присутствовать функционал полноценной шифровальной машинки коммерческого образца. На ней не будет коммутационной панели. Будет всего 8 роторов, одновременно можно будет использовать 3, благодаря им и производится шифрование букв. У каждого ротора будет 26 положений, столько же, сколько букв в английском алфавите. Значения настроек и буква для зашифровки будут вводиться в отдельные соответствующие поля. По нажатию кнопки все значения должны сформироваться в запрос определенной формы, отправится на сервер, обработаться алгоритмом и результат должен быть отправлен и отображен в клиенте. Пользование программой не должно доставлять трудностей, все должно быть предельно просто и понятно.

По средствам языков C++ и Python, была реализована шифровальная машина, которая функционирует и работает по архитектуре клиент – сервер. Проект удался и безусловно возбудил во мне интерес к криптографии, возможно, даже кто-то другой этим заинтересовался, благодаря данной программе. Есть возможность продолжить разработку этого проекта, совершенствуя его, есть много идей по дальнейшему развитию “Project Enigma”.

Оглавление

Введение

23 февраля 1918 года немецкий инженер Артур Шербиус получил патент на шифровальную машину, принцип работы которой заключался в роторах. Именно эта машина и была первой из семейства Энигм. Позже, совместно с Рихардом Риттером, Артур Шербиус создал фирму Шербиус и Риттер. В 20-х годах 20 века была создана “Энигма” (в переводе с немецкого:” Загадка”) такая, какой ее знают многие люди, увлекающиеся криптографией и не только.

Название “Энигма” скрывало в себе целое семейство шифровальных машин, которые различались по ряду признаков и появлялись постепенно по мере надобности. Самые первые машины “Энигма” были коммерческими, так как в то время, не армия, не флот, небыли заинтересованы в шифровальных машинах. Любая компания могла приобрести для себя Энигму и вести тайную переписку с партнерами, или филиалами, передавая конфиденциальную информацию, не боясь лишних глаз. Но позже удалось заключить договоры с вооруженными силами Германии. Не сложно догадаться, что самыми первыми структурами, которые были снабжены Энигмами, стали спец. службы Германии. Позже они в различных модификациях заполонили практически все вооруженные силы страны. Позже были разработаны аксессуары, которые, либо упрощали использование Энигмы, либо повышали секретность, либо увеличивали количество возможных комбинаций шифрования.

Идея проекта была предложена преподавателем по дисциплине “Программирование” Чабановым Владимиром Викторовичем. Она мне очень понравилась. Была поставлена цель: разработать шифровальную машину “Энигма” классического образца (с тремя роторами, без коммутационной панели), именно эта модификация являлась коммерческой.

Для достижения цели были созданы задачи:

1. Изучить историю создания и использования шифровальной машины “Энигма”.
2. Найти и разобраться в алгоритме работы и в принципе шифрования.
3. Изучить необходимые технологии для разработки, учитывая все требования.

ГЛАВА 1
ПОСТАНОВКА ЗАДАЧИ

1.1 Цель проекта

Цель проекта была поставлена очень быстро, не было сомнений, что надо создать симулятор шифровальной машины “Энигма”, для системы Windows, с основным языком программирования C++. Нужно-было добиться удобного графического интерфейса и архитектуры клиент-сервер. “Project Enigma” простимулирует интерес пользователя к изучению криптографии и рассмотрению прочих шифровальных систем и технологий. Программа позволит общаться с помощью шифрования, что будет препятствовать похищению важной информации злоумышленниками. Есть возможность внедрения данного алгоритма шифрования в чат, который будет обеспечивать довольно-таки высокий уровень защиты переписки.

1.2 Существующие аналоги

1.2.1 Enigmasim

Рис. 1 Рабочее пространство программы Enigmasim

1.2.2 Enigma Simulator by Terry Long

Рис. 2 Рабочее пространство программы Enigma Simulator by Terry Long

1.2.3 Enigma Machine by Matthias Schorer

Рис. 3 Рабочее пространство программы Enigma Machine by Matthias Schorer

1.3 Основные отличия от аналогов

Перед началом разработки проекта, мной были просмотрены други уже готовые программы. Мной сразу же была замечена проблема с удобством пользования. Было сложно сразу же интуитивно понять, как пользоваться программой в полной мере, эта проблема связана с изменением настроек шифрования. Эта проблема появилась из-за того, что разработчики сделали главный упор на графическую часть, забыв об удобстве пользования. Благодаря этому, я поставил еще одну задачу для своего проекта: сделать упор на удобство пользования, нежели на графическую составляющую. В моей программе легко понять, как совершать настройку и шифровать, не прибегая к какой-либо документации.

