МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

*Изображение выглядит как текст, Шрифт, снимок экрана, логотип

Автоматически созданное описание*

**РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА**

**по дисциплине: «Программирование»**

Выполнили:Проверил:

Студенты гр. АБ(с)-222, АВТФ *доцент кафедры ЗИ*

*Гатауллин Д. Р., Линкер Э. С. Архипова А. Б.*

«10» \_\_ 06\_\_\_ 2023г«\_\_\_» \_\_\_\_\_\_ 2023 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2023

СОДЕРЖАНИЕ

[Введение 3](#_Toc137111451)

[1. Теоретическая часть 5](#_Toc137111452)

[1.1 Алгоритм работы шифра Гроснфельда 5](#_Toc137111453)

[1.2 Алгоритм работы шифра с помощью квадрата Полибия 6](#_Toc137111454)

[1.3 Алгоритм работы шифра Атбаша 9](#_Toc137111455)

[1.4 Алгоритм работы шифра Виженера 10](#_Toc137111456)

[1.5 Алгоритм работы шифрования RSA 12](#_Toc137111457)

[1.6 Алгоритм работы шифра «перестановка скитала» 14](#_Toc137111458)

[2. Практическая часть 17](#_Toc137111459)

[2.1 Постановка задачи 17](#_Toc137111460)

[2.2 Характеристика задачи 17](#_Toc137111461)

[2.3 Алгоритм решения 18](#_Toc137111462)

[2.4 Руководство пользователя 29](#_Toc137111463)

[2.5 Руководство системного программиста 35](#_Toc137111464)

[2.6 Контрольный пример 39](#_Toc137111465)

[Список использованных источников 49](#_Toc137111466)

[Приложение 51](#_Toc137111467)

# ВВЕДЕНИЕ

Криптография — это тема, которая имеет большую актуальность в настоящее время. Каждый день мы сталкиваемся с необходимостью передавать конфиденциальную информацию, которую нужно защищать от постороннего доступа. В такой ситуации крайне важен правильный выбор алгоритма шифрования, а также уверенность в его надежности.

Шифрование существует уже несколько тысячелетий и началось с простых замен букв на цифры, но с приходом компьютеров и развитием криптографии ситуация стала более сложной. Сегодня у нас есть много различных методов шифрования и расшифрования, которые используются не только в военных целях, но и в коммерческих и технологических отраслях.

Цель данной работы заключается в создании программного продукта, способного реализовать различные методы шифрования и расшифрования данных.

Задачи, которые необходимо решить в процессе выполнения работы, включают в себя: исследование принципов работы различных алгоритмов шифрования и их эффективности, анализ их недостатков и преимуществ; классифицировать проанализированные алгоритмы шифрования и выбрать наиболее подходящие для реализации; реализовать многофайловый проект; протестировать программу на наличии уязвимостей и в случае наличия их – исправить; провести рефакторинг готового кода; ознакомиться с основными командами «GitHub», а затем опубликовать там свои исходные файлы; подготовить exe-файл для сдачи готового программного продукта.

Ожидаемыми результатами данной работы являются полностью рабочий и эффективный программный продукт, способный шифровать и расшифровывать данные при помощи выбранных алгоритмов шифрования. Программа должна быть доступна для использования на компьютерах с операционной системой Windows и иметь интуитивно понятный интерфейс для удобства работы пользователя.

Эффективность работы будет выражаться в скорости работы программы, а также в ее устойчивости и безопасности. Будет проведено тестирование на наличие уязвимостей и исправление найденных ошибок и недостатков, что обеспечит высокую степень защиты данных пользователя.

Новизна данной работы заключается в том, что создается многофайловый проект с использованием различных алгоритмов шифрования, что позволит пользователю выбирать наиболее подходящий способ защиты данных в зависимости от их важности.

Теоретической основой написания расчетно-графической работы явились электронные ресурсы и учебные пособия следующих авторов Гребенников В.В. [1], Ашаева А. [4], Владимиров С.М., Габидулин Э.М., Колыбельников А.И., Кшевецкий А.С. [6], Бещева Т. [9], Бондаренко М., Бондаренко С. [11], Васильев В.С. [12], а также электронные статьи [2-3, 5, 7-8, 10, 13].

Практической основой написания расчетно-графической работы явились учебные материалы по языку программирования C++14 [14-15], среде разработки Visual Studio 2022 [16].

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В программе используется 5 методов шифрования: Гроснфельда, квадрата Полибия, Атбаша, Виженера, RSA и перестановка «скитала». Шифр Гронсфельда, Виженера, RSA и перестановки «скитала» реализуют с помощью ключей шифрования.

## 1.1 Алгоритм работы шифра Гроснфельда [1-3]

В 1734-м начальником 1-й немецкой криптослужбы графом Гронсфельдом была предложена идея шифрования без применения таблицы, что было названо шифром Гронсфельда и было упрощением шифра Виженера. Словесный ключ-пароль заменялся цифровым, который хорошо запоминался, а большая громоздкая квадратная таблица заменялась одним алфавитом. Буквы исходного сообщения заменялись алфавитными буквами, стоящими от них правее или левее на такое число букв, которое составляло цифру пароля [1].

Шифр Гронсфельда — полиалфавитный подстановочный шифр, представляет собой модификацию шифра Цезаря числовым ключом. Для этого под буквами исходного сообщения записывают цифры числового ключа. Если ключ короче сообщения, то его запись циклически повторяют. Шифртекст получают примерно, как в шифре Цезаря, но отсчитывают по алфавиту не третью букву (как это делается в шифре Цезаря), а выбирают ту букву, которая смещена по алфавиту на соответствующую цифру ключа. Шифр Гронсфельда «размывает» характеристики частот появления символов в тексте, но некоторые особенности появления символов в тексте остаются. Главный недостаток шифра Гронсфельда состоит в том, что его ключ повторяется. Поэтому простой криптоанализ шифра может быть построен в два этапа:

1. Поиск длины ключа. Можно анализировать распределение частот в зашифрованном тексте с различным прореживанием. То есть брать текст, включающий каждую 2-ю букву зашифрованного текста, потом каждую 3-ю и т. д. Как только распределение частот букв будет сильно отличаться от равномерного (например, по энтропии), то можно говорить о найденной длине ключа.
2. Криптоанализ. Совокупность L шифров Цезаря (где L — найденная длина ключа), которые по отдельности легко взламываются [2].

Описание алгоритма шифрования и расшифрования:

1. Ключ

Длина ключа (K) должна быть равной длине исходного текста. Для этого циклически записывают ключ до тех пор, пока его длина не будет соответствовать длине исходного текста.

1. Шифрование

Каждый символ открытого текста нужно на (соответствующий символ ключа K) шагов сдвинуть вправо.

1. Дешифрирование

Каждый символ () зашифрованого текста нужно на (соответствующий символ ключа K) шагов сдвинуть влево [3].

## 1.2 Алгоритм работы шифра с помощью квадрата Полибия [4-5]

Древние греки придумали если не все, то очень многое, в том числе и слово «криптография» (с древнегреческого «скрытый» + «пишу»). Среди множества способов передать сообщение тайно особо выделяется способ, названный именем выдающегося древнегреческого историка II в. до н. э. Полибия, который описал его в своем труде «Всеобщая история» [4].

В криптографии квадрат Полибия, также известный как шахматная доска Полибия — оригинальный код простой замены, одна из древнейших систем кодирования, предложенная Полибием (греческий историк, полководец, государственный деятель, III век до н. э.). Данный вид кодирования изначально применялся для греческого алфавита, но затем был распространен на другие языки.

Очень часто данный вид шифрования применяется при проведении городских квестов.

Латинский алфавит

В современном латинском алфавите 26 букв, следовательно таблица должна состоять из 5 строк и 5 столбцов, так как 25=5\*5 наиболее близкое к 26 число. При этом буквы I, J не различаются (J отождествляется с буквой I), так как не хватает 1 ячейки:

1 2 3 4 5

1 A B C D E

2 F G H I/J K

3 L M N O P

4 Q R S T U

5 V W X Y Z

Русский алфавит

Идею формирования таблицы шифрования проиллюстрируем для русского языка. Число букв в русском алфавите отличается от числа букв в греческом алфавите, поэтому размер таблицы выбран другой (квадрат 6\*6=36, поскольку 36 наиболее близкое число к 33):

1 2 3 4 5 6

1 А Б В Г Д Е

2 Ё Ж З И Й К

3 Л М Н О П Р

4 С Т У Ф Х Ц

5 Ч Ш Щ Ъ Ы Ь

6 Э Ю Я — — —

Возможен также другой вариант составления, предусматривающий объединение букв Е и Ё, И и Й, Ъ и Ь. В данном случае получаем следующий результат:

1 2 3 4 5 6

1 А Б В Г Д Е/Ё

2 Ж З И/Й К Л М

3 Н О П Р С Т

4 У Ф Х Ц Ч Ш

5 Щ Ы Ь/Ъ Э Ю Я

Используя подобный алгоритм, таблицу шифрования можно задать для любого языка. Чтобы расшифровать закрытый текст, необходимо знать, таблицей шифрования какого алфавита он зашифрован.

Или есть такой вариант: Шифр «Квадрат Полибия».

«Квадрат Полибия» представляет собой квадрат 5×5, столбцы и строки которого нумеруются цифрами от 1 до 5. В каждую клетку этого квадрата записывается одна буква (в нашем алфавите 31 буква, Ъ и Ё исключены, кроме того, в одну клетку поместите буквы е-э, и-й, ж-з, р-с, ф-х, ш-щ). Буквы расположены в алфавитном порядке. В результате каждой букве соответствует пара чисел, и шифрованное сообщение превращается в последовательность пар чисел. Расшифровывается путём нахождения буквы, стоящей на пересечении строки и столбца.

1 2 3 4 5

1 А Б В Г Д

2 Е/Э Ж З И/Й К

3 Л М Н О П

4 Р/С Т У Ф/Х Ц

5 Ч Ш/Щ Ы Ю Я

Принцип шифрования

Сообщение преобразуется в координаты по квадрату Полибия, координаты записываются вертикально:

Таблица координат

Буква:

S O M E T E X T

Координата горизонтальная:

3 4 2 5 4 5 3 4

Координата вертикальная:

4 3 3 1 4 1 5 4

Затем координаты считывают по строкам:

34 25 45 34 43 31 41 54 [5]

## 1.3 Алгоритм работы шифра Атбаша [6-7]

В Ветхом Завете, в том числе в книге пророка Иеремии (VI век до н. э.), использовалась техника скрытия отдельных кусков текста, получившая название «атбаш».

* Иер. 25:26: и всех царей севера, близких друг к другу и дальних, и все царства земные, которые на лице земли, а царь Сесаха выпьет после них.
* Иер. 51:41: Как взят Сесах, и завоевана слава всей земли! Как сделался Вавилон ужасом между народами!

В этих отрывках слово «Сесах» относится к государству, не упоминаемому в других источниках, но если в написании слова «Сесах» на иврите заменить первую букву алфавита на последнюю, вторую на предпоследнюю и так далее, то получится «Бавель» – одно из названий Вавилона. Таким образом, с помощью техники «атбаш» авторы манускрипта скрывали отдельные названия, оставляя большую часть текста без шифрования. Возможно, это делалось в том числе и для того, чтобы не иметь проблем с распространением текстов на территории, подконтрольной Вавилону [6].

Шифр Атбаша можно рассматривать как частный случай аффинного шифра.

Согласно стандартному аффинному соглашению, алфавит из m букв отображается на числа 0, 1, ..., m – 1. (Еврейский алфавит имеет m = 22 и в стандартном латинском алфавите m = 26). Затем шифр Атбаша может быть зашифрован и расшифрован с использованием функции шифрования для аффинного шифра, установив a = b = (m – 1):

E(x) = D(x) = ((m – 1)x + (m – 1)) mod m.

Это можно упростить до

E(x) = (m – 1)(x + 1) mod m = – (x+1) mod m.

Если вместо этого m букв алфавита сопоставлены с 1, 2, ..., m, то функция шифрования и дешифрования для шифра Атбаш становится

E(x) = (–x mod m) + 1 [7].

## 1.4 Алгоритм работы шифра Виженера [8-10]

Шифр Виженера – это метод шифрования буквенного текста с использованием ключевого слова.

