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*»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБc-324», «АВТФ»  *Петров Максим Игоревич*  «\_\_\_» \_\_\_\_\_\_ 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Доцент кафедры ЗИ*  *Архипова Анастасия Борисовна*  «\_\_\_» \_\_\_\_\_\_ 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2024

СОДЕРЖАНИЕ

[Введение 3](#_Toc183012731)

[1. Теоретическая часть 4](#_Toc183012732)

[1.1 Алгоритм работы шифровки A1Z26 4](#_Toc183012733)

[1.2 Алгоритм работы шифровки Шамира 6](#_Toc183012734)

[1.3 Алгоритм работы шифровки Бэкона 8](#_Toc183012735)

[2. Практическая часть 11](#_Toc183012736)

[2.1 Постановка задачи 11](#_Toc183012737)

[2.2 Характеристика задачи 11](#_Toc183012738)

[2.3 Алгоритм решения 12](#_Toc183012739)

[2.4 Руководство пользователя 14](#_Toc183012740)

[2.5 Руководство системного программиста 18](#_Toc183012741)

[2.6 Контрольный пример 20](#_Toc183012742)

[2.7 Ручной расчет 21](#_Toc183012743)

[Список использованных источников 26](#_Toc183012744)

[Приложение 27](#_Toc183012745)

# ВВЕДЕНИЕ

В современном обществе все большую роль играют компьютеры, и вообще электронные средства передачи, хранения, и обработки информации. Для того чтобы информационные технологии можно было использовать в различных областях, необходимо обеспечить их надежность и безопасность. Поэтому широкое использование информационных технологий привело к бурному развитию различных методов защиты информации, из которых основными можно, пожалуй, назвать, помехоустойчивое кодирование и криптографию.

Цель работы — это разработка программы, которая будет шифровать и дешифровать текст. Задача заключается в том, чтобы разработать алгоритмы, они будут обеспечивать безопасность обработки данных, которые будут доступны тем, кто обладает специальным ключом.

Результаты при создании программного обеспечения, должен включать работу шифрования и дешифрования текстовых данных, и также применение этого программного обеспечения для работы с задачами в области информационной безопасности.

Теоретической основой написания расчетно-графической работы являлись следующие источники: Фионов А.Н. [1], Ильин М.Е. [4], И.И.Ефишов [5].

Практической основой написания расчетно-графической работы являлись следующие источники, помогающие разработать алгоритм решения задачи: Павловская Т. А. [2], Мартин Р. [3], Страуструп Б. [6], Мейерс С. [7 – 8], Саттер Г. [9], Прата С. [10].

## 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В программе используются 3 метода шифрования: Шифр A1Z26, Шифр Шамира, Шифр Бэкона.

# 1.1 Алгоритм работы шифровки A1Z26

1.1.1 Введение

Шифр A1Z26 (A1Z33), по-другому моноалфавитная замена — каждая буква заменяется на одну и только одну букву или символ. Такие шифры очень легко расшифровываются с помощью частотного анализа. Шифр A1Z26 представляет собой простой шифр замещения, декодируемый путем замены nᵗʰ буквы алфавита на заданное число n [4].

1.1.2 Исторический контекст

Шифр A1Z26 считается древнейшим и простейшим шифром, который зародился одновременно в нескольких развитых древних государствах примерно за 400 лет до н. э [4].

1.1.3 Принцип работы

Это простая подстановка, где каждая буква заменена её порядковым номером в алфавите. Только нижний регистр.

Алфавит выглядит следующим образом:

английские: a-z = 0-26  
английские: A-Z = 27-52  
русские: а-я = 53-85  
русские: А-Я = 86-118

остальные символы (с 33 кода ASCII до 64) он шифрует, просто прибавляя 100. Условно сообщение "Привет Hello". Нам нужно закодировать по шаблону выше у П код ASCII 207, а порядок по нашему шаблону 101. Мы знаем, что A большая русская начинается с 86, а в ASCII ее код 192, то есть разница между числами 106, и так будет со всеми буквами от А до Я. То есть для кодирования символов от А до Я нам нужно вычесть код ASCII от порядка в нашем алфавите. Для а - я логика идентична, их расположили в порядке от 53 до 85, а коды ASCII для них 224 - 255, разница для каждого = 224 - 53 = 171.  
П = 207 (код ASCII) - 106 (разница между кодом и порядковым номером) = 101  
р = 240 (код) - 171 (разница) = 69  
и = 232 (код) - 171 (разница)= 61.  
....  
Для латиницы a-z в ASCII это 97-122 , в алфавите это 1-26. находим разницу 97-1 = 96. A-Z в ASCII это 65-90, в алфавите это 27-52. разница: 65 - 27 = 38  
H = 72 - 38 = 34  
e = 101 - 96 = 5  
...

1.1.4 Криптоанализ

Несмотря на свою простоту, шифр A1Z26 имеет ограничения. Он не обеспечивает высокую степень безопасности, и его легко разгадать, особенно если злоумышленник знает о методе шифрования. Дешифровка аналогично, только мы к нашему числу прибавляем разницу, в зависимости от того в какой диапазон оно попало [4].

1.1.5 Применение

В наши дни данный шифр не используется для защиты определенной информации, так как очень прост в расшифровке. Но его часто используют в развлекательных целях: головоломках, игр и даже в мультсериалах. Данный метод шифрования может помочь студентам понять азы шифрования и дешифрования [4].

# 1.2 Алгоритм работы шифровки Шамира

1.2.1 Введение

Шифр Шамира этот шифр, предложенный Шамиром, был первым, позволяющим организовать обмен секретными сообщениями по открытой линии связи для лиц, которые не имеют никаких защищенных каналов и секретных ключей и, возможно, никогда не видели друг друга. Шифр, как правило, используется для передачи чисел, например, секретных ключей, значения которых меньше p [1].

1.2.2 Исторический контекст

В 1979 году израильский криптоаналитик Ади Шамир предложил пороговую схему разделения секрета. Она позволяет проводить разделение таким образом, что для восстановления секрета достаточно определённой доли сторон, а остальные не смогут получить никакой информации о секрете.

Также около 1980 года Шамир разработал трёхэтапный протокол Шамира — криптографический протокол, который позволяет двум сторонам безопасно обмениваться сообщениями без необходимости распространения ключей шифрования. Обмен сообщением между пользователями происходит в три прохода [1].

1.2.3 Принцип работы

Пусть есть два абонента A и B, соединенные линией связи. A хочет передать сообщение m абоненту B так, чтобы никто не узнал его содержание. A выбирает случайное большое простое число p и открыто передает его B. Затем A выбирает два числа cA и dA , такие, что

cAdA mod (p − 1) = 1

Эти числа A держит в секрете и передавать не будет. B тоже выбирает два числа cB и dB , такие, что

cBdB mod (p − 1) = 1

и держит их в секрете.