1.4 Техническое задание

Создать симулятор немецкой шифровальной машины “Энигма”. Должен присутствовать функционал полноценной шифровальной машинки коммерческого образца. На ней не будет коммутационной панели. Будет всего 8 роторов, одновременно можно будет использовать 3, благодаря им и производится шифрование букв. У каждого ротора будет 26 положений, столько же, сколько букв в английском алфавите. Значения настроек и буква для зашифровки будут вводиться в отдельные соответствующие поля. По нажатию кнопки все значения должны сформироваться в запрос определенной формы, отправится на сервер, обработаться алгоритмом и результат должен быть отправлен и отображен в клиенте. Пользование программой не должно доставлять трудностей, все должно быть предельно просто и понятно.

Программа должна работать по определенным правилам:

1. При включении программы, должны выставляться следующие настройки:
   - Номера барабанов, используемые по умолчанию: 1, 2, 3.
   - Положение роторов по умолчанию: 0, 0, 0.
   - Изначально используемый рефлектор – B.
2. При введении настроек в отведенные окна и нажатии на кнопку “Готово”, эти значения должны приниматься программой и отображаться. При введении некорректных данных, либо при заполнении не всех полей в настройках, программа должна выводить на экран окно с ошибкой, а в настройках должны оставаться начальные параметры. При нажатии на любую кнопку с буквой в окне программы, выше в окне она должна выводиться, а ниже зашифрованная буква, после обработки алгоритмом. Также при нажатии на кнопку алфавита, значение правого крайнего ротора должно увеличиться на единицу. При достижении крайним правым ротором значения 25, следующая его итерация будет 0, а значение среднего ротора увеличится на единицу. При достижении крайнего правого и среднего роторов значений 25, при нажатии, они обнулятся, а крайний левый ротор увеличит значение на единицу. По достижению всеми тремя роторами значения 25, при следующей итерации, они все станут ровны нулю.

ГЛАВА 2
ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

2.1 Анализ инструментальных средств

Изначально разработка производилась только по средствам кода на C++ в QT Creator. Все версии, кроме последней были написаны только на C++, а интерфейс создавался на специальном конструкторе в QT Creator. Когда работоспособность проекта была доведена до приемлемого уровня, началась разработка требуемой архитектуры клиент – сервер. Алгоритм остался на C++, но при создании клиента и сервера были использованы новые для меня технологии. Для создания клиента был язык разметки html, с ним был опыт в прошлом семестре, поэтому вспомнить было не сложно. Новым оказался язык python, пришлось адаптироваться, но все же был создан сервер при помощи flask. Удалось все это связать между собой и проект заработал.

2.2 Описание алгоритмов

Существует алгоритм, по которому и производится шифрование буквы.

