МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации



**РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА**

**по дисциплине: «*Программирование*»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБc-322», «АВТФ»  *Тишин Никита Александрович*  «20» июня 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Доцент кафедры ЗИ*  *Архипова Анастасия Борисовна*  «20» июня 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

/Новосибирск 2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc8861)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_Toc14389)

[1.1 Алгоритм работы шифра «AES128». 4](#_Toc15233)

[1.2 Алгоритм распределения ключей Хьюза. 9](#_Toc14995)

[1.3 Алгоритм работы шифра «Тарабарская грамота» 13](#_Toc25324)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 15](#_Toc30690)

[2.1 Постановка задачи 15](#_Toc21231)

[2.2 Характеристика задачи 15](#_Toc26667)

[2.3 Алгоритм решения 16](#_Toc6336)

[2.4 Руководство пользователя 18](#_Toc19413)

[2.4.4 Аварийные ситуации. 22](#_Toc25357)

[К аварийным ситуациям относятся: нехватка 22](#_Toc20270)

[2.5 Руководство системного программиста 23](#_Toc27612)

[2.6 Контрольный пример 25](#_Toc20306)

[2.7 Ручной расчёт 26](#_Toc8795)

[СПИСКОВ ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33](#_Toc29488)

[ПРИЛОЖЕНИЕ 34](#_Toc7252)

ВВЕДЕНИЕ

В современном мире, где электронные коммуникации стали неотъемлемой частью повседневной жизни, обеспечение безопасности передачи и хранения данных приобретает первостепенное значение. С увеличением объема информации, передаваемой и обрабатываемой в сети, возрастают и риски несанкционированного доступа. В этом контексте разработка эффективных методов шифрования и дешифрования текстовой информации становится критически важной для защиты данных.

Цель этой работы — создание программы, обеспечивающей надежное шифрование и дешифрование текстовых данных. Основная задача состоит в разработке алгоритмов и программного обеспечения, которые будут гарантировать высокий уровень безопасности обработки данных, оставаясь доступными для авторизованных пользователей..

Теоретической основой написания расчетно-графической работы явились работы следующих авторов Й. Бухманн [9], Б. Я. Рябко [10].

Практической основой написания расчетно-графической работы явились учебные материалы по языку программирования следующих авторов С.Прата [1], Т.А.Павлоская [2], Б.Страуструп [3-4], Р.Мартин [5], А. Александреску [6], С.Мейерс [7-8].

Ожидаемые результаты включают создание программного обеспечения, которое сможет эффективно шифровать и дешифровать текстовую информацию, а также успешное применение этого ПО для решения практических задач в области информационной безопасности. Новизна и эффективность данной работы заключаются в разработке продукта, который объединяет высокую степень защиты данных с удобством использования для конечных пользователей.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В программе используется 3 метода шифрования: AES128, Алгоритм распределения ключей Хьюза, Шифровка «Тарабарская грамота» (весь алфавит) . Все три шифра реализуют с помощью ключей шифрования. В «Тарабарской грамоте» ключ используется в формате таблицы замены символов.

1.1 Алгоритм работы шифра «AES128».

1.1.1Введение

AES-128 (Advanced Encryption Standard с ключом длиной 128 бит) является одним из самых популярных алгоритмов симметричного шифрования, широко используемых для защиты данных. Он был принят в качестве стандарта Федеральным агентством по стандартам и технологиям США (NIST) в 2001 году, после тщательного конкурса, в котором победил алгоритм, разработанный бельгийскими криптографами Джоаном Дайменом и Винсентом Рейменом (Rijndael).

1.1.2 Исторический контекст

История AES-128 начинается в 1997 году, когда Национальный институт стандартов и технологий США (NIST) объявил конкурс на разработку нового стандарта шифрования, который заменит устаревший DES (Data Encryption Standard). DES, принятый в 1977 году, к 1990-м годам стал уязвимым из-за значительного роста вычислительных мощностей, что позволило потенциальным злоумышленникам успешно проводить атаки на шифр.

Конкурс NIST привлек множество участников со всего мира, представивших различные криптографические алгоритмы. В 1998 году NIST получил 15 заявок на рассмотрение. После тщательного анализа и публичного обсуждения, в котором приняли участие эксперты в области криптографии, в 1999 году были отобраны пять финалистов: MARS, RC6, Serpent, Twofish и Rijndael.

В октябре 2000 года NIST объявил, что алгоритм Rijndael, разработанный бельгийскими криптографами Джоаном Дайменом и Винсентом Рейменом, станет новым стандартом шифрования. Rijndael был выбран за свою высокую криптостойкость, эффективность и гибкость, так как он поддерживал различные длины блоков и ключей. В декабре 2001 года алгоритм был официально принят в качестве Advanced Encryption Standard (AES).

Внедрение AES-128 стало важным шагом в области информационной безопасности. Он предоставил надёжный инструмент для защиты данных, который остаётся устойчивым к большинству известных атак и до сих пор широко используется во многих системах и приложениях, обеспечивая безопасность цифровой информации в современном мире.

1.1.3 Принцип работы

Принцип работы AES-128 заключается в многократном применении серии криптографических преобразований к 128-битным блокам данных с использованием 128-битного ключа. Алгоритм выполняет 10 раундов шифрования, каждый из которых включает четыре основные операции:

1. **SubBytes (Подстановка байтов)**: Каждый байт блока данных заменяется на другой байт по предопределенной таблице замен (S-box), что обеспечивает нелинейное преобразование.

2. **ShiftRows (Сдвиг строк)**: Байты в каждом ряду блока данных сдвигаются на разное количество позиций влево. Это перемешивает данные на уровне строк.

3. **MixColumns (Перемешивание столбцов)**: Столбцы блока данных перемешиваются путем умножения их на фиксированную матрицу. Эта операция обеспечивает диффузию, распространяя влияние каждого байта на весь блок.

4. **AddRoundKey (Добавление ключа раунда)**: К каждому байту блока данных применяется операция XOR с соответствующим байтом расширенного ключа для текущего раунда. Ключ раунда получается из начального ключа с помощью процедуры расширения ключа.

Начальный ключ используется для начальной операции AddRoundKey до начала первых 9 раундов. В последнем (10-м) раунде операция MixColumns пропускается.

Эти операции вместе обеспечивают высокую степень запутывания и рассеивания данных, делая шифротекст сложно обратимым без знания ключа.

1.1.4 Уязвимости

Хотя AES-128 считается очень безопасным, он имеет некоторые потенциальные уязвимости:

1. **Брутфорс-атаки**: Теоретически возможны, но из-за огромного пространства ключей (2^128 вариантов) практически неосуществимы с современными технологиями.

2. **Побочные атаки (side-channel attacks)**: Включают анализ времени выполнения, потребляемой мощности или электромагнитного излучения устройства для получения ключа.

3. **Квантовые атаки**: Алгоритм Гровера для квантовых компьютеров может теоретически сократить время брутфорс-атаки до 2^64 операций, но это все еще неосуществимо на практике с текущими квантовыми технологиями.

1.1.5 Применение в современности

AES-128 широко используется в различных областях современной технологии для обеспечения безопасности данных. Вот несколько примеров его применения:

1. Интернет-безопасность:

- HTTPS: AES-128 применяется в протоколе TLS (Transport Layer Security), который защищает данные при их передаче по Интернету, обеспечивая конфиденциальность и целостность данных на веб-сайтах с HTTPS.

2. Мобильные коммуникации:

- Wi-Fi: В современных протоколах безопасности Wi-Fi, таких как WPA2 и WPA3, используется AES-128 для шифрования трафика, что защищает беспроводные сети от несанкционированного доступа.

3. Защита данных в облачных сервисах:

- Облачные хранилища: Сервисы, такие как Google Drive, Dropbox и Microsoft OneDrive, используют AES-128 для шифрования данных, хранящихся на их серверах, чтобы гарантировать безопасность пользовательской информации.

4. Шифрование на уровне устройств:

- Смартфоны и планшеты: Операционные системы, такие как iOS и Android, применяют AES-128 для шифрования данных на устройстве, что защищает данные в случае потери или кражи устройства.

5. Банковские и платежные системы:

- Защита транзакций\*\*: Платежные системы, такие как EMV-чипы в банковских картах, используют AES-128 для обеспечения безопасности транзакций и предотвращения мошенничества.

6. VPN-сервисы:

- Виртуальные частные сети (VPN): Многие VPN-сервисы применяют AES-128 для шифрования трафика между устройством пользователя и VPN-сервером, что защищает данные от перехвата в открытых сетях.

Пример: Использование AES-128 в WhatsApp

WhatsApp использует протокол Signal, который включает шифрование на основе AES-128 для защиты сообщений, звонков и файлов, передаваемых между пользователями. Это обеспечивает высокую степень конфиденциальности и безопасности, так как сообщения шифруются на устройстве отправителя и расшифровываются только на устройстве получателя, исключая возможность перехвата и прочтения сообщений третьими лицами.

Эти примеры демонстрируют, как AES-128 обеспечивает безопасность данных в различных аспектах нашей цифровой жизни..

1.1.6 Заключение

AES-128 (Advanced Encryption Standard с ключом длиной 128 бит) — один из наиболее популярных и надежных алгоритмов симметричного шифрования, принятый в качестве стандарта в 2001 году. Он заменил устаревший DES и обеспечивает высокую степень безопасности через 10 раундов криптографических преобразований.

