Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Отчёт

по учебной (ознакомительной) практике

Студент:

гр. 250501 Гнетецкий Д. Г.

Руководитель:

старший преподаватель кафедры ЭВМ

Ковальчук А. М.

МИНСК 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc136333798)

[1 ПОСТАНОВКА ЗАДАЧИ 4](#_Toc136333799)

[2 ОБЗОР ЛИТЕРАТУРЫ 5](#_Toc136333800)

[2.1 Принципы работы алгоритма RSA 6](#_Toc136333801)

[2.2 Принцип работы шифра четырех квадратов 6](#_Toc136333802)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 7](#_Toc136333803)

[3.1 Структура входных данных 7](#_Toc136333804)

[3.2 Структура выходных данных 9](#_Toc136333805)

[4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 10](#_Toc136333806)

[4.1. Разработка схем алгоритмов 10](#_Toc136333807)

[4.2 Разработка алгоритмов 11](#_Toc136333808)

[4.3 Код программы 12](#_Toc136333809)

[5 РЕЗУЛЬТАТЫ РАБОТЫ 12](#_Toc136333810)

[ЗАКЛЮЧЕНИЕ 16](#_Toc136333811)

[ПРИЛОЖЕНИЕ А 18](#_Toc136333812)

[ПРИЛОЖЕНИЕ Б 19](#_Toc136333813)

[ПРИЛОЖЕНИЕ В 20](#_Toc136333814)

*P.S: Короче нужно соблюдать все госты. Делать все четко как тут, все картинки подписывать по центру, стараться не выходить за поля при растягивании картинки. Смотреть чтобы было тире. Слева для разделов отступ табуляцией. Везде вставлять пробельные строки. Код обязательно в приложение и при его вставке нужно как-то там уменьшить табуляцию на 0.5.*

*Страницы с кодом, титульный лист и содержание не нумеруются*

# ВВЕДЕНИЕ

Кафедра "Вычислительные машины, системы и сети" является выпускающей по специальности 40 02 01 и ориентирована на подготовку инженеров-системотехников в области информационных технологий и вычислительной техники. Кафедра была основана в 1964 году под названием "Математические и счетно-решающие приборы и устройства". В 1970 году она была переименована в кафедру "Электронные вычислительные машины".

Язык программирования C был разработан в 1972 году и считается одним из самых популярных и важных языков программирования в мире. Он отличается высокой эффективностью, гибкостью и способностью работать на низком уровне. Язык C предоставляет программистам прямой доступ к памяти компьютера, что позволяет писать оптимизированный и быстрый код. Однако, с этой возможностью также приходит ответственность за правильное управление памятью и обработку указателей. Программистам на C необходимо быть внимательными и осторожными при работе с памятью, чтобы избежать ошибок и утечек памяти.

Одной из ключевых особенностей C является его поддержка модульности и раздельной компиляции. Это означает, что программы могут быть разделены на отдельные модули, которые могут быть компилированы независимо друг от друга. Это упрощает разработку проектов и повторное использование кода.

Язык программирования С широко применяется во множестве областей. Он является основным языком для разработки операционных систем, компиляторов, встроенных систем и микроконтроллеров. Он также используется для создания системного и прикладного программного обеспечения, игр, алгоритмов и многого другого.

Шифрование текста является важным аспектом в современном информационном обществе, где защита конфиденциальности и безопасность данных играют ключевую роль. Существует множество алгоритмов шифрования, каждый из которых обладает своими особенностями и применяется в различных сферах. В данном отчете мы рассмотрим два из них, а именно алгоритм RSA и шифр четырех квадратов.

# 

# 1 ПОСТАНОВКА ЗАДАЧИ

Цель данного программного модуля заключается в разработке алгоритмов для шифрования и дешифрования текста с использованием двух алгоритмов: RSA и шифра четырех квадратов.

Модуль должен обеспечивать понятный пользовательский интерфейс с необходимыми пунктами меню. Информация о тексте и ключах шифрования должна храниться в нескольких файлах. Модуль должен реализовывать функциональность генерации случайных ключей для шифрования, пользовательского решения шифрования, а также получения текста из файлов и записи его в файлы.

Необходимо разработать алгоритмы шифрования и дешифрования текста с использованием RSA и шифра четырех квадратов, разработать пользовательский интерфейс с меню, позволяющий выбрать необходимую операцию (ввод текста, работа с файлами, шифрование, дешифрование и т.д.). Затем необходимо протестировать модуль на различных сценариях использования и проверить его корректность и надежность.

# 2 ОБЗОР ЛИТЕРАТУРЫ

## 2.1 Принципы работы алгоритма RSA

Необходимо выбрать два разных больших простых числа p, q. Вычислить n = p\*q. Затем взять небольшое нечётное число e, взаимно простое с функцией Эйлера φ(n) = (p − 1)(q − 1). После необходимо вычислить d — элемент, обратный по модулю φ(n), к числу e.

Пара {e, n} — это открытый ключ; пара {d, n} — закрытый ключ.

Шифрование сообщения «x» с помощью открытого ключа происходит по формуле: E(x) = xe mod n.

Дешифрование сообщения с помощью закрытого ключа происходит по формуле: D(y) = yd mod n.

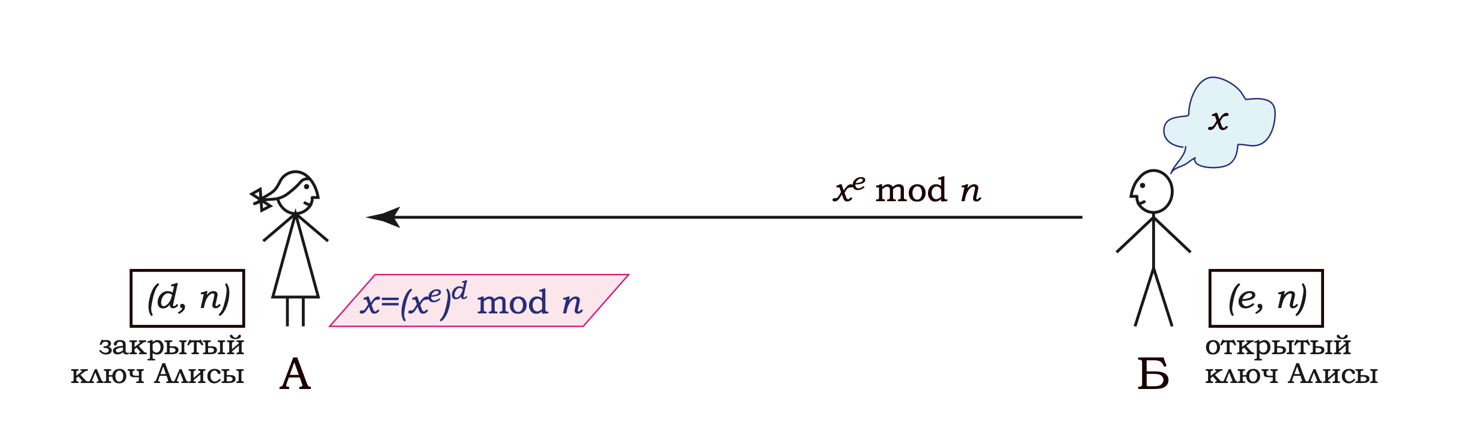
На рисунке 2.1 наглядно показан принцип шифрования сообщения: Боб посылает Алисе сообщение «x», зашифрованное её открытым ключом {e, n}; Алиса расшифровывает сообщение, используя свой закрытый ключ {d, n}.

Рисунок 2.1.1 – шифрование сообщения алгоритмом RSA

## 

Рисунок 2.1 – Принцип Шифрования алгоритмом RSA

## 2.2 Принцип работы шифра четырех квадратов

Шифр четырех квадратов является полиграфическим шифром, использующим матрицы и подстановки для шифрования и дешифрования текста. Полиграфический шифр - это метод шифрования, который основывается на замене букв или символов в открытом тексте на другие буквы или символы с помощью определенных правил и таблиц.

Шифр четырех квадратов основан на использовании четырех матриц, каждая из которых содержит буквы алфавита и некоторые символы. Обычно это матрицы 6x6.

Для шифрования сообщения, оно разбивается на пары букв. Затем каждая пара букв заменяется соответствующей парой букв из матриц. Необходимо выбрать первую букву в левом верхнем квадрате, вторую букву в правом нижнем квадрате и затей достроить до прямоугольника, найдя буквы, которыми будет выполняться шифрование. Данный принцип представлен на рисунке 2.2

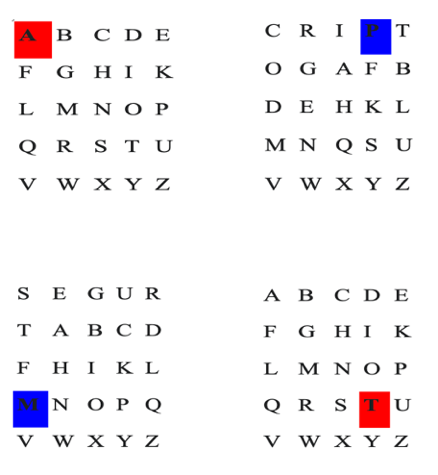


Рисунок 2.2 – Принцип шифрование шифром четырех квадратов

Для расшифрования сообщения необходимо выполнить обратную операцию.

# 3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

## 3.1 Структура входных данных

Программа принимает на вход массив структур, который считывается из файла. Этот массив содержит информацию о зашифрованных файлах. Каждая структура имеет следующие поля, представленные на рисунке 3.1.

struct dataCode{  
 char fileName[N]; //имя зашифрованного файла  
 long e; //открытая экспонента для шифрования алгоритмом RSA  
 long d; //закрытая экспонента для дешифрования  
 long n; //часть открытого и закрытого ключа в алгоритме RSA  
 int flag; //флажок для определения, каким алгоритмом зашифрован файл  
 char box2[SIZE][SIZE]; //ключ для шифрования шифром четырех квадратов  
 char box3[SIZE][SIZE]; //ключ для шифрования шифром четырех квадратов  
};

Рисунок 3.1 – Поля структуры

Таким образом, программа обрабатывает информацию о зашифрованных файлах, используя указанные поля в каждой структуре, чтобы правильно расшифровать каждый файл.

Входными данными программы также является массив строк char\*\* text, который требуется закодировать с использованием алгоритмов RSA или шифра четырех квадратов.

