**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Основы информатики»

Отчет по лабораторной работе №7

«Обработка символьных строк»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-11 |  | преподаватель каф. ИУ5 |
| Михалёв Ярослав |  | Козлов А.Д. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2021 г.

Постановка задачи

Провести кодирование и декодирование текста (массива символов) при помощи кода Цезаря с переменным сдвигом по таблице ASCII-кодов. Величина сдвига для каждой позиции в исходном тексте - сумма (по модулю 256) кодов символов слова кодового блокнота, стоящего в блокноте на той же позиции. Если кодовый блокнот имеет слов меньше, чем количество символов в исходном тексте, то по исчерпании слов в нём перейти к первому слову и продолжить. (На основе кодового блокнота целесообразно сначала сформировать по заданному правилу целочисленный массив ключей, который затем использовать при кодировании. Эти действия оформить в виде отдельной функции.)

Исследовать повторяемость символов в закодированном тексте (сколько каких кодов одного и того же исходного символа получено) в зависимости от кодового блокнота и длины исходного текста. Результаты исследования представить в виде таблицы (продумать формат таблицы). Исследование и вывод таблицы результатов следует выполнять в режиме диалога, последовательно вычисляя и выводя результаты для запрашиваемого символа. Статистические данные хранить в массиве int stat[256], Для большей достоверности статистических результатов в качестве исходного текста и кодового блокнота использовать текстовые файлы размером около 1 Кбайта.

Разработка алгоритма

**Входные данные**

* Input.txt – текстовый файл
* key.txt – кодовый блокнот

**Выходные данные**

* shifr.txt – зашифрованный текст
* output.txt – расшифрованный текст

**Этапы программы**

* По кодовому блокноту получить массив ключей
* Зашифровать текст
* Вывести статистику по символу
* Расшифровать текст

**Список функций**

Функции для работы с файлами

streampos FileSize(const char\* filename)

Возвращет размер файла (количество символов)

Входные данные

* const char\* filename – название файла

Выходные данные

* streampos fsize – размер файла (количество символов)

char\* ReadFile(const char\* filename)

Считывает текст из файла

Входные данные

* const char\* filename – название файла

Выходные данные

* char\* s - массив символов

void WriteFile(char text[], int size, const char\* filename)

Записывает текст в файл

Входные данные

* char text[] – массив символов
* int size – размер массива text
* const char\* filename – название выходного файла

Выходные данные

* Текстовый файл

Функции для работы с текстом

void PrintText(char s[], int len)

Выводит массив символов на экран

Входные данные

* char text[] – массив символов
* int len - размер массива text

void PrintMatrix(double\*\* A, int N)

Выводит матрицу на экран

Входные данные:

* double\*\* A – матрица
* int N – размер N

int\* CreateArray(int size)

Возвращает динамический массив размером size и заполняет его нулями

Входные данные:

* int N – размер массива

Выходные данные:

* int\* arr - динамический массив

double\*\* CreateMatrix(int N)

Возвращает нулевую матрицу размером N

Входные данные:

* int N – размер матрицы

Выходные данные:

* double\*\* A – единичная матрица

Главные функции

int\* GetKeyArray(const char\* filename)

Возвращает массив ключей

Входные данные

* const char\* filename – название текстового файла с кодовым блокнотом

Выходные данные

* int\* scod

void GetStatistic(int\*\* statistic, char symbol)

Выводит статистику по переданному символу

Входные данные

* int\*\* statistic – двумерный массив со статистикой по символам
* char symbol – символ, по которому нужно получить статистику

char\* Encode(const char\* input\_file, const char\* keys\_file, int\*\* statistic)

Шифрует текст

Входные данные

* const char\* input\_file – название файла с исходным текстом
* const char\* keys\_file – название файла с кодовым блокнотом
* int\*\* statistic - двумерный массив со статистикой по символам

Выходные данные

* char\* codtext – зашифрованный текст

char\* Decode(const char\* cipher\_file, const char\* keys\_file)