После этого A передает свое сообщение m, используя трехступенчатый протокол. Если m < p (m рассматривается как число), то сообщение m передается сразу , если же m ≥ p, то сообщение представляется в виде m1, m2, . . . , mt, где все mi < p, и затем передаются последовательно m1, m2, . . . , mt. При этом для кодирования каждого mi лучше выбирать случайно новые пары (cA, dA) и (cB, dB) — в противном случае надежность системы понижается. В настоящее время такой шифр, как правило, используется для передачи чисел, например, секретных ключей, значения которых меньше p. Таким образом, мы будем рассматривать только случай m < p. Дадим описание протокола.

1. A вычисляет число x1 = mcA mod p, где m — исходное сообщение, и пересылает x1 к B.

2. B, получив x1, вычисляет число x2 = xcB 1 mod p и передает x2 к A.

3. A вычисляет число x3 = xdA 2 mod p и передает его B.

4. B, получив x3 , вычисляет число x4 = xdB 3 mod p. [1]

1.2.4 Криптоанализ

Уязвимости криптографических алгоритмов могут возникать из-за слабостей в их математических основах, ошибок в реализации или использования слабых ключей. Например, использование предсказуемых или повторяющихся ключей может значительно облегчить работу злоумышленника. Кроме того, ошибки в реализации, такие как неправильное управление памятью или незащищенные процессы генерации ключей, могут создавать дополнительные возможности для атак [1].

1.2.5 Применение

В системах, где требуется высокая степень безопасности, ключи могут храниться в распределённом виде. Например, в банковских системах или системах управления доступом, где доступ к критически важным данным должен быть ограничен. В системах электронного голосования шифр Шамира может использоваться для обеспечения анонимности и безопасности голосов. Алгоритм может быть использован в различных криптографических протоколах, таких как протоколы аутентификации и обмена ключами, где требуется надежная защита секретной информации.

# 1.3 Алгоритм работы шифровки Бэкона

1.3.1 Введение

Шифр Бэкона, разработанный в начале XVII века, является одним из самых уникальных методов стеганографии и криптографии того времени. Этот шифр позволяет передавать скрытые сообщения под видом обычного текста, используя различное форматирование букв для представления двоичного кода. В основе метода лежит идея, что каждая буква алфавита может быть представлена комбинацией из пяти символов 'А' и 'B', что в современной интерпретации соответствует двоичной системе счисления [5].

1.3.2 Исторический контекст

Фрэнсис Бэкон (1561–1626) еще в ранней юности, когда ему было всего семнадцать лет, изобрел шифр, который, представляет собой высшую ступень совершенства шифра, давая возможность выражать все через все. В это время Бэкон служил в составе английской миссии в Париже, где, вероятнее всего, впервые и познакомился с дипломатическими шифрами. Однако первое краткое упоминание о своем шифре ученый привел только через двадцать семь лет в работе «О преумножении наук». Более подробно двухлитерный шифр описан в трактате «О достоинстве и приумножении наук». Исторически шифр Бэкона использовался не только для отправки секретных сообщений, но и как способ скрытой передачи знаний и идей, что было особенно важно в эпоху, когда прямое высказывание своих мыслей могло привести к преследованиям [5].

1.3.3 Принцип работы

Основная идея шифра Бэкона заключается в использовании двух различных шрифтов или стилей для представления каждой буквы алфавита как комбинации двух символов, которые в современном понимании можно сравнить с двоичной системой счисления, где 'А' и 'B' соответствуют '0' и '1'. Таким образом, каждая буква кодируется последовательностью из пяти символов, что позволяет создавать скрытые сообщения. В своем оригинальном виде Фрэнсис Бэкон предложил алфавит из 24 букв, где каждая буква заменяется на комбинацию из пяти символов 'А' или 'B'. Например, буква 'A' кодируется как 'AAAAA', 'B' - как 'AAAAB', и так далее до 'Y' ('BABBB') и 'Z' ('BBAAA') [5].

1.3.4 Криптоанализ

Шифр Бэкона несмотря на свою изобретательность, шифр имеет ограничения, связанные с необходимостью договоренности между отправителем и получателем о значении 'A' и 'B' в контексте конкретного сообщения, а также сложности генерации и передачи длинных скрытых сообщений из-за ограниченности текстового пространства для кодирования. Также обладает рядом уязвимостей, которые могут снизить его эффективность при недостаточно аккуратном использовании [5].

1.3.5 Применение

Шифр Бэкона, хотя и является достаточно сложным для реализации в повседневной жизни из-за его скрытого характера, нашел свое применение в различных областях, включая литературу, искусство и даже в современных цифровых технологиях. Также он может использоваться как обучающий инструмент для объяснения принципов шифрования и стеганографии, а также как элемент головоломок [5].

# 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

# 2.1 Постановка задачи

Необходимо сделать программу, которая будет шифровать и дешифровать данные одним из алгоритмов, которые представлены. Программа должна выполнять 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 «*Крона*» 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.

# 2.3 Алгоритм решения

1. Запустить приложение «Крона»;

2. Вывод: «Enter the password:»;

3. Ввод пароля;

4. Вывод меню:

«1 - Encryption using Bacon's cipher»;

«2 - A1Z26 cipher»;

«3 - Encryption using Shamir's algorithm»;

«0 - Exit the program»;

5. Если выбран пункт «1 - Encryption using Bacon's cipher»:

5.1 Вывод: «Enter the message:»;

5.2 Ввод сообщения;

5.3 Вывод: «Enter the filename to save the message:»;

5.4 Ввод названия файла, в котором будет храниться текст;

5.5 Вывод зашифрованного сообщения «Encrypted message:»;

5.6 Вывод: «Enter the filename to save the encrypted message:»;

5.7 Ввод названия файла, в котором будет храниться расшифрованный текст;

5.8 Программа предлагает расшифровать текст: «Decrypt the message? Enter /y/ or /Y/ to confirm:»;

5.8.1 Если выбрано расшифрование, то ввод: «Enter the filename where the encrypted message is stored:»;

5.8.2 Вывод: «Decrypted message:»;

5.8.3 Ввод: «Enter the filename to save the decrypted message:»;

5.8.4 Ввод названия файла, куда будет записано расшифрованное сообщение;

5.9 Ввод: «Select the cipher number:»;

6. Если выбран пункт «2 - A1Z26 cipher»:

6.1 Вывод: «Enter the message:»;

6.2 Ввод сообщения;

6.3 Вывод: «Enter the filename to save the message:»;

6.4 Ввод названия файла, в котором будет храниться текст;