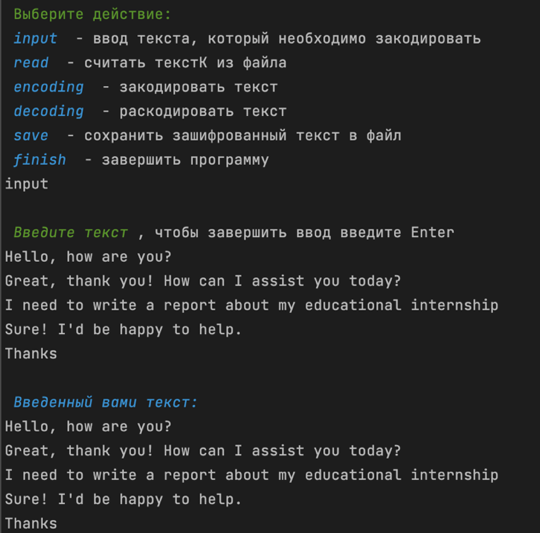
На рисунке 3.2 показан процесс ввода данных с клавиатуры

Рисунок 3.2 – Ввод данных с клавиатуры

На рисунке 3.3 показан процесс чтения данных из файла

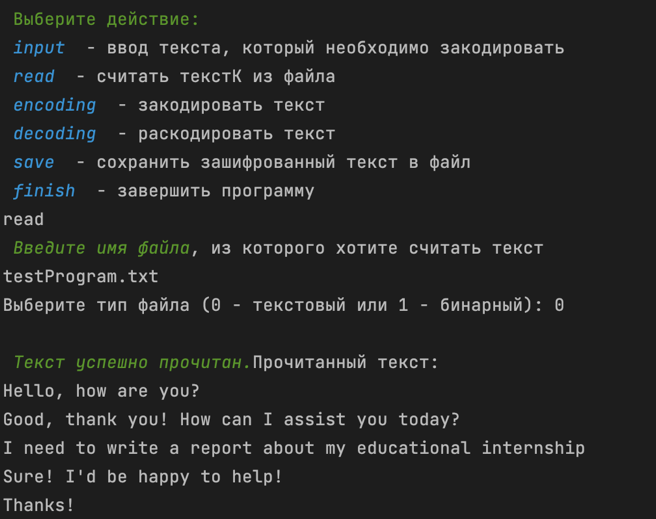


Рисунок 3.3 – Чтение данных из файла

## 3.2 Структура выходных данных

В данном подразделе представлены выходные данные программы. Текст, зашифрованный с помощью алгоритма RSA хранится в двумерном массиве чисел long \*\*codeText, а данные, зашифрованные шифром четырех квадратов в массиве строк char \*\*newText.

На рисунке 3.4 показан результат шифрования текста при помощи алгоритма RSA. Исходный текст был успешно зашифрован с использованием открытого ключа, и получен зашифрованный текст, который представлен в виде цифровой последовательности. Это демонстрирует эффективность алгоритма RSA при обеспечении конфиденциальности данных.

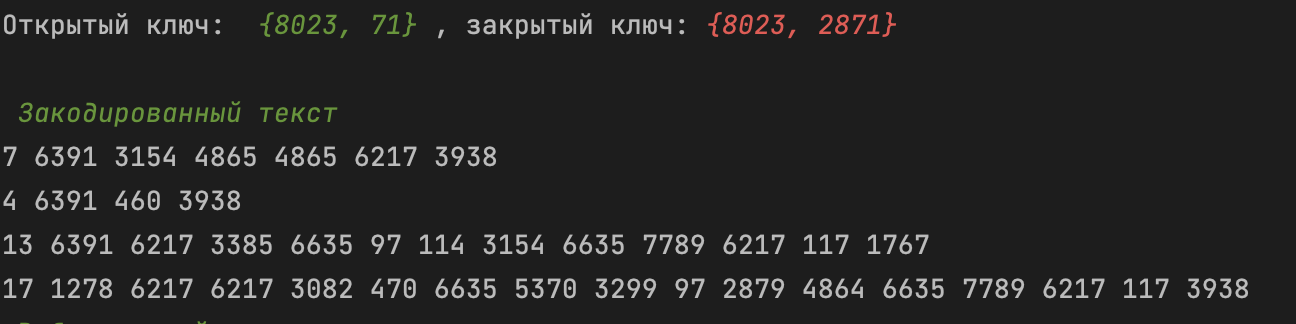


Рисунок 3.4 – Зашифрованный текст алгоритмом RSA

Рисунок 3.5 отображает результат шифрования текста с помощью шифра четырех квадратов. Исходный текст был преобразован с использованием матриц символов, что привело к получению символов, а ни цифр, как в первом алгоритме.

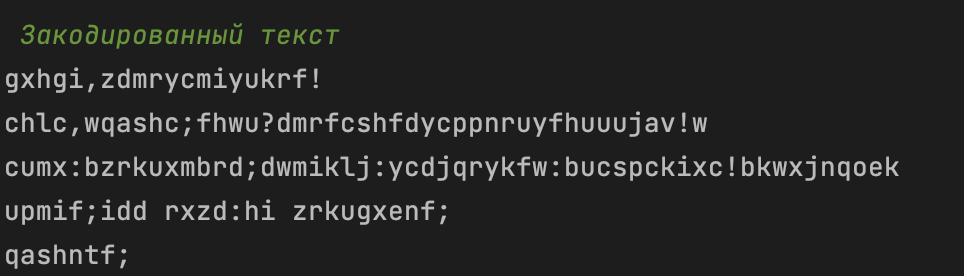


Рисунок 3.5 – Зашифрованный текст шифром четырех квадратов

Выходные данные, представленные в данном разделе, подтверждают успешное шифрование текста с помощью алгоритма RSA и шифра четырех квадратов. Эти скриншоты позволяют визуально оценить работу программы и убедиться в правильности применения выбранных алгоритмов шифрования.

# 4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

## 4.1. Разработка схем алгоритмов

Схема алгоритма программы main.c представлена в приложении А.

Функция decodeText() дешифрует текст, закодированный шифром четырех квадратов. Схема алгоритма представлена в приложении Б.

## 4.2 Разработка алгоритмов

Алгоритм по шагам функции coding(), которая шифрует текст алгоритмом RSA:

Переменные, передаваемые в функцию:

* mas - указатель на двумерный массив символов, который надо зашифровать
* e - значение открытой экспоненты
* n - значение модуля функции Эйлера в алгоритме RSA
* rows - количество строк в массиве

Выходные данные: текст, закодированный алгоритмом RSA, который хранится в двумерном массиве code типа long\*\*

1. Начало.
2. Проверить, если mas равен NULL, вывести сообщение "Вы не ввели текст, который надо зашифровать" и вернуть NULL.
3. Выделить память для двумерного массива cods размером rows типа long\*\*.
4. Цикл по строкам от j = 0 до rows.
5. Объявить переменную k равной 1, это будет позиция для записи элемента в строку cods[j].
6. Объявить переменную size, длину текущей строки, и инициализировать ее длиной строки mas[j] с помощью функции strlen().
7. Выделить память для строки cods[j] размером size + 1 элементов типа long. Плюс один надо, чтобы хранить длину строки.
8. Записать в cods[j][0] значение size + 1, представляющее количество элементов в строке.
9. Цикл по i = 0 до нуль-терминатора по строке mas[j].
10. Преобразовать символ mas[j][i] в ASCII-код и сохранить в переменную s.
11. Закодировать символ s с использованием функции powerMod(s, e, n) и сохранить результат в cods[j][k].
12. Увеличить значение k на 1 для перехода к следующему элементу.
13. Конец цикла по i.
14. Конец цикла по j.
15. Вернуть cods.
16. Конец.

Алгоритм по шагам функции decodingRSA(), которая дешифрует текст, зашифрованный алгоритмом RSA:

Переменные, передаваемые в функцию:

* codeText - указатель на двумерный массив типа long
* d - значение закрытой экспоненты
* n - значение модуля функции Эйлера в алгоритме RSA
* rows - количество строк в массиве

Выходные данные: текст, закодированный алгоритмом RSA, который хранится в двумерном массиве code типа long\*\*

1. Начало.
2. Выделить память для массива строк decodedText размером rows типа char\*\*.
3. Цикл по строкам от i = 0 до rows.
4. Получить количество элементов cols в текущей строке из codeText[i][0].
5. Выделить память для строки decodedText[i] размером (cols + 1) элементов типа char. Плюс один элемент для символа нуль-терминатора '\0'.
6. Цикл по элементам от j = 1 до cols.
7. Декодировать текущий элемент codeText[i][j] с использованием функции powerMod(codeText[i][j], d, n) и сохранить результат в переменной s.
8. Преобразовать числовое значение s в символ и сохранить его в decodedText[i][j-1].
9. Конец цикла по j.
10. Записать в последний элемент decodedText[i][cols] нуль-терминатор '\0'.
11. Конец цикла по i.
12. Вернуть decodedText.
13. Конец.

## 4.3 Код программы

Код программы представлен в приложении В.