Несмотря на свою стойкость, AES-128 уязвим для побочных атак, таких как анализ электромагнитного излучения, и потенциальных квантовых атак, что требует дополнительных мер защиты.

В современной практике AES-128 широко применяется для обеспечения безопасности в интернет-соединениях (HTTPS), беспроводных сетях (Wi-Fi), облачных хранилищах, мобильных устройствах, банковских транзакциях и VPN-сервисах. Например, WhatsApp использует AES-128 для шифрования сообщений, обеспечивая их конфиденциальность.

Таким образом, AES-128 остается критически важным инструментом для защиты данных в цифровом мире. .

1.2 Алгоритм распределения ключей Хьюза.

1.2.1 Введение.

Алгоритм распределения ключей Хьюза представляет собой улучшенный вариант классического алгоритма Диффи-Хеллмана, позволяющий абоненту А заранее генерировать ключ и передавать его абоненту В. Это улучшение обеспечивает дополнительную гибкость и эффективность в сценариях, где требуется предварительное шифрование сообщений до установления соединения между абонентами.

1.2.2 Исторический контекст и сущность алгоритма.

#### **Исторический контекст**

Алгоритм распределения ключей Хьюза возник как улучшение классического алгоритма Диффи-Хеллмана, предложенного Уитфилдом Диффи и Мартином Хеллманом в 1976 году. Диффи-Хеллман стал революцией в криптографии, так как впервые позволил безопасно обмениваться криптографическими ключами по незащищенному каналу. Однако алгоритм имел ограничения, особенно в сценариях, требующих предварительного шифрования сообщений. В ответ на эти вызовы был разработан алгоритм Хьюза, предоставляющий более гибкий способ распределения ключей.

**Сущность алгоритма Хьюза**

Алгоритм Хьюза позволяет абоненту А заранее генерировать закрытый ключ и передавать его абоненту В.

1.2.3 Принцип работы алгоритма.

Этот вариант алгоритма Диффи - Хеллмана позволяет абоненту А генерировать ключ и послать его абоненту В.

Абонент А выбирает случайное большое целое число х и генерирует закрытый ключ - k = g^x mod n.

Абонент В выбирает случайное большое целое число у и посылает абоненту А - Y = g^y mod n.

Абонент А посылает абоненту В - X = Y^x mod n.

Абонент В вычисляет - z = y^-1 mod (n - 1);

k' = X^z mod n.

Если все процедуры выполнены, то закрытый ключ k = k'.

1.2.4 Современное состояние и анализ уязвимостей.

Как и многие классические методы шифрования, шифр зафиксированной перестановки имеет свои уязвимости. Основная проблема заключается в том, что метод предполагает фиксированный шаблон перестановки, что может быть использовано для дешифровки текста при помощи анализа паттернов и частотного анализа. Современные криптографические исследования показывают, что для повышения стойкости шифра необходимо вводить дополнительные сложности, такие как случайное изменение шаблона или использование более сложных схем перестановок.

1.2.5 Преимущества и недостатки.

Преимуществом этого протокола распределения ключей над алгоритмом Диффи — Хеллманом состоит в том, что закрытый ключ k можно вычислить заранее, до взаимодействия, и абонент А может шифровать сообщения с помощью закрытого ключа k до установления соединения абонента А с абонентом В. Абонент А может послать сообщение сразу множеству абонентов, а закрытый ключ передать позднее каждому по отдельности.

1.2.6 Современные применения и модификации.

Алгоритм распределения ключей Хьюза находит применение в современных технологиях, требующих безопасного обмена ключами и защиты данных:

1. Криптографические протоколы: Используется в протоколах обеспечения безопасности, таких как TLS/SSL, SSH для установки защищенных соединений в сетях.

2. Блокчейн технологии: Применяется для обеспечения безопасного обмена данными и транзакциями в децентрализованных системах, таких как Bitcoin и Ethereum.

3. Интернет вещей (IoT): Обеспечивает безопасность передачи данных между устройствами IoT, защищая от несанкционированного доступа и манипуляций.

4. Защищенные коммуникации: Используется для шифрования и защиты конфиденциальных данных в мессенджерах, электронной почте и других системах обмена сообщениями.

Алгоритм Хьюза обеспечивает надежный механизм для установления общего ключа между сторонами, что делает его важным инструментом в сфере кибербезопасности и защиты данных в современном цифровом мире.

1.3 Алгоритм работы шифра «Тарабарская грамота»

1.3.1 Введение.

Тарабарская грамота — это метод скрытия текста путем замены букв на другие буквы по заранее определенному шаблону. Этот метод шифрования, известный со времен Древней Руси, использовался для передачи скрытых сообщений. Несмотря на свою простоту, тарабарская грамота привлекает внимание благодаря своей исторической значимости и легкости использования.

1.3.2 Принцип работы.

Принцип тарабарской грамоты основан на замене каждой буквы в тексте на другую букву согласно заранее определенным правилам. Основные шаги включают:

* Создание шаблона замены: Определение набора замен для каждой буквы алфавита.
* Шифрование текста: Замена каждой буквы в исходном тексте на соответствующий символ из шаблона.
* Расшифровка текста : Обратная замена символов на исходные буквы согласно тому же шаблону.

1.3.3 Исторический контекст.

Тарабарская грамота имеет древние корни и использовалась для передачи скрытых сообщений в различных культурных средах. Этот метод шифрования стал известен благодаря своей простоте и эффективности для скрытия информации.

1.3.4 Преимущества и недостатки.

Преимущества Тарабарской грамоты включают её простоту и доступность, что делает её привлекательной для обучения основам шифрования и создания простых кодов. Этот метод также имеет историческую значимость и широко используется в развлекательных задачах и играх. Однако Тарабарская грамота имеет и недостатки : уязвимость к частотному анализу, шаблон замены можно легко обнаружить. Для повышения безопасности рекомендуется комбинировать Тарабарскую грамоту с другими методами шифрования или использовать более современные криптографические алгоритмы.

1.3.5 Применение

Тарабарская грамота используется в различных областях, включая обучение, развлекательные игры и создание простых шифров. В современном мире она применяется для обучения основам шифрования, создания тайных кодов и разработки головоломок. Несмотря на свою простоту, тарабарская грамота остается популярной из-за своей легкости использования и исторической значимости.

2. 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

2.2.1 Программа «AIRShifr» предназначена для автоматизации шифрования и расшифрования исходных данных авторизованного в системе пользователя.

2.2.2 Программа используется пользователем для защиты персональной информации.

2.2.3 Периодичность решения задачи по запросу пользователя.

2.2.4 Прекращение автоматизированного решения задачи происходит при отключении источника электропитания ЭВМ.

2.2.5 Связь с другими задачами отсутствует.

2.2.6 Специальных ограничений на временные характеристики решения задачи не налагается.

2.2.7 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. Запустите приложение «AIRShifr»;

2. Вывод: «Введите пароль: »;

3. Ввод пароля;

4. Вывод меню:

«1. AES128»

«2. Тарабарская грамота»

«3. Hughes»

«4. Выход»

5. Если выбран пункт – «1. AES128»:

«1. Закодировать»

«2. Декодировать»

«3. Назад»

5.1.1 Вывод: «Введите ключдля шифрования (длиной не больше 16 символов): »;

5.1.2 Ввод текста;

5.1.3 Вывод: «Введите текст для шифрования: »;

5.1.4 Ввод текста;

5.1.5 Вывод расшифрованного текста в консоль;

5.1.6 Запись расшифрованного текста в файлt;

5.1.7 Возвращаемся на п. 4;

5.2.1 Переход на п. 4;

6. Если выбран пункт – «2. Тарабарская грамота»:

«Выберете язык:»

«1. Закодировать»

«2. Декодировать»

«3. Назад»

6.1 Выбор языка(не влияет на дальнейший функционал);

6.1.1 Вывод: «Введите текст для шифрования: »;

6.1.2 Ввод текста;

6.1.3 Вывод зашифрованного текста в консоль;

6.1.4 Запись зашифрованного текста в файл encryptedGramota.txt;

6.1.5 Вывод расшифрованного текста в консоль;

6.1.6 Запись расшифрованного текста в файл decryptGramota.txt;

6.1.7 Возвращаемся на п. 4;

6.2.1 Переход на п. 4;

7.Если выбран пункт – «3. Алгоритм распределения ключей Хьюза»:

«1. Закодировать»

«2. Декодировать»

«3. Назад»

7.1 Отрабатывает алгоритм омена ключами.

7.2 Выводит зашифрованное сообщение

7.3 Выводит расшифрованное соощение.

8. Если выбран пункт – «4. Выход»;

8.1 Вывод: «Работа программы завершена»;

8.2 Программа завершает свою работу;

Этот алгоритм обеспечивает простое и интуитивно понятное взаимодействие с приложением для выполнения шифрования и дешифрования текста.

2.4 Руководство пользователя

2.4.1 Введение

2.4.1.1 Программа «AIRShifr» предназначена для автоматизации шифрования и расшифрования исходных данных авторизованного в системе пользователя.