6.5 Вывод зашифрованного сообщения «Encrypted message:»;

6.6 Вывод: «Enter the filename to save the encrypted message:»;

6.7 Ввод названия файла, в котором будет храниться расшифрованный текст;

6.8 Программа предлагает расшифровать текст: «Decrypt the message? Enter /y/ or /Y/ to confirm:»;

6.8.1 Если выбрано расшифрование, то ввод: «Enter the filename where the encrypted message is stored:»;

6.8.2 Вывод: «Decrypted message:»;

6.8.3 Ввод: «Enter the filename to save the decrypted message:»;

6.8.4 Ввод названия файла, куда будет записано расшифрованное сообщение;

6.9 Ввод: «Select the cipher number:»;

7. Если выбран пункт «3 - Encryption using Shamir's algorithm»:

7.1 Вывод: «Enter the message:»;

7.2 Ввод сообщения;

7.3 Вывод: «Enter the filename to save the message:»;

7.4 Ввод названия файла, в котором будет храниться текст;

7.5 Вывод зашифрованного сообщения «Encrypted message:»;

7.6 Вывод: «Enter the filename to save the encrypted message:»;

7.7 Ввод названия файла, в котором будет храниться расшифрованный текст;

7.8 Программа предлагает расшифровать текст: «Decrypt the message? Enter /y/ or /Y/ to confirm:»;

7.8.1 Если выбрано расшифрование, то ввод: «Enter the filename where the encrypted message is stored:»;

7.8.2 Вывод: «Decrypted message:»;

7.8.3 Ввод: «Enter the filename to save the decrypted message:»;

7.8.4 Ввод названия файла, куда будет записано расшифрованное сообщение;

8. Если выбран пункт «0 - Exit the program»;

8.1 Программа завершает свою работу;

# 2.4 Руководство пользователя

2.4.1 Введение

2.4.1.1 Программа «Крона» 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/файла;

- шифрование/расшифрование данных;

- вывод результата в консоль/файл.

2.4.1.3 Программа реализована на языке C++. Работает в любой среде совместимой с «Windows». Дисковой памяти для запуска программы требуется не менее 300 Mb. Оперативной памяти для запуска 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 назначения.

После запуска на экран выводится окно программы. Как выглядит окно приведено на русинке 2.1. Для начала работы с программой необходимо ввести правильный пароль («1q2w3e»)



Рисунок 2.1 – Окно программы

Если введен некорректный пароль программа запросит ввести его вновь (рисунок 2.2).

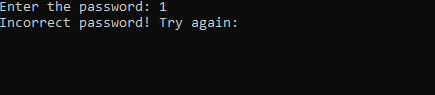


Рисунок 2.2 – Ошибка «Неверный пароль! Попробуйте снова»

В случае если ввести пароль неправильно второй раз программа предупредит о том, что осталась последняя попытка. После она завершится (рисунок 2.3).

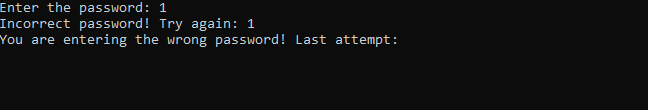


Рисунок 2.3 – Ошибка «Неверный пароль! Последняя попытка»

После верно введенного пароля, программа предлагает выбор алгоритмов шифрования (рисунок 2.4).

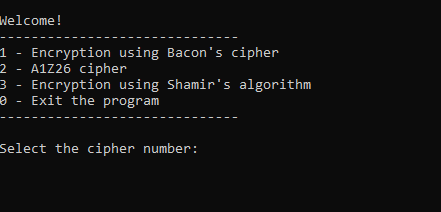


Рисунок 2.4 – Вывод меню алгоритмов шифрования

При выборе «Bacon» начинается диалог с пользователем (рисунок 2.5). Пользователю предлагают ввести текст и записать его в файл, а после зашифровать его. После нам предлагают расшифровать текст по файлу, в котором хранится зашифрованное сообщение.

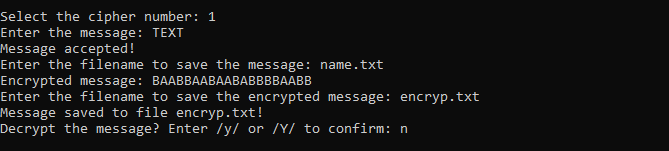


Рисунок 2.5 – Вывод «Bacon»

Те же способы взаимодействия с пользователем в остальных двух алгоритмах (рисунок 2.6 и 2.7).

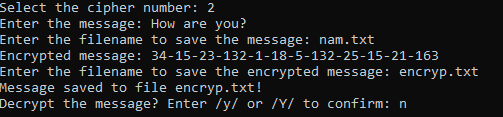


Рисунок 2.6 – Вывод «A1Z26»

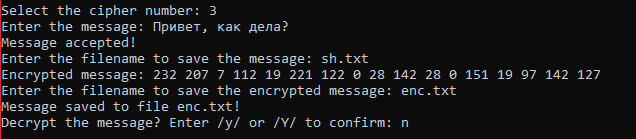


Рисунок 2.7 – Вывод «Shamir»

2.4.3 Сообщение пользователю

При работе с программой могут возникнуть некоторые ошибки, которые представлены в таблице 2.1.

Таблица 2.1 – Ошибки, которые могут появиться

|  |  |
| --- | --- |
| Текст сообщение программы | Ответ пользователя |
| Incorrect password! Try again:  You are entering the wrong password! Last attempt: | Выводится при неверном вводе пароля. |
| Invalid cipher number selected! | Выводится, если был выбран некорректный пункт |
| Error, invalid characters entered: | Выводится, если происходит ввод символа, который не шифруется. |
| Error, unable to open the file | Выводится при неверном пути для считывания из файла. |

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 C++ в среде Visual Studio Code.