# 5 РЕЗУЛЬТАТЫ РАБОТЫ

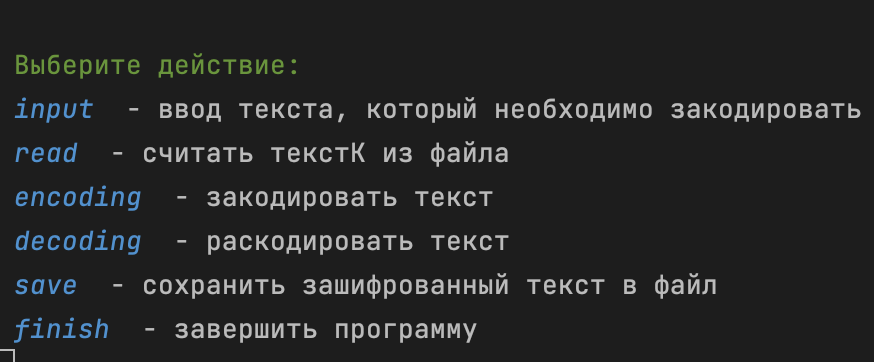


Рисунок 5.1 – Интерфейс программы

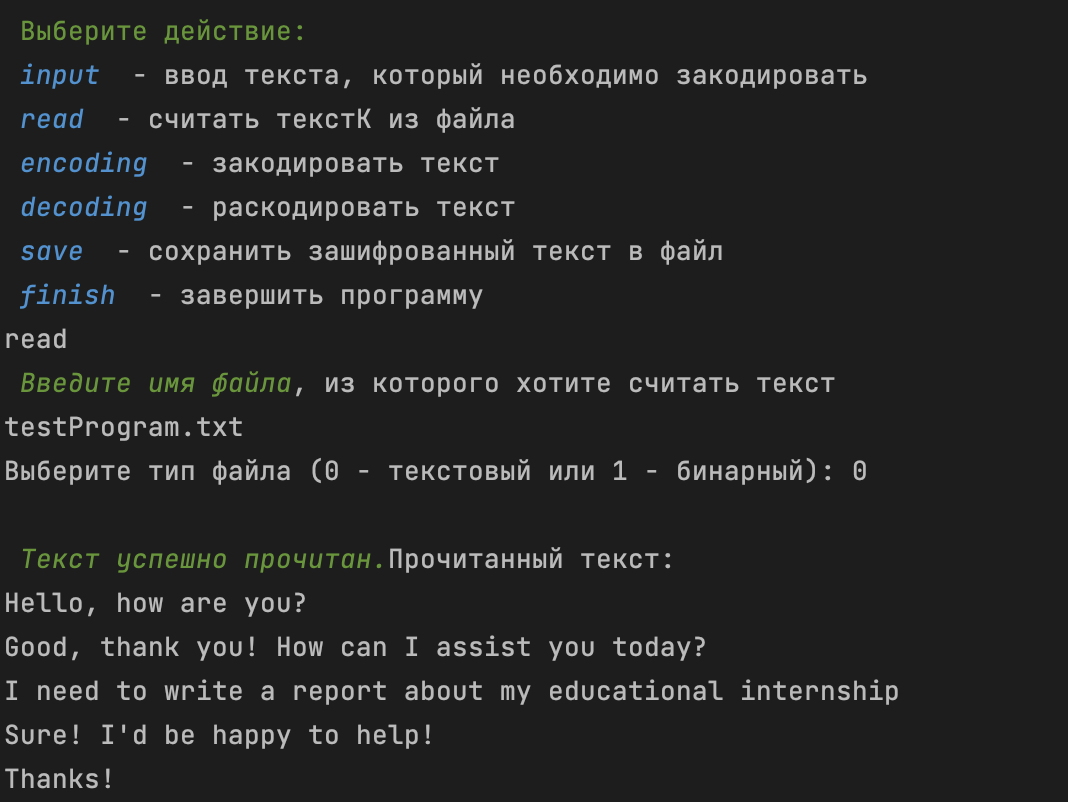


Рисунок 5.2 – Чтение данных из файла

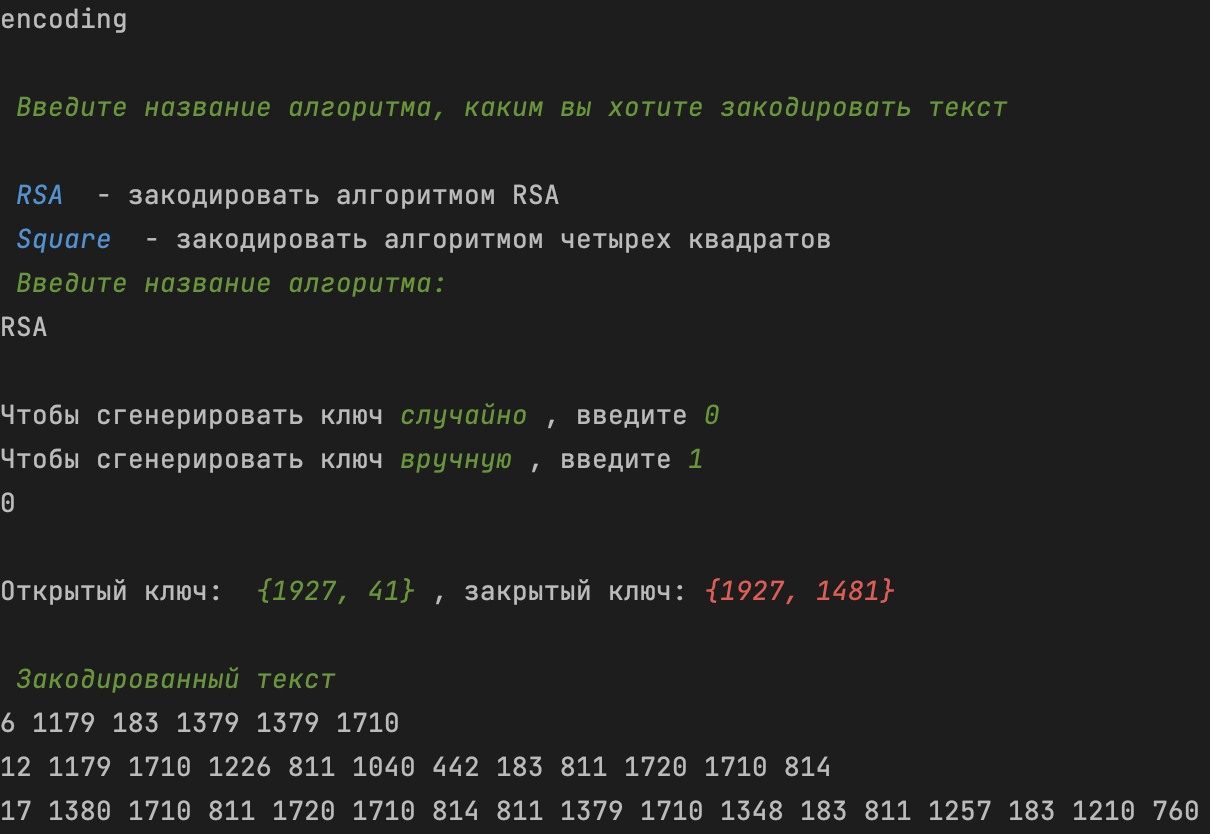


Рисунок 5.3 – Кодирование текста алгоритмом RSA

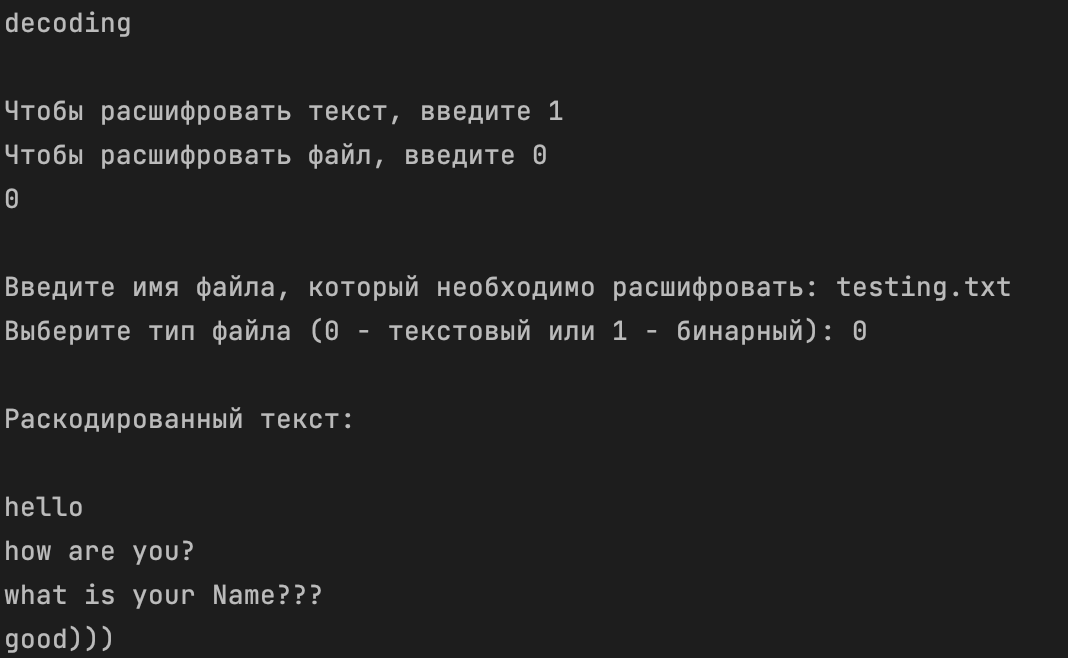


Рисунок 5.4 – Декодирование файла

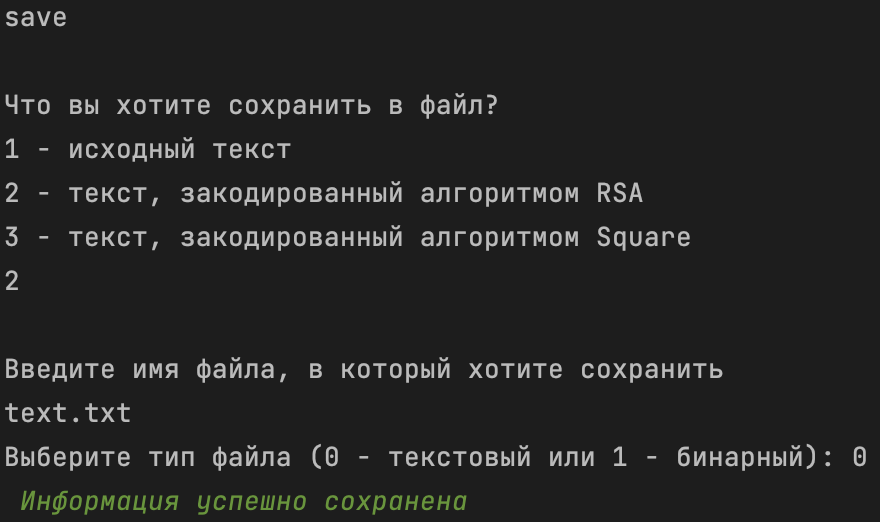
**

Рисунок 5.5 – Сохранение данных в файл

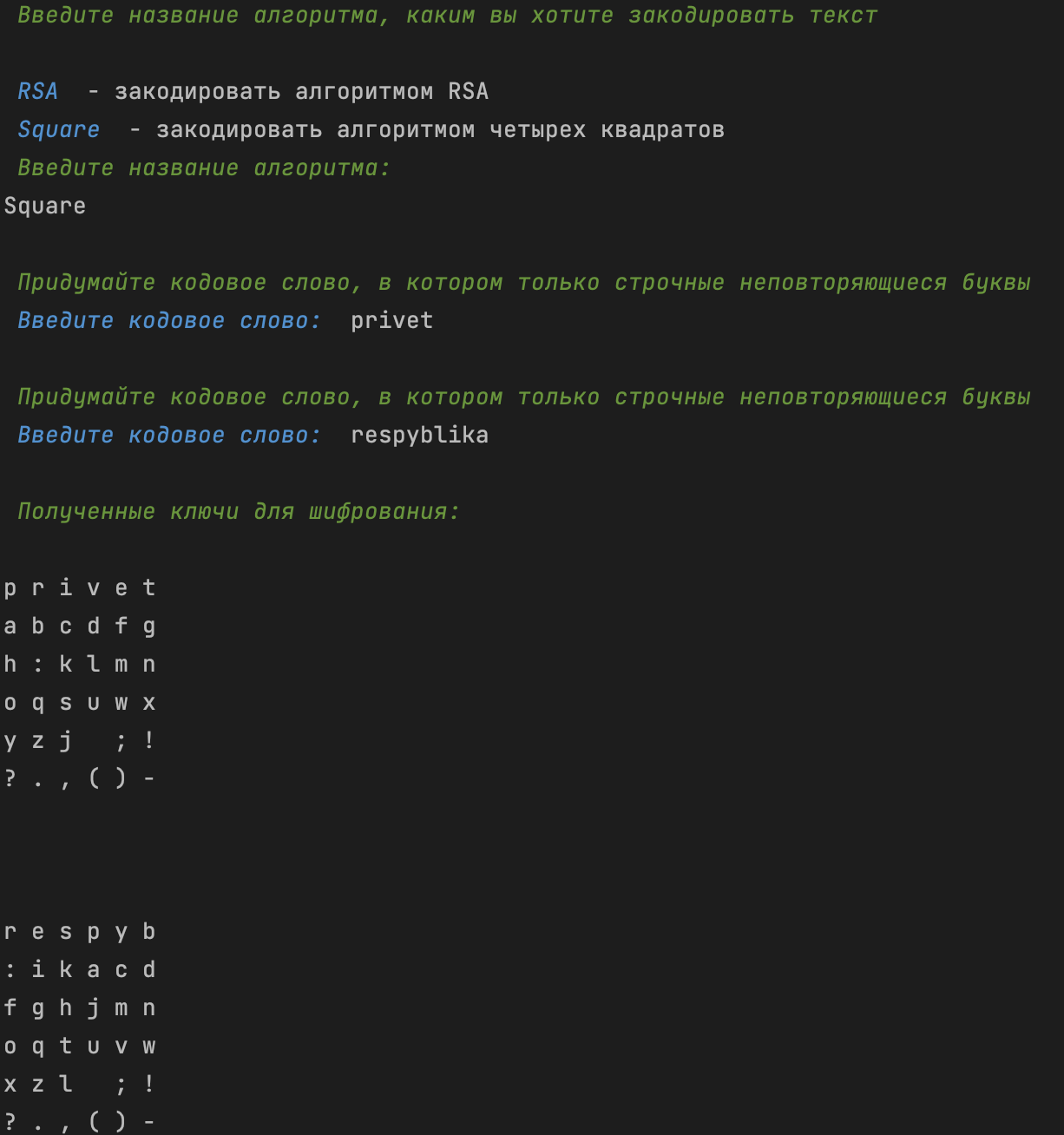


Рисунок 5.6 – Генерация ключа для шифра четырех квадратов

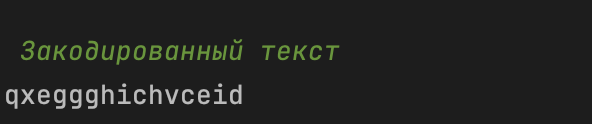


Рисунок 5.7 – Результат кодирования шифром четырех квадратов

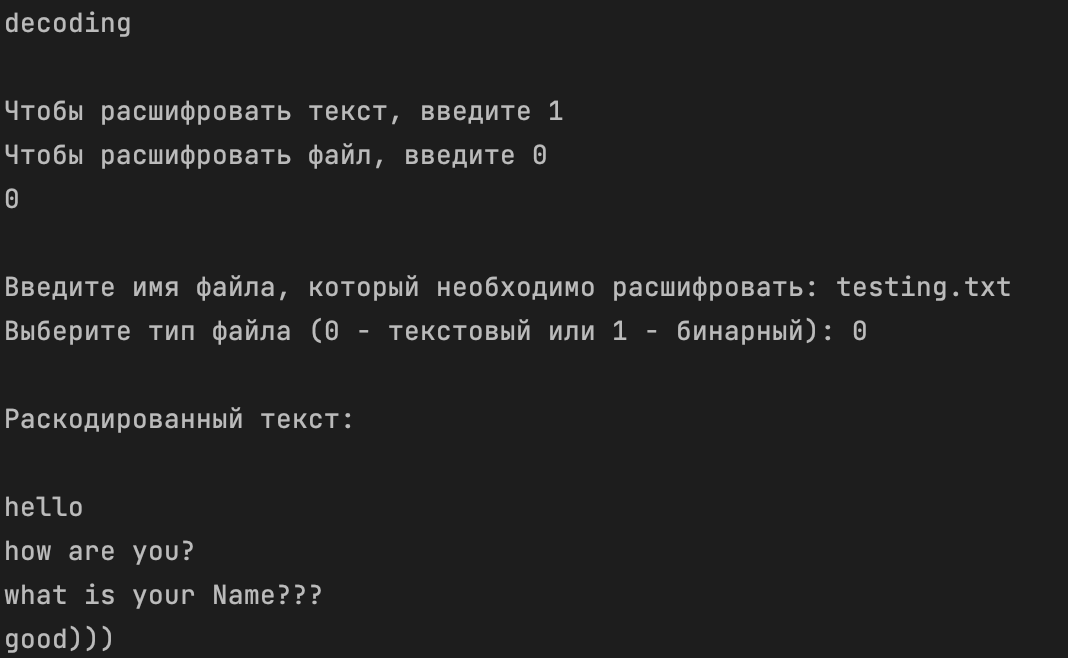


Рисунок 5.8 – Декодирование файла

# ЗАКЛЮЧЕНИЕ

В рамках данной работы был успешно разработан алгоритм и его программная реализация на языке С для шифрования текста алгоритмом RSA и шифром четырех квадратов, с возможностью работы с файлами. Разработанный алгоритм обеспечивает эффективное и безопасное шифрование данных, а также точную обработку информации, что позволяет обеспечить высокую степень защиты и надежность программы.

Выполнение данной учебной практики позволило не только разработать алгоритмы шифрования, но и улучшить навыки работы с криптографическими алгоритмами, а также понять принципы функционирования и применения шифров в информационных системах. Полученные знания и опыт могут быть использованы в дальнейшей работе над созданием и развитием безопасных систем передачи и хранения данных.

В ходе проведенных экспериментов было выявлено, что время шифрования текста размером 400 символов алгоритмом RSA составляет приблизительно 0.000063 секунды, в то время как алгоритмом четырех квадратов - 0.000070 секунды.