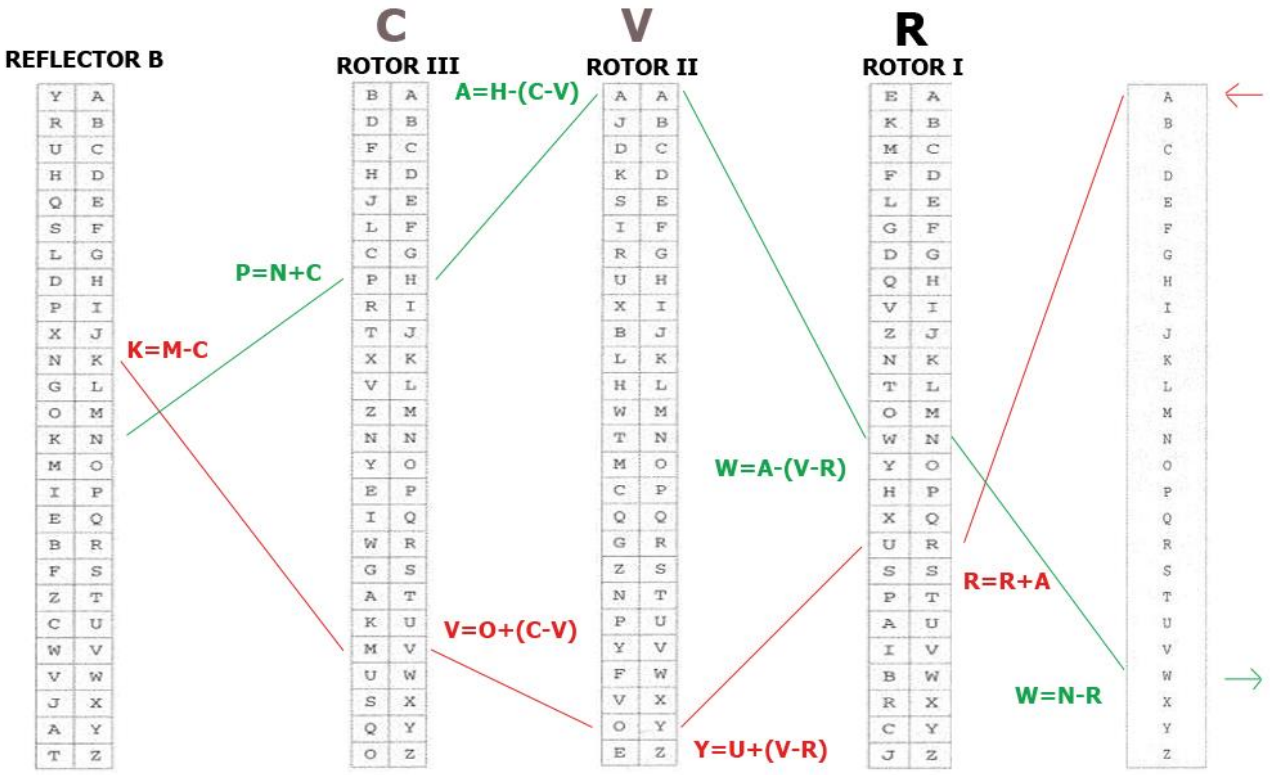

Рис. 4 Алгоритм шифрования и расшифровки букв.

Можно заметить, что шифрование — это взаимодействие значений по особой математической формуле. На фото видно, что с правой стороны находится ввод и красной стрелкой указано, как производится шифрование. При шифровании учитывается порядковый номер введенной буквы (начиная с нуля), а также учитывается текущее положение роторов. Над роторами указаны буквы – это и есть его положение, в формуле он учитывается по тому же принципу, как и введенная буква, она переводится в свой порядковый номер (начиная с нуля). Благодаря формулам, показанным на фото, вычисляется номер буквы, которая будет взята на роторе, и она переведется в парную букву, которая изначально предусмотрена в конструкции ротора.

INPUT	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Rotor I	E	K	M	F	L	G	D	Q	V	Z	N	T	O	W	Y	H	X	U	S	P	A	I	B	R	C	J
Rotor II	A	J	D	K	S	I	R	U	X	B	L	H	W	T	M	C	Q	G	Z	N	P	Y	F	V	O	E
Rotor III	B	D	F	H	J	L	C	P	R	T	X	V	Z	N	Y	E	I	W	G	A	K	M	U	S	Q	O
Rotor IV	E	S	O	V	P	Z	J	A	Y	Q	U	I	R	H	X	L	N	F	T	G	K	D	C	M	W	B
Rotor V	V	Z	B	R	G	I	T	Y	U	P	S	D	N	H	L	X	A	W	M	J	Q	O	F	E	C	K
Rotor VI	J	P	G	V	O	U	M	F	Y	Q	B	E	N	H	Z	R	D	K	A	S	X	L	I	C	T	W
Rotor VII	N	Z	J	H	G	R	C	X	M	Y	S	W	B	O	U	F	A	I	V	L	P	E	K	Q	D	T
Rotor VIII	F	K	Q	H	T	L	X	O	C	B	J	S	P	D	Z	R	A	M	E	W	N	I	U	Y	G	V
Beta rotor	L	E	Y	J	V	C	N	I	X	W	P	B	Q	M	D	R	T	A	K	Z	G	F	U	H	O	S
Gamma rotor	F	S	O	K	A	N	U	E	R	H	M	B	T	I	Y	C	W	L	Q	P	Z	X	V	G	J	D

Рис. 5 Комбинации парных букв для каждого из роторов

На данном изображении можно увидеть комбинации всех роторов, которые использовались в машинах “Энигма”. Опираясь на эти данные, мной и был создан проект: реализован основной алгоритм и каждый из роторов (1 – 8). Алгоритм можно увидеть ниже, во вкладке “Код основных модулей проекта”.

2.3 Описание основных модулей

Если рассматривать финальный проект в целом, то существует три модуля: клиент, сервер и алгоритм шифрования. Их код можно увидеть ниже, во вкладке “Код основных модулей проекта”.