Этот метод является простой формой многоалфавитной замены. Шифр Виженера изобретался многократно. Впервые этот метод описал Джованни-Баттиста Беллазо (Giovan Battista Bellaso) в книге La cifra del. Sig. Giovan Battista Bellasо в 1553 году, однако в 19 веке получил имя Блеза Виженера, швейцарского дипломата. Метод прост для понимания и реализации, он является недоступным для простых методов криптоанализа.

Квадрат Виженера или таблица Виженера, может быть использована для зашифрования и расшифрования.

В шифре Цезаря каждая буква алфавита сдвигается на несколько позиций; например, в шифре Цезаря при сдвиге +3, A стало бы D, B стало бы E и так далее. Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Для зашифрования может использоваться таблица алфавитов, называемая квадрат Виженера. Применительно к латинскому алфавиту таблица Виженера составляется из строк по 26 символов, причем каждая следующая строка сдвигается на несколько позиций. Таким образом, в таблице получается 26 различных шифров Цезаря. На разных этапах кодировки шифр Виженера использует различные алфавиты из этой таблицы. На каждом этапе шифрования используются различные алфавиты, выбираемые в зависимости от символа ключевого слова. Например, предположим, что исходный текст имеет вид:

ATTACKATDAWN

Человек, посылающий сообщение, записывает ключевое слово("LEMON") циклически до тех пор, пока его длина не будет соответствовать длине исходного текста:

LEMONLEMONLE

Первый символ исходного текста A зашифрован последовательностью L, которая является первым символом ключа. Первый символ L шифрованного текста находится на пересечении строки L и столбца A в таблице Виженера. Точно так же для второго символа исходного текста используется второй символ ключа; т. е. второй символ шифрованного текста X получается на пересечении строки E и столбца T. Остальная часть исходного текста шифруется подобным способом [8].

Расшифрование

Расшифрование слова осуществляется с помощью таблицы Виженера. Необходимо найти строку, которая соответствует первому символу ключевого слова. Строка будет содержать первый символ зашифрованного текста.

Столбец, который содержит данный символ, будет соответствовать первому символу исходного текста. Последующие значения будут расшифровываться аналогичным образом.

Предоставляя зашифрованный текст, необходимо задать ключевое слово. Оно понадобится для того, чтобы расшифровать код с помощью русского шифра Виженера в том числе. Для того чтобы убедиться в правильности кодировки, лучше дважды проверить текст. Если текст будет неправильно закодирован, его невозможно правильно расшифровать.

При использовании квадрата Виженера с пробелами и пунктуацией процесс расшифровки значительно усложнится. Важно знать о том, что частое повторение кодового слова позволит легче расшифровать текст. Поэтому кодовая информация должна быть длинной [9].

Пример

Исходный текст: ATTACKATDAWN

Ключ: LEMONLEMONLE

Зашифрованный текст: LXFOPVEFRNHR [8]

Интересно, что в оригинальном изложении идеи у Альберти описывается специальный диск с вращающимися кольцами с алфавитом, и только чуть позже, уже другим автором, была предложена таблица, которую мы используем выше. Диск, описанный в трактате 1466 года, в итоге использовался, например, в армии конфедератов во время гражданской войны в Америке [10].

## 1.5 Алгоритм работы шифрования RSA [6]

В 1978 г. Рональд Рив´ест, Ади Шамир и Леонард Адлеман (англ. Ronald Linn Rivest, Adi Shamir, Leonard Max Adleman) предложили алгоритм, обладающий рядом интересных для криптографии свойств. На его основе была построена первая система шифрования с открытым ключом, получившая название по первым буквам фамилий авторов – система RSA.

Рассмотрим принцип построения криптосистемы шифрования RSA с открытым ключом.

1. Создание пары из закрытого и открытого ключей.
2. Случайно выбрать большие простые различные числа 𝑝 и 𝑞, для которых .
3. Вычислить произведение
4. Вычислить функцию Эйлера
5. Выбрать случайное целое число 𝑒 ∈ [3, 𝜙(𝑛) −1], взаимно простое с 𝜙(𝑛):
6. Вычислить число 𝑑 такое, что
7. Закрытым ключом будем называть пару чисел 𝑛 и 𝑑, открытым ключом – пару чисел 𝑛 и 𝑒.
8. Шифрование с использованием открытого ключа.
9. Сообщение представляют целым числом
10. Шифртекст вычисляется как

Шифртекст – также целое число из диапазона

1. Расшифрование с использованием закрытого ключа.

Владелец закрытого ключа вычисляет

Покажем корректность схемы шифрования RSA. В результате расшифрования шифртекста 𝑐 (полученного путём шифрования открытого текста 𝑚) легальный пользователь имеет:

Если 𝑚 и 𝑝 являются взаимно простыми, то из малой теоремы Ферма следует, что:

Если же 𝑚 и 𝑝 не являются взаимно простыми, то есть 𝑝 является делителем 𝑚 (помним, что 𝑝 – простое число), то и

В результате, для любых 𝑚 верно, что . Аналогично доказывается, что Из китайской теоремы об остатках следует:

.

Пример. Создание ключей, шифрование и расшифрование в криптосистеме RSA.

1. Генерирование параметров.
2. Выберем числа 𝑝 = 13, 𝑞 = 11, 𝑛 = 143.
3. Вычислим 𝜙(𝑛) = (𝑝 − 1)(𝑞 − 1) = 12 · 10 = 120.
4. Выберем 𝑒 = 23 : gcd(𝑒, 𝜙(𝑛)) = 1, 𝑒 ∈ [3, 119].
5. Найдём 𝑑 = mod 𝜙(𝑛) = mod 120 = 47.
6. Открытый и закрытый ключи:

PK = (𝑒: 23, 𝑛: 143), SK = (𝑑: 47, 𝑛: 143).

1. Шифрование.
2. Пусть сообщение 𝑚 = 22 ∈ [1, 𝑛 − 1].
3. Вычислим шифртекст:

𝑐 = mod 𝑛 = mod 143 = 55 mod 143.

1. Расшифрование.
2. Полученный шифртекст 𝑐 = 55.
3. Вычислим открытый текст:

𝑚 = mod 𝑛 = mod 143 = 22 mod 143 [6].

## 1.6 Алгоритм работы шифра «перестановка скитала» [11-13]

В Древней Спарте использовали палицу, которая называется скитала (если верить сотрудникам музея Сен-Сир, где выставлена такая штука, она сделана приблизительно в пятом веке до нашей эры).

На этот посох наматывалась по спирали полоска пергамента с зашифрованным посланием. Смысл такого “гаджета” был в том, что прочитать эту полоску мог лишь обладатель скиталы аналогичного размера. При правильном размере витка буквы послания совпадали, и получался связный текст. Устройство было очень простым и практичным, хотя особо надежным его назвать никак нельзя.

Согласно легенде, этот “шифр” сумел разгадать еще Аристотель, а в 1841 году в июльском журнале Graham's Magazine его редактор Эдгар Аллан По опубликовал статью «Несколько слов о тайнописи» (A FEW WORDS ON SECRET WRITING), в которой рассказал про скиталу и поведал об остроумном методе дешифровки скиталы любого диаметра.

По словам родоначальника детективного жанра, для “взлома” скиталы нужно взять, скажем, шестифутовый конус и намотать на него ленту с текстом, а затем перемещать вдоль длины конуса, пока текст не станет читаемым. Страсть Эдгара По к различным головоломкам и шифрам нашла свое отражение не только в его бессмертных произведениях. Будучи редактором журнала, он вел активное общение с читателями, призывая присылать ему ребусы и зашифрованные послания, которые он старался разгадывать совместно с подписчиками издания, совершенствуя тем самым свои навыки в криптологии [11].

Шифровка

Вводится сообщение для шифровки.

Высчитывается k – длина сообщения.

Вводится «ключ» m – количество строк матрицы Скитала.

Высчитывается количество столбцов n по формуле |(k-1) div n| + 1. Пример, разобранный ниже, поможет разобраться в происхождении формулы.

Определяется новое положение буквы сообщения index по формуле:

index = |m \* (i mod n)| + |i div n|, тут i – текущее положение буквы в исходном сообщении [12].

Типичный текст отправленного зашифрованного сообщения мог бы выглядеть на полоске бумаги так:

ПЕАОСНДЬАП\_ЛИН\*ША\*И\_\*ТК\*

Чтобы расшифровать сообщение, получателю необходимо было знать лишь количество строк m, помещающихся на скитале. В данном случае, например, m=3. Тогда количество столбцов n при длине сообщения в k символов (у нас k=24) вычисляется по простой формуле:

n = [(k-1)/m] + 1 = [(24-1)/3] + 1 = 8 ([x] - целая часть числа)

Имея набор из различных скитал, получатель наматывал полоску на подходящую и расшифровывал сообщение:

Исходное сообщение "Подпишитесь на канал" содержит знаки пробелов и символы, призванные дополнить его до 24 символов, что конечно даёт врагу информацию о длине сообщения. Шифровали на стороне отправителя так: плотно наматывали полоску на скиталу и записывали сообщение слева направо, сдвигаясь п строкам вниз. После разматывания текст представлял собой белиберду.

Таким образом, в этой системе шифрования ключом было количество строк скиталы, которое и было необходимо держать в секрете, периодически менять и т. д. Скитала фактически реализовывала шифр маршрутной перестановки. Его принцип в том, что открытый текст записывается в некую фигуру по определенной траектории, а выписывается по другой. Конечно, криптостойкость данного шифра была невысока даже для полного перебора значений m и n, но доподлинно известно, что скитала использовалась в войне Спарты против Афин в 5 веке до н. э. [13]

# ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1 Постановка задачи

Необходимо разработать программу, которая должна шифровать и дешифровать исходные данные одним из представленных алгоритмов.

Программа должна выполнять следующие основные действия:

* обеспечить процедуру проверки пользовательских данных путем авторизации в системе;
* предусмотреть ввод исходного текста с клавиатуры или загрузку данных из файла;
* реализовать шифрование данных и демонстрацию полученных

результатов;

* реализовать дешифрование (расшифрование) данных и демонстрацию полученных результатов (консоль, файл).

Методы шифрования реализовать в виде отдельных функций/модулей.

## 2.2 Характеристика задачи

* + 1. Программа «MDTFCipher» предназначена для автоматизации шифрования и расшифрования исходных данных авторизованного в системе пользователя.
    2. Программа используется пользователем для защиты персональной информации.
    3. Периодичность решения задачи по запросу пользователя.
    4. Прекращение автоматизированного решения задачи происходит при отключении источника электропитания ЭВМ.
    5. Связь с другими задачами отсутствует.
    6. Специальных ограничений на временные характеристики решения задачи не налагается.
    7. Специальных требований на уровень подготовки пользователя не налагается. Но лицо, работающее с программой, должно иметь минимальное представление о компьютере (знание необходимых операций).