На основании полученных результатов можно сделать вывод о том, что оба алгоритма обеспечивают высокую скорость шифрования. Однако, алгоритм RSA показывает незначительное преимущество по времени работы по сравнению с алгоритмом четырех квадратов.

Следует отметить, что эффективность алгоритма RSA обусловлена его математическим подходом и возможностью обеспечить высокий уровень безопасности. В то же время, алгоритм четырех квадратов предлагает более простой подход к шифрованию, что может быть удобным в некоторых случаях.

В целом, эффективность алгоритмов шифрования зависит от конкретных требований и контекста использования. При выборе между алгоритмом RSA и алгоритмом четырех квадратов следует учитывать как требования к безопасности, так и требования к скорости работы системы.

Программа была написана на операционной системе Windows 11, в среде разработке CLion – 2023. Минимальные требования для запуска программы – Windows 7, Visual Studio 2015.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Луцик, Ю. А. Основы алгоритмизации и программирования : язык Си : учебно-метод. пособие / Ю. А. Луцик, А. М. Ковальчук, Е. А. Сасин. – Минск : БГУИР, 2015. – 170с. : ил.

2. Керниган, Б. Язык программирования С / Б. Керниган, Д. Ритчи. – 3-е изд. – М. : Издательский дом «Вильямс», 2013. – 304 с.

3. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – СПб. : БХВ-Петербург, 2008. – 440 с.

4. Стивен Прата Язык программирования С. Лекции и упражнения, 6-е изд. : Пер. с англ. —М. :ООО “И.Д. Вильямс”, 2015. — 928 с.

5. Теоретическая информатика: [Электронный ресурс]. – Режим доступа:

<https://users.mathcs.spbu.ru/~okhotin/teaching/tcs2_2019/okhotin_tcs2alg_2019_l5.pdf>Дата доступа: 19.04.2023.