Клиент сделан очень просто, что обеспечивает быстрое понимание принципов взаимодействия с программой. Клиент создан на html, он позволяет вводить значения и формировать запрос особого формата для успешной передачи его серверу.

Сервер также не представляет из себя нечто сложное и не понятное, так как при его создании, я только знакомился с языком python. Сервер постоянно запущен и ожидает поступления запроса от клиента. Когда поступает запрос от клиента, сервер из запроса формирует переменные с переданными значениями и запускает алгоритм, передавая значения, полученные из запроса. После того, как алгоритм отработал, он возвращает ответ, который и предается в клиент.

Принцип работы алгоритма был приведен мной выше в разделе “Описание алгоритмов”.

ГЛАВА 3
ТЕСТИРОВАНИЕ ПРОГРАММЫ

3.1 Тестирование исходного кода

Тестирование кода производилось независимым тестеровщиком. Он тестировал код промежуточной версии программы, в которой еще не было архитектуры клиент – сервер, а графический интерфейс был реализован по средствам QT Creator. Но алгоритм работы полностью перекочевал в финальную версию проекта, представленную на данный момент, лишь были добавлены необходимые условия.

Тестеровщик подготовил отчет о тестировании:

Обнаруженные недостатки:

- В приложении очень много почти одинакового кода, различия лишь в определённых местах. Например, функция «rotor1back» (строки 365-444) и функция «rotor2back» (строки 445-524). Как можно заметить единственным их отличием является число в условиях. Ладно бы было всего 2 функции, но их 8, следовательно, отличных решением было бы объединить эти функции в одну, где вместо цифр в условии поставить переменные, значения которых мы бы передавали при вызове функции, что очень сильно уменьшило бы код и ускорило бы его изменение. Теперь посмотрим на функцию «rotor1go» (строки 1005-1084) и функцию «rotor2go» (строки 1085-1164) в них похожая ситуация, только теперь не в условии, а в присвоении к переменной числа, следовательно, можно сделать то же, что мы делали для предыдущих 2х функций, только в переменные взять число, которое присваиваем, кстати таких функций тоже 8. Теперь рассмотрим «rotor1back» и «rotor1up», они практически одинаковы, просто в одной функции различны цифры в условиях, а в другой в присваивании к переменной, следовательно, эти две функции можно объединить, просто цифры в условиях и в присваиваниях заменить на переменные, которые мы бы передавали при вызове функции, а также одну bool переменную, которая отвечала бы за то, вверх мы идем или вниз, и в зависимости от ситуации присваивала бы одним переменным значение от 0, до 25,а другим, те которые мы вводили при вызове. Таким образом мы в 1 функцию запишем 16 функций. Еще можно заменить все 25 if на 1, путём добавления цикла, который будет идти от 0, до 25 включительно и в зависимости от того, идем мы вверх или вниз мы бы условие или присваивание заменили бы на итерацию цикла, которая идет

в данный момент, а числа, которые мы передаем, мы могли бы записать в конструкторе в массивы и передавать их в функции при вызове. Это позволит в цикле не использовать 25 переменных, а использовать массив и в зависимости какая итерация брался бы определенный элемент массива. Кстати, это не все функции, которые можно объединить похожим образом, вот

список (всё что в одном пункте можно объединить в единую функцию), можно собрать):

- «delimiter_x_rotor», «delimiter_y_rotor», «delimiter_z_rotor», «delimiter_x_rotor_back», «delimiter_y_rotor_back», «delimiter_z_rotor_back»
- «delimiter_x_number», «delimiter_y_number», «delimiter_z_number»

- «enigma.cpp» подключение ненужных библиотек на строке 4 (#include <iostream>) и строке 5 (#include "string")
- «enigma.cpp» использование ненужной системы имен на строке 8 (using namespace std;)
- «enigma.cpp» объявление прототипов функций (строки 10-38) в .cpp файле. Перенести в .h файл.
- «enigma.cpp» объявление глобальных переменных (строки 40-42) в .cpp файле. Перенести в .h файл
- «enigma.cpp» присваивание глобальным переменным (в строке 41) не в конструкторе. Перенести в конструктор.
- В коде нет ни единого комментария (кода в программе около 2570 строк)

Вывод: В целом код написан хорошо. Все условия выполняться правильно и предусмотрены все случаи развития событий. У каждой функции и переменной свое имя, которое имеет смысл. Нету лишних функций и переменных, которые бы не задействовались в коде. В UI тоже всё хорошо, в целом понятный и интуитивный интерфейс, а также есть справка, выход и проверка значений при вводе.