## 2.3 Алгоритм решения

1. Запустить приложение «MDTFCipher.exe»;
2. Вывод: "Enter password (0 – Exit):";
   1. Если выбран пункт – "0 – Exit";
      1. Выход из приложения с сообщением "Exiting program";
3. Иначе ввод пароля;
   1. Если выбран пункт – "0 – Exit";
      1. Выход из приложения с сообщением "Exiting program";
4. Вывод меню: "Choose mode (0 - Exit, 1 - Manual input, 2 - Encrypt file, 3 - Decrypt file):";
   1. Если выбран пункт – "1 - Manual input";
      1. Вывод подменю: "Enter valid choice (0 - Back, 1 - Atbash Method, 2 - Gronsfeld Method, 3 - RSA Method, 4-Skitala Method, 5-Vigener Method, 6-Polybius Method):";
         1. Если выбран пункт – "0 - Back";
            1. Возврат в предыдущее меню;
         2. Если выбран пункт – "1 - Atbash Method";
            1. Вывод подменю: "Choose mode (0 - Back, 1 - Encrypt, 2 - Decrypt):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

Если выбран пункт – "1 - Encrypt";

Вывод: "Enter text to encrypt:";

Ввод текста;

Вывод зашифрованного текста;

Если выбран пункт – "2 - Decrypt";

Вывод: "Enter text to encrypt:";

Ввод текста;

Вывод расшифрованного текста;

* + - 1. Если выбран пункт – "2 - Gronsfeld Method";
         1. Вывод подменю: "Choose mode (0 - Back, 1 - Encrypt, 2 - Decrypt):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

Если выбран пункт – "1 - Encrypt";

Вывод: "Enter text to encrypt:";

Ввод текста;

Вывод: "Enter key:";

Ввод ключа;

Вывод зашифрованного текста;

Если выбран пункт – "2 - Decrypt";

Вывод: "Enter text to encrypt:";

Ввод текста;

Вывод: "Enter key:";

Ввод ключа;

Вывод расшифрованного текста;

* + - 1. Если выбран пункт – "3 - RSA Method";
         1. Вывод подменю: "Choose mode (0 - Back, 1 - Encrypt, 2 - Decrypt):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

Если выбран пункт – "1 - Encrypt";

Вывод: "Enter text to encrypt:";

Ввод текста;

Вывод: "Enter two different prime numbers separated by space:";

Ввод двух простых чисел через пробел;

Вывод зашифрованного текста;

Если выбран пункт – "2 - Decrypt";

Вывод: "Enter text to encrypt:";

Ввод текста;

Вывод: "Enter two different prime numbers separated by space:";

Ввод двух простых чисел через пробел;

Вывод расшифрованного текста;

* + - 1. Если выбран пункт – "4-Skitala Method";
         1. Вывод подменю: "Choose mode (0 - Back, 1 - Encrypt, 2 - Decrypt):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

Если выбран пункт – "1 - Encrypt";

Вывод: "Enter text to encrypt:";

Ввод текста;

Вывод: "Enter the diameter (enter the number >=1):";

Ввод диаметра;

Вывод зашифрованного текста;

Если выбран пункт – "2 - Decrypt";

Вывод: "Enter text to encrypt:";

Ввод текста;

Вывод: "Enter the diameter (enter the number >=1):";

Ввод диаметра;

Вывод расшифрованного текста;

* + - 1. Если выбран пункт – "5-Vigener Method";
         1. Вывод подменю: "Choose mode (0 - Back, 1 - Encrypt, 2 - Decrypt):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

Если выбран пункт – "1 - Encrypt";

Вывод: "Enter text to encrypt:";

Ввод текста;

Вывод: "Enter the key in English (for English text) and in Russian (for Russian text)";

Ввод ключа;

Вывод зашифрованного текста;

Если выбран пункт – "2 - Decrypt";

Вывод: "Enter text to encrypt:";

Ввод текста;

Вывод: "Enter the key in English (for English text) and in Russian (for Russian text)";

Ввод ключа;

Вывод расшифрованного текста;

* + - 1. Если выбран пункт – "6-Polybius Method";
         1. Вывод подменю: "Choose mode (0 - Back, 1 - Encrypt, 2 - Decrypt):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

Если выбран пункт – "1 - Encrypt";

Вывод: "Enter text to encrypt:";

Ввод текста;

Вывод зашифрованного текста;

Если выбран пункт – "2 - Decrypt";

Вывод: "Enter text to encrypt:";

Ввод текста;

Вывод расшифрованного текста;

* 1. Если выбран пункт – "2 - Encrypt file";
     1. Вывод подменю: "Enter valid choice (0 - Back, 1 - Atbash Method, 2 - Gronsfeld Method, 3 - RSA Method, 4-Skitala Method, 5-Vigener Method, 6-Polybius Method):";
        1. Если выбран пункт – "1 - Atbash Method";
           1. Вывод: "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, из которого считываем данные;
        2. Вывод: "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, куда мы хотим записать зашифрованный текст;
        2. Ввод текста;
        3. Вывод: "Enter text to encrypt:";
        4. Ввод текста;
        5. Вывод: "Enter text to encrypt:";
        6. Ввод текста;
        7. Вывод "File is encrypted!"
      1. Если выбран пункт – "2 - Gronsfeld Method";
         1. Вывод: "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, из которого считываем данные;
        2. Вывод: "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, куда мы хотим записать зашифрованный текст;
        2. Вывод: "Enter text to encrypt:";
        3. Ввод текста;
        4. Вывод: "Enter key:";
        5. Ввод ключа;
        6. Вывод "File is encrypted!"
      1. Если выбран пункт – "3 - RSA Method";
         1. Вывод: "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, из которого считываем данные;
        2. Вывод: "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, куда мы хотим записать зашифрованный текст;
        2. Вывод: "Enter text to encrypt:";
        3. Ввод текста;
        4. Вывод: "Enter two different prime numbers separated by space:";
        5. Ввод двух простых чисел через пробел;
        6. Вывод "File is encrypted!"
      1. Если выбран пункт – "4-Skitala Method";
         1. Вывод: "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, из которого считываем данные;
        2. Вывод: "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, куда мы хотим записать зашифрованный текст;
        2. Вывод: "Enter text to encrypt:";
        3. Ввод текста;
        4. Вывод: "Enter the diameter (enter the number >=1):";
        5. Ввод диаметра;
        6. Вывод "File is encrypted!"
      1. Если выбран пункт – "5-Vigener Method";
         1. Вывод: "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, из которого считываем данные;
        2. Вывод: "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, куда мы хотим записать зашифрованный текст;
        2. Вывод: "Enter text to encrypt:";
        3. Ввод текста;
        4. Вывод: "Enter the key in English (for English text) and in Russian (for Russian text)";
        5. Ввод ключа;
        6. Вывод "File is encrypted!"
      1. Если выбран пункт – "6-Polybius Method";
         1. Вывод: "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, из которого считываем данные;
        2. Вывод: "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, куда мы хотим записать зашифрованный текст;
        2. Вывод: "Enter text to encrypt:";
        3. Ввод текста;
        4. Вывод "File is encrypted!"
  1. Если выбран пункт – "3 - Decrypt file";
     1. Вывод подменю: "Enter valid choice (0 - Back, 1 - Atbash Method, 2 - Gronsfeld Method, 3 - RSA Method, 4-Skitala Method, 5-Vigener Method, 6-Polybius Method):";
        1. Если выбран пункт – "1 - Atbash Method";
           1. Вывод: "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, из которого считываем данные;
        2. Вывод: "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, куда мы хотим записать расшифрованный текст;
        2. Ввод текста;
        3. Вывод: "Enter text to encrypt:";
        4. Ввод текста;
        5. Вывод: "Enter text to encrypt:";
        6. Ввод текста;
        7. Вывод "File is decrypted!"
      1. Если выбран пункт – "2 - Gronsfeld Method";
         1. Вывод: "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, из которого считываем данные;
        2. Вывод: "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, куда мы хотим записать расшифрованный текст;
        2. Вывод: "Enter text to encrypt:";
        3. Ввод текста;
        4. Вывод: "Enter key:";
        5. Ввод ключа;
        6. Вывод "File is decrypted!"
      1. Если выбран пункт – "3 - RSA Method";
         1. Вывод: "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, из которого считываем данные;
        2. Вывод: "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, куда мы хотим записать расшифрованный текст;
        2. Вывод: "Enter text to encrypt:";
        3. Ввод текста;
        4. Вывод: "Enter two different prime numbers separated by space:";
        5. Ввод двух простых чисел через пробел;
        6. Вывод "File is decrypted!"
      1. Если выбран пункт – "4-Skitala Method";
         1. Вывод: "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, из которого считываем данные;
        2. Вывод: "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, куда мы хотим записать расшифрованный текст;
        2. Вывод: "Enter text to encrypt:";
        3. Ввод текста;
        4. Вывод: "Enter the diameter (enter the number >=1):";
        5. Ввод диаметра;
        6. Вывод "File is decrypted!"
      1. Если выбран пункт – "5-Vigener Method";
         1. Вывод: "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, из которого считываем данные;
        2. Вывод: "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, куда мы хотим записать расшифрованный текст;
        2. Вывод: "Enter text to encrypt:";
        3. Ввод текста;
        4. Вывод: "Enter the key in English (for English text) and in Russian (for Russian text)";
        5. Ввод ключа;
        6. Вывод "File is decrypted!"
      1. Если выбран пункт – "6-Polybius Method";
         1. Вывод: "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, из которого считываем данные;
        2. Вывод: "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back):";

Если выбран пункт – "0 - Back";

Возврат в предыдущее меню;

* + - * 1. Ввод имени текстового файла, куда мы хотим записать расшифрованный текст;
        2. Вывод: "Enter text to encrypt:";
        3. Ввод текста;
        4. Вывод "File is decrypted!"

## 2.4 Руководство пользователя

2.4.1 Введение

* + - 1. Программа «MDTFCipher» предназначена для шифрования и расшифрования различного рода символов (в некоторых алгоритмах шифруется только буквы), которые могут быть введены и выведены, как и в консоль, так и из файла в файл (в формате .txt).

* + - 1. Программа предоставляет пользователю следующие возможности:

‒ полный доступ в систему при вводе правильного пароля;

‒ ввод и вывод в консоль, выгрузка и запись текста в файлы (в формате .txt);

‒ шифрование и расшифрование текста шестью способами ("RSA", "Атбаш", "Гронсфельд", "Квадрат Полибия", "Вижинер", "Скитала").

* + - 1. Программа реализована на алгоритмическом языке C++14. Работает в любой среде совместимой с «Windows». Дисковой памяти для запуска программы требуется не менее 5 Mb. Оперативной памяти для нормальной работы программы требуется не менее 4 Mb.

2.4.2 Описание операций

Основные функции программы «MDTFCipher» соответствуют рисункам 2.3–2.8. Для удобства пользователя и более легкого изучения системы большинство форм и диалогов имеют идентичный интерфейс. Далее описаны все функции системы, а также формы и диалоги для ввода и вывода информации. Для каждой формы приведены основные компоненты и их назначение.

После запуска приложения на экран выводится главное окно программы. Вид окна приведен на рисунке 2.1. Для начала работы с программой необходимо ввести правильный пароль (в данном случае «mdt!f 228 1213 2»)

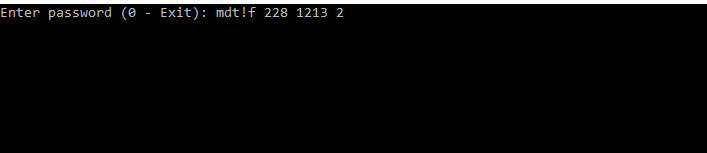


Рисунок 2.1 - Главное окно программы

В случае неверного ввода пароля программа выдаст ошибку и попросит повторить вход (рисунок 2.2)

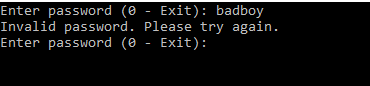


Рисунок 2.2 – Вывод ошибки в пароли

Далее программа предлагает пользователю на выбор любой режим записи и чтения текста (рисунок 2.3)

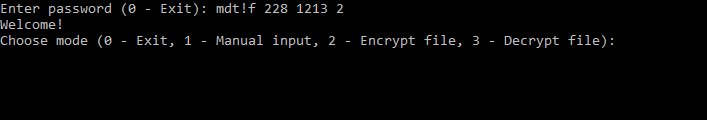


Рисунок 2.3 – Вывод меню выбора

режима и выбор пункта

После выбора любого режима программа предоставляет пользователю на выбор один из шести предложенных алгоритма шифрования (рисунок 2.4)

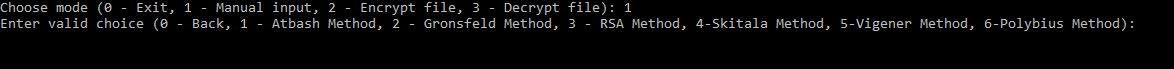


Рисунок 2.4 – Вывод меню выбора

алгоритма и выбор пункта

После выбора любого алгоритма шифрования и выбора режима «1 - Manual input» программа предоставляет пользователю выбрать режим для консоли (рисунок 2.5)

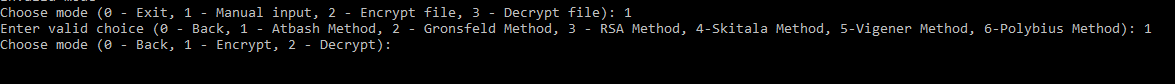


Рисунок 2.5 – Вывод меню выбора

режима для консоли и выбор пункта

Примерный вид программы во время шифрования и расшифрования в консоли на рисунке 2.6. Иногда пользователю предстоит вводить дополнительный ключ (рисунки 2.7–2.8).