Модули программы:

cryptosystem.cpp. Содержит функцию main, представляющую функционал по вводу пароля, выводу меню выбора алгоритмов шифрования, с соответствующим вызовом функций, объявленных в заголовочном файле cryptosystem.h:

Объявление функции для работы с файлами

string FileInput(const string &filename) – функция чтения из файла;

string FileOutput(const string &filename, const string &str) – функция

записи в файл;

void input\_and\_check(string &message, const string &choice\_cipher, const string &message\_or\_key) – функция для ввода и проверки сообщения;

void Enc\_and\_Desc(const string &choice\_cipher) – функция для

шифрования и дешифрования;

extern const string passwd – объявление переменной пароля;

bacon.h

vector<char> checkinputbacon(const string& message) – функция для

проверки ввода для шифра Бэкона;

string baconEncryption(string& message) – функция шифровки

сообщения с помощью шифра Бэкона

string baconDecryption(const string& message) – функция расшифровки сообщения с помощью шифра Бэкона

a1z26.h

string a1z26Encryption(string text) – функция шифрования сообщения с

помощью шифра A1Z26;

string a1z26Decryption(string crypto) – функция расшифровки

сообщения с помощью шифра A1Z26 ;

shamir.h

vector<char> checkinputshamir(const string& message) – функция для

проверки ввода для шифра Шамира;

void generateShamirKeys() – функция для генерации ключей для

помощью алгоритма Шамира;

string shamirEncryption(const string& message) – функция шифрования

сообщения с помощью алгоритма Шамира;

string shamirDecryption(const string& message) – функция расшифровки

сообщения с помощью алгоритма Шамира;

Связь модулей представлена на рисунке 2.7

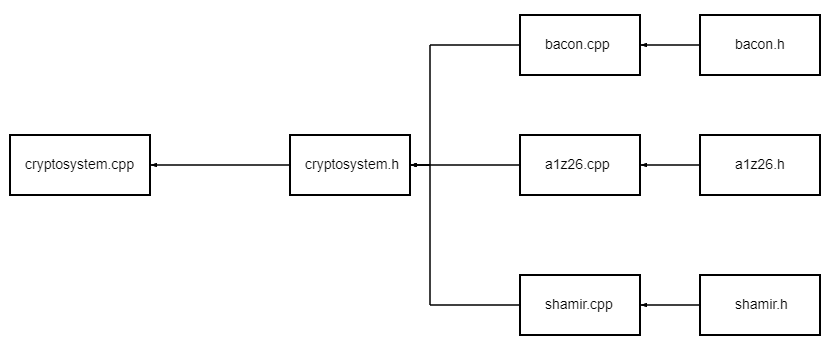


Рисунок 2.7 – Связь модулей

# 2.6 Контрольный пример

После запуска на экран выводится окно программы, где необходимо ввести правильный пароль (рисунок 2.1).

После верно введенного пароля, программа предлагает выбор алгоритмов шифрования (рисунок 2.4).

Если пользователь выберет шифр Бекона, программа предложит ввести сообщение. После чего программа предложит нам сохранить текст в файл и зашифрует его, и предложит сохранить зашифрованное сообщение. Далее перед нами будет стоять выбор расшифровать сообщение, для этого нужно указать верный путь к зашифрованному сообщению, или нет. Если мы выбрали расшифровать сообщение, то на экране будет вывод нашего зашифрованного сообщения (рисунок 2.8).

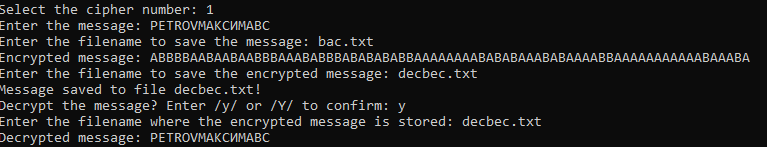


Рисунок 2.8 – Работа с шифром Бекона

Если пользователь выберет шифр A1Z26, программа предложит ввести сообщение. После чего программа предложит нам сохранить текст в файл и зашифрует его, и предложит сохранить зашифрованное сообщение. Далее перед нами будет стоять выбор расшифровать сообщение, для этого нужно указать верный путь к зашифрованному сообщению, или нет. Если мы выбрали расшифровать сообщение, то на экране будет вывод нашего зашифрованного сообщения (рисунок 2.9).

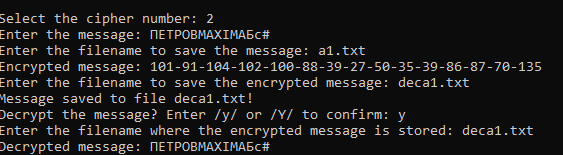


Рисунок 2.9 – Работа с шифром A1Z26

Если пользователь выберет шифр Шамира, программа предложит ввести сообщение. После чего программа предложит нам сохранить текст в файл и зашифрует его, и предложит сохранить зашифрованное сообщение. Далее перед нами будет стоять выбор расшифровать сообщение, для этого нужно указать верный путь к зашифрованному сообщению, или нет. Если мы выбрали расшифровать сообщение, то на экране будет вывод нашего зашифрованного сообщения (рисунок 2.10).

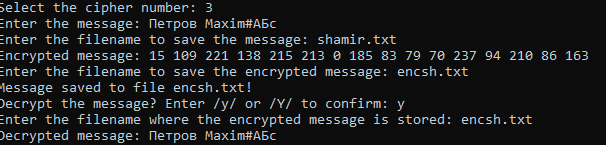


Рисунок 2.10 – Работа с шифром Шамира

# 2.7 Ручной расчет

2.7.1 Шифровка с помощью шифра Бекона

Текст – PETROVМАКСИМABC;

Так как данный шифр шифрует только буквы, то он не будет принимать различные специальные символы или буквы.

Разделим строку на отдельные символы:

Шифровка английских символов:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Текст | P | E | T | R | O | V |
| Шифр | ABBBB | AABAA | BAABB | BAAAB | ABBBA | BABAB |

|  |  |  |  |
| --- | --- | --- | --- |
| Текст | A | B | C |
| Шифр | AAAAA | AAAAB | AAABA |

Шифровка русских символов:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Текст | М | А | К | С | И | М |
| Шифр | ABBAA | ААААА | АBАBА | BАААB | АBААА | ABBAA |

Теперь соберем все коды Бэкона в одну строку:

ABBBB AABAA BAABB BAAAB ABBBA BABAB ABBAA АAAAA АBАBА BАААB АBААА ABBAA AAAAA AAAAB AAABA

Получаем результат ручного расчета: «ABBBBAABAABAABBBAAABABBBABABABABBAAАAAAAАBАBАBАААBАBАААABBAAAAAAAAAAABAAABA»

Результат ручного расчета соответствует контрольному примеру.



Рисунок 2.11 – Вывод контрольного примера шифра «Bacon»

2.7.2 Шифровка с помощью шифра A1Z26

Текст – ПЕТРОВMAXIMАБс#

Порядковый номер из ASCII и разница между порядковым номером в алфавите вычисляем так, начальная буква английского алфавита (a – z) равна 1, порядковый номер ASCII равен 97, далее мы находим разницу 97–1 равно 96. Теперь вычисляем разницу для (A – Z), начальная буква 27, порядковый номер ASCII равен 65, 65–27 равно 38. Далее нам нужно вычислить разницу для русских букв (А – Я), начальная буква 86, порядковый номер ASCII равен 192, после этого найдем разность 192–86 равно 106. И последнее для (а – я), начальная буква 53, порядковый номер ASCII равен 224, и находим разность 224–53 равно 171.