По окончанию работы над проектом, большинство проблем, либо потеряли актуальности, либо были исправлены. В основном все замечания связаны с объёмами кода, это произошло из-за того, что алгоритм осваивался на ходу и точно так же на ходу формировался код.

3.2 Тестирование интерфейса пользователя и юзабилити

Промежуточная версия также тестировалась, основываясь на составленных тест кейсах. Тест кейсы были рассчитаны на тестирование интерфейса программы и качества взаимодействия пользователя с программой. Хоть и тестировалась промежуточная версия, концепция интерфейса перекочевала в финальную версию, включая проверку значений.

Тестеровщик предоставил отчет о проведенном тестировании:

Содержание тестов(сценариев) /Test Cases (Scenarios Summary)

Идентификатор **Test ID**	**Цельтеста/Purpose of test**
Тест 1/**Error! Reference source not found.**	Запуск и закрытие программы.
Тест 2/**Error! Reference source not found.**	Работа алгоритма шифрования и расшифровки.
Тест 3/**Error! Reference source not found.**	Работа алгоритма проверки введенных параметров.
Тест 4/Test Name 4	Работоспособность кнопок ввода.
Тест 5/Test Name 5	Работоспособность алгоритма перещелкивания роторов.

Шаг /Step No.	Действие (операция) /Process (Actions)	Ожидаемый результат /Expected Results	Результат /Actual Results	Пройден /не пройден/ не доступен*	Комментарии/Notes
1	Отладка и запуск программы	Программа запускается, выставляются изначальные настройки роторов, написанные в техническом задании. Корректно отображаются все кнопки и окна на экране.	Программа запускается в левом верхнем углу, правее надписи «Номера роторов» отображаются цифры «1 2 3», правее надписи «Положение роторов» отображаются значения «0 0 0», справа от надписи «Рефлекторы» отображается «B».	Пройден	Надпись «Положение роторов» перекрывает первое значение положения ротора.
2	Нажатые на кнопку “Выход”	Окно программы закрывается.	После нажатия кнопки «Выход» окно программы закрывается	Пройден	-
3	Нажатые на кнопку “Понял” в окне с информацией.	Окно программы закрывается.	В основном окне после нажатия на кнопку со знаком «?» открывается окно с поясняющей информацией к программе. Это окно закрывается после нажатия на кнопку «Понял».	Пройден	-

Шаг /Step No.	Действие (операция) /Process (Actions)	Ожидаемый результат /Expected Results	Результат /Actual Results	Пройден /не пройден/ не доступен*	Комментарии/Notes
1	Набор слов на клавиатуре программы при различных настройках роторов и рефлекторов.	При вводе определенной буквы, она должна зашифроваться. При вводе зашифрованной буквы с теми же настройками роторов и рефлекторов, при которых производилось шифрования, зашифрованная буква должна превратиться в искомую.	При нажатии на виртуальной клавиатуре какой-либо кнопки с изображением буквы, выше кнопки с изображение латинской буквы “U” появляется зашифрованная буква, а выше на одной лини нажатая буква.	Пройден	Возможно, для удобства пользователя, стоило обозначить, где зашифрованная буква, а где искомая.

Шаг /Step No.	Действие (операция) /Process (Actions)	Ожидаемый результат /Expected Results	Результат /Actual Results	Пройден /не пройден/ не доступен*	Комментарии/ Notes
1	Не заполнить хотя бы одно окно ввода в настройках.	Вывод на экран ошибки и выставление настроек по умолчанию.	Выводиться окно с ошибкой, предупреждающее о некорректном вводе. Настройки выставляются по умолчанию.	Пройден	-
2	Ввести номера роторов, чтобы хотя бы два из них совпали.	Вывод на экран ошибки и выставление настроек по умолчанию.	Выводиться окно с ошибкой, предупреждающее о некорректном вводе. Настройки выставляются по умолчанию.	Пройден	-
3	Ввести номера роторов меньше единицы, или больше восьми	Вывод на экран ошибки и выставление настроек по умолчанию.	Выводиться окно с ошибкой, предупреждающее о некорректном вводе. Настройки выставляются по умолчанию.	Пройден	-