2.4.1.2 Программа предоставляет пользователю следующие возможности:

– полный доступ в систему;

– выбор алгоритма шифрования;

– выбор шифрования из консоли;

– шифрование/расшифрование данных;

– вывод результатов;

– запись зашифрованного и расшифрованного текста в выходной файл;

2.4.1.3 Программа реализована на языке C++. Работает в любой среде совместимой с «Linux», на которой есть компилятор g++ под нужную архитектуру процессора. Дисковой памяти для запуска программы требуется не менее 300 Mb (компилятор g++ + библиотеки + статически линкованная программа). Оперативной памяти для нормальной работы программы требуется не менее 16 Mb.

2.4.2 Описание операций

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 программы на экран выводится текст с запросом пароля. Консольный вывод приведён на рисунке 2.1. Для начала работы с программой необходимо ввести правильный пароль (в данном случае «1423»)

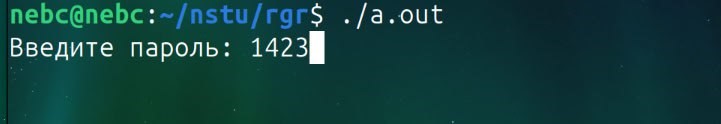


Рисунок 2.1 – Главное окно программы

В случае правильного ввода пароля выводится меню программы (рисунок 2.2), в случае неправильного — сообщение о ошибке (рисунок 2.3).

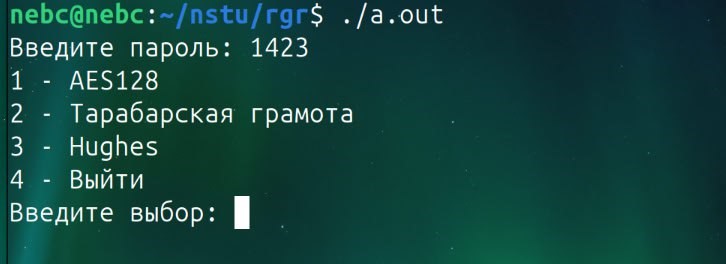


Рисунок 2.2 – Вывод меню и выбор пункта

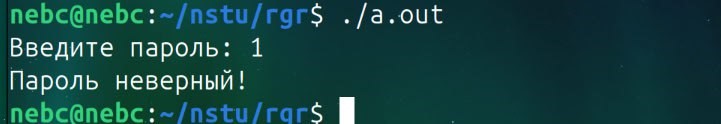


Рисунок 2.3 – Ошибка «Неверный пароль»

Далее программа предлагает пользователю на выбор один из трёх алгоритмов шифрования (рисунок 2.2).

При выборе «AES128» начинается следующий диалог с пользователем (рисунок 2.4). Здесь нужно выбрать закодировать или декодировать сообщение.



Рисунок 2.4 – Вывод AES128

У шифра «Тарабарская грамота» диалог с пользователем такой же (рисунок 2.5). Характерной особенностью этого шифра является то, что он шифрует только буквы.

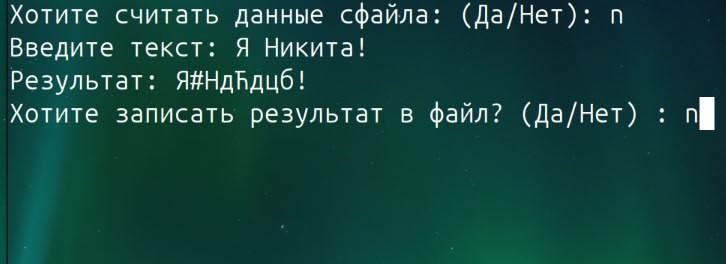


Рисунок 2.5 – Вывод Тарабарская грамота

При выборе «Шифр зафиксированной перестановкой» начинается следующий диалог с пользователем (рисунок 2.6). Здесь нужно ввести сообщение, которое мы хотим зашифровать как пример чтобы посмотреть как происходит обмен закрытого ключа и открытых ключей. Для этого два абонента должны договорится о входных данных чисел a и p.

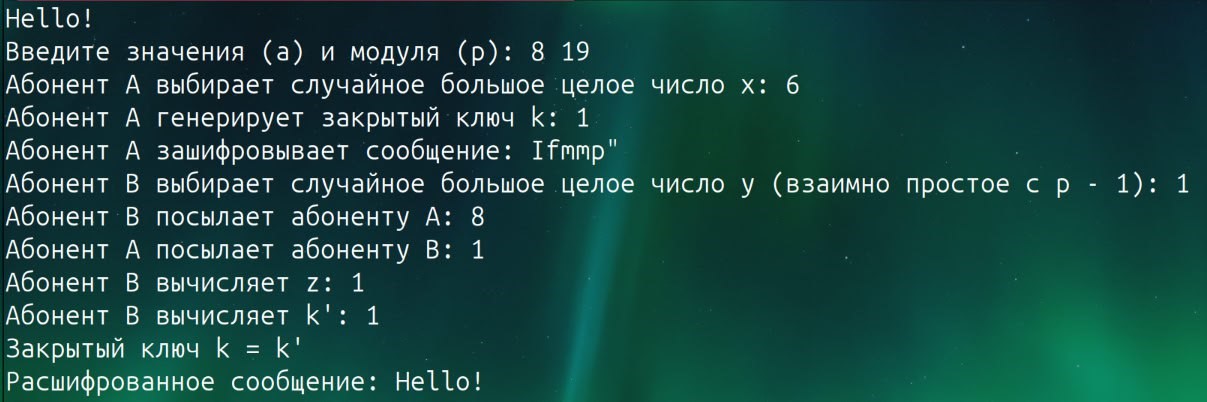


Рисунок 2.6 – Вывод алгоритма распределения ключей Хьюза

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 |
| Ошибка : Пароль неверный! | Выводится при неверном вводе пароля. |
| Ошибка: Неверный выбор. Пожалуйста, введите число от 1 до 4: | Выводится при неверном выборе команды в главном меню или в меню при работе с шифром. |
| Ошибка: Неверный выбор. Пожалуйста, введите число от 1 до 3: | Выводится при не правильно вводе закодировать, декодировать или выйти в меню. |
| Ошибка : Не удалось открыть файл для записи. | Выводится при невозможности открыть файл для записи зашифрованного или |
| Ошибка : Не удалось открыть файл для чтения. | Выводится при невозможности открыть файл для считывания текста при дешифровании. |
| Ошибка : Ключ длиной ольше чем 16 символов. | Выводит при вводе ключа больше чем 16 символов при выборе AES128. |
| Ошибка: Неверный выбор. Пожалуйста, введите число от 1 до 4: | Выводится при неправильно выборе шифра. |

2.4.4 Аварийные ситуации.

К аварийным ситуациям относятся: нехватка оперативной памяти для запуска. Если исполняемый модуль программы не запускается, либо не выполнены требования условия работы программы, необходимо обратиться к разработчику программы.

2.5 Руководство системного программиста

Программа реализована на языке С++ в среде Visual Studio Code.

Модули программы:

Tishn.cpp – файл является точкой входа для запуска приложения. Вызывается функция main, которая в свою очередь проверяет пароль на корректность, далее реализует выбор в меню;

hughes.h

std::pair<int64\_t, int64\_t> init(); // инициализация простого числа p и любого целого числа a;

bool isPrime(int64\_t n); // проверка числа на простоту

int64\_t rn(int64\_t p); // функция для генерации рандомного числа

int64\_t gcd(int64\_t a, int64\_t b); // нахождение НОД

int64\_t f(int64\_t x, int64\_t y, int64\_t n, int64\_t r); // функция для вызова быстрого возведения в степень

int64\_t powMod(int64\_t x, int64\_t y, int64\_t n); // быстрое возведение в степень

int64\_t extendedEuclidean(int64\_t a, int64\_t b, int64\_t& x, int64\_t& y); // расширенный алгоритм Евклида.

int64\_t findModularInverse(int64\_t a, int64\_t m); // нахождение обратного элемента по модулю

std::string caesar\_cipher(const std::string& message, int64\_t k); // шифр цезаря

std::string caesar\_decipher(const std::string& message, int64\_t k\_prime);

int64\_t huges();

littera.h —

extern map<wchar\_t, wchar\_t> ru\_encrypt; // алфавит русский

extern map<char, char> ru\_alphabet; // алфавит английский

void littera(); // функция с кодированием и декодированием

common\_aes.h

pair<string, string> init\_encrypted(); // инициализация ключа и текста

pair<string, string> init\_decrypt(); // инициализация ключа и текста

pair<vector<vector<unsigned char>>, vector<vector<vector<unsigned char>>>> create\_matrices(const string& input\_message, const string& input\_key); // создание матрицы

void transpose\_matrix(vector<vector<unsigned char>>& matrix); // транспонирование матрицы

void key\_expansion(vector<vector<unsigned char>>& key\_matrix, vector<vector<vector<unsigned char>>>& extended\_keys); // расширение ключа

vector<vector<unsigned char>> xor\_matrices(const vector<vector<unsigned char>>& matrix1, const vector<vector<unsigned char>>& matrix2); // xor матриц

void print\_matrix\_hex(const vector<vector<unsigned char>>& matrix); // вывод матрицы в 16 ричном виде.