Таблица расчета для русских символов.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Текст | П | Е | Т | Р | О | В | А | Б | с |
| Код | 207 | 197 | 210 | 208 | 206 | 194 | 192 | 193 | 240 |
| Разница | 106 | 106 | 106 | 106 | 106 | 106 | 106 | 106 | 171 |
| Шифр | 101 | 91 | 104 | 102 | 100 | 88 | 86 | 87 | 70 |

Таблица расчетов английских символов.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Текст | M | A | X | I | M | # |
| Код | 77 | 65 | 88 | 73 | 77 | 35 |
| Разница | 38 | 38 | 38 | 38 | 38 | 100 |
| Шифр | 39 | 27 | 50 | 35 | 39 | 135 |

Получаем результат: «101-91-104-102-100-88-39-27-50-35-39-86-87-70-135»

Результат ручного расчета соответствует контрольному примеру.



Рисунок 2.12 – Вывод контрольного примера шифра «A1Z26»

2.7.3 Шифровка с помощью шифра Шамира

Текст – Петров Maxim#АБс

Для расчета у нас есть простое число p = 257, cA это секретный ключ, который равен 5. M = номер алфавита.

Таблица шифрования по формуле C = M^cA mod p

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Текст | П | е | т | р | о | в | А | Б | с |
| M | 68 | 91 | 104 | 102 | 100 | 88 | 53 | 54 | 103 |
| M5 mod 257 | 15 | 109 | 221 | 138 | 215 | 213 | 210 | 86 | 163 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Текст | M | a | x | i | m | # | Space |
| M | 13 | 27 | 50 | 35 | 39 | 154 | 0 |
| M5 mod 257 | 185 | 83 | 79 | 70 | 237 | 94 | 0 |

Получаем результат: «15 109 221 138 215 213 0 185 83 79 70 237 94 210 86 163»

Результат ручного расчета соответствует контрольному примеру.



Рисунок 2.12 – Вывод контрольного примера шифра «Shamir»

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рябко Б.Я., Фионов А.Н. Криптографические методы защиты информации. 2-е издание, стереотип. – М.: Горячая линия – Телеком, 2012. – 229 c.
2. Павловская Т. А. C/C++. Программирование на языке высокого уровня: [для вузов] / Т.А.Павловская. - СПб. [и др.]: Питер, 2010. - 460 с.
3. Мартин Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста. — СПб.: Питер, 2013. — 464 с.
4. Ильин М.Е., Калинкина Т.И. Криптографическая защита информации в объектах информационной инфраструктуры. 46-е издание, Москва: Издательский центр «Академия», 2019. – 272 с.
5. Ефишов И.И. К двухлитерному шифру Фрэнсиса Бэкона. Дискуссии. 2016. №3.
6. Страуструп Б. Программирование: принципы и практика с использованием С++, 2-е изд. : Пер. с англ. - М. : ООО "И.Д. Вильяме", 2016. - 1 328 с.
7. Мейерс С. Эффективное использование С++. — М.: ДМК Пресс, 2014. — 294 с.
8. Мейерс С. Эффективное использование STL. Библиотека программиста / С. Мейерс. СПб.: Питер, 2002. — 224 с.
9. Саттер Г., Александреску А. Стандарты программирования на С++. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2008. — 224 с.
10. Прата С. Язык программирования C++ (C++11). Лекции и упражнения, 6-е издание — М.: Вильямс, 2012. — 1248 с.

# ПРИЛОЖЕНИЕ

**cryptosystem.cpp**

#include "Cryptosystem.h"

#include "bacon.h"

#include "a1z26.h"

#include "shamir.h"

const string passwd = "1q2w3e";

string FileInput(const string &filename) { // Ф-ия вывода из файла

    string str;

    ifstream input;

    input.open(filename);

    if (input.is\_open()) {

        getline(input, str);

        input.close();

        return str;

    } else {

        return "Error: Unable to open the file";

    }

}

string FileOutput(const string &filename, const string &str) { // ф-ия ввода в файл

    ofstream output;

    output.open(filename);

    if (output.is\_open()) {

        output << str;

        output.close();

        return "Completed";

    } else {

        return "Error: Unable to open the file";

    }

}

void input\_and\_check(string& message, const string &choice\_cipher, const string &message\_or\_key) { // ф-ия ввода сообщения с клавиатуры и его проверка

    if (message\_or\_key == "message") {

        cout << "Enter the message: ";

    } else {

        cout << "Enter the key word: ";

    }

    cin.ignore();

    while (true) {

        getline(cin, message);

        vector<char> errorinput;

        if (choice\_cipher == "Bacon") {

            errorinput = checkinputbacon(message);

        } else if (choice\_cipher == "A1Z26") {

            errorinput = checkinputa1z26(message);

        } else {

            // Для шифра Шамира проверка может быть иной

            errorinput = checkinputshamir(message);

        }

        if (!errorinput.empty()) {

            cout << "Error, invalid characters entered: ";

            for (const auto elem : errorinput) {

                cout << elem << " ";

            }

            cout << endl << "Please try again: ";

        } else {

            if (message\_or\_key == "message") {

                cout << "Message accepted!" << endl;

            } else {

                cout << "Key word accepted!" << endl;

            }

            break;

        }

    }

}

void Enc\_and\_Desc(const string &choice\_cipher) {

    string message, wordkey, filename;

    if (choice\_cipher == "Bacon") {

        input\_and\_check(message, "Bacon", "message");

    } else if (choice\_cipher == "A1Z26") {

        input\_and\_check(message, "A1Z26", "message");

        input\_and\_check(wordkey, "A1Z26", "key");

    } else if (choice\_cipher == "Shamir") {

        input\_and\_check(message, "Shamir", "message");

        // Генерация ключей для шифра Шамира

        generateShamirKeys();

    }

    cout << "Enter the filename to save the message: ";

    cin >> filename;

    FileOutput(filename, message);

    // ШИФРОВКА //

    message = FileInput(filename);

    string Encrypted;

    if (choice\_cipher == "Bacon") {

        Encrypted = baconEncryption(message);

    } else if (choice\_cipher == "A1Z26") {

        Encrypted = a1z26Encryption(message, wordkey);

    } else if (choice\_cipher == "Shamir") {

        Encrypted = shamirEncryption(message);

    }

    cout << "Encrypted message: " << Encrypted << endl;

    cout << "Enter the filename to save the encrypted message: ";

    cin >> filename;

    string check = FileOutput(filename, Encrypted); // записываем в файл

    if (check != "Completed") {

        cout << "Error, unable to open the file " << filename << "!" << endl;

    } else {

        cout << "Message saved to file " << filename << "!" << endl;