Шаг /Step No.	Действие (операция) /Process (Actions)	Ожидаемый результат /Expected Results	Результат /Actual Results	Пройден /не пройден/ не доступен*	Комментарии/ Notes
4	Ввести номера роторов меньше нуля, либо больше двадцати пяти.	Вывод на экран ошибки и выставление настроек по умолчанию.	Выводиться окно с ошибкой, предупреждающее о некорректном вводе. Настройки выставляются по умолчанию.	Пройден	В этой строке в первом столбце «Действие(операция)» возможно, описка вместо «номера роторов» следовало проверить «положение роторов», этот тест также был проведён. Тест «Пройден». Результат: Выводиться окно с ошибкой, предупреждающее о некорректном вводе. Настройки выставляются по умолчанию.
5	Ввести букву или цифру рефлектора кроме B, илиC.	Вывод на экран ошибки и выставление настроек по умолчанию.	Выводиться окно с ошибкой, предупреждающее о некорректном вводе. Настройки выставляются по умолчанию.	Пройден	-

Шаг /Step No.	Действие (операция) /Process (Actions)	Ожидаемый результат /Expected Results	Результат /Actual Results	Пройден /не пройден/ не доступен*	Комментарии/Notes
1	Нажатие на всевозможные кнопки в окне проекта с целью выяснения их работоспособности.	Все кнопки функционируют и выполняют свою роль.	Все кнопки главного окна приложения функционируют и выполняют свою роль.	Пройден	Не понятно, какую функцию выполняет кнопка со знаком «?» в окне информации к приложению.

Шаг /Step No.	Действие (операция) /Process (Actions)	Ожидаемый результат /Expected Results	Результат /Actual Results	Пройден /не пройден/ не доступен*	Комментарии/Notes
1	Выставить положение роторов: 1, 1, 25 Номера роторов и рефлектора любые в пределах ограничений. Нажать на любую букву на клавиатуре в окне программы.	Крайний правый ротор обнуляется, средний ротор становится двойкой.	Значение крайнего правого ротор обнуляется, средний ротор становится двойкой.	Пройден	-
2	Выставить положение роторов: 1, 25, 25 Номера роторов и рефлектора любые в пределах ограничений. Нажать на любую букву на клавиатуре в окне программы.	Крайний правый ротор обнуляется, средний ротор обнулится, крайний левый ротор становится двойкой.	Значение среднего и крайнего правого ротора становятся равными нулю, значение крайнего левого ротора становиться равным двум.	Пройден	-
3	Выставить положение роторов: 25, 25, 25 Номера роторов и рефлектора любые в пределах ограничений. Нажать на любую букву на клавиатуре в окне программы.	Крайний правый ротор обнуляется, средний ротор обнулится, крайний левый ротор обнулится.	Значения всех роторов становиться равным нулю.	Пройден	-

ГЛАВА 4
ПЕРСПЕКТИВЫ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОЕКТА

Перспективы технического развития

Алгоритм шифровальной машины “Энигма” можно встроить в мессенджер, который будет шифровать сообщения до того, как отправить получателю. Можно реализовать импровизированную сеть, в которой будет находиться несколько пользователей, и система будет автоматически шифровать и расшифровывать сообщения, так как при создании сети, клиенты бы обменивались конфигурацией настроек шифрования. При этом, даже при взломе чата, злоумышленник не сможет понять сообщений, как как у него не будет параметров шифрования. Но есть некоторые минусы, которые не позволяют использовать данный алгоритм в качестве единой системы шифрования. Шифровальная машина “Энигма” не умеет зашифровывать цифры, не получится зашифровать мультимедиа файлы. Но я считаю, что можно совместить несколько систем шифрования и использовать Энигму только для текста, а файлы шифровать другими системами. Это может усложнить злоумышленнику просмотр информации. Даже, если удастся взломать одну из систем, то полную картину диалога все-равно увидеть не удастся. Возможно реагировать другие модификации шифровальной машины, реализовать некоторые аксессуары. Можно адаптировать Энигму под другие языки, но для этого придется проделать не малую работу и это уже не будет Энигмой, это будет новая шифровальная система, использующая алгоритм Энигмы.

ЗАКЛЮЧЕНИЕ

В результате, я считаю, что проект удался. Были реализованы все требования преподавателя и была выполнена главная цель проекта, но есть, к чему стремиться, есть перспективы к развитию, возможно они туманны, но выполнена задача: заинтересовать других студентов, привить интерес криптографии. Возможно, после этого кому-то захочется изучить иные системы шифрования. “Энигма” стала для меня отправной точкой в криптографии, я точно этим заинтересовался. После работы над этим проектом, у меня появилось желание изучить более совершенные системы.