Расшифровывает текст

Входные данные

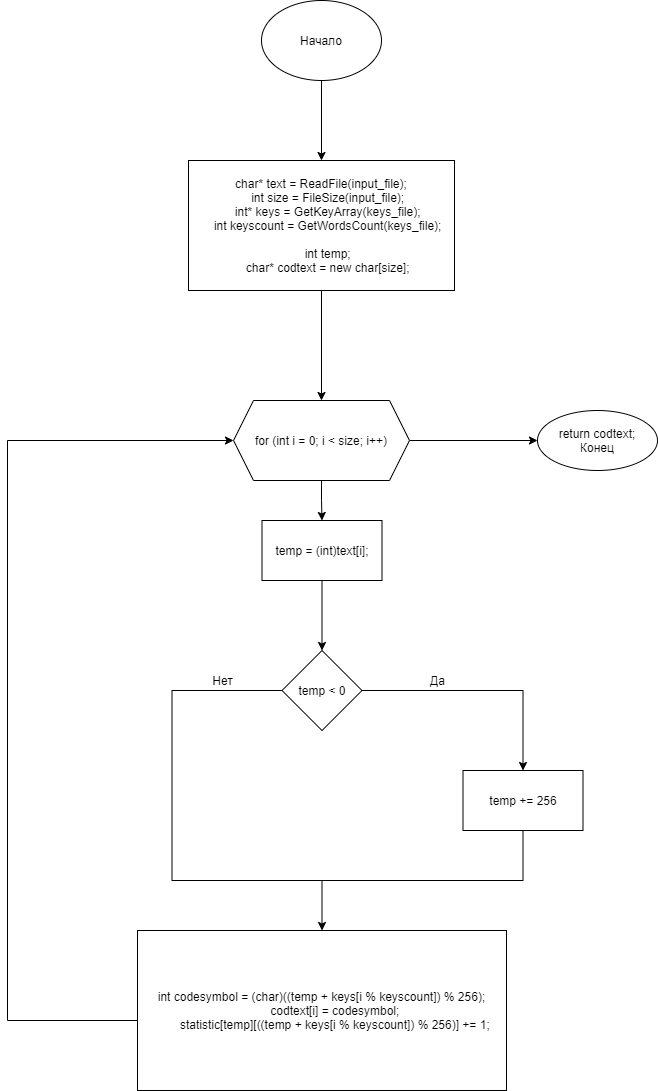
* const char\* cipher\_file – название файла с зашифрованным текстом
* const char\* keys\_file – название файла с кодовым блокнотом

Выходные данные

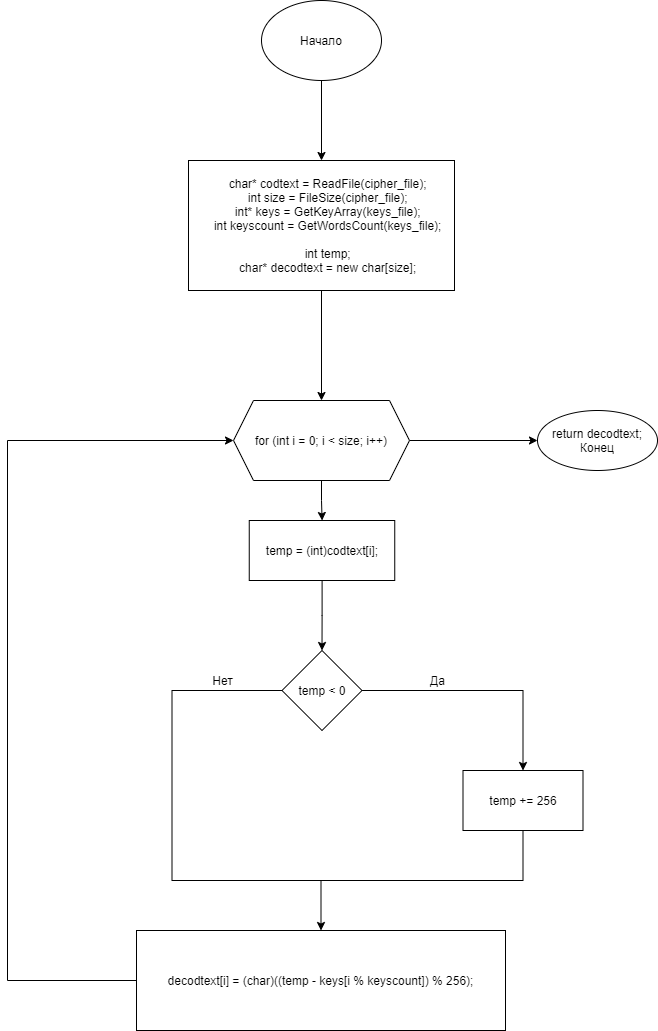
* char\* decodtext – расшифрованный текст