aes128\_decrypt.h

void invSubBytes(vector<vector<unsigned char>>& matrix); // замена байт

void invShiftRows(vector<vector<unsigned char>>& state); // сдвиг битов

unsigned char gmul(unsigned char a, unsigned char b); // g поле

void invMixColumns(vector<vector<unsigned char>>& state); // умножени в побитовом поле

void aes\_decrypt(); // расшифрование

aes128\_encrypted.h

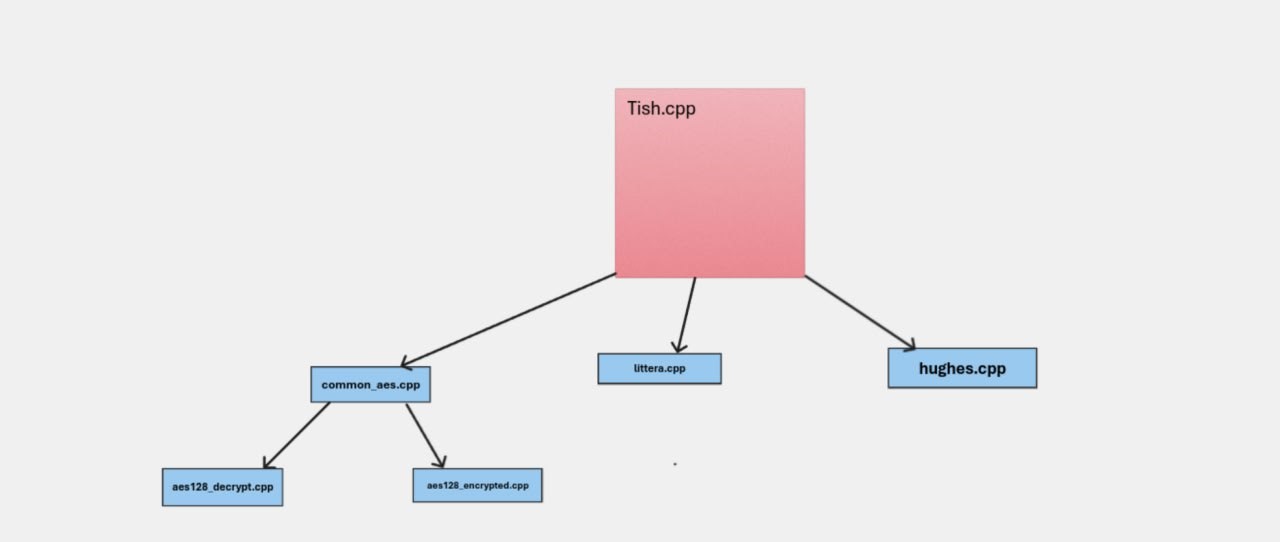
void sub\_bytes(vector<vector<unsigned char>>& matrix); // замена байтов из sBox

void shiftRows(vector<vector<unsigned char>>& state); // сдвиг байтов

void mixColumns(vector<vector<unsigned char>>& state); // умножение в побитовом поле

void wride\_ciphertext(vector<unsigned char> ciphertext); запись зашифрованного сообщени в файл

void aes\_encrypted(); // зашшифрование



Связь модулей программы между собой представлена на рисунке 2.7.

Рисунок 2.7 – Связь модулей программы

2.6 Контрольный пример

После запуска на экран выводится сообщение о просьбе ввести пароль (рисунок 2.1).

После ввода корректного пароля пользователь попадает в меню выбора способа шифрования текста (рисунок 2.2).

Если пользователь выберет шифр Тарабарская грамота, программа попросит выбрать закодировать или декодировать текст. После этих действий зашифрованный и расшифрованный текст будет выведен в консоль и каждый из текстов записан в свой файл (рисунок 2.8).

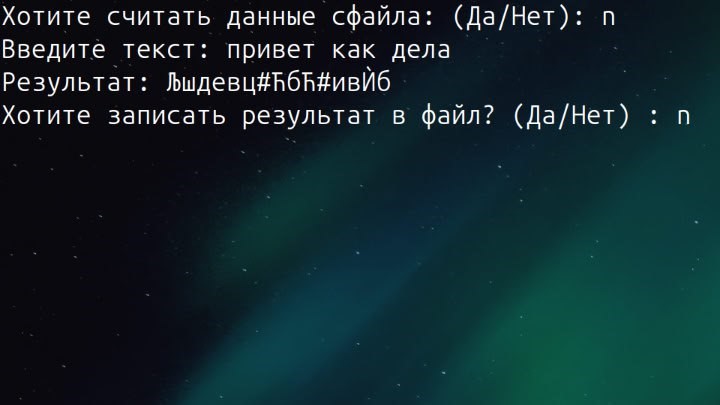


Рисунок 2.8 – Работа с шифром Тарабарская грамота.

Если пользователь выберет шифр «AES128», программа попросит выбрать закодировать или декодировать текст. После этих действий зашифрованный и расшифрованный текст будет выведен в консоль и каждый из текстов записан в свой файл (рисунок 2.9).

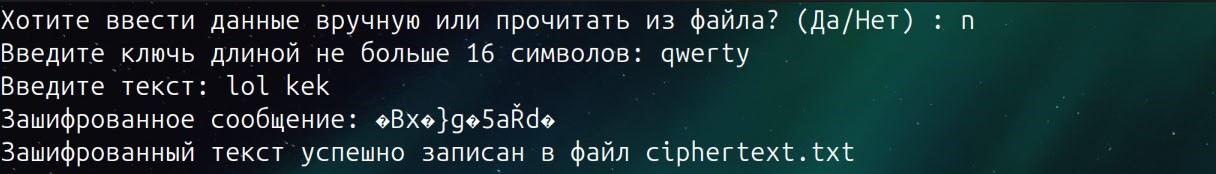


Рисунок 2.9 – Работа с шифром AES128

2.7 Ручной расчёт

2.7.1 AES128

У нас сообщение «Two One Nine Two».

Ч а т с м и к и н г Фу 54 68 61 74 73206D 79 20 4B 75 6E 67 2046 75

Шестнадцатеричный ключ (128 бит): 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75

Открытый текст на английском языке: Two One Nine Two (16 символов ASCII, по 1 байту каждый)

Перевод в шестнадцатеричный формат:

В обе стороны от 54 77 6F 20 4F 6E 65 204E 69 6E 65 20 54 77 6F

Обычный текст в шестнадцатеричном формате (128 бит): 54 77 6F 20 4F 6E 65 20 4E 69 6E 65 20 54 77 6F

Пример AES - Первая круглая клавиша

Шестнадцатеричный ключ (128 бит): 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75 вт[0] = (54, 68, 61, 74), w[1] = (73, 20, 6D, 79), w[2] = (20, 4B, 75, 6E), w[3] = (67, 20, 46, 75)

g(w[3]):

циклический сдвиг w влево на байт[3]: (20, 46, 75, 67)

Подстановка байтов (S-Box): (B7,5A,9D,85)

Добавление круглой константы (01, 00,00, 00) дает: g(w[3]) = (B6,5A, 9D, 85)

w[4] = w[0] @ g(w[3]) = (E2,32, FC, F1):

w[5] = w[4] @ w[1] = (91, 12, 91,88), w[6] = w[5] @ w[2] = (B1, 59, E4, E6), w[7] = w[6] @ w[3] = (D6, 79, A2,93)

ключ первого раунда: E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93

Пример AES - Добавляем круглую клавишу, округляем 0

Матрицу состояний и круглую клавишу №0:

(54 4F4E 20

77 6E 69 54

6F 65 6E 77

20 20 65 6F

(54 73 20 67 68 20 4B 20 616D 75 46 74 79 6E 75

ИСКЛЮЧИТЕ соответствующие записи, например, 69 @ 4B = 22

0110 1001

0100 1011

0010 0010

новая матрица состояний имеет вид

(00 3C 6E 47 1F4E 22 74 0E 08 1B 31 54 59 0B1A

Пример - Раунд 1, байты подстановки

•Матрица текущего состояния равна

(00 3C6E 47 1F4E 22 74 0E 08 1B 31 54 59 0B1A

• замените каждую запись (байт) текущей матрицы состояний соответствующей записью в S-поле AES

• например: байт 6E заменяется записью S-поля в строке 6 и столбце E, т.е. на 9F

•

это приводит к новой матрице состояний

63 EB 9F A0 C0 2F 93 92

AB 30 AFC7

20 CB 2B A2

• этот нелинейный уровень предназначен для защиты от атак дифференциального и линейного криптоанализа

Пример AES - Раунд 1, сдвиг строки

• текущая матрица состояний

63 EB 9F A0 C0 2F 93 92 AB 30 AFC7 20 CB 2B A2

• четыре строки циклически сдвигаются влево на смещения 0,1,2 и 3 • новая матрица состояний имеет вид

63 EB 9F A0 2F 93 92 C0 AF C7 AB 30 A2 20 CB2B)

этот этап линейного перемешивания приводит к распределению битов по нескольким кругам

Пример AES - Раунд 1, столбец Mix

Столбец Mix умножает фиксированную матрицу на матрицу текущего состояния:

(02030101\ 63 EB 9F A0 01020301 2F 93 92 C0 01010203 AF C7 AB 30 03010102 A2 20 CB2B

(BA 84 E81B

75 A48D 40

F48D 06 7D

7A 32 0E5D

запись BA является результатом (02 • 63) @ (03• 2F) @ (01 • AF) @ (01 • A2):

• 02 63 = 00000010 01100011 = 11000110

• 03 2F = (02 2F) @ 2F = (00000010• 00101111) 00101111 = 01110001

• 01 AF = AF = 10101111 и 01 • A2 = A2 = 10100010

следовательно

11000110 01110001 10101111 10100010 10111010

Пример AES - Добавляем круглую клавишу, раунд 1

Матрицу состояний и круглую клавишу №1:

(BA 84 E81B

75 A48D 40

F48D 06 7D

7A 32 0E5D

(E2 91 B1 D6) 32 12 59 79 FC91 E4 A2 F1 88 E6 93

• XOR возвращает новую матрицу состояний

58 15 59 CD

47 B6 D4 39

08 1C E2 DF

8BBAE8CE

Результат AES после первого раунда: 58 47 08 8B 15 B6 1C BA 59 D4 E2 E8 CD 39 DF CE

2.7.2 Шифр Тарабарская грамота

Таблица для замены букв в алфавите представлена на рисунке 2.12.