Источники

1. https://habr.com/Алгоритм Энигмы/ 2014г.
2. https://www.codesandciphers.org.uk/Technical Specifications of the Enigma
3. https://habr.com/Криптоанализ “Энигмы”
4. https://codex.so/Простой веб-сервер с использованием Python и Flask.

ПРИЛОЖЕНИЕ

КОД ОСНОВНЫХ МОДУЛЕЙ ПРОЕКТА

1. Основной алгоритм шифрования и расшифровки (C++)

```
alphago();
a = z_rotor + a;
if (a < 0)
        a = a + 26;
if (a > 25)
        a = a - 26;
if (z_number == 1)
        rotor1go();
else
        if (z_number == 2)
                rotor2go();
        else
                if (z_number == 3)
                        rotor3go();
                else
                        if (z_number == 4)
                                rotor4go();
                        else
                                if (z_number == 5)
                                        rotor5go();
                                else
                                        if (z_number == 6)
                                                rotor6go();
                                        else
                                                if (z_number == 7)
                                                        rotor7go();
                                                else
                                                        if (z_number == 8)
                                                                rotor8go();
a = a + (y_rotor - z_rotor);
if (a < 0)
        a = a + 26;
if (a > 25)
        a = a - 26;
if (y_number == 1)
        rotor1go();
else
        if (y_number == 2)
                rotor2go();
        else
                if (y_number == 3)
                        rotor3go();
                else
                        if (y_number == 4)
                                rotor4go();
                        else
                                if (y_number == 5)
                                        rotor5go();
                                else
                                        if (y_number == 6)
                                                rotor6go();
                                        else
                                                if (y_number == 7)
                                                        rotor7go();
                                                else
                                                        if (y_number == 8)
                                                                rotor8go();
```

```
a = a + (x_rotor - y_rotor);

        if (a < 0)
                a = a + 26;
        if (a > 25)
                a = a - 26;
        if (x_number == 1)
                rotor1go();
        else
                if (x_number == 2)
                        rotor2go();
                else
                        if (x_number == 3)
                                rotor3go();
                        else
                                if (x_number == 4)
                                        rotor4go();
                                else
                                        if (x_number == 5)
                                                rotor5go();
                                        else
                                                if (x_number == 6)
                                                        rotor6go();
                                                else
                                                        if (x_number == 7)
                                                                rotor7go();
                                                        else
                                                                if (x_number == 8)
                                                                        rotor8go();
        a = a - x_rotor;
        if (a < 0)
                a = a + 26;
        if (a > 25)
                a = a - 26;
        if (reflector == "B")
                reflectorB();
        else
                reflectorC();
        a = a + x_rotor;
        if (a > 25)
                a = a - 26;
        if (a < 0)
                a = a + 26;
        if (x_number == 1)
                rotor1back();
        else
                if (x_number == 2)
                        rotor2back();
                else
                        if (x_number == 3)
                                rotor3back();
                        else
                                if (x_number == 4)
                                        rotor4back();
                                else
                                        if (x_number == 5)
                                                rotor5back();
                                        else
                                                if (x_number == 6)
                                                        rotor6back();
                                                else
                                                        if (x_number == 7)
                                                                rotor7back();
                                                        else
                                                                if (x_number == 8)
```

```
                                                                rotor8back();
a = a - (x_rotor - y_rotor);

if (a > 25)
        a = a - 26;
if (a < 0)
        a = a + 26;
if (y_number == 1)
        rotor1back();
else
        if (y_number == 2)
                rotor2back();
        else
                if (y_number == 3)
                        rotor3back();
                else
                        if (y_number == 4)
                                rotor4back();
                        else
                                if (y_number == 5)
                                        rotor5back();
                                else
                                        if (y_number == 6)
                                                rotor6back();
                                        else
                                                if (y_number == 7)
                                                        rotor7back();
                                                else
                                                        if (y_number == 8)
                                                                rotor8back();
a = a - (y_rotor - z_rotor);
if (a > 25)
        a = a - 26;
if (a < 0)
        a = a + 26;
if (z_number == 1)
        rotor1back();
else
        if (z_number == 2)
                rotor2back();
        else
                if (z_number == 3)
                        rotor3back();
                else
                        if (z_number == 4)
                                rotor4back();
                        else
                                if (z_number == 5)
                                        rotor5back();
                                else
                                        if (z_number == 6)
                                                rotor6back();
                                        else
                                                if (z_number == 7)
                                                        rotor7back();
                                                else
                                                        if (z_number == 8)
                                                                rotor8back();




a = a - z_rotor;
if (a > 25)
```

```
        a = a - 26;
if (a < 0)
        a = a + 26;
alphaback();
```