**Схема алгоритма**

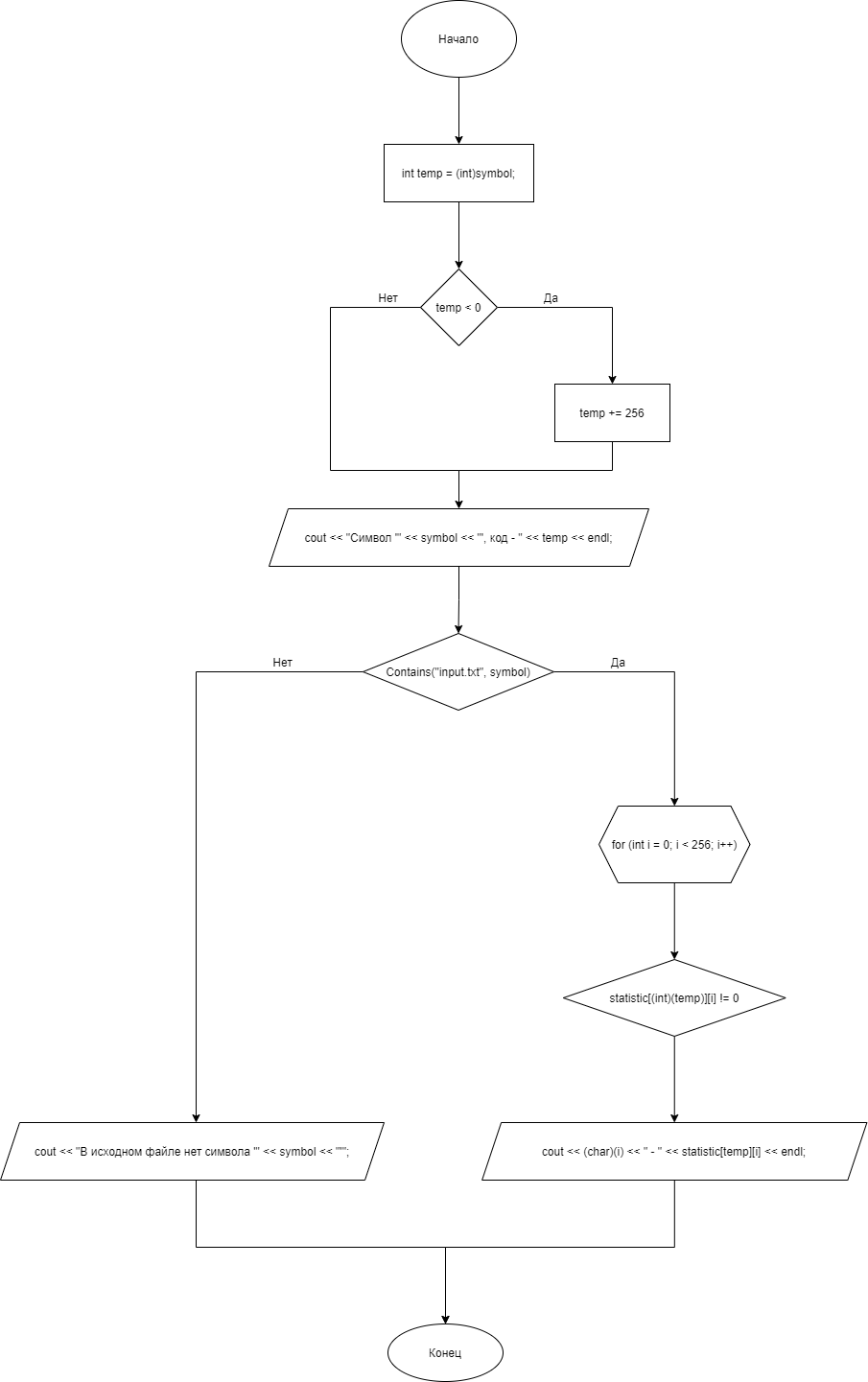
Encode



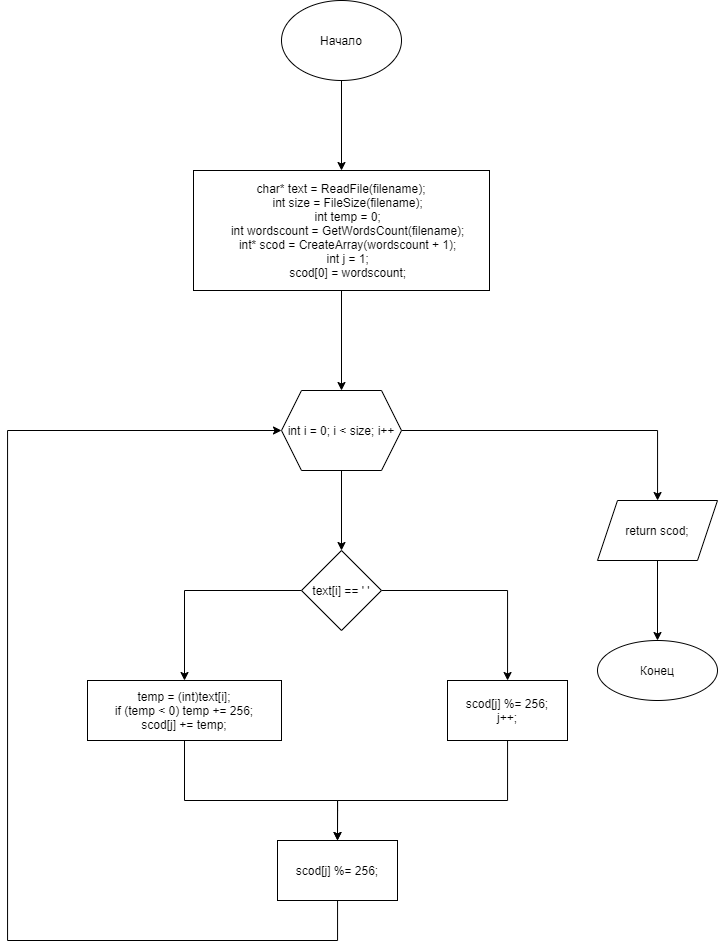
Decode



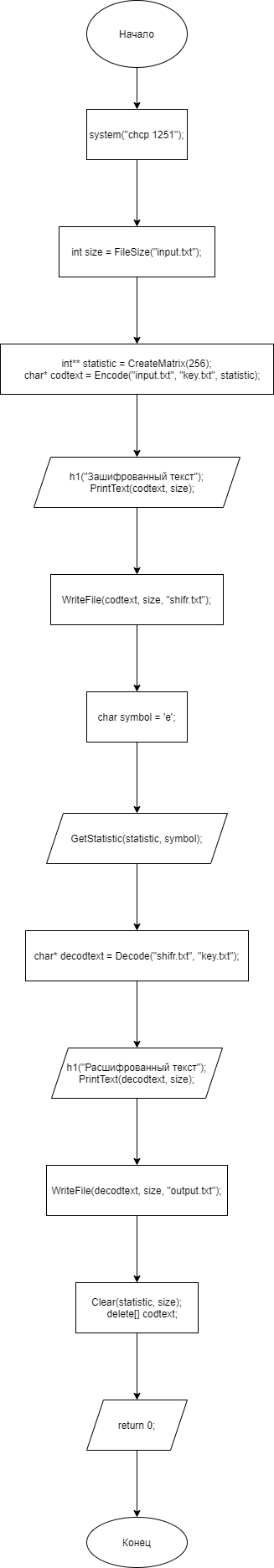
GetStatistic



GetKeyArray



Main



Текст программы

Main.cpp

#include <iostream>

#include <fstream>

**using** **namespace** std**;**

// Выводит текст

void PrintText**(**char s**[],** int len**)**

**{**

**for** **(**int j **=** 0**;** j **<** len**;** j**++)**

cout **<<** s**[**j**];**

**}**

// Выводит массив

void PrintArray**(**int arr**[],** int size**)**

**{**

**for** **(**int j **=** 0**;** j **<** size**;** j**++)**

cout **<<** arr**[**j**]** **<<** " "**;**

**}**

// Выводит квадратную матрицу A

void PrintMatrix**(**int**\*\*** A**,** int size**)**

**{**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

**{**

**for** **(**int j **=** 0**;** j **<** size**;** j**++)**

cout **<<** A**[**i**][**j**]** **<<** " "**;**

cout **<<** endl**;**

**}**

**}**

// Выводит заголовок

void h1**(**const char text**[])**

**{**

cout **<<** "\n\n\n" **<<** text **<<** "\n\n\n"**;**

**}**

// Возвращет размер файла (количество символов)