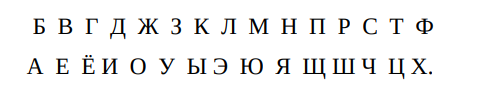


Рисунок 2.12 – Таблица замен букв в Тарабарской грамоте

У нас текст «Я студент».

Каждую букву сообщения меняем на букву по таблице:

* Буква 'Я' меняем на → «Н»;
* Пробел меняем на → «#»;
* Буква 'с' меняем на → «ч»;
* Буква 'т' меняем на → «ц»;
* Буква 'у' меняем на → «з»;
* Буква 'д' меняем на → «и»;
* Буква 'е' меняем на → «в»;
* Буква 'н' меняем на → «я»;
* Буква 'т' меняем на → «ц»;

Расшифровка будет такая же, но в обратную сторону.

Результат точно такой же как в контрольном примере.

2.7.3 Шифр зафиксированной перестановкой

У нас сообщение «Hello, bro».

Для начала нам нужно выбрать два целых числа a и p таких что а любое целое число и a < p, p в свою очередь простое число и p меньше чем 2^1024, потому что потом математически алгоритм восстановить нельзя.

Пусть a = 11; p = 17;

Абонент A выбирает случайное число x = 6;

Генерирует закрытый ключ k = 8;

По закрытому ключу 8 сдвинем каждый элемент сообщения на восемь символовм по ascii. Получим сообщение: Pmttw4(Jzw

Абонент B генерирует случайное число y (взаимно простое с p – 1) = 9;

Абонент B посылает абоненту А: 6

Абонент А посылает абоненту B: 8

Абонент B вычисляет z: 9

Абонент B вычисляет k': 8

Закрытый ключ k = k'

Расшифрованное сообщение: Hello, Bro

СПИСКОВ ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Стивен Прата. Язык программирования C++ (C++11). Лекции и упражнения, 6-е издание — М.: Вильямс, 2012. — 1248 с.
2. Павловская, Т. А. C/C++. Программирование на языке высокого уровня: [для вузов] / Т.А.Павловская. - СПб. [и др.]: Питер, 2010. - 460с.
3. Страуструп, Б. Программирование: принципы и практика использования С++, испр. Изд. / Б. Страуструп. – М. : ООО «И. Д. Вильямс», 2011. – 1248 с.
4. Страуструп, Б. Язык программирования С++. Специальное издание/ Б. Страуструп. – М. : Издательство Бином, 2011. – 1248 с.
5. Мартин Р. Чистый код. Создание, анализ и рефакторинг. Библиотека программиста. — СПб.: Питер, 2014. — 464 с.
6. Андрей Александреску. Современное проектирование на C++. Обобщенное программирование и прикладные шаблоны проектирования. Перевод с английского — Издательский дом «Вильямс», 2002 г. - 336 с.
7. Мейерс, С. Эффективное использование STL. Библиотека программиста / С. Мейерс. СПб.: Питер, 2002. — 224 с.
8. Мейерс С. Эффективное использование С++. 35 новых рекомендаций по улучшению ваших программ и проектов. — М.: ДМК Пресс, 2014. — 294с.
9. Бухманн Й. Введение в криптографию . - 2 изд. - Шпрингер, 2004. - 338 с.
10. Рябко, Б. Я. Основы современной криптографии и стеганографии [Текст] : монография / Б. Я. Рябко, А. Н. Фионов. - Москва : Горячая линия-Телеком, 2010. - 232 с.

ПРИЛОЖЕНИЕ

**Абс-322.Тишин.РГР.cpp**

#include <iostream>

#include <limits>

#include <stdexcept>

#include "AES/include/aes128\_encrypted.h"

#include "AES/include/aes128\_decrypt.h"

#include "Littera/include/littera.h"

#include "Hughes/include/hughes.h"

using namespace std;

#define password "1423"

bool login() {

string input\_password;

try {

cout << "Введите пароль: ";

getline(cin, input\_password);

if (input\_password == password) {

return true;

} else {

throw runtime\_error("Пароль неверный!");

}

} catch (const runtime\_error& e) {

cout << e.what() << endl;

return false;

} catch (const exception& e) {

cout << "Ошибка: " << e.what() << endl;

return false;

}

}

int getChoice(int min, int max) {

int choice;

cout << "Введите выбор: ";

while (!(cin >> choice) || choice < min || choice > max) {

cout << "Ошибка: Неверный выбор. Пожалуйста, введите число от " << min << " до " << max << ": ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

return choice;

}

void aes\_menu() {

try {

cout << "1 - Закодировать" << endl;

cout << "2 - Расскодировать" << endl;

cout << "3 - Выйти" << endl;

int choice = getChoice(1, 3);

printf("\033[2J");

printf("\033[0;0f");

switch (choice) {

case 1:

aes\_encrypted();

break;

case 2:

aes\_decrypt();

break;

case 3:

return;

default:

throw runtime\_error("Неверный выбор!");

}

} catch (const runtime\_error& e) {

cout << e.what() << endl;

} catch (const exception& e) {

cout << "Ошибка: " << e.what() << endl;

}

}

void littera\_menu() {

try {

cout << "1 - Закодировать" << endl;

cout << "2 - Расскодировать" << endl;

cout << "3 - Выйти" << endl;

int choice = getChoice(1, 3);

printf("\033[2J");

printf("\033[0;0f");

switch (choice) {

case 1:

case 2:

littera();

break;

case 3:

return;

default:

throw runtime\_error("Неверный выбор!");

}

} catch (const runtime\_error& e) {

cout << e.what() << endl;

} catch (const exception& e) {

cout << "Ошибка: " << e.what() << endl;

}

}

void hughes\_menu() {

try {

cout << "1 - Закодировать" << endl;

cout << "2 - Расскодировать" << endl;

cout << "3 - Выйти" << endl;

int choice = getChoice(1, 3);

printf("\033[2J");

printf("\033[0;0f");

switch (choice) {

case 1:

case 2:

huges();

break;

case 3:

return;

default:

throw runtime\_error("Неверный выбор!");

}

} catch (const runtime\_error& e) {

cout << e.what() << endl;

} catch (const exception& e) {

cout << "Ошибка: " << e.what() << endl;

}

}

void menu() {

try {

while (true) {

cout << "1 - AES128" << endl;

cout << "2 - Тарабарская грамота" << endl;

cout << "3 - Hughes" << endl;

cout << "4 - Выйти" << endl;

int choice = getChoice(1, 4);

printf("\033[2J");

printf("\033[0;0f");

switch (choice) {

case 1:

aes\_menu();

break;

case 2:

littera\_menu();

break;

case 3:

hughes\_menu();

break;

case 4:

return;

default:

throw runtime\_error("Неверный выбор!");

}

}

} catch (const runtime\_error& e) {

cout << e.what() << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

} catch (const exception& e) {

cout << "Ошибка: " << e.what() << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

}

int main() {

try {

bool log = login();

if (log) {

menu();

}

} catch (const exception& e) {

cout << "Ошибка в основной функции: " << e.what() << endl;

}

return 0;

}

**aes128\_common.cpp**

#include "../include/box.h"

#include <algorithm>

#include <string>

#include <iostream>

#include <vector>

#include <fstream>

#include <algorithm>

#include <cctype>

using namespace std;

pair<string, string> init\_encrypted() {

string choice;

string key, text;

while(true) {

cout << "Хотите ввести данные вручную или прочитать из файла? (Да/Нет) : ";

getline(cin, choice);

transform(choice.begin(), choice.end(), choice.begin(),

[](unsigned char c) {return tolower(c); });

a

if(choice == "да" || choice == "yes" || choice == "д" || choice == "y"){

string key\_file\_path, text\_file\_path;

cout << "Введите путь к файлу с ключом: ";

getline(cin, key\_file\_path);

cout << "Введите путь к файлу с текстом: ";

getline(cin, text\_file\_path);

ifstream key\_file(key\_file\_path);

ifstream text\_file(text\_file\_path);

while ((!key\_file.is\_open() || !text\_file.is\_open()))

{

cerr << "Ошибка открытия файла. Пожалуйста, проверьте пути и повторите попытку." << endl;

cout << "Путь к файлу с ключом:";

getline(cin, key\_file\_path);

cout << "Путь к файлу с текстом: ";

getline(cin, text\_file\_path);

ifstream key\_file(key\_file\_path);

ifstream text\_file(text\_file\_path);