# ПРИЛОЖЕНИЕ А

*(обязательное)*

Схема алгоритма main.c

# ПРИЛОЖЕНИЕ Б

*(обязательное)*

Схема алгоритма функции decodeSquare()

# ПРИЛОЖЕНИЕ В

*(обязательное)*

Код программы

**Код файла headerPractice.h**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include <stdbool.h>

#define SIZE 6 //SIZE - размеры квадратов для шифрования

#define N 21 //N - длина имени файла

#define MAX\_LENGTH 80 //MAX\_LENGTH - максимальная длина строки

struct dataCode{

char fileName[N]; //имя зашифрованного файла

long e; //открытая экспонента для шифрования RSA

long d; //закрытая экспонента для дешифрования

long n; //часть открытого и закрытого ключа RSA

int flag; //флажок для выбора алгоритма шифрования

char box2[SIZE][SIZE]; //ключ для шифрования шифром четырех квадратов

char box3[SIZE][SIZE]; //ключ для шифрования шифром четырех квадратов

};

enum choiceCommand {input, read, encoding, decoding, save, finish};

///// //////////////////Функции для шифрования алгоритмом RSA////////////////////

//функция проверки числа на простоту

long isPrimes (long x);

//функция создания простого числа

void createPrime (long \*x);

//функция создания чисел p, q, n, fi

void createNumbers (long \*p, long \*q, long \*n, long \*fi);

//функция создания числа e - части открытого ключа

void createEilerNumber (long \*e, long fi);

//функция нахождения НОД алгоритмом Евклида

long NOD (long x, long y);

//расширенный алгоритм Евклида для нахождения обратного по модулю числа

void search\_d(long\* x, long e, long fi);

//функция бинарного возведения в степень по модулю

long powerMod(long x, long y, long n);

//функция кодирования текста алгоритмом RSA

long\*\* coding (char\*\* mas, long e, long n, long rows);

//функция декодирования текста алгоритмом RSA

char\*\* decodingRSA(long\*\* codeText, long d, long n, long rows);

///////////////Функции для шифрования алгоритмом четырех квадратов///////////////

//функция для создания квадрата

void writeBox(char box[SIZE][SIZE], int n);

//функция вывода шифра для шифра четырех квадратов

void outputBox (char box[SIZE][SIZE]);

//кодирование текста алгоритмом четырех квадратов

char\*\* codingSquare(char , char, char, char, char\*\* , int);

//функция декодирования текста алгоритмом четырех квадратов

char\*\* decodeSquare(char , char , char , char , char\*\*, long);

//функция проверки, является ли символ буквой

int checkLetter (char s);

//функция проверки текста на возможность шифрования алгоритмом четырех квадратов

int checkTextForSquare (char\*\* text, long rows);

//функция вставки элемента в случайную позицию

void pushRandom(char box[SIZE][SIZE], char s, int i, int j);

//функция поиска индексов символа в квадрате

void searchLetterIndex (char box[SIZE][SIZE], int \*i, int\* j, int n, char s);

//функция выбора команды

void choiceTask (enum choiceCommand \*x , const char\* tasks[], bool \*taskIsFound);

///////////Основные функции вызов функций для кодирования и декодирования/////////

//функция чтения квадрата из файла

void readInfo(char box[SIZE][SIZE], int n, const char\* name);

//основная функция для кодирования текста

struct dataCode encodingText (char\*\*, int, long \*\*\*, char \*\*\*);

//декодирование текста из файла

char\*\* decodingTextFromFile(long\*\*, char\*\*, struct dataCode\*, int, char , long\*);

//функция декодирования текста, введенного с клавиатуры

char\*\* decodingInputText(char box[SIZE][SIZE], long\* rows);

//////////////////////////Функции работы с файлами///////////////////////////////

//функция сохранения массива структур с данными о зашифрованных файлах

void saveStructInfo (struct dataCode\* key, long count);

//функция чтения массива структур с данными о зашифрованных файлах

void readStructInfo(struct dataCode\*\* key, long\* count);

//функция сохранения двумерного массива чисел в файл

void saveNumberToFile(long\*\* code, long rows, char\* fileName);

//функция чтение зашифрованного текста RSA из файла

long\*\* loadNumberFromFile(long\*\* code, long\* rows, char\* fileName);

//функция сохранения текста в файл

void saveTextToFile(char\*\* text, long rows, char\* fileName);

//функция считывания текста в двумерный массив из файла

char\*\* loadTextFromFile(long\* rows, char\* fileName);

//функция перевода двумерного массива чисел в массив строк

long\*\* myAtoi2DText(char\*\* text, long rows);

/////////////Вспомогательные функции которые вызывают другие функции/////////////

//функция для кодирования текста

void caseEncoding (struct dataCode \*\*keys, long \*countKeys, long\*\*\* codeText, char\*\* text, int n, char\*\*\* newText);

//функция для декодирования текста

void caseDecoding (char \*\*\*text, long \*\*codeText, char \*\*newText, struct dataCode \*keys, long countKeys, long \*rows);

//функция для сохранения информации в файл

void saveInformation (char\*\*, char\*\*, long\*\*, long, struct dataCode \*, long );

//функция чтения текста из файла

void caseReadText (char\*\*\* text, long\* rows);

//функция очистки памяти

void freeMemory(char\*\*, char\*\*, long\*\*, struct dataCode\*, long, long);

/////////////////////Вспомогательные функции из библиотеки///////////////////////

//функция ввода строки произвольной длины

void inputStr (char\*\* mas);

//функция ввода числа типа int

void inputInt (int\* x, int a, int b);

//функция вода числа типа long

void inputLong (long\* x, int a, int b);

//функция ввода массива строк с клавиатуры

void input2DString (char\*\*\* text, long\* n);

//функция вывода двумерного массива чисел long\*\* на экран

void output2DNumbers (long\*\* mas, long n);

//функция вывода массива строк на экран

void output2DString (char\*\* mas, int n);

//функция удаления символа '\n' из строки

void deleteSymbolN (char\*\* str);

**Код файла main.c**

#include "headerPractice.h"

int main()

{

//text - исходный текст, newText - текст зашифрованный Square

char \*\*text = NULL, \*\*newText = NULL;

//keys - структура с данными о зашифрованных файлах

struct dataCode\* keys;

//codeText - текст RSA, n - кол-во строк исходного текста

long \*\*codeText = NULL, countKeys, rows, n;

//countKeys - кол-во структур, rows - кол-во строк текста

const char\* commands[] = {"input", "read", "encoding", "decoding", "save", "finish"};

//command - перечисления для читабельности кода

enum choiceCommand command;

; //TaskIsFound - переменная для определения правильности команды

bool taskIsFound

; //end - переменная для продолжения/завершения программы

int end = 1

//считываю информацию о зашифрованных файлах в keys

readStructInfo(&keys, &countKeys);

do

{

choiceTask(&command, commands, &taskIsFound); //выбор команды

if (taskIsFound)

{

switch(command)

{

case input:

input2DString(&text, &n); //ввод текста с клавиатуры

break;

case read:

caseReadText(&text, &n); //чтение текста из файла

break;

case encoding: //кодирование текста

caseEncoding(&keys, &countKeys, &codeText, text, n, &newText);

break;

case decoding: //декодирование текста

caseDecoding(&text, codeText, newText, keys, countKeys, &rows);

break;

case save: //сохранение данных в файл

saveInformation(text, newText, codeText, n, keys, countKeys);

break;

case finish:

saveStructInfo(keys, countKeys); //сохранение данных

end = 0;

break;

}

}

else

{

printf("\n\033[3;31m Команда введена неверно \033[0m\n");

}

}while(end);

freeMemory(text, newText, codeText, keys, countKeys, n); //очистка памяти

return 0;

}

**Код файла functionsPractice.c**

#include "headerPractice.h"

///////////////////////функции для шифрования алгоритмом RSA/////////////////////

//функция проверки числа на простоту

long isPrimes (long x)

{

if(x == 0) return 0;

for(int i = 2; i < x; i++)

if (x%i == 0) return 0;

return 1;

}

//функция создания простого числа

void createPrime (long \*x)

{

srand(time(NULL));

\*x = rand()%10;

\*x = (\*x)\*(\*x) - (\*x) + 41;

}

//функция создания чисел p, q, n, fi

void createNumbers (long \*p, long \*q, long \*n, long \*fi)

{

createPrime(p); //создать простое число p

do

{

createPrime(q); //создать простое число q, не равное p

}while(\*q == \*p);

\*n = (\*p) \* (\*q); //создать число n - часть приватного и публичного ключа

\*fi = ((\*p) - 1)\*((\*q) - 1); //создать число fi, которое будем нужно для создания ключей

}

//функция создания числа e - части открытого ключа

void createEilerNumber (long \*e, long fi)

{

do //создать простое число e, взаимно простое с fi, e >= fi

{

createPrime(e);

}while(NOD(\*e, fi) != 1 && \*e >= fi);

}

//функция нахождения НОД алгоритмом Евклида

long NOD (long x, long y)

{

if (y == 0)

return x;

else

return NOD (y, x%y);

}

//расширенный алгоритм Евклида для нахождения обратного по модулю числа

void search\_d(long\* x, long e, long fi)

{

long m0 = fi; //сохраняю значение fi

long y = 0, xPrev = 1; //инициализирую начальные значения коэффициентов

while (e > 1)

{

long q = e / fi; //вычислить частное от деления e на fi

long t = fi; //временная переменная для сохранения значения fi

fi = e % fi; //вычисля остаток от деления e на fi

e = t; //обновить значение e для следующей итерации

t = y; //сохранить предыдущее значение y

y = xPrev - q \* y;

xPrev = t;

}

if (xPrev < 0)

{

xPrev += m0; //если результат отрицательный, сделать положительным

}

\*x = xPrev;

}

//функция бинарного возведения в степень по модулю

long powerMod(long x, long y, long n)

{

if(y == 0)

return 1; //при y=0 вернуть 1

if (y%2 == 0) //если степень четная, рекурсивно вызвать для степени y/2

{

long z = powerMod(x, y/2, n);

return (z\*z)%n; //вернуть остаток от деления

}

else

return (x \* powerMod(x, y-1, n))%n; // рекурсивно вызвать для y-1

}

//функция кодирования текста алгоритмом RSA

long\*\* coding (char\*\* mas, long e, long n, long rows)

{

if (mas == NULL) //проверка, ввел ли пользователь текст

{

printf("\nВы не ввели текст, который надо зашифровать\n");

return NULL;

}

long \*\*cods = (long \*\*)calloc(rows, sizeof(long\*)); //выделить память

for(int j = 0; j < rows; j++) //цикл по строкам

{

int k = 1; //k - позиция с которой начинается запись

long size = strlen(\*(mas + j)); //определяем длину строки

\*(cods + j) = (long \*)calloc(size + 1, sizeof(long));

cods[j][0] = size + 1; //записываю первым элементов кол-во элементов в строке

for (int i = 0; mas[j][i] != '\0'; i++)

{

long s = (long) (mas[j][i]); //перевожу символ в ASCII - код

cods[j][k] = powerMod(s, e, n); //кодирую символ

k++; //переход на следующий элемент для записи

}

}

return cods;

}

//функция декодирования текста алгоритмом RSA

char\*\* decodingRSA(long\*\* codeText, long d, long n, long rows)

{

char\*\* decodedText = (char\*\*)malloc(rows \* sizeof(char

for (long i = 0; i < rows; i++)

{

long cols = codeText[i][0]; //получаю кол-во элементов в строке

decodedText[i] = (char\*)malloc((cols + 1) \* sizeof(char));

for (long j = 1; j < cols; j++)

{

long s = powerMod(codeText[i][j], d, n); //декодирую элемент

decodedText[i][j-1] = (char)s; //преобразую число в символ

}

decodedText[i][cols] = '\0';