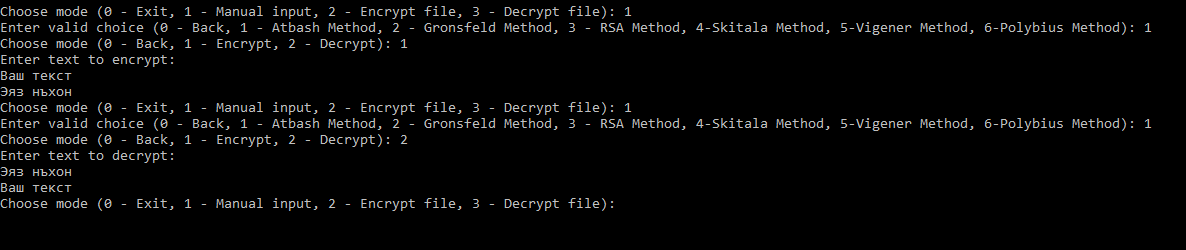


Рисунок 2.6 – Вид программы во время шифрования и расшифрования в консоли

Примерный вид программы во время шифрования файла на рисунке 2.7. Пользователь должен указать путь на существующий текстовый файл и путь записи в виде текстового файла.

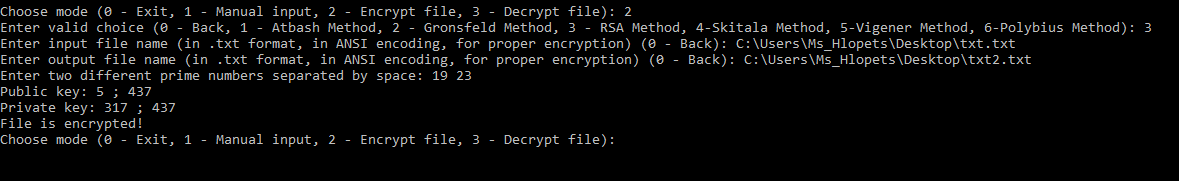


Рисунок 2.7 – Вид программы во время шифрования файла

Примерный вид программы во время расшифрования файла на рисунке 2.8. Пользователь должен указать путь на существующий текстовый файл и путь записи в виде текстового файла.

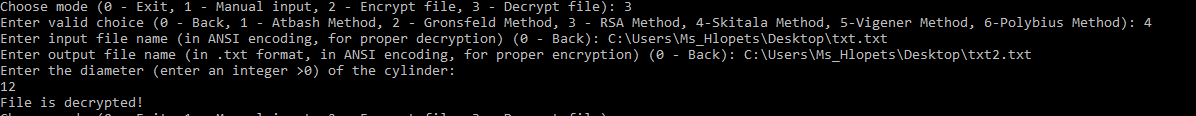


Рисунок 2.8 – Вид программы во время расшифрования файла

В случае некорректного ввода в любой из операций программа выдаст соответствующую ошибку и попросит попробовать ещё раз (рисунок 2.9)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2.9 – Примерный вывод ошибки

2.4.3 Сообщения пользователю

При работе с программой могут появиться следующие сообщения, представленные в таблице 2.1.

Таблица 2.1 – Сообщения пользователю

|  |  |
| --- | --- |
| Текст сообщения программы | Ответ пользователя |
| Invalid password. Please try again | Неправильный пароль. Ввести правильный пароль. |
| Invalid mode | Некорректный выбор режима. Ввести корректный режим. |
| Invalid choice | Некорректный выбор алгоритма шифрования. Ввести корректный алгоритм шифрования. |
| Invalid Key (The number must be a natural number). Please try again | Некорректный ввод ключа. Ввести корректный ключ. |
| Invalid input. Please enter two different prime numbers separated by space | Некорректный ввод ключа. Ввести корректный ключ. |
| Invalid input. Please enter two prime larger numbers separated by space ((p-1)\*(q-1)>254) | Ошибка при вводе ключа: функция Эйлера ключа должна быть больше 254. Ввести корректный ключ. |
| Invalid input. Please enter two smaller prime numbers separated by space (p\*q<3025550010) | Ошибка при вводе ключа: произведение ключа должно быть меньше 3025550010. Ввести корректный ключ. |
| Invalid input. Please enter two different prime numbers separated by space | Ошибка при вводе ключа: ключ должен состоять из двух разных простых чисел. Ввести корректный ключ. |
| Error, please enter only numbers! | Некорректный ввод ключа. Ввести корректный ключ. |
| Error, please enter the number >=1! | Некорректный ввод ключа. Ввести корректный ключ. |
| Input file doesn't exist. Please try again | Ошибка при открытии файла: файл не найден или не открывается. Ввести корректное название и проверить целостность файла. |
| Input file is not a text file. Please try again/ Output file is not a text file. Please try again | Ошибка при открытии файла: файл не в формате .txt. Ввести название файла в формате .txt. |
| Error, please enter at least one English or Russian letter | Ошибка при вводе ключа: в ключе должны иметься русские или английские буквы. |

2.4.4 Аварийные ситуации

К аварийным ситуациям относятся: нехватка оперативной памяти для создания окна программы. Если исполняемый модуль программы не запускается, либо не выполнены требования условия работы программы, либо один из файлов поврежден, необходимо обратиться к разработчику программы.

## 2.5 Руководство системного программиста

Программа реализована на языке C++14 в среде Microsoft Visual Studio 2022.

Модули программы:

cipher.h – заголовочный файл, содержит объявления функций, использованных в основной функции;

cipher.cpp – файл содержит определение функций, объявленных в заголовочном файле cipher.h:

void encrypt\_decryptFile(std::string& mode\_2) – функция расшифрования и шифрования из файла в файл;

void encrypt\_decryptConsole(std::string& inputFileName, std::string& outputFileName, std::string& mode\_1, std::string& choice) – функция расшифрования и шифрования в консоли;

password.h – заголовочный файл, содержит объявление функции, использованную в основной функции;

password.cpp – файл содержит определение функции, объявленной в заголовочном файле password.h:

std::string get\_password(std::string key, uint\_fast64\_t p, uint\_fast64\_t q) – функция, которая получает пароль из файла password.txt (при помощи расшифровки);

atb.h, gro.h, polybius.h, rsa.h, skitala.h, vigener.h – заголовочные файлы, содержат объявления функций, использованных в cipher.cpp (некоторые из функций gro.h, rsa.h и atb.h также используются в password.cpp);

atb.cpp, gro.cpp, polybius.cpp, rsa.cpp, skitala.cpp, vigener.cpp – файлы содержат определения функций, объявленных в заголовочных файлах atb.h, gro.h, polybius.h, rsa.h, skitala.h, vigener.h соответственно:

atb.cpp:

void atb\_encrypt(std::string& inputFileName, std::string& outputFileName) – функция для зашифрования;

void atb\_decrypt(std::string& inputFileName, std::string& outputFileName) – функция для расшифрования;

void changeatb(int& asciiCode) – функция, которая шифрует и расшифрует символы;

rsa.cpp:

uint\_fast64\_t calculate\_gcd(uint\_fast64\_t a, uint\_fast64\_t b) – функция подсчёта НОД;

uint\_fast64\_t calculate\_e(uint\_fast64\_t phi) – функция подсчёта взаимнопростого числа;

uint\_fast64\_t calculate\_d(uint\_fast64\_t e, uint\_fast64\_t phi) – функция подсчёта обратного числа по модулю;

bool is\_prime(uint\_fast64\_t num) – функция проверки на простоту;

bool can\_encrypt1(uint\_fast64\_t p, uint\_fast64\_t q) – функция проверки, что (p-1) \* (q-1) >= 255;

bool is\_different(uint\_fast64\_t p, uint\_fast64\_t q) – функция проверки, что числа разные;

bool can\_encrypt2(uint\_fast64\_t p, uint\_fast64\_t q) – функция проверки, что p \* q <= 3025550009;

uint\_fast64\_t fast\_pow(uint\_fast64\_t a, uint\_fast64\_t b, uint\_fast64\_t n) – функция быстрого возведения в степень по модулю;

void rsa\_encrypt(std::string& inputFileName, std::string& outputFileName) – функция зашифрования;

void rsa\_decrypt(std::string& inputFileName, std::string& outputFileName) – функция расшифрования;

skitala.cpp:

void skital\_encrypt(std::string& inputFileFileName, std::string& outputFileFileName) – функция зашифрования;

void skital\_decrypt(std::string& inputFileFileName, std::string& outputFileFileName) – функция расшифрования;

vigener.cpp:

void vigener\_encrypt(string& inputFileFileName, string& outputFileFileName) – функция зашифрования;

void vigener\_decrypt(string& inputFileFileName, string& outputFileFileName) – функция расшифрования;

main.cpp. Содержит функцию main, представляющую функционал по вводу пароля, выводу меню режима и выбора шифров, с соответствующим вызовом функций, и вводом необходимых данных.

Программа содержит ряд сообщений, предназначенных для сигнализации ошибок, которые представлены в таблице 2.2:

Таблица 2.2 – Сообщения для сигнализации ошибок

|  |  |
| --- | --- |
| Текст сообщения программы | Расшифровка |
| Invalid password. Please try again | Неправильный пароль. Программа попросит повторить ввод ещё раз. |
| Invalid mode | Некорректный выбор режима. Программа попросит повторить выбор ещё раз. |
| Invalid choice | Некорректный выбор алгоритма шифрования. Программа попросит повторить выбор ещё раз. |
| Invalid Key (The number must be a natural number). Please try again | Некорректный ввод ключа. Программа попросит повторить ввод ещё раз. |
| Invalid input. Please enter two different prime numbers separated by space | Некорректный ввод ключа. Программа попросит повторить ввод ещё раз. |
| Invalid input. Please enter two prime larger numbers separated by space ((p-1)\*(q-1)>254) | Ошибка при вводе ключа: функция Эйлера ключа должна быть больше 254. Программа попросит повторить ввод ещё раз. |
| Invalid input. Please enter two smaller prime numbers separated by space (p\*q<3025550010) | Ошибка при вводе ключа: произведение ключа должно быть меньше 3025550010. Программа попросит повторить ввод ещё раз |
| Invalid input. Please enter two different prime numbers separated by space | Ошибка при вводе ключа: ключ должен состоять из двух разных простых чисел. Программа попросит повторить ввод ещё раз |  |
| Error, please enter only numbers! | Некорректный ввод ключа. Программа попросит повторить ввод ещё раз. |  |
| Error, please enter the number >=1! | Некорректный ввод ключа. Программа попросит повторить ввод ещё раз. |  |
| Input file doesn't exist. Please try again | Ошибка при открытии файла: файл не найден или не открывается. Программа попросит повторить ввод ещё раз. |  |
| Input file is not a text file. Please try again/ Output file is not a text file. Please try again | Ошибка при открытии файла: файл не в формате .txt. Программа попросит повторить ввод ещё раз. |  |
| Error, please enter at least one English or Russian letter | Ошибка при вводе ключа. Программа попросит повторить ввод. |  |

Связь модулей программы между собой представлена на рисунке 2.10.