        // Расшифровка //

        cout << "Decrypt the message? Enter /y/ or /Y/ to confirm: ";

        char choice;

        cin >> choice;

        if (choice == 'y' || choice == 'Y') {

            Encrypted.clear();

            cout << "Enter the filename where the encrypted message is stored: ";

            cin >> filename;

            Encrypted = FileInput(filename);

            if (Encrypted != "Error: Unable to open the file") {

                string Decrypted;

                if (choice\_cipher == "Bacon") {

                    Decrypted = baconDecryption(Encrypted);

                } else if (choice\_cipher == "A1Z26") {

                    Decrypted = a1z26Decryption(Encrypted, wordkey);

                } else if (choice\_cipher == "Shamir") {

                    Decrypted = shamirDecryption(Encrypted);

                }

                cout << "Decrypted message: " << Decrypted << endl;

                cout << "Enter the filename to save the decrypted message: ";

                cin >> filename;

                check = FileOutput(filename, Decrypted); // записываем в файл

                if (check != "Completed") {

                    cout << "Error, unable to open the file " << filename << "!" << endl;

                } else {

                    cout << "Message saved to file " << filename << "!" << endl;

                }

            } else {

                cout << "Error, unable to open the file " << filename << "!" << endl;

            }

        }

    }

}

int main() {

    system("cls");

    SetConsoleCP(1251);

    SetConsoleOutputCP(1251);

    srand(time(0));

    string password;

    cout << "Enter the password: ";

    cin >> password;

    if (password != passwd) {

        cout << "Incorrect password! Try again: ";

        cin >> password;

        if (password != passwd) {

            cout << "You are entering the wrong password! Last attempt: ";

            cin >> password;

            if (password != passwd) {

                cout << "All attempts used! Exiting the program...";

                exit(0);

            }

        }

    }

    system("cls");

    cout << "\nWelcome!\n" << "------------------------------\n"

         << "1 - Encryption using Bacon's cipher\n"

         << "2 - A1Z26 cipher\n"

         << "3 - Encryption using Shamir's algorithm\n"

         << "0 - Exit the program\n"

         << "------------------------------\n";

    while (true) {

        int choice;

        while (true) {

            try { // обработка ошибки

                cout << "\nSelect the cipher number: ";

                cin >> choice;

                if (cin.fail()) {

                    throw invalid\_argument("Enter a number only!");

                }

                break;

            } catch (invalid\_argument& ex) { // ловим ошибку, выводим её пользователю и запрашиваем ввод заново

                cin.clear();

                cin.ignore();

                cout << "Error: " << ex.what() << endl;

            }

        }

        if (choice == 1) {

            Enc\_and\_Desc("Bacon");

        } else if (choice == 2) {

            Enc\_and\_Desc("A1Z26");

        } else if (choice == 3) {

            Enc\_and\_Desc("Shamir");

        } else if (choice == 0) {

            cout << "Exiting the program...";

            exit(0);

        } else {

            cout << "Invalid cipher number selected!" << endl;

        }

    }

}

**bacon.cpp**

#include <map>

#include "Cryptosystem.h"

// Функция для проверки ввода для шифра Бэкона

vector<char> checkinputbacon(const string& message) {

    vector<char> invalidChars;

    for (char c : message) {

        // Допустимые символы: латинские и кириллические буквы, кириллические буквы, пробел

        if (!((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') ||

              (c >= 'А' && c <= 'Я') || (c >= 'а' && c <= 'я') ||

               c == ' ')) {

            invalidChars.push\_back(c);

        }

    }

    return invalidChars;

}

string baconEncryption(string& message) {

    map<char, string> baconMap = {

        // English

        {'A', "AAAAA"}, {'B', "AAAAB"}, {'C', "AAABA"}, {'D', "AAABB"},

        {'E', "AABAA"}, {'F', "AABAB"}, {'G', "AABBA"}, {'H', "AABBB"},

        {'I', "ABAAA"}, {'J', "ABAAB"}, {'K', "ABABA"}, {'L', "ABABB"},

        {'M', "ABBAA"}, {'N', "ABBAB"}, {'O', "ABBBA"}, {'P', "ABBBB"},

        {'Q', "BAAAA"}, {'R', "BAAAB"}, {'S', "BAABA"}, {'T', "BAABB"},

        {'U', "BABAA"}, {'V', "BABAB"}, {'W', "BABBA"}, {'X', "BABBB"},

        {'Y', "BBAAA"}, {'Z', "BBAAB"},

        {'a', "aaaaa"}, {'b', "aaaab"}, {'c', "aaaba"}, {'d', "aaabb"},

        {'e', "aabaa"}, {'f', "aabab"}, {'g', "aabba"}, {'h', "aabbb"},

        {'i', "abaaa"}, {'j', "abaab"}, {'k', "ababa"}, {'l', "ababb"},

        {'m', "abbaaa"}, {'n', "abbab"}, {'o', "abbba"}, {'p', "abbbb"},

        {'q', "baaaa"}, {'r', "baaab"}, {'s', "baaba"}, {'t', "baabb"},

        {'u', "babaa"}, {'v', "babab"}, {'w', "babba"}, {'x', "babbb"},

        {'y', "bbaaa"}, {'z', "bbaab"},

        {'А', "ААААА"}, {'Б', "ААААB",}, {'В', "АААBА"}, {'Г', "АААBB"},

        {'Д', "ААBАА"}, {'Е', "ААBАB"}, {'Ё', "ААBАB"}, {'Ж', "ААBBА"},

        {'З', "ААBBB"}, {'И', "АBААА"}, {'Й', "АBААB"}, {'К', "АBАBА"},

        {'Л', "АBАBB"}, {'М', "АBBАА"}, {'Н', "АBBАB"}, {'О', "АBBBА"},

        {'П', "АBBBB"}, {'Р', "BАААА"}, {'С', "BАААB"}, {'Т', "BААBА"},

        {'У', "BААBB"}, {'Ф', "BАBАА"}, {'Х', "BАBАB"}, {'Ц', "BАBBА"},

        {'Ч', "BАBBB"}, {'Ш', "BBААА"}, {'Щ', "BBААB"}, {'Ъ', "BBАBА"},

        {'Ы', "BBАBB"}, {'Ь', "BBBАА"}, {'Э', "BBBАB"}, {'Ю', "BBBBА"},

        {'Я', "BBBBB"},

        {'а', "ааааа"}, {'б', "ааааb",}, {'в', "аааbа"}, {'г', "аааbb"},

        {'д', "ааbаа"}, {'е', "ааbаb"}, {'ё', "ааbаb"}, {'ж', "ааbbа"},

        {'з', "ааbbb"}, {'и', "аbааа"}, {'й', "аbааb"}, {'к', "аbаbа"},

        {'л', "аbаbb"}, {'м', "аbbаа"}, {'н', "аbbаb"}, {'о', "аbbbа"},

        {'п', "аbbbb"}, {'р', "bаааа"}, {'с', "bаааb"}, {'т', "bааbа"},

        {'у', "bааbb"}, {'ф', "bаbаа"}, {'х', "bаbаb"}, {'ц', "bаbbа"},

        {'ч', "bаbbb"}, {'ш', "bbааа"}, {'щ', "bbааb"}, {'ъ', "bbаbа"},

        {'ы', "bbаbb"}, {'ь', "bbbаа"}, {'э', "bbbаb"}, {'ю', "bbbbа"},

        {'я', "bbbbb"}

    };

    string encryptedMessage;

    for (char c : message) {

        if (baconMap.find(c) != baconMap.end()) {

            encryptedMessage += baconMap[c];