}

return decodedText;

}

///////////////////функции для шифрования алгоритмом четырех квадратов///////////

//функция для создания квадрата

void writeBox(char box[SIZE][SIZE], int n)

{

int k = 0; //индекс для записи букв в квадрат

char\* word = NULL; //word - кодовое слово

char letters[26] = "abcdefghijklmnopqrstuvwxyz"; //алфавит

int used[26] = {0}, flag; //used отвечает за повторение букв

do

{

flag = 0;

printf("\033[3;34m Введите кодовое слово: \033[0m");

inputStr(&word); //считываю кодовое слово

for (int i = 0; word[i] != '\0'; i++) //цикл для проверки

{

for (int j = i + 1; word[j - 1] != '\0'; j++)

{

if ((word[i] == word[j]) || !checkLetter(word[i]))

{

flag = 1;

break;

}

}

if (flag) {break;}

}

if(flag){printf("\n1mКодовое слово введено неверно\n");}

} while (flag);

for (int i = 0; word[i] != '\0'; i++) //записываю буквы слова в квадрат

{

used[word[i] - 'a'] = 1; //отмечаю букву как использованную

box[k / SIZE][k % SIZE] = word[i]; //записываю букву в квадрат

k++;

}

for (int i = 0; i < 26; i++) //записываю оставшиеся буквы

{

if (used[letters[i] - 'a'] == 0) //если буква не была использована

{

box[k / SIZE][k % SIZE] = letters[i]; //записываю букву в квадрат

used[letters[i] - 'a'] = 1; //отмечаю букву как использованную

k++;

}

}

pushRandom (box, ' ', 4, 2); //случайным образом вставляю символы

pushRandom (box, ';', 4, 3);

pushRandom (box, '!', 4, 4);

pushRandom (box, '?', 4, 5);

pushRandom (box, '.', 5, 0);

pushRandom (box, ',', 5, 1);

pushRandom (box, '(', 5, 2);

pushRandom (box, ')', 5, 3);

pushRandom (box, '-', 5, 4);

pushRandom (box, ':', 5, 5);

}

//функция вывода шифра для шифра четырех квадратов

void outputBox (char box[SIZE][SIZE])

{

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

{

printf("\n");

for (int j = 0; j < SIZE; j++)

{

printf("%c ", box[i][j]);

}

}

printf("\n\n\n");

}

//кодирование текста алгоритмом четырех квадратов

char\*\* codingSquare(char box1[SIZE][SIZE], char box2[SIZE][SIZE], char box3[SIZE][SIZE], char box4[SIZE][SIZE], char \*\*text, int size)

{

for(int c = 0; c < size; c++) //цикл по строкам

{

int n = strlen(text[c]); //получаю длину текста

int i = 0, j = 0, k = 0, l = 0; //переменные для хранения индексов букв

if (strlen(text[c]) % 2 == 1) //если длина текста нечетная

{

n += 2; //расширяю строку

char\* tmp = (char\*)realloc(text[c], (n) \* sizeof(char));

if (tmp != NULL)

{

text[c] = tmp;

text[c][n - 2] = ' '; //добавляю пробел

text[c][n - 1] = '\0'; //добавляю символ конца строки

}

else

{

printf("\n\033[1;41m\033[1mОшибка памяти\033[0m\n");

exit(EXIT\_FAILURE);

}

}

for (int z = 0; z < n - 1; z++)

{

if (checkLetter(text[c][z])) //если символ является буквой

{

i = (text[c][z] - 'a') / SIZE; //вычисляю индекс строки

j = (text[c][z] - 'a') % SIZE; //вычисляю индекс столбца

}

else

{

searchLetterIndex(box1, &i, &j, SIZE, text[c][z]); //ищу индексы

}

z++;

if (checkLetter(text[c][z])) //если символ является буквой

{

k = (text[c][z] - 'a') / SIZE; //вычисляю индекс строки

l = (text[c][z] - 'a') % SIZE; //вычисляю индекс столбца

}

else

{

searchLetterIndex(box4, &k, &l, SIZE, text[c][z]); //ищу индексы

}

char x1, x2;

x1 = box2[i][l]; //получаю символ из квадрата box2

x2 = box3[k][j]; //получаю символ из квадрата box3

text[c][z - 1] = x1; //заношу закодированные буквы

text[c][z] = x2; //заношу закодированные буквы в массив

}

}

return text;

}

//функция декодирования текста алгоритмом четырех квадратов

char\*\* decodeSquare(char box1[SIZE][SIZE], char box2[SIZE][SIZE], char box3[SIZE][SIZE], char box4[SIZE][SIZE], char\*\* text, long rows)

{

for (int c = 0; c < rows; c++) //цикл по строкам

{

int i = 0, j = 0, k = 0, l = 0, n = strlen(text[c]);

for (int z = 0; z < n; z++)

{

searchLetterIndex(box2, &i, &j, SIZE, text[c][z]); //ищу индексы

searchLetterIndex(box3, &k, &l, SIZE, text[c][z + 1]); //ищу индексы

char x1 = box1[i][l]; //получаю символ из квадрата box1

char x2 = box4[k][j]; //получаю символ из квадрата box4

text[c][z] = x1; //делаю замены в исходном массиве

text[c][z + 1] = x2; //делаю замены в исходном массиве

z++;

}

}

return text;

}

//функция проверки, является ли символ буквой

int checkLetter (char s)

{

int flag = 1;

if(s < 'a' || s > 'z') //если не строчная буква, вернуть 0, иначе 1

flag = 0;

return flag;

}

//функция проверки текста на возможность шифрования алгоритмом четырех квадратов

int checkTextForSquare (char\*\* text, long rows)

{

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

{

for (int j = 0; text[i][j] != '\0'; j++)

{

if(!checkLetter(text[i][j]) && text[i][j] != '\0')

{

return 0;

}

}

}

return 1;

}

//функция вставки элемента в случайную позицию

void pushRandom(char box[SIZE][SIZE], char s, int i, int j)

{

int l, m;

srand(time(NULL));

l = rand() % SIZE; //генерирую случайную строку в квадрате

m = rand() % SIZE; //генерирую случайный столбец в квадрате

char letter = box[l][m]; //сохраняю символ, который будет заменен

box[l][m] = s; //заменяю символ случайной позиции на новый символ s

box[i][j] = letter; //заменяю символ в позиции (i, j) на сохраненный символ

}

//функция поиска индексов символа в квадрате

void searchLetterIndex (char box[SIZE][SIZE], int \*i, int\* j, int n, char s)

{

int flag = 0;

for ((\*i) = 0; (\*i) < n; (\*i)++) //цикл по строкам

{

for ((\*j) = 0; (\*j) < n; (\*j)++) //цикл по символам

{

if (box[(\*i)][(\*j)] == s) //если нашел символ, выйти из цикла

{

flag = 1;

break;

}

}

if (flag == 1) {break;}

}

}

//функция выбора команды

void choiceTask (enum choiceCommand \*doTask, const char\* tasks[], bool \*taskIsFound)

{

char\* task = NULL;

printf("\n\033[32m Выберите действие: \033[0m\n");

printf("\033[3;34m input \033[0m - ввод текста \n");

printf("\033[3;34m read \033[0m - считать текстК из файла\n");

printf("\033[3;34m encoding \033[0m - закодировать текст\n");

printf("\033[3;34m decoding \033[0m - раскодировать текст\n");

printf("\033[3;34m save \033[0m - сохранить зашифрованный текст в файл\n");

printf("\033[3;34m finish \033[0m - завершить программу\n");

rewind(stdin);

inputStr(&task);

for(\*doTask = input; \*doTask < finish; (\*doTask)++)

{

if(strcmp(task, tasks[\*doTask]) == 0)

{

\*taskIsFound = true;

break;

}

}

}

//функция чтения квадрата из файла

void readInfo(char box[SIZE][SIZE], int n, const char\* name)

{

FILE\* file;

char s; //символ в который считывается

int j; //параметр по символам строки

file = fopen(name, "r");

if (file == NULL)

{

printf("Ошибка открытия файла");

exit(EXIT\_FAILURE);

}

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

{

j = 0;

while ((s = getc(file)) != '\n' && !feof(file))

{

box[i][j] = s; //сохраняю считанный символ

j++;

}

}

fclose(file);

}

//////////Основные функции вызов функций для кодирования и декодирования//////////

//основная функция для кодирования текста

struct dataCode encodingText (char\*\* text, int n, long \*\*\*codeText, char \*\*\*newText)

{

struct dataCode key = {0};

if(text == NULL)

{

printf("\033[1;31m Вы не ввели текст: \033[0m");

return key;

}

//key - новая структура хранящая данные о зашифрованном тексте

char\* task = NULL; //task - переменная для выбора задачи

int choice;

long p, q, fi; //p, q, fi - переменные для шифрования алгоритмом RSA

long \*\*code = (long\*\*)calloc(n, sizeof(long\*));//code - двумерный массив кода

char \*\*codingText = (char\*\*)calloc(n, sizeof(char\*));

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

\*(codingText + i) = (char\*)calloc(strlen(text[i]), sizeof(char));

char box1[SIZE][SIZE], box2[SIZE][SIZE], box3[SIZE][SIZE], box4[SIZE][SIZE];

printf("\n\033[3;32m Введите название алгоритма \033[0m\n");

printf("\n\033[3;34m RSA \033[0m - закодировать алгоритмом RSA");

printf("\n\033[3;34m Square \033[0m - закодировать алгоритмом Square");

do

{

printf("\n\033[3;32m Введите название алгоритма: \033[0m\n");

inputStr(&task);

if (strcmp(task, "RSA") == 0)

{

key.flag = 1; //помечаю, каким алгоритмом зашифровано

printf("\nЧтобы сгенерировать ключ случайно, введите 0");

printf("\nЧтобы сгенерировать ключвручную, введите 1\n");

inputInt(&choice, 0, 1); //выбор, как создать ключ

if (choice)

{

do //создаю нужные числа для шифрования RSA

{

printf("\033[3;34m Введите простое число p - \033[0m");

inputLong(&p,3, 100000);

} while (!isPrimes(p)); //создаю число p

do

{

printf("\033[3;34m Введите простое число q - \033[0m");

inputLong(&q, 3, 100000);