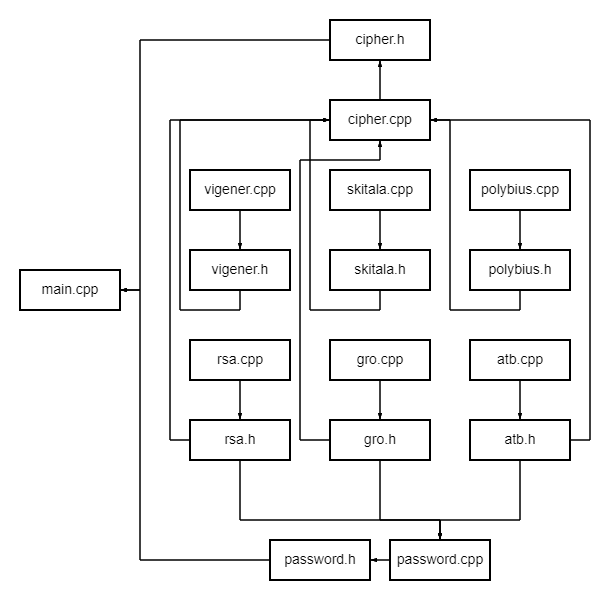


Рисунок 2.10 – Связь модулей программы

## 2.6 Контрольный пример

После запуска приложения на экран выводится главное окно программы. Вид окна приведен на рисунке 2.11. Для начала работы с программой необходимо ввести правильный пароль (в данном случае «mdt!f 228 1213 2»)

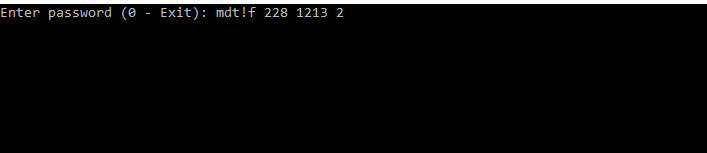


Рисунок 2.11 - Главное окно программы

В случае неверного ввода пароля программа выдаст ошибку и попросит повторить вход (рисунок 2.12)

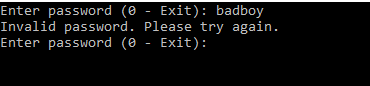


Рисунок 2.12 – Вывод ошибки в пароли

Далее программа предлагает пользователю на выбор любой режим записи и чтения текста (рисунок 2.13)

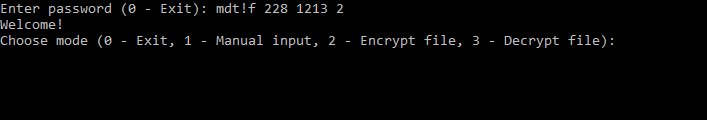


Рисунок 2.13 – Вывод меню выбора

режима и выбор пункта

После выбора любого режима программа предоставляет пользователю на выбор один из шести предложенных алгоритма шифрования (рисунок 2.14)

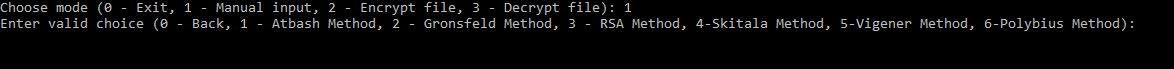


Рисунок 2.14 – Вывод меню выбора

алгоритма и выбор пункта

После выбора любого алгоритма шифрования и выбора режима «1 - Manual input» программа предоставляет пользователю выбрать режим для консоли (рисунок 2.15)

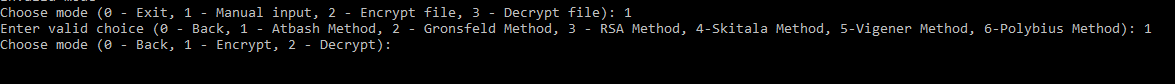


Рисунок 2.15 – Вывод меню выбора

режима для консоли и выбор пункта

Примерный вид программы во время шифрования и расшифрования в консоли на рисунке 2.16. Иногда пользователю предстоит вводить дополнительный ключ (рисунки 2.17–2.18).

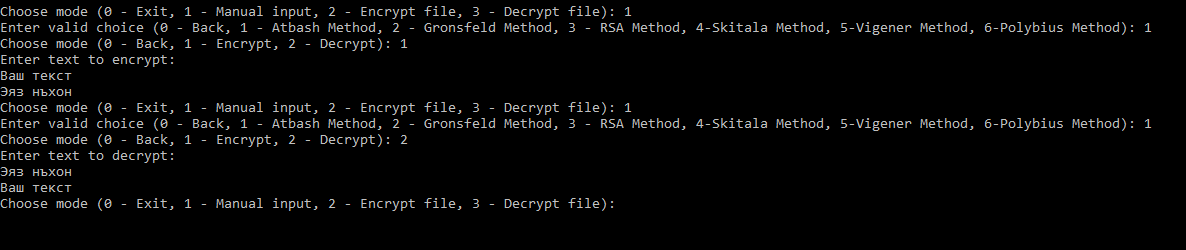


Рисунок 2.16 – Вид программы во время шифрования и расшифрования в консоли

Примерный вид программы во время шифрования файла на рисунке 2.17. Пользователь должен указать путь на существующий текстовый файл и путь записи в виде текстового файла.

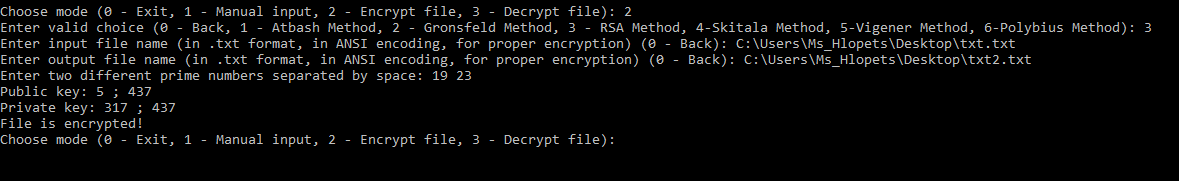


Рисунок 2.17 – Вид программы во время шифрования файла

Примерный вид программы во время расшифрования файла на рисунке 2.18. Пользователь должен указать путь на существующий текстовый файл и путь записи в виде текстового файла.

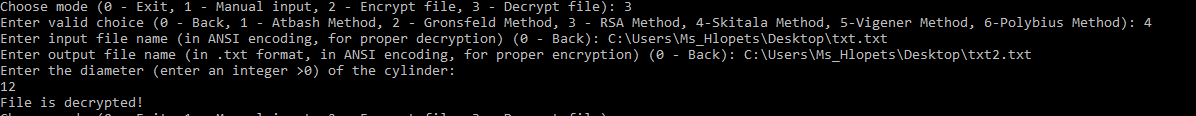


Рисунок 2.18 – Вид программы во время расшифрования файла

В случае некорректного ввода в любой из операций программа выдаст соответствующую ошибку и попросит попробовать ещё раз (рисунок 2.19)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2.19 – Примерный вывод ошибки

Ручной расчёт

Linker Эдуард 123 !@#

1) Скитала:

Ключ: 3

Шифровка:

Диаметр — это ключ

Находим количество рядов по формуле: (длина текста + диаметр - 1) / диаметр = (21+3-1)/3=7 (int); дальше шифруем так: пишем 0 букву (L), потом пишем букву с индексом 0(L)+7=7(Э), дальше пишем 7(Э)+7=14(1), дальше 14(1)+7=21 - конец цикла, начинаем заново с 1;

Пишем 1 букву (i), дальше пишем 1(i)+7=8, дальше 8(д)+7=15(2), ...

В итоге получаем строку: LЭ1iд2nу3kа eр!rд@ #

Расшифровка:

Пишем 0 букву (L), дальше 0+7=7, значит на 7 позицию пишем (Э), дальше 7+7=14, значит 14 позицию пишем (1), дальше 14+7=21 - конец цикла, начинаем заново с 1;

Пишем 1 букву (i), дальше 1+7=8, значит на 8 позицию пишем (д), ...

В итоге получаем строку: Linker Эдуард 123 !@#

2) Виженер:

Ключ: фы as

Шифровка:

Разделяем ключ на: key1 = as (для английского), key2 = фы (для русского);

Индекс ключа = 0, но при каждом выводе зашифрованного символа увеличивается на 1;

Находим part\_key\_eng (часть англ. ключа) = key1[индекс ключа % длину key1] = 0 % 2 = 0, это 0 индекс в англ. алфавите, следовательно part\_key\_eng = 97('a');

Находим индекс в англ алфавите для нашей буквы из ключа part\_key\_eng в 97('a')-'a'=0;

Находим индекс в англ алфавите для нашей 0 буквы ('L'): 'L'-'A'=11;

Складываем индексы по модулю длины алфавита для того, чтобы получить новый индекс: (0+11)%26 = 11;

С помощью нового индекса получаем зашифрованную букву: 11 +'A' = 76('L');

Далее

Индекс ключа = 1;

Находим part\_key\_eng = key1[индекс ключа % длину key1] = 1 % 2 = 1, это 1 индекс в англ. алфавите, следовательно part\_key\_eng = 115('s');

Находим индекс в англ алфавите для нашей буквы из ключа part\_key\_eng в 115('s')-'a'=18;

Находим индекс в англ алфавите для нашей 1 буквы ('i'): 'i'-'a'=8;

Складываем индексы по модулю длины алфавита для того, чтобы получить новый индекс: (18+8)%26 = 0;

С помощью нового индекса получаем зашифрованную букву: 0 +'a' = 97('a');

Далее

Индекс ключа = 2;

Находим part\_key\_eng = key1[индекс ключа % длину key1] = 2 % 2 = 0, это 1 индекс в англ. алфавите, следовательно part\_key\_eng = 97('a');

Находим индекс в англ алфавите для нашей буквы из ключа part\_key\_eng в 97('a')-'a'=0;

Находим индекс в англ алфавите для нашей 2 буквы ('n'): 'n'-'a'=13;

Складываем индексы по модулю длины алфавита для того, чтобы получить новый индекс: (0+13)%26 = 13;

С помощью нового индекса получаем зашифрованную букву: 13 +'a' = 97('n');

...

В итоге получим: Lancej Сязыдя 123 !@#

Расшифровка:

Индекс ключа = 0;

Разделяем ключ на: key1 = as (для английского), key2 = фы (для русского);

Находим part\_key\_eng = key1[индекс ключа % длину key1] = 0 % 2 = 0, это 0 индекс в англ. алфавите, следовательно part\_key\_eng = 97('a');

Находим индекс в англ алфавите для нашей буквы из ключа part\_key\_eng в 97('a')-'a'=0;

Находим индекс в англ алфавите для нашей 0 буквы ('L'): 'L'-'A'=11;

Отнимаем от индекса буквы индекс ключа + 26 (длина англ. алфавита) по модулю длины алфавита для того, чтобы получить новый индекс: (11-0+26)%26 = 11;

С помощью нового индекса получаем букву: 11 +'A' = 76('L');

Далее

Индекс ключа = 1;

Находим part\_key\_eng = key1[индекс ключа % длину key1] = 1 % 2 = 1, это 1 индекс в англ. алфавите, следовательно part\_key\_eng = 115('s');

Находим индекс в англ алфавите для нашей буквы из ключа part\_key\_eng в 115('s')-'a'=18;

Находим индекс в англ алфавите для нашей 1 буквы ('a'): 'a'-'a'=0;

Отнимаем от индекса буквы индекс ключа + 26 (длина англ. алфавита) по модулю длины алфавита для того, чтобы получить новый индекс: (0-18+26)%26 = 8;

С помощью нового индекса получаем букву: 0 +'a' = 105('i');

Далее

Индекс ключа = 2;

Находим part\_key\_eng = key1[индекс ключа % длину key1] = 2 % 2 = 0, это 1 индекс в англ. алфавите, следовательно part\_key\_eng = 97('a');

Находим индекс в англ алфавите для нашей буквы из ключа part\_key\_eng в 97('a')-'a'=0;

Находим индекс в англ алфавите для нашей 2 буквы ('n'): 'n'-'a'=13;

Отнимаем от индекса буквы индекс ключа + 26 (длина англ. алфавита) по модулю длины алфавита для того, чтобы получить новый индекс: (13-0+26)%26 = 13;

С помощью нового индекса получаем букву: 13 +'a' = 110('n');

...

В итоге получим: Linker Эдуард 123 !@#

3) Квадрат Полибиуса:

Шифровка:

Каждый символ заменяем четырьмя цифрами (первые две цифры - номер строки квадрата (если ascii-код <100, то к '0' добавляем первую цифру ascii-кода, если ascii-код >=100, то мы к первой цифре ascii-кода добавляем '0'), 3 и 4 цифры - номер столбца квадрата (если ascii-код <100, то к '0' добавляем вторую цифру ascii-кода, если ascii-код >=100, то пишем вторую и третью цифры));

L - 0706 (ascii-код = 76); i - 1005 (ascii-код = 105), n - 1010 (ascii-код = 110) ...

В итоге получаем строку: 070610051010100710011014030220212028204320242040202803020409050005010302030306040305

Расшифровка:

Если встречается 4 подряд цифры, то проверяется какая первая цифра, если она равна 0, то получаем ascii-код так: вторая цифра умножается на десять + четвертая цифра, если первая цифра не равна нулю, то получаем ascii-код так: первая цифра умножается на сто + третья цифра умножается на десять + четвертая цифра, с помощью ascii-кода выписываем символ;

0706 преобразуем в 76, с помощью ascii-кода выписываем L, 1005 преобразуем в 105, с помощью ascii-кода выписываем i, 1010 преобразуем в 110, с помощью ascii-кода выписываем n...