2. Код сервера (Python)

```
# -*- coding: utf-8 -*-
from flask import Flask, request, abort
from flask_cors import CORS
import sys, os, getopt
import subprocess

app = Flask(__name__)
CORS(app)

@app.route('/')
def hello_world():
    if request.method == 'GET':
        return run_algoritm(request), 200
    else:
        abort(404)

@app.errorhandler(404)
def page_not_found(error):
    return 'Page not found!', 404


def run_algoritm(request):
    x_number = request.args.get('x_number', default='')
    y_number = request.args.get('y_number', default='')
    z_number = request.args.get('z_number', default='')
    x_rotor = request.args.get('x_rotor', default='')
    y_rotor = request.args.get('y_rotor', default='')
    z_rotor = request.args.get('z_rotor', default='')
    reflector = request.args.get('reflector', default='')
    b_string = request.args.get('b_string', default='')

    dir_path = os.path.dirname(os.path.realpath(__file__))
    script = os.path.join(dir_path, "Project.exe")
    proc = subprocess.Popen([script, x_number, y_number, z_number, x_rotor, y_rotor, z_rotor,
reflector, b_string], stdout=subprocess.PIPE, shell=True)
    out, errs = proc.communicate()

    return out
```

3. Код клиента (HTML)

```
<html>
<head>
   <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
   <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">
</head>
<body>
<div id="vueApp">
   <form @submit.prevent="submit">
      <div class="form-group row">
         <div class="col">
            <input type="text" class="form-control" placeholder="Номер первого ротора" v-
model="x_number" required>
         </div>
         <div class="col">
            <input type="text" class="form-control" placeholder="Номер второго ротора" v-
model="y_number" required>
         </div>
         <div class="col">
            <input type="text" class="form-control" placeholder="Номер третьего ротора" v-
model="z_number" required>
         </div>
      </div>
      <div class="form-group row">
         <div class="col">
            <input type="text" class="form-control" placeholder="Положение первого ротора"
v-model="x_rotor" required>
         </div>
         <div class="col">
            <input type="text" class="form-control" placeholder="Положение второго ротора" v-
model="y_rotor" required>
         </div>
         <div class="col">
            <input type="text" class="form-control" placeholder="Положение третьего ротора"
v-model="z_rotor" required>
         </div>
      </div>
      <div class="form-group row">
         <div class="col">
            <input type="text" class="form-control" placeholder="Рефлектор" v-
model="reflector" required>
         </div>
         <div class="col">
            <input type="text" class="form-control" placeholder="Буква" v-model="b_string"
required>
         </div>
      </div>
      <div class="form-group row">
         <div class="col-auto">
            <button type="submit" class="btn btn-primary mb-2">Готово</button>
         </div>
      </div>
   </form>
```

```
   <div>
      <p>Результат: <strong>{{result}}</strong></p>
   </div>
</div>

<script>
   var app = new Vue({
      el: '#vueApp',
      data: {
         result: null,
         x_number: null,
         y_number: null,
         z_number: null,
         x_rotor: null,
         y_rotor: null,
         z_rotor: null,
         reflector: null,
         b_string: null
      },
           methods: {
         async submit() {
            var query = 'x_number=' + this.x_number + '&y_number=' + this.y_number +
'&z_number=' + this.z_number
               + '&x_rotor=' + this.x_rotor + '&y_rotor=' + this.y_rotor + '&z_rotor=' +
this.z_rotor
               + '&reflector=' + this.reflector + '&b_string=' + this.b_string;

            try {
               var response = await axios.get('http://127.0.0.1:5000?' + query);
                                       if (this.z_rotor > 24)
                                        {
                                               this.z_rotor = 0;
                                               this.y_rotor++;
                                        }
                                        if (this.y_rotor > 25)
                                        {
                                               this.z_rotor = 0;
                                               this.y_rotor = 0;
                                               this.x_rotor++;
                                        }
                                        if (this.x_rotor > 25)
                                        {
                                               this.z_rotor = 0;
                                               this.y_rotor = 0;
                                               this.x_rotor = 0;
                                        }
                                        this.z_rotor++;
               this.result = response.data;
            } catch (error) {
               console.error(error);
            }

         }
      }
   })
</script>
</body>
</html>
```