} while (!isPrimes(q)); //создаю число q

key.n = p \* q; //создаю n - модуль для расшифровки

fi = (p - 1) \* (q - 1); //создаю функцию Эйлера

createEilerNumber(&(key.e), fi); //создаю e - открытую экспоненту

search\_d(&(key.d), key.e, fi); //создаю d - закрытую экспоненту

}

else //иначе сгенерировать ключи случайно

{

createNumbers(&p, &q, &key.n, &fi);

createEilerNumber(&key.e, fi); //создать закрытую экспоненту e

search\_d(&key.d, key.e, fi); //создать число d - часть ключа

}

}

else if (strcmp(task, "Square") == 0)

{

if(!checkTextForSquare(text, n))

{

printf("Данный текст не подходит для шифрования алгоритмом Square");

\*newText = NULL;

\*codeText = NULL;

return key;

}

key.flag = 0;

readInfo(box1, SIZE, "box1.txt"); //считать box1, box4 из файла

readInfo(box4, SIZE, "box4.txt");

writeBox(box2, SIZE); //сгенерировать box2 и box3 пользователем

writeBox(box3, SIZE);

printf("\n\033[3;32m Полученные ключи для шифрования: \033[0m\n");

outputBox(box2);

outputBox(box3);

for (int i = 0; i < SIZE; i++) //сохранить ключи (box2) в структуру

for(int j = 0; j < SIZE; j++)

key.box2[i][j] = box2[i][j];

for (int i = 0; i < SIZE; i++) //сохранить ключи (box3) в структуру

for(int j = 0; j < SIZE; j++)

key.box3[i][j] = box3[i][j];

}

else

{

printf("\nНазвание алгоритма неверное");

key.flag = 2;

}

}while(key.flag == 2);

if (key.flag)

{

code = coding(text, key.e, key.n, n); //закодировать текст

codingText = NULL;

}

else

{

//закодировать текст алгоритмом четырех квадратов

codingText = codingSquare(box1, box2, box3, box4, text, n);

code = NULL; //указатель на другой массив поставить в NULL

}

\*newText = codingText;

\*codeText = code;

return key;

}

//декодирование текста из файла

char\*\* decodingTextFromFile(long\*\* codeText, char\*\* newText, struct dataCode\* keys, int count, char box[SIZE][SIZE], long\* rows)

{

int flag = 0; //flag - флаг для проверки наличия файла

int i;

char \*\*text = NULL; //text - расшифрованный текст

char\* fileName = NULL; //fileName - имя файла для расшифровки

printf("\nВведите имя файла, который необходимо расшифровать: ");

inputStr(&fileName); //ввод имени файла для расшифровки

for (i = 0; i < count; i++) //проверка, есть ли файл с таким именем

{

if (strcmp(fileName, keys[i].fileName) == 0)

{

flag = 1;

break;

}

}

if (flag) //если файл найден

{

FILE\* file;

file = fopen(keys[i].fileName, "rt");

if (file != NULL)

{

int dataType = keys[i].flag

if (dataType == 1)

{

//загрузка числовых данных из файла

codeText = loadNumberFromFile(codeText, rows, fileName);

//расшифровка алгоритмом RSA

text = decodingRSA(codeText, keys[i].d, keys[i].n, \*rows);

}

else if (dataType == 0)

{

//загрузка текстовых данных из файла

newText = loadTextFromFile(rows, fileName);

//расшифровка методом четырех квадратов

text = decodeSquare(box, keys[i].box2, keys[i].box3, newText);

}

fclose(file);

}

else

{

printf("\nОшибка при открытии файла для чтения\n");

}

}

else

{

printf("\nФайл с таким именем не найден\n");

}

return text;

}

//функция декодирования текста, введенного с клавиатуры

char\*\* decodingInputText(char box[SIZE][SIZE], long\* rows)

{

char\* string; //string - строка для выбора алгоритма шифрования

char\*\* text; //text - текст для расшифровки

char\* name1, \*name2; //имя файлов для хранения ключей box2 и box3

char box2[SIZE][SIZE], box3[SIZE][SIZE]; //ключи для шифрования

long\*\* code; //числовое представление текста для алгоритма RSA

long d, n; //ключи для расшифровки алгоритма RSA

int flag = 0; //flag - флаг для проверки корректного ввода

do

{

printf("\nКаким алгоритмом зашифрован текст? (RSA или Square)\n");

inputStr(&string); //ввод алгоритма шифрования

if (strcmp("RSA", string) == 0 || strcmp("Square", string) == 0)

{

flag = 1;

}

else

{

printf("\n\033[3;31m Команда введена неверно");

}

} while (!flag);

if (strcmp("RSA", string) == 0)

{

printf("\nВведите через пробел цифры");

input2DString(&text, rows); //ввод закодированного текста

code = myAtoi2DText(text, \*rows); //преобразование текста в числа

printf("\nВведите ключ для расшифровки\nВведите число d - ");

inputLong(&d, 0, 10000); //ввод значения d

printf("Введите число n - ");

inputLong(&n, 0, 10000); //ввод значения n

text = decodingRSA(code, d, n, \*rows); //расшифровка текста

}

else

{

printf("\nВведите закодированный текст\n");

input2DString(&text, rows); //ввод закодированного текста

printf("\nВведите имя файла, в котором хранится box2 - ");

inputStr(&name1); //ввод имени файла для box2

readInfo(box2, SIZE, name1); //чтение информации из файла в box2

printf("\nВведите имя файла, в котором хранится box3 - ");

inputStr(&name2); //ввод имени файла для box3

readInfo(box3, SIZE, name2); //чтение информации из файла в box3

text = decodeSquare(box, box2, box3, box, text, \*rows);

}

return text;

}

/////////////////////////////Функции работы с файлами////////////////////////////

//функция сохранения массива структур с данными о зашифрованных файлах в файл

void saveStructInfo (struct dataCode\* key, long count)

{

FILE\* file;

file = fopen("keyInfo", "wb");

if (file != NULL)

{

fwrite(&count, sizeof(long), 1, file); //записываю кол-во элементов

fwrite(key, sizeof(struct dataCode), count, file); //записываю весь массив

}

else

printf("\nОшибка при открытии бинарного файла для записи ключей");

fclose(file);

}

//функция чтения массива структур с данными о зашифрованных файлах

void readStructInfo(struct dataCode\*\* key, long\* count)

{

FILE\* file;

file = fopen("keyInfo", "rb");

if (file != NULL)

{

fread(count, sizeof(long), 1, file); //считываю количество элементов

\*key = (struct dataCode\*)calloc((\*count), sizeof(struct dataCode));

fread(\*key, sizeof(struct dataCode), \*count, file); //Считываю массив

fclose(file);

}

else

printf (“\nОшибка при открытии бинарного файла для чтения ключей");

}

//функция сохранения двумерного массива чисел в файл

void saveNumberToFile(long\*\* code, long rows, char\* fileName)

{

int format; //переменная для выбора формата: бинарный или текстовый

printf("Выберите тип файла (0 - текстовый или 1 - бинарный): ");

inputInt(&format, 0, 1);

FILE\* file;

if (!format)

{

file = fopen(fileName, "wt"); //открываю текстовый файл для записи

if (file != NULL)

{

fprintf(file, "%ld\n", rows); //записываю количество строк

for (long i = 0; i < rows; i

{

long cols = code[i][0]; //получаю количество элементов

fprintf(file, "%ld ", cols); //записываю количество элементов

for (long j = 1; j <= cols; j

{

fprintf(file, "%ld ", code[i][j]); //Записываю элементы строки

}

fprintf(file, "\n");

}

fclose(file);

}

else

{

printf("\nОшибка при открытии текстового файла для записи кода\n");

}

}

else

{

file = fopen(fileName, "wb"); //открываю бинарный файл для записи

if (file != NULL)

{

fwrite(&rows, sizeof(long), 1, file); //записываю количество строк

for (long i = 0; i < rows; i++)

{

long cols = code[i][0];

fwrite(&cols, sizeof(long), 1, file);//записываю кол-во элементов

for (long j = 1; j <= cols; j++)

{

fwrite(&code[i][j], sizeof(long), 1, file

}

}

fclose(file);

}

else

{

printf("\nОшибка при открытии бинарного файла для записи кода\n");

}

}

}

//функция чтение зашифрованного текста RSA из файла

long\*\* loadNumberFromFile(long\*\* code, long\* rows, char\* fileName)

{

int format; //переменная для выбора типа файла: бинарный или текстовый

printf("Выберите тип файла (0 - текстовый или 1 - бинарный): ");

inputInt(&format, 0, 1);

FILE\* file;

if (!format) //если текстовый файл

{

file = fopen(fileName, "rt"); //открываю текстовый файл для чтения

if (file != NULL)

{

if (fscanf(file, "%ld", rows) == 1) //читаю количество строк из файла

{

code = (long\*\*)malloc((\*rows) \* sizeof(long

for (long i = 0; i < \*rows; i++)

{

long cols;

if (fscanf(file, "%ld", &cols) == 1)//читаю кол-во элементов

{

code[i] = (long\*)malloc((cols + 1) \* sizeof(long));

code[i][0] = cols; //сохраняю количество

for (long j = 1; j <= cols; j++)

{

fscanf(file, "%ld", &code[i][j]); //читаю элементы

}

}

else

{

printf("Ошибка чтения файла\n");

fclose(file);

return NULL;

}

}

}

else

{

printf("Ошибка чтения файла\n");

}

fclose(file);

}

else

{

printf("Ошибка при открытии текстового файла для чтения\n");

}

}

else

{

file = fopen(fileName, "rb"); //открываю бинарный файл для чтения

if (file != NULL)

{

if (fread(rows, sizeof(long), 1, file) == 1) //читаю количество строк

{

code = (long\*\*)malloc((\*rows) \* sizeof(long\*));

for (long i = 0; i < \*rows; i++)

{

long cols;

if (fread(&cols, sizeof(long), 1, file) == 1)

{

code[i] = (long\*)malloc((cols + 1) \* sizeof(long));

code[i][0] = cols;

for (long j = 1; j <= cols; j++)

{

if (fread(&code[i][j], sizeof(long), 1, file) != 1)

{ //читаю элементы строки из файла

printf("Ошибка чтения файла\n");

fclose(file);

return NULL;

}

}

}

else

{

printf("Ошибка чтения файла\n");

fclose(file);

return NULL;

}

}

}

else

{

printf("Ошибка чтения файла\n");

}

fclose(file);

}

else

{

printf("Ошибка при открытии бинарного файла для чтения\n");