В итоге получаем строку: Linker Эдуард 123 !@;

Gataullin Денис «№; 123

4) Атбаш

Шифровка:

Берём и смотрим по алфавиту для английских и русских букв: просто считаем с начала позицию нашей шифруемой буквы и заменяем на букву с такой же позицией, но с конца (для цифр аналогичная история, т. е. 0–9, 1–8...), специальные символы с пробелами не изменяются;

В итоге получаем строку: Tzgzfoorm Ыътчо «№; 876

Расшифровка:

Делаем то же самое, как и в шифровке, но наоборот;

Получаем строку: Gataullin Денис «№; 123

5) Гронсфельд

Ключ = 123;

Шифровка:

Записываем под каждым символом по символу из ключа (если ключ меньше, чем строка, то мы повторяем ключ):

G a t a u l l i n

1 2 3 1 2 3 1 2 3 1

Д е н и с « № ; 1 2 3

2 3 1 2 3 1 2 3 1 2 3 1 1

Далее просто каждый символ смещаем на то количество букв в алфавите, чему равен ключ символа, т. е. G смещается по алфавиту на одну букву и т. д. Специальные символы, пробелы и цифры не шифруются;

В итоге получаем строку: Hcwbwomkq Жиокф «№; 123

Расшифровка:

В расшифровке делаем туже операцию, но сдвигаем буквы влево;

Получаем исходную строку: Gataullin Денис «№; 123

6) RSA

Простые числа – 17 и 19;

Шифровка:

Вычитываем произведение: 17\*19=323;

Вычитываем функцию эйлера для этого произведения: (17-1)(19-1)=288;

Находим первое число, которое обратно взаимнопростое с 288: 2, 3, 4 – не подходят, 5 подходит;

Находим обратный элемент 5 по модулю 288: 288, 5 (2, -115); 5, 288%5 (-1, 2); 3, 5%3 (1, -1); 2, 3%2 (0, 1); 1, 0 (1, 0) => обратный элемент равен -115 mod 288= 173;

Открытый ключ: (5; 323);

Закрытый ключ: (173; 323);

Получаем по таблице ASCII номер каждого символа (кроме пробела), допустим, G = 71. Далее шифруем при помощи 71^5 mod 323 = 124, допустим ещё Д = 196. Шифруем 196^5 mod 323 = 195 и так далее;

Расшифровка:

Теперь выполняем вот такие операции: 124^173 mod 323 = 71 (G), 195^173 mod 323 = 196 (Д).

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* 1. Гребенников В.В. Европейская криптология. История спецсвязи. // Персональный сайт Владимира Гребенникова. — 2013. [Электронный ресурс] URL: <http://cryptohistory.ru> (дата обращения 08.06.2023 г.).
  2. GitHub [Электронный ресурс] URL: <https://github.com/IkigaiYoshi/AnalysisOfGronsfeldCipher> (дата обращения: 08.06.2023).
  3. Шифр Гронсфельда // Большой Справочник [Электронный ресурс] URL: https://newstest.ru/Шифр\_Гронсфельда (дата обращения: 08.06.2023).
  4. Ашаева А. ФОЦПМРЬОЯКЦА: разбираемся в шифрах // Музей криптографии [Электронный ресурс]. URL: <https://theoryandpractice.ru/posts/19385-fotspmroyaktsa-razbiraemsya-v-shifrakh> (дата обращения: 08.06.2023).
  5. Квадрат Полибия (Тюремная азбука) // NIGHTQUESTS [Электронный ресурс] URL: <https://nightquests.ru/knowledgebases/kvadrat-polibiya-tyuremnaya-azbuka/> (дата обращения: 08.06.2023).
  6. Криптографические методы защиты информации: учебное пособие / Владимиров С.М., Габидулин Э.М., Колыбельников А.И., Кшевецкий А.С., под ред. Владимиров С.М., Габидулин Э.М., Колыбельников А.И., Кшевецкий А.С. — 3-е изд. — М.: Московский физико-технический институт, 2021. - 418 с.
  7. Атбаш // АльфапедиЯ [Электронный ресурс]. Дата обновления: 13.06.2021 — URL: <https://alphapedia.ru/w/Atbash> (дата обращения: 08.06.2023).
  8. Шифр Вижинера, пример,Криптоанализ, Реализация // Intellect.icu — 2016. — 19 сентября [Электронный ресурс]. Дата обновления: 13.06.2021 — URL: <https://intellect.icu/shifr-vizhinera-primer-kriptoanaliz-realizatsiya-6420> (дата обращения: 08.06.2023).
  9. Бещева Т. Шифр Виженера. Квадрат Виженера. Шифрование текста // FB.RU. — 2018. — 25 апреля [Электронный ресурс] URL: <https://yandex.ru/turbo/fb.ru/s/article/381590/shifr-vijenera-kvadrat-vijenera-shifrovanie-teksta> (дата обращения 08.06.2023 г.).
  10. Как шифр из XVI века стал основой для невзламываемого шифра XX века // Kaspersky daily — 2018. — 11 сентября [Электронный ресурс]. — URL: <https://www.kaspersky.ru/blog/vigenere-cipher-history/8856/> (дата обращения: 08.06.2023).
  11. Бондаренко М., Бондаренко С. Шифры из прошлого: тайнопись и загадки докомпьютерной эпохи // 3D NEWS — 2018. — 08 июля [Электронный ресурс] URL: <https://3dnews.ru/916293/offsyanka/shifri-iz-proshlogo-taynopis-i-zagadki-dokompyuternoy-epohi?ext=subscribe&source=subscribeRu> (дата обращения 08.06.2023 г.).
  12. Васильев В.С. Алгоритм шифра Скитала // Персональный сайт Владимира Васильева. — 2018. — 3 марта [Электронный ресурс] URL: <https://pro-prof.com/forums/topic/cipher-scytale-algorithm> (дата обращения 08.06.2023 г.).
  13. Как спартанцы шифровали военную тайну? // Дзен [Электронный ресурс]. — 2021. — 8 марта — URL: <https://dzen.ru/a/YEXnuFgoVzbdPpv6> (дата обращения: 08.06.2023).
  14. C++ reference [Электронный ресурс] URL: <https://en.cppreference.com/w> (дата обращения: 08.06.2023).
  15. Самоучитель "Уроки по C++": учебное пособие — Самиздат, 2022. – 1655 с.
  16. Документация по Visual Studio // Microsoft [Электронный ресурс] — URL: <https://learn.microsoft.com/ru-ru/visualstudio/windows/?view=vs-2022> (дата обращения: 08.06.2023).

# ПРИЛОЖЕНИЕ

Текст программы

main.cpp – главный модуль программы

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

#include <windows.h>

#include "cipher.h"

#include "password.h"

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string pas, password, key;

uint\_fast64\_t p, q;

Invalidpassword:

cout << "Enter password (0 - Exit): ";

getline(cin, pas);

stringstream ss(pas);

if (pas == "0") goto exit;

ss >> password >> key >> p >> q;

if (password != get\_password(key, p, q)) {

cout << "Invalid password. Please try again." << endl;

goto Invalidpassword;

}

else {

cout << "Welcome!" << endl;

string mode\_2;

cout << "Choose mode (0 - Exit, 1 - Manual input, 2 - Encrypt file, 3 - Decrypt file): ";

getline(cin, mode\_2);

while (mode\_2 != "0") {

if (mode\_2 == "1") {

cout << "Enter valid choice (0 - Back, 1 - Atbash Method, 2 - Gronsfeld Method, 3 - RSA Method, 4-Skitala Method, 5-Vigener Method, 6-Polybius Method): ";

string choice;

getline(cin, choice);

try {

if (choice != "0" && choice != "1" && choice != "2" && choice != "3" && choice != "4" && choice != "5" && choice != "6")

throw "Invalid choice";

}

catch (const char\* msg) {

cout << msg << endl;

continue;

}

if (choice != "0") {

string inputFileName = "input.txt";

string outputFileName = "output.txt";

string mode\_1;

Invalidmode:

cout << "Choose mode (0 - Back, 1 - Encrypt, 2 - Decrypt): ";

getline(cin, mode\_1);

if (mode\_1 != "0" && mode\_1 != "1" && mode\_1 != "2") {

cout << "Invalid mode" << endl;

goto Invalidmode;

}

if (mode\_1 == "0") continue;

if (mode\_1 == "1" || mode\_1 == "2") encrypt\_decryptConsole(inputFileName, outputFileName, mode\_1, choice);

}

}

else if (mode\_2 == "2" || mode\_2 == "3") encrypt\_decryptFile(mode\_2);

else {

cout << "Invalid mode" << endl;

}

cout << "Choose mode (0 - Exit, 1 - Manual input, 2 - Encrypt file, 3 - Decrypt file): ";

getline(cin, mode\_2);

}

}

exit:

cout << "Exiting program";

return 0;

}

password.h

#ifndef PASSWORD\_H

#define PASSWORD\_H

#include <string>

std::string get\_password(std::string key, uint\_fast64\_t p, uint\_fast64\_t q);

#endif

password.cpp

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

#include "rsa.h"

#include "gro.h"

#include "atb.h"

using namespace std;

void check\_numbers(uint\_fast64\_t& p, uint\_fast64\_t& q)

{

if (p <= 0 || q <= 0) {

p = 17;

q = 19;

}

if (!is\_prime(p) || !is\_prime(q)) {

p = 17;

q = 19;

}

if (!is\_different(p, q)) {

p = 17;

q = 19;

}

if (!can\_encrypt1(p, q)) {

p = 17;

q = 19;

}

if (!can\_encrypt2(p, q)) {

p = 17;

q = 19;

}

}

string pass1\_decrypt(string inputString, string key) {

istringstream input(inputString);

if (!(checkIfNumeric(key))) key = "666";

int keyLength = key.length();

int\* keyArray = new int[keyLength];

for (int i = 0; i < keyLength; i++)

{

keyArray[i] = key[i] - '0';

}

string decryptedText = "";

string line;

while (getline(input, line))

{

for (int i = 0; i < line.length(); i++)

{

char symbol = line[i];

if (symbol == ' ')

{

decryptedText += " ";

continue;

}

int asciiCode = (int)symbol - keyArray[i % keyLength];

changegro(asciiCode, symbol);

decryptedText += (char)asciiCode;

}

}

delete[] keyArray;

return decryptedText;

}

string pass2\_decrypt(string inputString)

{

istringstream iss(inputString);

string line;

string decrypted\_pass = "";

while (getline(iss, line)) {

for (int i = 0; i < line.length(); i++) {

char symbol = line[i];

int asciiCode = (int)symbol;

changeatb(asciiCode);

decrypted\_pass += (char)asciiCode;

}

}

return decrypted\_pass;

}

string pass3\_decrypt(string inputFileName, uint\_fast64\_t p, uint\_fast64\_t q)

{

ifstream inputFile(inputFileName);

string line;

check\_numbers(p, q);

uint\_fast64\_t n = p \* q;

uint\_fast64\_t phi = (p - 1) \* (q - 1);

uint\_fast64\_t e = calculate\_e(phi);

uint\_fast64\_t d = calculate\_d(e, phi);

string decrypted\_pass = "";

string crypted\_m\_str;

stringstream ss(line);

while (getline(inputFile, line))

{

string crypted\_m\_str;

uint\_fast64\_t crypted\_m;

stringstream ss(line);

while (getline(ss, crypted\_m\_str, ' '))

{

crypted\_m = stoull(crypted\_m\_str);

uint\_fast64\_t m = fast\_pow(crypted\_m, d, n);

char symbol = (char)m;

decrypted\_pass += symbol;

}

}

inputFile.close();

return decrypted\_pass;

}

string get\_password(string key, uint\_fast64\_t p, uint\_fast64\_t q) {

string inputFileName = "password.txt";

if (key == "" || p <= 0 || q <= 0)

return "Invalid";

else

return (pass1\_decrypt(pass2\_decrypt(pass3\_decrypt(inputFileName, p, q)), key));