        } else {

            encryptedMessage += c; // Non-alphabet characters remain unchanged

        }

    }

    return encryptedMessage;

}

// Функция расшифровки сообщения с помощью шифра Бэкона

string baconDecryption(const string& message) {

    map<string, char> baconMap = {

        {"AAAAA", 'A'}, {"AAAAB", 'B'}, {"AAABA", 'C'}, {"AAABB", 'D'},

        {"AABAA", 'E'}, {"AABAB", 'F'}, {"AABBA", 'G'}, {"AABBB", 'H'},

        {"ABAAA", 'I'}, {"ABAAB", 'J'}, {"ABABA", 'K'}, {"ABABB", 'L'},

        {"ABBAA", 'M'}, {"ABBAB", 'N'}, {"ABBBA", 'O'}, {"ABBBB", 'P'},

        {"BAAAA", 'Q'}, {"BAAAB", 'R'}, {"BAABA", 'S'}, {"BAABB", 'T'},

        {"BABAA", 'U'}, {"BABAB", 'V'}, {"BABBA", 'W'}, {"BABBB", 'X'},

        {"BBAAA", 'Y'}, {"BBAAB", 'Z'},

        {"aaaaa", 'a'}, {"aaaab", 'b'}, {"aaaba", 'c'}, {"aaabb", 'd'},

        {"aabaa", 'e'}, {"aabab", 'f'}, {"aabba", 'g'}, {"aabbb", 'h'},

        {"abaaa", 'i'}, {"abaab", 'j'}, {"ababa", 'k'}, {"ababb", 'l'},

        {"abbaaa", 'm'}, {"abbab", 'n'}, {"abbba", 'o'}, {"abbbb", 'p'},

        {"baaaa", 'q'}, {"baaab", 'r'}, {"baaba", 's'}, {"baabb", 't'},

        {"babaa", 'u'}, {"babab", 'v'}, {"babba", 'w'}, {"babbb", 'x'},

        {"bbaaa", 'y'}, {"bbaab", 'z'},

        {"ААААА", 'А'}, {"ААААB", 'Б'}, {"АААBА", 'В'}, {"АААBB", 'Г'},

        {"ААBАА", 'Д'}, {"ААBАB", 'Е'}, {"ААBАB", 'Ё'}, {"ААBBА", 'Ж'},

        {"ААBBB", 'З'}, {"АBААА", 'И'}, {"АBААB", 'Й'}, {"АBАBА", 'К'},

        {"АBАBB", 'Л'}, {"АBBАА", 'М'}, {"АBBАB", 'Н'}, {"АBBBА", 'О'},

        {"АBBBB", 'П'}, {"BАААА", 'Р'}, {"BАААB", 'С'}, {"BААBА", 'Т'},

        {"BААBB", 'У'}, {"BАBАА", 'Ф'}, {"BАBАB", 'Х'}, {"BАBBА", 'Ц'},

        {"BАBBB", 'Ч'}, {"BBААА", 'Ш'}, {"BBААB", 'Щ'}, {"BBАBА", 'Ъ'},

        {"BBАBB", 'Ы'}, {"BBBАА", 'Ь'}, {"BBBАB", 'Э'}, {"BBBBА", 'Ю'},

        {"BBBBB", 'Я'},

        {"ааааа", 'а'}, {"ааааb", 'б'}, {"аааbа", 'в'}, {"аааbb", 'г'},

        {"ааbаа", 'д'}, {"ааbаb", 'е'}, {"ааbаb", 'ё'}, {"ааbbа", 'ж'},

        {"ааbbb", 'з'}, {"аbааа", 'и'}, {"аbааb", 'й'}, {"аbаbа", 'к'},

        {"аbаbb", 'л'}, {"аbbаа", 'м'}, {"аbbаb", 'н'}, {"аbbbа", 'о'},

        {"аbbbb", 'п'}, {"bаааа", 'р'}, {"bаааb", 'с'}, {"bааbа", 'т'},

        {"bааbb", 'у'}, {"bаbаа", 'ф'}, {"bаbаb", 'х'}, {"bаbbа", 'ц'},

        {"bаbbb", 'ч'}, {"bbааа", 'ш'}, {"bbааb", 'щ'}, {"bbаbа", 'ъ'},

        {"bbаbb", 'ы'}, {"bbbаа", 'ь'}, {"bbbаb", 'э'}, {"bbbbа", 'ю'},

        {"bbbbb", 'я'}

    };

    string decryptedMessage;

    string filteredMessage;

    // Удаление недопустимых символов

    for (char c : message) {

        if (c == 'A' || c == 'B' || c == 'a' || c == 'b' || c == 'А' || c == 'B' || c == 'а' || c == 'b') {

            filteredMessage += c;

        }

    }

    // Дешифровка

    for (size\_t i = 0; i < filteredMessage.length(); i += 5) {

        string code = filteredMessage.substr(i, 5);

        if (baconMap.find(code) != baconMap.end()) {

            decryptedMessage += baconMap[code];

        } else {

            decryptedMessage += '?'; // Неизвестный код

        }

    }

    return decryptedMessage;

}

**a1z26.cpp**

#include <sstream>

#include "Cryptosystem.h"

typedef unsigned char uc;

int toInt(uc c) {

    int res = int(c);

    if (c >= 'a' && c <= 'z') {

        res = int(c) - 96;

    }

    if (c >= 'A' && c <= 'Z') {

        res = int(c) - 38;

    }

    if (c >= uc('А') && c <= uc('Я')) {

        res = int(c) - 106;

    }

    if (c >= uc('а') && c <= uc('я')) {

        res = int(c) - 171;

    }

    if (c >= ' ' && c <= '@') {

        res = int(c) + 100;

    }

    return res;

}

char toChar(string c) {

    int sym = stoi(c);

    uc res = 0;

    if (sym >= 1 && sym <= 26) {

        res = uc(sym + 96);