}

}

return code;

}

//функция сохранения текста в файл

void saveTextToFile(char\*\* text, long rows, char\* fileName)

{

int format; //переменная для определения типа файла

printf("Выберите тип файла (0 - текстовый или 1 - бинарный): ");

inputInt(&format, 0, 1);

FILE\* file;

if (!format)

{

file = fopen(fileName, "w"); //открываю текстовый файл для записи

if (file != NULL)

{

fprintf(file, "%ld\n", rows); //записываю количество строк

for (long i = 0; i < rows; i++)

{

fprintf(file, "%s\n", text[i]); //записываю строку

}

fclose(file);

}

else

{

printf("Ошибка при открытии текстового файла для записи\n");

}

}

else

{

file = fopen(fileName, "wb"); //открываю бинарный файл для записи

if (file != NULL)

{

fwrite(&rows, sizeof(long), 1, file); //записываю количество строк

for (long i = 0; i < rows; i++)

{

long len = strlen(text[i]); //получаю длину строки

fwrite(&len, sizeof(long), 1, file); //записываю длину строки

fwrite(text[i], sizeof(char), len, file);//записываю строку

}

fclose(file);

}

else

{

printf("Ошибка при открытии бинарного файла для записи\n");

}

}

}

//функция считывания текста в двумерный массив из файла

char\*\* loadTextFromFile(long\* rows, char\* fileName)

{

int format; //переменная для определения типа файла

printf("Выберите тип файла (0 - текстовый или 1 - бинарный): ");

inputInt(&format, 0, 1);

FILE\* file;

if (!format)

{

file = fopen(fileName, "r"); //открываю текстовый файл для чтения

}

else

{

file = fopen(fileName, "rb"); //открываю бинарный файл для чтения

}

if (file != NULL)

{

if (!format)

{

fscanf(file, "%ld\n", rows); //считываю количество строк

}

else

{

fread(rows, sizeof(long), 1, file); //считываю количество строк

}

char\*\* text = (char\*\*)malloc(\*rows \* sizeof(char

if (text != NULL)

{

for (long i = 0; i < \*rows; i++)

{

char buffer[MAX\_LENGTH]; //буфер для считывания строки

if (!format)

{

fgets(buffer, MAX\_LENGTH, file); //считываю строку

int j = 0;

while (buffer[j] != '\n')

j++;

buffer[j] = '\0'; //удаляю \n заменяя на \0

text[i] = strdup(buffer); //выделяю память и копирую строку

}

else

{

long len;

fread(&len, sizeof(long), 1, file); //считываю длину

text[i] = (char\*)malloc((len + 1) \* sizeof(char));

fread(text[i], sizeof(char), len, file); //считываю строку

text[i][len] = '\0';

}

}

}

else

{

printf("Ошибка при выделении памяти\n");

}

fclose(file);

return text;

}

else

{

printf("\nФайл с таким именем не найден\n");

return NULL;

}

}

//функция перевода двумерного массива чисел в массив строк

long\*\* myAtoi2DText(char\*\* text, long rows)

{

long\*\* numbers = (long\*\*)malloc(rows \* sizeof(long\*)); //выделяю память

for (int i = 0; i < rows; i++) //цикл по строкам

{

int count = 0; //кол-во чисел в текущей строке

char\* copy = strdup(text[i]); //создаю копию строки

char\* token = strtok(copy," " ); //разбиваю строку на слова

while (token != NULL)

{

count++;

token = strtok(NULL, " ");

}

free(copy);

numbers[i] = (long\*)malloc((count + 1) \* sizeof(long

numbers[i][0] = count + 1; //записываю первым элементов кол-во чисел

char\* tokenPtr = strtok(text[i], " "); //разбиваю на слова

int j = 1;

while (tokenPtr != NULL)

{

numbers[i][j++] = atol(tokenPtr); //слово (число) преобразую в long

tokenPtr = strtok(NULL, " ");

}

}

return numbers;

}

////////////////Вспомогательные функции которые вызывают другие функции//////////

//функция для кодирования текста

void caseEncoding (struct dataCode \*\*keys, long \*countKeys, long\*\*\* codeText, char\*\* text, int n, char\*\*\* newText)

{

(\*countKeys)++;

(\*keys) = (struct dataCode\*)realloc((\*keys), (\*countKeys) \* sizeof(struct dataCode)); //расширяю массив структур

\*(\*keys + (\*countKeys) - 1) = encodingText(text, n, codeText, newText); //кодирую текст

if((\*keys + (\*countKeys) - 1)->flag && text != NULL) //вывод на экран результата шифрования

{

printf("\n\n\033[3;32m Закодированный текст \033[0m");

output2DNumbers(\*codeText, n);

}

if (((\*keys + (\*countKeys) - 1)->flag == 0) && text != NULL)

{

printf("\n\n\033[3;32m Закодированный текст \033[0m");

output2DString(\*newText, n);

}

}

//функция для декодирования текста

void caseDecoding (char \*\*\*text, long \*\*codeText, char \*\*newText, struct dataCode \*keys, long countKeys, long \*rows)

{

int format;

char box[SIZE][SIZE];

readInfo(box, SIZE, "box1.txt");

printf("\nЧтобы расшифровать текст, введите 1");

printf("\nЧтобы расшифровать файл, введите 0\n");

inputInt(&format, 0, 1);

if(format)

{

\*text = decodingInputText(box, rows);

}

else

{

\*text = decodingTextFromFile(codeText, newText, keys, countKeys, box, rows);

}

printf("\nРаскодированный текст:\n");

output2DString(\*text, \*rows);

}

//функция для сохранения информации в файл

void saveInformation (char\*\* text, char\*\* newText, long\*\* code, long rows, struct dataCode \*keys, long countKeys)

{

if(text == NULL && newText == NULL && code == NULL)

{

printf("\033[1;31m Вы не ввели текст: \033[0m");

return;

}

int format;

char\* fileName;

printf("\nЧто вы хотите сохранить в файл?");

printf("\n1 - исходный текст");

printf("\n2 - текст, закодированный алгоритмом RSA");

printf("\n3 - текст, закодированный алгоритмом Square\n");

inputInt(&format, 1, 3);

printf("\nВведите имя файла, в который хотите сохранить\n");

inputStr(&fileName);

strcpy((keys + countKeys - 1)->fileName, fileName); //сохраняю имя файла

if (format == 1)

{

saveTextToFile(text, rows, fileName); //сохраняю исходный текст

}

if (format == 2)

{

saveNumberToFile(code, rows, fileName);//сохраняю текст зашифрованный RSA

}

else

{

saveTextToFile(newText, rows, fileName);

}

printf("\033[3;32m Информация успешно сохранена \033[0m");

}

//функция чтения текста из файла

void caseReadText (char\*\*\* text, long\* rows)

{

char\* fileName;

printf("Введите имя файла, из которого хотите считать текст\n");

inputStr(&fileName);

\*text = loadTextFromFile(rows, fileName); //читаю текст из файла

if(\*text != NULL)

{

printf("\n\033[3;32m Текст успешно прочитан.\033[0mПрочитанный текст:");

output2DString(\*text, \*rows);

}

}

//функция очистки памяти

void freeMemory(char\*\* text, char\*\* newText, long\*\* codeText, struct dataCode\* keys, long countKeys, long n)

{

if (text != NULL)

{

for (long i = 0; i < n; i++)

{

free(text[i]);

}

free(text);

}

if (newText != NULL)

{

for (long i = 0; i < n; i++)

{

free(newText[i]);

}

free(newText);

}

if (codeText != NULL)

{

for (long i = 0; i < n; i++)

{

free(codeText[i]);

}

free(codeText);

}

if (keys != NULL)

{

free(keys);

}

}

//функция ввода строки произвольной длины

void inputStr (char\*\* mas)

{

int n = 1, i = 0; //n - длина строки, i - параметр цикла

char s;

\*mas = (char\*)calloc(n, sizeof(char)); //выделение памяти на один элемент

while ((s = getchar()) != '\n')

{

\*(\*mas + i) = s; //запись символа в строку

\*mas = (char\*)realloc(\*mas,(++n) \* sizeof(char));

i++;

}

\*(\*mas + i) = '\0'; //запись нуль-терминатора

}

//функция ввода числа типа int

void inputInt (int\* x, int a, int b)

{

while(!scanf("%d", x) || (((\*x) < a) || ((\*x) > b)))

{

rewind(stdin);

printf("\nОшибка ввода числа, попробуйте еще раз: ");

}

rewind(stdin);

}

//функция вода числа типа long

void inputLong (long\* x, int a, int b)

{

while(!scanf("%ld", x) || (((\*x) < a) || ((\*x) > b)))

{

rewind(stdin);

printf("\033[1;31m Ошибка ввода числа, попробуйте еще раз \033[0m\n");

}

rewind(stdin);

}

//функция ввода массива строк с клавиатуры

void input2DString (char\*\*\* text, long\* n)

{

printf("\nВведите текст, чтобы завершить ввод введите Enter\n");

int i = 0;

\*text = (char\*\*)calloc(1, sizeof(char\*));

\*(\*text + i) = (char\*)calloc(80, sizeof(char));

do

{

fgets(\*(\*text + i), 80, stdin);

if(\*(\*(\*text + i)) == '\n') break;

deleteSymbolN(&(\*(\*text + i)));

i++;

\*text = (char\*\*)realloc(\*text, (i+1) \* sizeof(char\*));

\*(\*text + i) = (char\*)calloc(80, sizeof(char));

}while(true);

\*n = i;

printf("\033[3;34m Введенный вами текст: \033[0m");

output2DString(\*text, \*n);

}

//функция вывода двумерного массива чисел long\*\* на экран

void output2DNumbers (long\*\* mas, long n)

{

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

{

printf("\n");

for(int j = 0; j < mas[i][0]; j++)

printf("%ld ", mas[i][j]);

}

}

//функция вывода массива строк на экран

void output2DString (char\*\* mas, int n)

{

if(mas == NULL)

{

return;

}

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

{

printf("\n");

for(int j = 0; mas[i][j] != '\0'; j++)

printf("%c", mas[i][j]);

}

}

//функция удаления символа '\n' из строки

void deleteSymbolN (char\*\* str)

{

int n = strlen(\*str);

\*(\*str + n - 1) = '\0';

\*str = (char\*)realloc(\*str, n \* sizeof(char));

}