}

cipher.h

#ifndef CIPHER\_H

#define CIPHER\_H

#include <string>

void encrypt\_decryptFile(std::string& mode\_2);

void encrypt\_decryptConsole(std::string& inputFileName, std::string& outputFileName, std::string& mode\_1, std::string& choice);

#endif

cipher.cpp

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <thread>

#include <sstream>

#include "cipher.h"

#include "rsa.h"

#include "atb.h"

#include "gro.h"

#include "skitala.h"

#include "vigener.h"

#include "polybius.h"

using namespace std;

bool fileDifferent(string& inputFileName, string& outputFileName) {

return (inputFileName == outputFileName);

}

bool fileExists(string& inputFileName) {

ifstream file(inputFileName);

return file.good();

}

void encryptFile(string& inputFileName, string& outputFileName, string& choice)

{

string line;

if (choice == "0")

return;

else if (choice == "1") {

atb\_encrypt(inputFileName, outputFileName);

return;

}

else if (choice == "2") {

gro\_encrypt(inputFileName, outputFileName);

return;

}

else if (choice == "3") {

rsa\_encrypt(inputFileName, outputFileName);

return;

}

else if (choice == "4") {

skital\_encrypt(inputFileName, outputFileName);

return;

}

else if (choice == "5") {

vigener\_encrypt(inputFileName, outputFileName);

return;

}

else if (choice == "6") {

polybius\_encrypt(inputFileName, outputFileName);

return;

}

}

void decryptFile(string& inputFileName, string& outputFileName, string& choice)

{

if (choice == "0")

return;

else if (choice == "1") {

atb\_decrypt(inputFileName, outputFileName);

return;

}

else if (choice == "2") {

gro\_decrypt(inputFileName, outputFileName);

return;

}

else if (choice == "3") {

rsa\_decrypt(inputFileName, outputFileName);

return;

}

else if (choice == "4") {

skital\_decrypt(inputFileName, outputFileName);

return;

}

else if (choice == "5") {

vigener\_decrypt(inputFileName, outputFileName);

return;

}

else if (choice == "6") {

polybius\_decrypt(inputFileName, outputFileName);

return;

}

}

void encrypt\_decryptFile(string& mode\_2) {

while (true) {

cout << "Enter valid choice (0 - Back, 1 - Atbash Method, 2 - Gronsfeld Method, 3 - RSA Method, 4-Skitala Method, 5-Vigener Method, 6-Polybius Method): ";

string choice;

getline(cin, choice);

try {

if (choice != "0" && choice != "1" && choice != "2" && choice != "3" && choice != "4" && choice != "5" && choice != "6")

throw "Invalid choice";

}

catch (const char\* msg) {

cout << msg << endl;

continue;

}

if (choice != "0") {

string inputFileName, outputFileName;

Inputfile:

cout << "Enter input file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back): ";

getline(cin, inputFileName);

if (inputFileName == "0") return;

if (!fileExists(inputFileName)) {

cout << "Input file doesn't exist. Please try again." << endl;

goto Inputfile;

}

if ((inputFileName.substr(inputFileName.find\_last\_of(".") + 1)) != "txt") {

cout << "Input file is not a text file. Please try again." << endl;

goto Inputfile;

}

Outfile:

cout << "Enter output file name (in .txt format, in ANSI encoding, for proper encryption) (0 - Back): ";

getline(cin, outputFileName);

if (outputFileName == "0") goto Inputfile;

if (fileDifferent(inputFileName, outputFileName)) {

cout << "The output file must't have the same name as the input file. Please try again." << endl;

goto Outfile;

}

if ((outputFileName.substr(outputFileName.find\_last\_of(".") + 1)) != "txt") {

cout << "Output file is not a text file. Please try again." << endl;

goto Outfile;

}

if (mode\_2 == "2") {

encryptFile(inputFileName, outputFileName, choice);

cout << "File is encrypted!" << endl;

}

else {

decryptFile(inputFileName, outputFileName, choice);

cout << "File is decrypted!" << endl;

}

}

return;

}

}

void encrypt\_decryptConsole(string& inputFileName, string& outputFileName, string& mode\_1, string& choice) {

string text;

cout << "Enter text to encrypt:" << endl;

getline(cin, text);

ofstream inputFile(inputFileName);

if (inputFile.is\_open())

{

inputFile << text;

inputFile.close();

}

if (mode\_1 == "1") encryptFile(inputFileName, outputFileName, choice);

else decryptFile(inputFileName, outputFileName, choice);

ifstream outputFile(outputFileName);

if (outputFile.is\_open())

{

string en\_decryptedText;

while (getline(outputFile, en\_decryptedText))

{

cout << en\_decryptedText << endl;

}

outputFile.close();

}

}

atb.h

#ifndef ATB\_H

#define ATB\_H

#include <string>

void atb\_encrypt(std::string& inputFileName, std::string& outputFileName);

void atb\_decrypt(std::string& inputFileName, std::string& outputFileName);

void changeatb(int& asciiCode);

#endif

atb.cpp

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

void changeatb(int& asciiCode)

{

int temp = 0;

if ((asciiCode >= 65 && asciiCode <= 90) || (asciiCode >= 97 && asciiCode <= 122)) {

if (asciiCode < 97) temp = 32;

asciiCode = 122 - (asciiCode + temp - 97);

asciiCode -= temp;

}

if (asciiCode >= 192 && asciiCode <= 255) {

if (asciiCode < 224) temp = 32;

asciiCode = 255 - (asciiCode + temp - 224);

asciiCode -= temp;

}

if (asciiCode >= 48 && asciiCode <= 57) {

asciiCode = 57 - (asciiCode - 48);

}

}

void atb\_encrypt(string& inputFileFileName, string& outputFileFileName)

{

wifstream inputFile(inputFileFileName);

wofstream outputFile(outputFileFileName);

wstring line;

while (getline(inputFile, line)) {

for (int i = 0; i < line.length(); i++) {

wchar\_t symbol = line[i];

if (symbol == ' ') {

outputFile << " ";

continue;

}

int asciiCode = (int)symbol;

changeatb(asciiCode);

outputFile << (wchar\_t)asciiCode;

}

outputFile << endl;

}

inputFile.close();

outputFile.close();

}

void atb\_decrypt(string& inputFileFileName, string& outputFileFileName)

{

atb\_encrypt(inputFileFileName, outputFileFileName);

}

polybius.h

#ifndef POLYBIUS\_H

#define POLYBIUS\_H

#include <string>

using namespace std;

void polybius\_encrypt(string& inputFileFileName, string& outputFileFileName);

void polybius\_decrypt(string& inputFileFileName, string& outputFileFileName);

#endif

polybius.cpp

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

void polybius\_encrypt(string& inputFileFileName, string& outputFileFileName)

{

ifstream inputFile(inputFileFileName);

ofstream outputFile(outputFileFileName);

string line;

while (getline(inputFile, line)) {

for (size\_t i = 0; i < line.length(); ++i) {

int code\_ascii = static\_cast<unsigned char>(line[i]);

if (code\_ascii < 100) { outputFile << '0' << code\_ascii / 10 << '0' << code\_ascii % 10; }

else { outputFile << code\_ascii / 100 << '0' << (code\_ascii / 10) % 10 << code\_ascii % 10; }

}

outputFile << endl;

}

inputFile.close();

outputFile.close();

}

void polybius\_decrypt(string& inputFileFileName, string& outputFileFileName)

{

ifstream inputFile(inputFileFileName);

ofstream outputFile(outputFileFileName);

string line;

int code\_ascii;

while (getline(inputFile, line)) {

for (size\_t i = 0; i < line.length(); ++i) {

if ((line[i] - '0' >= 0) && (line[i] - '0' <= 9) && (line[i + 1] - '0' >= 0) && (line[i + 1] - '0' <= 9) && (line[i + 2] - '0' >= 0) && (line[i + 2] - '0' <= 9) && (line[i + 3] - '0' >= 0) && (line[i + 3] - '0' <= 9)) {

if (line[i] == '0') {

code\_ascii = int(line[i + 1] - '0') \* 10 + int(line[i + 3] - '0');

outputFile << unsigned char(code\_ascii);

}

else {

code\_ascii = int(line[i] - '0') \* 100 + int(line[i + 2] - '0') \* 10 + int(line[i + 3] - '0');

outputFile << unsigned char(code\_ascii);

}

i += 3;

}

else { outputFile << line[i]; }

}

outputFile << endl;

}

inputFile.close();

outputFile.close();

}

gro.h

#ifndef GRO\_H

#define GRO\_H

#include <string>

bool checkIfNumeric(std::string str);

void gro\_encrypt(std::string& inputFileName, std::string& outputFileName);

void gro\_decrypt(std::string& inputFileName, std::string& outputFileName);

void changegro(int& asciiCode, wchar\_t symbol);

#endif

gro.cpp

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

bool checkIfNumeric(string str)

{

return !str.empty() && str.find\_first\_not\_of("0123456789") == string::npos;

}

void changegro(int& asciiCode, wchar\_t symbol) {

int temp = 0;

if ((symbol >= 65 && symbol <= 90) || (symbol >= 97 && symbol <= 122)) {

if (symbol < 97) temp = 32;

asciiCode += temp;

if (asciiCode > 122) asciiCode -= 26;

if (asciiCode < 97) asciiCode += 26;

asciiCode -= temp;

}

else if (symbol >= 192 && symbol <= 255) {

if (symbol < 224) temp = 32;

asciiCode += temp;

if (asciiCode > 255) asciiCode -= 32;

if (asciiCode < 224) asciiCode += 32;

asciiCode -= temp;

}

else asciiCode = int(symbol);

}

void gro\_universal(string& inputFileName, string& outputFileName, int temp) {

wifstream inputFile(inputFileName);

wofstream outputFile(outputFileName);

string key;

do {

cout << "Enter key: ";

getline(cin, key);

if (!(checkIfNumeric(key)))

cout << "Invalid Key (The number must be a natural number). Please try again." << endl;

else {

int keyLength = key.length();

int\* keyArray = new int[keyLength];

for (int i = 0; i < keyLength; i++)

{

keyArray[i] = key[i] - '0';

}

wstring line;

while (getline(inputFile, line))

{

for (int i = 0; i < line.length(); i++)

{

wchar\_t symbol = line[i];

if (symbol == ' ')

{

outputFile << " ";

continue;

}

int asciiCode = (int)symbol + temp \* keyArray[i % keyLength];

changegro(asciiCode, symbol);

outputFile << (wchar\_t)asciiCode;

}

outputFile << endl;

}

inputFile.close();

outputFile.close();

delete[] keyArray;

}

} while (!(checkIfNumeric(key)));

}

void gro\_encrypt(string& inputFileName, string& outputFileName) {

gro\_universal(inputFileName, outputFileName, 1);

}

void gro\_decrypt(string& inputFileName, string& outputFileName) {

gro\_universal(inputFileName, outputFileName, -1);

}

vigener.h

#ifndef VIGENER\_H

#define VIGENER\_H

#include <string>

using namespace std;

void vigener\_encrypt(string& inputFileFileName, string& outputFileFileName);

void vigener\_decrypt(string& inputFileFileName, string& outputFileFileName);

#endif

vigener.cpp

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

void key(string& temp\_key, string& key1, string& key2){

for (size\_t i = 0; i < temp\_key.length(); ++i) {

if (temp\_key[i] >= 'a' && temp\_key[i] <= 'z') {

key1 += temp\_key[i];

}

else if (temp\_key[i] >= 'A' && temp\_key[i] <= 'Z') {

key1 += temp\_key[i];

}

else if (temp\_key[i] >= 'а' && temp\_key[i] <= 'я') {

key2 += temp\_key[i];

}

else if (temp\_key[i] >= 'А' && temp\_key[i] <= 'Я') {

key2 += temp\_key[i];