}

getline(key\_file, key);

getline(text\_file, text);

key\_file.close();

text\_file.close();

if (key.length() > 16) {

cerr << "Длина ключа больше чем 16 символов. Пожалуйста, проверьте содержимое файла с ключом." << endl;

exit(EXIT\_FAILURE);

}

else {

break;

}

}

else {

while (true) {

cout << "Введите ключь длиной не больше 16 символов: ";

getline(cin, key);

if (key.length() > 16) {

cout << "Длина ключа больше чем 16 символов. \nВведите ключ: ";

getline(cin, key);

} else {

break;

}

}

cout << "Введите текст: ";

getline(cin, text);

break;

}

}

return {key, text};

}

pair<string, string> init\_decrypt() {

string input\_key, ciphertext;

cout << "Введите ключ длиной не больше 16 символов: ";

while (true) {

getline(cin, input\_key);

if (input\_key.length() > 16) {

cout << "Длина ключа больше чем 16 символов. \nВведите ключ: ";

}

else {

break;

}

}

ifstream inputFile("ciphertext.txt", ios::binary);

if (inputFile.is\_open()) {

unsigned char ch;

while (inputFile.read(reinterpret\_cast<char\*>(&ch), sizeof(ch))) {

ciphertext.push\_back(ch);

}

inputFile.close();

}

else {

cout << "Не удалось открыть файл для чтения" << endl;

}

return {input\_key, ciphertext};

}

// Функция для создания матриц из ключа и сообщения

pair<vector<vector<unsigned char>>, vector<vector<vector<unsigned char>>>> create\_matrices(const string& input\_message, const string& input\_key) {

vector<vector<vector<unsigned char>>> message;

vector<vector<unsigned char>> matrix(4, vector<unsigned char>(4, 0));

size\_t position = 0;

// Создание матриц из сообщения

while (position < input\_message.length()) {

for (int i = 0; i < 4 && position < input\_message.size(); ++i) {

for (int j = 0; j < 4 && position < input\_message.size(); ++j) {

matrix[i][j] = input\_message[position++];

}

}

message.push\_back(matrix);

}

// Создание матрицы из ключа

vector<vector<unsigned char>> key\_matrix(4, vector<unsigned char>(4, 0));

for (int i = 0; i < 4; ++i) {

for (int j = 0; j < 4; ++j) {

int index = i \* 4 + j;

if (index < input\_key.length()) {

key\_matrix[i][j] = input\_key[index];

}

}

}

return {key\_matrix, message};

}

void transpose\_matrix(vector<vector<unsigned char>>& matrix) {

int n = matrix.size();

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

for (int j = i + 1; j < n; ++j) {

swap(matrix[i][j], matrix[j][i]);

}

}

}

void key\_expansion(vector<vector<unsigned char>>& key\_matrix, vector<vector<vector<unsigned char>>>& extended\_keys) {

vector<vector<unsigned char>> exp\_key;

vector<vector<unsigned char>> state;

for (const auto& row : key\_matrix) {

exp\_key.push\_back(row);

state.push\_back(row);

}

for (int j = 0; j < 10; j++) {

rotate(state[3].begin(), state[3].begin() + 1, state[3].end());

for (int i = 0; i < 4; i++) {

state[3][i] = sBox[state[3][i] / 16][state[3][i] % 16];

state[3][i] = state[3][i] ^ r\_w[j][i];

exp\_key[0][i] = exp\_key[0][i] ^ state[3][i];

exp\_key[1][i] = exp\_key[0][i] ^ state[1][i];

exp\_key[2][i] = exp\_key[1][i] ^ state[2][i];

exp\_key[3][i] = exp\_key[2][i] ^ exp\_key[3][i];

}

extended\_keys.push\_back(exp\_key);

state = exp\_key;

}

}

vector<vector<unsigned char>> xor\_matrices(const vector<vector<unsigned char>>& matrix1, const vector<vector<unsigned char>>& matrix2) {

int n = matrix1.size();

vector<vector<unsigned char>> result(n, vector<unsigned char>(n));

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

for (int j = 0; j < n; ++j) {

result[i][j] = matrix1[i][j] ^ matrix2[i][j];

}

}

return result;

}

void print\_matrix\_hex(const vector<vector<unsigned char>>& matrix) {

for (const auto& row : matrix) {

for (unsigned char value : row) {

printf("%02X ", value);

}

cout << "\n";

}

cout << "\n\n";

}

**Aes128\_encrypt.cpp**

#include "../include/aes128\_common.h"

#include "../include/box.h"

#include <iostream>

#include <string>

#include <vector>

#include <algorithm>

#include <fstream>

using namespace std;

void sub\_bytes(vector<vector<unsigned char>>& matrix) {

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

matrix[i][j] = sBox[matrix[i][j] / 16][matrix[i][j] % 16];

}

}

}

void shiftRows(vector<vector<unsigned char>>& state) {

for (int i = 1; i < 4; ++i) {

rotate(state[i].begin(), state[i].begin() + i, state[i].end());

}

}

void mixColumns(vector<vector<unsigned char>>& state) {

for (int i = 0; i < 4; ++i) {

unsigned char a[4];

unsigned char b[4];

unsigned char h;

for (int c = 0; c < 4; ++c) {

a[c] = state[c][i];

h = (unsigned char)((signed char)state[c][i] >> 7);

b[c] = state[c][i] << 1;

b[c] ^= 0x1B & h;

}

state[0][i] = b[0] ^ a[1] ^ b[1] ^ a[2] ^ a[3];

state[1][i] = a[0] ^ b[1] ^ a[2] ^ b[2] ^ a[3];

state[2][i] = a[0] ^ a[1] ^ b[2] ^ a[3] ^ b[3];

state[3][i] = a[0] ^ b[0] ^ a[1] ^ a[2] ^ b[3];

}

}

void wride\_ciphertext(vector<unsigned char> ciphertext) {

ofstream outputFile("ciphertext.txt");

if (outputFile.is\_open()) {

for (unsigned char ch : ciphertext) {

outputFile << ch;

}

outputFile.close();

cout << "Зашифрованный текст успешно записан в файл ciphertext.txt" << endl;

}

}

void aes\_encrypted() {

string input\_key, input\_message;

tie(input\_key, input\_message) = init\_encrypted();

vector<vector<unsigned char>> key\_matrix;

vector<vector<vector<unsigned char>>> message\_matrices;

tie(key\_matrix, message\_matrices) = create\_matrices(input\_message, input\_key);

vector<vector<vector<unsigned char>>> extended\_keys;

key\_expansion(key\_matrix, extended\_keys);

for (int i = 0; i < 10; i++) {

transpose\_matrix(extended\_keys[i]);

}

transpose\_matrix(key\_matrix);

for (int i = 0; i < message\_matrices.size(); i++) {

transpose\_matrix(message\_matrices[i]);

}

vector<unsigned char> ciphertext;

for (int i = 0; i < message\_matrices.size(); i++) {

vector<vector<unsigned char>> result\_matrix = xor\_matrices(key\_matrix, message\_matrices[i]);

for (int j = 0; j < 9; j++) {

sub\_bytes(result\_matrix);

shiftRows(result\_matrix);

mixColumns(result\_matrix);

result\_matrix = xor\_matrices(result\_matrix, extended\_keys[j]);

}

sub\_bytes(result\_matrix);

shiftRows(result\_matrix);

result\_matrix = xor\_matrices(result\_matrix, extended\_keys[9]);

transpose\_matrix(result\_matrix);

for (const auto& row : result\_matrix) {

for (const auto ch : row) {

ciphertext.push\_back(ch);

}

}

}

cout << "Зашифрованное сообщение: ";

for(char ch : ciphertext) {

cout << ch;

}

cout << endl;

wride\_ciphertext(ciphertext);

}

**aes128\_decrypt.cpp**

#include "../include/aes128\_common.h"

#include "../include/box.h"

#include <algorithm>

#include <string>

#include <iostream>

#include <vector>

#include <fstream>

using namespace std;

void invSubBytes(vector<vector<unsigned char>>& matrix) {

for(int i = 0; i < 4; i++) {

for(int j = 0; j < 4; j++) {

matrix[i][j] = invsBox[matrix[i][j] / 16][matrix[i][j] % 16];

}

}

}

void invShiftRows(vector<vector<unsigned char>>& state) {

for (int i = 1; i < 4; ++i) {

rotate(state[i].rbegin(), state[i].rbegin() + i, state[i].rend());

}

}

unsigned char gmul(unsigned char a, unsigned char b) {

unsigned char p = 0;

unsigned char counter;

unsigned char hi\_bit\_set;

for (counter = 0; counter < 8; counter++) {

if (b & 1) {

p ^= a;

}

hi\_bit\_set = (a & 0x80);

a <<= 1;

if (hi\_bit\_set) {

a ^= 0x1b;

}

b >>= 1;

}

return p;

}

void invMixColumns(vector<vector<unsigned char>>& state) {

for (int i = 0; i < 4; ++i) {

unsigned char a[4];

unsigned char b[4];

for (int c = 0; c < 4; ++c) {

a[c] = state[c][i];