    }

    if (sym >= 27 && sym <= 53) {

        res = uc(sym + 38);

    }

    if (sym >= 86 && sym <= 118) {

        res = uc(sym + 106);

    }

    if (sym >= 53 && sym <= 85) {

        res = uc(sym + 171);

    }

    if (sym >= 132 && sym <= 164) {

        res = uc(sym - 100);

    }

    return res;

}

string a1z26Encryption(string text) {

    string crypto;

    for (int i = 0; i < text.size(); i++) {

        crypto += to\_string(toInt(text[i])) + '-';

    }

    crypto.pop\_back();

    return crypto;

}

string a1z26Decryption(string crypto) {

    string text;

    string sym;

    for (int i = 0; i < crypto.size(); i++) {

        if (crypto[i] != '-') {

            sym += crypto[i];

        }

        else {

            text += toChar(sym);

            sym = "";

        }

    }

    text += toChar(sym);

    sym = "";

    return text;

}

**shamir.cpp**

#include <sstream>

#include "Cryptosystem.h"

typedef unsigned char uc;

// Глобальные переменные для алгоритма Шамира

static int p;   // Простое число

static int cA;  // Секретный ключ Алисы

static int dA;  // Обратный ключ Алисы

// Функция для вычисления наибольшего общего делителя (НОД)

int gcd(int a, int b) {

    while (b != 0) {

        int t = b;

        b = a % b;

        a = t;

    }

    return a;

}

// Функция для вычисления мультипликативного обратного числа по модулю m

int modInverse(int a, int m) {

    int m0 = m, t, q;

    int x0 = 0, x1 = 1;

    if (m == 1)

        return 0;

    while (a > 1) {

        q = a / m;

        t = m;

        m = a % m;

        a = t;

        t = x0;

        x0 = x1 - q \* x0;

        x1 = t;

    }

    if (x1 < 0)

        x1 += m0;

    return x1;

}

// Функция для быстрого возведения в степень по модулю

long long modPow(long long base, long long exponent, long long modulus) {

    if (modulus == 1)

        return 0;

    long long result = 1;

    base = base % modulus;

    while (exponent > 0) {

        if (exponent % 2 == 1)

            result = (result \* base) % modulus;

        exponent = exponent >> 1; // Делим показатель на 2

        base = (base \* base) % modulus;

    }

    return result;

}

// Функция для генерации ключей для алгоритма Шамира

void generateShamirKeys() {

    // Выбираем простое число p

    p = 257; // Пример простого числа

    // Подбираем секретный Алисы cA

    cA = 5;

    // Вычисляем обратный ключ dA, такой что cA \* dA ? 1 (mod p-1)

    dA = modInverse(cA, p - 1);

}

// Функция для проверки ввода для шифра Шамира

vector<char> checkinputshamir(const string& message) {

    vector<char> invalidChars;

    for (char c : message) {

        // Допустимые символы: буквы (A-Z, a-z) и пробелы

        if (!((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') ||

              (c == ' ') || (c >= 'А' && c <= 'Я') || (c >= 'а' && c <= 'я') ||

              (c == '!') ||(c == '@') || (c == '"') || (c == '#') ||

              (c == '$') || (c == ';') || (c == '%') || (c == '^') ||

              (c == ':') || (c == '&') || (c == '?') || (c == '\*') ||

              (c == '~') || (c == '(') || (c == ')') || (c == '-') ||

              (c == '\_') || (c == '+') || (c == '=') || (c== '<') ||

              (c== '>') || (c == '/') || (c == '|') || (c == '.') ||

              (c == ',') || (c == '`'))) {

            invalidChars.push\_back(c);

        }

    }

    return invalidChars;

}

// Функция шифрования сообщения с помощью алгоритма Шамира

string shamirEncryption(const string& message) {

    // Преобразуем сообщение в числа

    vector<int> m\_numbers;

    for (uc c : message) {

        if (c >= 'A' && c <= 'Z') {

            m\_numbers.push\_back(c - 'A' + 1); // Преобразуем A-Z в 1-26

        } else if (c >= 'a' && c <= 'z') {

            m\_numbers.push\_back(c - 'a' + 27); // Преобразуем a-z в 27-52

        } else if (c == ' ') {

            m\_numbers.push\_back(0); // Представляем пробел как ''

        } else if (c >= uc('А') && c <= uc('Я')) {

            m\_numbers.push\_back(c - uc('А') + 53); // Преобразуем А-Я в 53-85

        } else if (c >= uc('а') && c <= uc('я')) {

            m\_numbers.push\_back(c - uc('а') + 86); // Преобразуем а-я в 86-118

        } else {

            m\_numbers.push\_back(119 + (int)c); // Специальные символы обрабатываем как 119-<n>, где <n> - порядковый номер символа

        }

    }

    // Шифруем: C = M^cA mod p

    vector<int> C\_numbers;

    for (int m : m\_numbers) {

        int C = modPow(m, cA, p);

        C\_numbers.push\_back(C);

    }

    // Преобразуем зашифрованные числа в строку

    string encryptedMessage;

    for (int C : C\_numbers) {

        encryptedMessage += to\_string(C) + " ";

    }

    // Удаляем последний пробел

    if (!encryptedMessage.empty()) {

        encryptedMessage.pop\_back();

    }

    return encryptedMessage;

}

// Функция расшифровки сообщения с помощью алгоритма Шамира

string shamirDecryption(const string &message) {

    // Преобразуем зашифрованное сообщение из строки в числа

    vector<int> C\_numbers;

    stringstream ss(message);

    string token;

    while (ss >> token) {

        try {

            int C = stoi(token);

            C\_numbers.push\_back(C);

        } catch (invalid\_argument&) {

            // Если встретился некорректный токен, можно обработать ошибку

        }

    }

    // Расшифровываем: M = C^dA mod p

    vector<int> m\_numbers;

    for (int C : C\_numbers) {

        int m = modPow(C, dA, p);

        m\_numbers.push\_back(m);

    }

    // Преобразуем числа обратно в символы

    string decryptedMessage;

    for (int m : m\_numbers) {

        if (m == 0) {

            decryptedMessage += ' ';

        } else if (m >= 1 && m <= 26) {

            decryptedMessage += (char)('A' + m - 1);

        } else if (m >= 27 && m <= 52) {

            decryptedMessage += (char)('a' + m - 27);

        } else if (m >= 53 && m <= 85) {

            decryptedMessage += (char)('А' + (m - 53));

        } else if (m >= 86 && m <= 118) {

            decryptedMessage += (char)('а' + (m - 86)); // Русские строчные буквы

        } else if (m >= 119) {

            decryptedMessage += (char)(m - 119); // Обратное преобразование для спец. символов

        }

    }

    return decryptedMessage;

}