}

}

}

void func\_enc\_dec(ifstream& inputFile, ofstream& outputFile, string& key1, string& key2, int& enc\_dec) {

string line;

int keyIndex = 0;

unsigned char part\_key\_eng = ' ', part\_key\_rus = ' ';

while (getline(inputFile, line)) {

for (size\_t i = 0; i < line.length(); ++i) {

unsigned char letter = line[i];

if ((letter >= 'A' && letter <= 'Z') || (letter >= 'a' && letter <= 'z') || (letter >= int(unsigned char('А')) && letter <= int(unsigned char('Я'))) || (letter >= int(unsigned char('а')) && letter <= int(unsigned char('я')))) {

if (key1.length() != 0) { part\_key\_eng = key1[keyIndex % key1.length()]; }

if (key2.length() != 0) { part\_key\_rus = key2[keyIndex % key2.length()]; }

if (key1.length() != 0) {

if (part\_key\_eng >= 'a' && part\_key\_eng <= 'z') {

part\_key\_eng -= 'a';

}

else if (part\_key\_eng >= 'A' && part\_key\_eng <= 'Z') {

part\_key\_eng -= 'A';

}

}

if (key2.length() != 0) {

if (part\_key\_rus >= int(unsigned char('а')) && part\_key\_rus <= int(unsigned char('я'))) {

part\_key\_rus -= int(unsigned char('а'));

}

else if (part\_key\_rus >= int(unsigned char('А')) && part\_key\_rus <= int(unsigned char('Я'))) {

part\_key\_rus -= int(unsigned char('А'));

}

}

if (key1.length() != 0) {

if (letter >= 'a' && letter <= 'z') {

letter -= 'a';

if (enc\_dec == 1) {

letter = (letter + part\_key\_eng) % 26;

}

else { letter = (letter - part\_key\_eng + 26) % 26; }

letter += 'a';

}

else if (letter >= 'A' && letter <= 'Z') {

letter -= 'A';

if (enc\_dec == 1) {

letter = (letter + part\_key\_eng) % 26;

}

else { letter = (letter - part\_key\_eng + 26) % 26; }

letter += 'A';

}

}

if (key2.length() != 0) {

if (letter >= int(unsigned char('А')) && letter <= int(unsigned char('Я'))) {

letter -= int(unsigned char('А'));

if (enc\_dec == 1) {

letter = (letter + part\_key\_rus) % 32;

}

else { letter = (letter - part\_key\_rus + 32) % 32; }

letter += int(unsigned char('А'));

}

else if (letter >= int(unsigned char('а')) && letter <= int(unsigned char('я'))) {

letter -= int(unsigned char('а'));

if (enc\_dec == 1) {

letter = (letter + part\_key\_rus) % 32;

}

else { letter = (letter - part\_key\_rus + 32) % 32; }

letter += int(unsigned char('а'));

}

}

outputFile << letter;

keyIndex++;

}

else { outputFile << letter; }

}

outputFile << endl;

}

inputFile.close();

outputFile.close();

}

void vigener\_encrypt(string& inputFileFileName, string& outputFileFileName)

{

ifstream inputFile(inputFileFileName);

ofstream outputFile(outputFileFileName);

string key1, key2, temp\_key;

int enc\_dec = 1;

jump:

cout << "enter the key in English (for English text) and in Russian (for Russian text)" << endl;

getline(cin, temp\_key);

key(temp\_key, key1, key2);

if ((key1.length() == 0) && (key2.length() == 0)) { cout << "Error, please enter at least one English or Russian letter" << endl; goto jump; }

func\_enc\_dec(inputFile, outputFile, key1, key2, enc\_dec);

}

void vigener\_decrypt(string& inputFileFileName, string& outputFileFileName)

{

ifstream inputFile(inputFileFileName);

ofstream outputFile(outputFileFileName);

string key1, key2, temp\_key;

int enc\_dec = 0;

jump:

cout << "enter the key in English (for English text) and in Russian (for Russian text)" << endl;

getline(cin, temp\_key);

key(temp\_key, key1, key2);

if ((key1.length() == 0) && ((key2.length() == 0))) { cout << "Error, please enter at least one English or Russian letter" << endl; goto jump; }

func\_enc\_dec(inputFile, outputFile, key1, key2, enc\_dec);

}

skitala.h

#ifndef SKITALA\_H

#define SKITALA\_H

#include <string>

void skital\_encrypt(std::string& inputFileFileName, std::string& outputFileFileName);

void skital\_decrypt(std::string& inputFileFileName, std::string& outputFileFileName);

#endif

skitala.cpp

#include <iostream>

#include <fstream>

#include <limits>

#include <iostream>

void skital\_encrypt(std::string& inputFileFileName, std::string& outputFileFileName)

{

std::ifstream inputFile(inputFileFileName);

std::ofstream outputFile(outputFileFileName);

std::string line;

jump:

std::string temp\_diameter = "";

int diameter = 0;

std::cout << "Enter the diameter (enter the number >=1): " << std::endl;

std::cin >> temp\_diameter;

for (int i = 0; i < temp\_diameter.length(); ++i) {

if ((temp\_diameter[i] - '0' >= 0) && (temp\_diameter[i] - '0' <= 9)) {

diameter = diameter \* 10 + (temp\_diameter[i] - '0');

}

}

if (diameter < 1) { std::cout << "Error, please enter the number >=1!" << std::endl; goto jump; }

getline(std::cin, line);

while (getline(inputFile, line)) {

std::string encryptedLine;

int lineLength = line.length();

int numRows = (lineLength + diameter - 1) / diameter;

for (int i = 0; i < numRows; i++) {

for (int j = i; j < lineLength; j += numRows) {

encryptedLine += line[j];

}

}

outputFile << encryptedLine << std::endl;

}

inputFile.close();

outputFile.close();

}

void skital\_decrypt(std::string& inputFileFileName, std::string& outputFileFileName)

{

std::ifstream inputFile(inputFileFileName);

std::ofstream outputFile(outputFileFileName);

std::string line;

jump:

std::string temp\_diameter = "";

int diameter = 0;

std::cout << "Enter the diameter (enter the number >=1): " << std::endl;

std::cin >> temp\_diameter;

for (int i = 0; i < temp\_diameter.length(); ++i) {

if ((temp\_diameter[i] - '0' >= 0) && (temp\_diameter[i] - '0' <= 9)) {

diameter = diameter \* 10 + (temp\_diameter[i] - '0');

}

}

if (diameter < 1) { std::cout << "Error, please enter the number >=1!" << std::endl; goto jump; }

getline(std::cin, line);

while (getline(inputFile, line)) {

std::string decryptedLine(line.length(), ' ');

int lineLength = line.length();

int numRows = ((lineLength - 1) / diameter) + 1;

int index = 0;

for (int i = 0; i < numRows; i++) {

for (int j = i; j < lineLength; j += numRows) {

decryptedLine[j] = line[index++];

}

}

outputFile << decryptedLine << std::endl;

}

inputFile.close();

outputFile.close();

}

rsa.h

#ifndef RSA\_H

#define RSA\_H

#include <string>

uint\_fast64\_t calculate\_gcd(uint\_fast64\_t a, uint\_fast64\_t b);

uint\_fast64\_t calculate\_e(uint\_fast64\_t phi);

uint\_fast64\_t calculate\_d(uint\_fast64\_t e, uint\_fast64\_t phi);

bool is\_prime(uint\_fast64\_t num);

bool can\_encrypt1(uint\_fast64\_t p, uint\_fast64\_t q);

bool is\_different(uint\_fast64\_t p, uint\_fast64\_t q);

bool can\_encrypt2(uint\_fast64\_t p, uint\_fast64\_t q);

uint\_fast64\_t fast\_pow(uint\_fast64\_t a, uint\_fast64\_t b, uint\_fast64\_t n);

void rsa\_encrypt(std::string& inputFileName, std::string& outputFileName);

void rsa\_decrypt(std::string& inputFileName, std::string& outputFileName);

#endif

rsa.cpp

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

#include <wchar.h>

using namespace std;

bool checkIfNumericRSA(string str)

{

if (str.length() < 19)

return !str.empty() && str.find\_first\_not\_of(" 0123456789") == std::string::npos;

else

return false;

}

uint\_fast64\_t calculate\_gcd(uint\_fast64\_t p, uint\_fast64\_t q)

{

if (q == 0)

return p;

else

return calculate\_gcd(q, p % q);

}

uint\_fast64\_t calculate\_e(uint\_fast64\_t phi)

{

uint\_fast64\_t e = 2;

while (e < phi && calculate\_gcd(e, phi) != 1)

{

e++;

}

return e;

}

uint\_fast64\_t calculate\_d(uint\_fast64\_t e, uint\_fast64\_t phi)

{

uint\_fast64\_t d = 1;

while ((d \* e) % phi != 1)

{

d++;

}

return d;

}

bool is\_prime(uint\_fast64\_t num)

{

if (num == 1)

return false;

for (uint\_fast64\_t i = 2; i <= sqrt(num); i++)

{

if (num % i == 0)

return false;

}

return true;

}

bool can\_encrypt1(uint\_fast64\_t p, uint\_fast64\_t q)

{

if ((p-1) \* (q-1) >= 255)

return true;

else

return false;

}

bool is\_different(uint\_fast64\_t p, uint\_fast64\_t q)

{

if (p != q)

return true;

else

return false;

}

bool can\_encrypt2(uint\_fast64\_t p, uint\_fast64\_t q)

{

if (p \* q <= 3025550009)

return true;

else

return false;

}

void get\_primes(uint\_fast64\_t & p, uint\_fast64\_t & q)

{

string primes;

cout << "Enter two different prime numbers separated by space: ";

Invalidinput :

getline(cin, primes);

stringstream ss(primes);

ss >> p >> q;

if (p <= 0 || q <= 0) {

cout << "Invalid input. Please enter two different prime numbers separated by space: ";

goto Invalidinput;

}

if (!is\_prime(p) || !is\_prime(q)) {

cout << "Invalid input. Please enter two different prime numbers separated by space: ";

goto Invalidinput;

}

if (!is\_different(p, q)) {

cout << "Invalid input. Please enter two different prime numbers separated by space: ";

goto Invalidinput;

}

if (!can\_encrypt1(p, q)) {

cout << "Invalid input. Please enter two prime larger numbers separated by space ((p-1)\*(q-1)>254): ";

goto Invalidinput;

}

if (!can\_encrypt2(p, q)) {

cout << "Invalid input. Please enter two smaller prime numbers separated by space (p\*q<3025550010): ";

goto Invalidinput;

}

}

uint\_fast64\_t fast\_pow(uint\_fast64\_t a, uint\_fast64\_t b, uint\_fast64\_t n)

{

uint\_fast64\_t res = 1;

while (b > 0)

{

if (b & 1)

{

res = (res \* a) % n;

}

a = (a \* a) % n;

b >>= 1;

}

return res;

}

void rsa\_encrypt(string& inputFileName, string& outputFileName)

{

wifstream inputFile(inputFileName);

wofstream outputFile(outputFileName);

wstring line;

uint\_fast64\_t p, q;

get\_primes(p, q);

uint\_fast64\_t n = p \* q;

uint\_fast64\_t phi = (p - 1) \* (q - 1);

uint\_fast64\_t e = calculate\_e(phi);

uint\_fast64\_t d = calculate\_d(e, phi);

cout << "Public key: " << e << " ; " << n << endl;

cout << "Private key: " << d << " ; " << n << endl;

while (getline(inputFile, line))

{

for (wchar\_t& symbol : line)

{

uint\_fast64\_t m = symbol;

uint\_fast64\_t crypted\_m = fast\_pow(m, e, n);

outputFile << crypted\_m << " ";

}

outputFile << "\n";

}

inputFile.close();

outputFile.close();

}

void rsa\_decrypt(string& inputFileName, string& outputFileName)

{

ifstream inputFile(inputFileName);

ofstream outputFile(outputFileName);

string line;

uint\_fast64\_t p, q;

get\_primes(p, q);

uint\_fast64\_t n = p \* q;

uint\_fast64\_t phi = (p - 1) \* (q - 1);

uint\_fast64\_t e = calculate\_e(phi);

uint\_fast64\_t d = calculate\_d(e, phi);

while (getline(inputFile, line))

{

string crypted\_m\_str;

uint\_fast64\_t crypted\_m;

stringstream ss(line);

while (getline(ss, crypted\_m\_str, ' '))

{

if (checkIfNumericRSA(crypted\_m\_str)) {

crypted\_m = stoull(crypted\_m\_str);

uint\_fast64\_t m = fast\_pow(crypted\_m, d, n);

char symbol = (char)m;

outputFile << symbol;

}

}

outputFile << "\n";

}

inputFile.close();

outputFile.close();

}