}

b[0] = gmul(a[0], 0x0e) ^ gmul(a[1], 0x0b) ^ gmul(a[2], 0x0d) ^ gmul(a[3], 0x09);

b[1] = gmul(a[0], 0x09) ^ gmul(a[1], 0x0e) ^ gmul(a[2], 0x0b) ^ gmul(a[3], 0x0d);

b[2] = gmul(a[0], 0x0d) ^ gmul(a[1], 0x09) ^ gmul(a[2], 0x0e) ^ gmul(a[3], 0x0b);

b[3] = gmul(a[0], 0x0b) ^ gmul(a[1], 0x0d) ^ gmul(a[2], 0x09) ^ gmul(a[3], 0x0e);

for (int c = 0; c < 4; ++c) {

state[c][i] = b[c];

}

}

}

void aes\_decrypt() {

string input\_key, ciphertext;

tie(input\_key, ciphertext) = init\_decrypt();

vector<vector<unsigned char>> key\_matrix;

vector<vector<vector<unsigned char>>> ciphertext\_matrices;

tie(key\_matrix, ciphertext\_matrices) = create\_matrices(ciphertext, input\_key);

vector<vector<vector<unsigned char>>> extended\_keys;

extended\_keys.push\_back(key\_matrix);

key\_expansion(key\_matrix, extended\_keys);

for (int i = 0; i <= 10; i++) {

transpose\_matrix(extended\_keys[i]);

}

transpose\_matrix(key\_matrix);

for (int i = 0; i < ciphertext\_matrices.size(); i++) {

transpose\_matrix(ciphertext\_matrices[i]);

}

vector<unsigned char> plaintext;

for (int i = 0; i < ciphertext\_matrices.size(); i++) {

vector<vector<unsigned char>> result\_matrix = xor\_matrices(extended\_keys[10], ciphertext\_matrices[i]);

for (int j = 9; j > 0; j--) {

invShiftRows(result\_matrix);

invSubBytes(result\_matrix);

result\_matrix = xor\_matrices(result\_matrix, extended\_keys[j]);

invMixColumns(result\_matrix);

}

invShiftRows(result\_matrix);

invSubBytes(result\_matrix);

result\_matrix = xor\_matrices(result\_matrix, extended\_keys[0]);

transpose\_matrix(result\_matrix);

for (const auto& row : result\_matrix) {

for (const auto ch : row) {

plaintext.push\_back(ch);

}

}

}

cout << "Расшифрованное сообщение: ";

for(char ch : plaintext) {

cout << ch;

}

cout << endl;

ofstream outputFile("decrypted\_message.txt");

if (outputFile.is\_open()) {

for (unsigned char ch : plaintext) {

outputFile << ch;

}

outputFile.close();

cout << "Расшифрованный текст успешно записан в файл decrypted\_message.txt" << endl;

}

else {

cout << "Не удалось открыть файл для записи" << endl;

}

}

**box.cpp**

#include "../include/box.h"

const std::vector<std::vector<unsigned char>> sBox =

{/\* 0 1 2 3 4 5 6 7 8 9 a b c d e f \*/

{0x63,0x7c,0x77,0x7b,0xf2,0x6b,0x6f,0xc5,0x30,0x01,0x67,0x2b,0xfe,0xd7,0xab,0x76}, /\*0\*/

{0xca,0x82,0xc9,0x7d,0xfa,0x59,0x47,0xf0,0xad,0xd4,0xa2,0xaf,0x9c,0xa4,0x72,0xc0}, /\*1\*/

{0xb7,0xfd,0x93,0x26,0x36,0x3f,0xf7,0xcc,0x34,0xa5,0xe5,0xf1,0x71,0xd8,0x31,0x15}, /\*2\*/

{0x04,0xc7,0x23,0xc3,0x18,0x96,0x05,0x9a,0x07,0x12,0x80,0xe2,0xeb,0x27,0xb2,0x75}, /\*3\*/

{0x09,0x83,0x2c,0x1a,0x1b,0x6e,0x5a,0xa0,0x52,0x3b,0xd6,0xb3,0x29,0xe3,0x2f,0x84}, /\*4\*/

{0x53,0xd1,0x00,0xed,0x20,0xfc,0xb1,0x5b,0x6a,0xcb,0xbe,0x39,0x4a,0x4c,0x58,0xcf}, /\*5\*/

{0xd0,0xef,0xaa,0xfb,0x43,0x4d,0x33,0x85,0x45,0xf9,0x02,0x7f,0x50,0x3c,0x9f,0xa8}, /\*6\*/

{0x51,0xa3,0x40,0x8f,0x92,0x9d,0x38,0xf5,0xbc,0xb6,0xda,0x21,0x10,0xff,0xf3,0xd2}, /\*7\*/

{0xcd,0x0c,0x13,0xec,0x5f,0x97,0x44,0x17,0xc4,0xa7,0x7e,0x3d,0x64,0x5d,0x19,0x73}, /\*8\*/

{0x60,0x81,0x4f,0xdc,0x22,0x2a,0x90,0x88,0x46,0xee,0xb8,0x14,0xde,0x5e,0x0b,0xdb}, /\*9\*/

{0xe0,0x32,0x3a,0x0a,0x49,0x06,0x24,0x5c,0xc2,0xd3,0xac,0x62,0x91,0x95,0xe4,0x79}, /\*a\*/

{0xe7,0xc8,0x37,0x6d,0x8d,0xd5,0x4e,0xa9,0x6c,0x56,0xf4,0xea,0x65,0x7a,0xae,0x08}, /\*b\*/

{0xba,0x78,0x25,0x2e,0x1c,0xa6,0xb4,0xc6,0xe8,0xdd,0x74,0x1f,0x4b,0xbd,0x8b,0x8a}, /\*c\*/

{0x70,0x3e,0xb5,0x66,0x48,0x03,0xf6,0x0e,0x61,0x35,0x57,0xb9,0x86,0xc1,0x1d,0x9e}, /\*d\*/

{0xe1,0xf8,0x98,0x11,0x69,0xd9,0x8e,0x94,0x9b,0x1e,0x87,0xe9,0xce,0x55,0x28,0xdf}, /\*e\*/

{0x8c,0xa1,0x89,0x0d,0xbf,0xe6,0x42,0x68,0x41,0x99,0x2d,0x0f,0xb0,0x54,0xbb,0x16} /\*f\*/

};

const std::vector<std::vector<unsigned char>> invsBox =

{/\* 0 1 2 3 4 5 6 7 8 9 a b c d e f \*/

{0x52,0x09,0x6a,0xd5,0x30,0x36,0xa5,0x38,0xbf,0x40,0xa3,0x9e,0x81,0xf3,0xd7,0xfb}, /\*0\*/

{0x7c,0xe3,0x39,0x82,0x9b,0x2f,0xff,0x87,0x34,0x8e,0x43,0x44,0xc4,0xde,0xe9,0xcb}, /\*1\*/

{0x54,0x7b,0x94,0x32,0xa6,0xc2,0x23,0x3d,0xee,0x4c,0x95,0x0b,0x42,0xfa,0xc3,0x4e}, /\*2\*/

{0x08,0x2e,0xa1,0x66,0x28,0xd9,0x24,0xb2,0x76,0x5b,0xa2,0x49,0x6d,0x8b,0xd1,0x25}, /\*3\*/

{0x72,0xf8,0xf6,0x64,0x86,0x68,0x98,0x16,0xd4,0xa4,0x5c,0xcc,0x5d,0x65,0xb6,0x92}, /\*4\*/

{0x6c,0x70,0x48,0x50,0xfd,0xed,0xb9,0xda,0x5e,0x15,0x46,0x57,0xa7,0x8d,0x9d,0x84}, /\*5\*/

{0x90,0xd8,0xab,0x00,0x8c,0xbc,0xd3,0x0a,0xf7,0xe4,0x58,0x05,0xb8,0xb3,0x45,0x06}, /\*6\*/

{0xd0,0x2c,0x1e,0x8f,0xca,0x3f,0x0f,0x02,0xc1,0xaf,0xbd,0x03,0x01,0x13,0x8a,0x6b}, /\*7\*/

{0x3a,0x91,0x11,0x41,0x4f,0x67,0xdc,0xea,0x97,0xf2,0xcf,0xce,0xf0,0xb4,0xe6,0x73}, /\*8\*/

{0x96,0xac,0x74,0x22,0xe7,0xad,0x35,0x85,0xe2,0xf9,0x37,0xe8,0x1c,0x75,0xdf,0x6e}, /\*9\*/

{0x47,0xf1,0x1a,0x71,0x1d,0x29,0xc5,0x89,0x6f,0xb7,0x62,0x0e,0xaa,0x18,0xbe,0x1b}, /\*a\*/

{0xfc,0x56,0x3e,0x4b,0xc6,0xd2,0x79,0x20,0x9a,0xdb,0xc0,0xfe,0x78,0xcd,0x5a,0xf4}, /\*b\*/

{0x1f,0xdd,0xa8,0x33,0x88,0x07,0xc7,0x31,0xb1,0x12,0x10,0x59,0x27,0x80,0xec,0x5f}, /\*c\*/

{0x60,0x51,0x7f,0xa9,0x19,0xb5,0x4a,0x0d,0x2d,0xe5,0x7a,0x9f,0x93,0xc9,0x9c,0xef}, /\*d\*/

{0xa0,0xe0,0x3b,0x4d,0xae,0x2a,0xf5,0xb0,0xc8,0xeb,0xbb,0x3c,0x83,0x53,0x99,0x61}, /\*e\*/

{0x17,0x2b,0x04,0x7e,0xba,0x77,0xd6,0x26,0xe1,0x69,0x14,0x63,0x55,0x21,0x0c,0x7d} /\*f\*/

};

const std::vector<std::vector<unsigned char>> r\_w =

{

{0x01 , 0x00, 0x00, 0x00},

{0x02 , 0x00, 0x00, 0x00},

{0x04 , 0x00, 0x00, 0x00},

{0x08 , 0x00, 0x00, 0x00},

{0x10 , 0x00, 0x00, 0x00},

{0x20 , 0x00, 0x00, 0x00},

{0x40 , 0x00, 0x00, 0x00},

{0x80 , 0x00, 0x00, 0x00},

{0x1b , 0x00, 0x00, 0x00},

{0x36 , 0x00, 0x00, 0x00}

};

Hughes.cpp

#include <iostream>

#include <string>

#include <random>

#include <tuple>

#include <cmath>

#include <cstdint>

using namespace std;

// Declare isPrime function

bool isPrime(int n);

pair<int64\_t, int64\_t> init() {

int64\_t a, p;

cout << "Введите значения (a) и модуля (p): ";

while (true) {

cin >> a >> p;

if (a < p && p < pow(2, 1024)) {

if (isPrime(p)) {

break;

} else {

cout << "Не подходит под параметры выполнения алгоритма" << endl;

}

} else {

cout << "Не подходит под параметры выполнения алгоритма" << endl;

}

}

return make\_pair(a, p);

}

bool isPrime(int n) {

if (n <= 1) return false; // 1 и меньшие числа не являются простыми

// Проверяем делители от 2 до sqrt(n)

for (int i = 2; i <= sqrt(n); i++) {

if (n % i == 0) {

// Если число делится без остатка, то оно не простое

return false;

}

}

return true; // Если не нашлось делителей, то число простое

}

int64\_t rn(int64\_t p) {

random\_device rd;

mt19937 gen(rd());

uniform\_int\_distribution<int64\_t> dist(1, p - 1);

int64\_t num = dist(gen);

return num;

}

int64\_t gcd(int64\_t a, int64\_t b) {

if (b == 0)

return a;

return gcd(b, a % b);

}

int64\_t f(int64\_t x, int64\_t y, int64\_t n, int64\_t r) {

if (y == 0) return r;

if (y & 0x01) return f((x \* x) % n, y >> 1, n, (r \* x) % n);

return f((x \* x) % n, y >> 1, n, r);

}

int64\_t powMod(int64\_t x, int64\_t y, int64\_t n) {

return f(x, y, n, 1);

}

// Расширенный алгоритм Евклида

int64\_t extendedEuclidean(int64\_t a, int64\_t b, int64\_t& x, int64\_t& y) {

if (b == 0) {

x = 1;

y = 0;

return a;

}

int64\_t x1, y1;

int64\_t gcd = extendedEuclidean(b, a % b, x1, y1);

x = y1;

y = x1 - (a / b) \* y1;

return gcd;

}

// Функция для нахождения обратного элемента

int64\_t findModularInverse(int64\_t a, int64\_t m) {

int64\_t x, y;

int64\_t gcd = extendedEuclidean(a, m, x, y);

x = (x % m + m) % m;

return x;

}

string caesar\_cipher(const string& message, int64\_t k) {

string result = "";

for (char c : message) {

result += char((c + k) % 256);

}

return result;

}

string caesar\_decipher(const string& message, int64\_t k\_prime) {

string result = "";

for (char c : message) {

result += char((c - k\_prime + 256) % 256);

}

return result;

}

void huges() {

string message;

getline(cin, message);

int64\_t a, p;

tie(a, p) = init();

int64\_t x = rn(p);

int64\_t y;

while (true) {

y = rn(p);

if (gcd(y, p - 1) == 1) {

break;

}

}

cout << "Абонент А выбирает случайное большое целое число х: " << x << endl;

int64\_t k = powMod(a, x, p);

cout << "Абонент А генерирует закрытый ключ k: " << k << endl;

string encrypted = caesar\_cipher(message, k);

cout << "Абонент А зашифровывает сообщение: " << encrypted << endl;

cout << "Абонент B выбирает случайное большое целое число у (взаимно простое с p - 1): " << y << endl;

int64\_t Y = powMod(a, y, p);

cout << "Абонент B посылает абоненту А: " << Y << endl;

int64\_t X = powMod(Y, x, p);

cout << "Абонент А посылает абоненту B: " << X << endl;

int64\_t z = findModularInverse(y, p - 1);

cout << "Абонент B вычисляет z: " << z << endl;

int64\_t k\_prime = powMod(X, z, p);

cout << "Абонент B вычисляет k': " << k\_prime << endl;

if (k == k\_prime) {

cout << "Закрытый ключ k = k'" << endl;

string decrypted = caesar\_decipher(encrypted, k\_prime);

cout << "Расшифрованное сообщение: " << decrypted << endl;

} else {

cout << "Закрытый не равен k = k'" << endl;

}

cout << endl << endl;

}

Littera.cpp

#include <iostream>

#include <string>

#include <map>

#include <locale>

#include <algorithm>

#include <cctype>

#include <fstream>

using namespace std;

map<char, char> ru\_alphabet = {

{'б', 'а'}, {'в', 'е'}, {'г', 'ё'}, {'д', 'и'}, {'ж', 'о'},

{'з', 'у'}, {'к', 'ы'}, {'л', 'э'}, {'м', 'ю'}, {'н', 'я'},

{'п', 'щ'}, {'р', 'ш'}, {'с', 'ч'}, {'т', 'ц'}, {'ф', 'х'},

{'а', 'б'}, {'е', 'в'}, {'ё', 'г'}, {'и', 'д'}, {'о', 'ж'},

{'у', 'з'}, {'ы', 'к'}, {'э', 'л'}, {'ю', 'м'}, {'я', 'н'},

{'щ', 'п'}, {'ш', 'р'}, {'ч', 'с'}, {'т', 'ц'}, {'х', 'ф'},

{' ', '#'}, {'#', ' '}

};

map<char, char> eng\_alphabet = {

{'a', 'b'}, {'b', 'c'}, {'c', 'd'}, {'d', 'e'}, {'e', 'f'},

{'f', 'g'}, {'g', 'h'}, {'h', 'i'}, {'i', 'j'}, {'j', 'k'},

{'k', 'l'}, {'l', 'm'}, {'m', 'n'}, {'n', 'o'}, {'o', 'p'},

{'p', 'q'}, {'q', 'r'}, {'r', 's'}, {'s', 't'}, {'t', 'u'},

{'u', 'v'}, {'v', 'w'}, {'w', 'x'}, {'x', 'y'}, {'y', 'z'},

{'z', 'a'},

{'A', 'B'}, {'B', 'C'}, {'C', 'D'}, {'D', 'E'}, {'E', 'F'},

{'F', 'G'}, {'G', 'H'}, {'H', 'I'}, {'I', 'J'}, {'J', 'K'},

{'K', 'L'}, {'L', 'M'}, {'M', 'N'}, {'N', 'O'}, {'O', 'P'},

{'P', 'Q'}, {'Q', 'R'}, {'R', 'S'}, {'S', 'T'}, {'T', 'U'},

{'U', 'V'}, {'V', 'W'}, {'W', 'X'}, {'X', 'Y'}, {'Y', 'Z'},

{'Z', 'A'},

{' ', '#'}, {'#', ' '}

};

string init\_littera() {

string choice, text;

cout << "Хотите считать данные сфайла: (Да/Нет): ";

getline(cin, choice);

transform(choice.begin(), choice.end(), choice.begin(),

[](unsigned char c) {return tolower(c); });

if(choice == "да" || choice == "yes" || choice == "д" || choice == "y") {

string text\_file\_path;

cout << "Введите путь к файлу с текстом: ";

getline(cin, text\_file\_path);

ifstream text\_file(text\_file\_path);

while((!text\_file.is\_open())) {

cerr << "Ошибка открытия файла. Пожалуйста, проверьте пути и повторите попытку." << endl;

cout << "Путь к файлу с текстом: ";

getline(cin, text\_file\_path);

ifstream text\_file(text\_file\_path);

}

getline(text\_file, text);

text\_file.close();

}

else{

cout << "Введите текст: ";

getline(cin, text);

}

return text;

}

void write\_text(const string& text) {

string choice;

cout << "Хотите записать результат в файл? (Да/Нет) : ";

getline(cin, choice);

transform(choice.begin(), choice.end(), choice.begin(),

[](unsigned char c) {return tolower(c); });

if(choice == "да" || choice == "yes" || choice == "д" || choice == "y") {

string file\_path;

cout << "Введите путь куда сохранить текст: ";

getline(cin, file\_path);

ofstream file(file\_path);

if (file.is\_open()) {

file << text;

file.close();

cout << "Текст успешно записан в файл." << endl;

}

else {

cout << "Не удалось открыть файл." << endl;

}

}

}

void littera() {

setlocale(LC\_ALL, "");

string text = init\_littera();

for (char& ch : text) {

if (ru\_alphabet.find(ch) != ru\_alphabet.end()) {

ch = ru\_alphabet[ch];

}

else if(eng\_alphabet.find(ch) != eng\_alphabet.end()) {

ch = eng\_alphabet[ch];

}

}

cout << "Результат: " << text << endl;

write\_text(text);

}