streampos FileSize**(**const char**\*** filename**)** **{**

streampos fsize **=** 0**;**

ifstream file**(**filename**,** ios**::**binary**);**

fsize **=** file**.**tellg**();**

file**.**seekg**(**0**,** ios**::**end**);**

fsize **=** file**.**tellg**()** **-** fsize**;**

file**.**close**();**

**return** fsize**;**

**}**

// Читает текст

char**\*** ReadFile**(**const char**\*** filename**)**

**{**

int size **=** FileSize**(**filename**);** // Количество символов в файле

char**\*** s **=** **new** char**[**size **+** 1**];**

char ch**;**

ifstream file**;**

file**.**open**(**filename**,** ios**::**binary**);**

**if** **(!**file**.**is\_open**())**

**{**

cout **<<** "Файла " **<<** filename **<<** " не существует" **<<** endl**;**

exit**(**0**);**

**}**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

**{**

file**.**get**(**ch**);**

s**[**i**]** **=** ch**;**

**}**

file**.**close**();**

**return** s**;**

**}**

// Записывает текст в файл

void WriteFile**(**char text**[],** int size**,** const char**\*** filename**)**

**{**

ofstream f**;**

f**.**open**(**filename**,** ios**::**binary**);**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

f **<<** text**[**i**];**

f**.**close**();**

**}**

// Проверяет, содержит ли текст данный символ

bool Сontains**(**const char**\*** filename**,** char symbol**)**

**{**

char**\*** text **=** ReadFile**(**filename**);**

int size **=** FileSize**(**filename**);**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

**if** **(**text**[**i**]** **==** symbol**)**

**return** **true;**

**return** **false;**

**}**

// Создаёт массив размера size и заполняет его нулями

int**\*** CreateArray**(**int size**)**

**{**

int**\*** arr **=** **new** int**[**size **+** 1**];**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

arr**[**i**]** **=** 0**;**

**return** arr**;**

**}**

// Создаёт квадратную матрицу размером size

int**\*\*** CreateMatrix**(**int size**)**

**{**

int**\*\*** A **=** **new** int**\*** **[**size**];**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

**{**

A**[**i**]** **=** **new** int**[**size**];**

**for** **(**int j **=** 0**;** j **<** size**;** j**++)**

A**[**i**][**j**]** **=** 0**;**

**}**

**return** A**;**

**}**

// Возвращает количество слов в тексте

int GetWordsCount**(**const char**\*** filename**)**

**{**

char**\*** text **=** ReadFile**(**filename**);**

int size **=** FileSize**(**filename**);**

int count **=** 0**;**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

**if** **(**text**[**i**]** **==** ' '**)**

count**++;**

count**++;**

**return** count**;**

**}**

// Возвращает массив ключей

int**\*** GetKeyArray**(**const char**\*** filename**)**

**{**

char**\*** text **=** ReadFile**(**filename**);**

int size **=** FileSize**(**filename**);**

int temp **=** 0**;**

// Количество слов в тексте (размер массива ключей)

int wordscount **=** GetWordsCount**(**filename**);**

// Массив ключей(ключ - сумма кодов одного слова)

int**\*** scod **=** CreateArray**(**wordscount **+** 1**);**

int j **=** 1**;**

scod**[**0**]** **=** wordscount**;**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

**{**

**if** **(**text**[**i**]** **==** ' '**)**

**{**

scod**[**j**]** **%=** 256**;**

j**++;**

**}**

**else**

**{**

temp **=** **(**int**)**text**[**i**];**

**if** **(**temp **<** 0**)** temp **+=** 256**;**

scod**[**j**]** **+=** temp**;**

**}**

scod**[**j**]** **%=** 256**;**

**}**

**return** scod**;**

**}**

// Выводит статистику по переданному символу

void GetStatistic**(**int**\*\*** statistic**,** char symbol**)**

**{**

int temp **=** **(**int**)**symbol**;**

**if** **(**temp **<** 0**)** temp **+=** 256**;**

cout **<<** "Символ '" **<<** symbol **<<** "', код - " **<<** temp **<<** endl**;**

**if** **(**Сontains**(**"input.txt"**,** symbol**))**

**{**

**for** **(**int i **=** 0**;** i **<** 256**;** i**++)**

**if** **(**statistic**[(**int**)(**temp**)][**i**])**

cout **<<** **(**char**)(**i**)** **<<** " - " **<<** statistic**[(**int**)(**temp**)][**i**]** **<<** endl**;**

**}**

**else**

**{**

cout **<<** "В исходном файле нет символа '" **<<** symbol **<<** "'"**;**

**}**

**}**

// Шифрует текст

char**\*** Encode**(**const char**\*** input\_file**,** const char**\*** keys\_file**,** int**\*\*** statistic**)**

**{**

char**\*** text **=** ReadFile**(**input\_file**);**

int size **=** FileSize**(**input\_file**);**

int**\*** keys **=** GetKeyArray**(**keys\_file**);**

int keyscount **=** GetWordsCount**(**keys\_file**);**

int temp**;**

char**\*** codtext **=** **new** char**[**size**];**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

**{**

temp **=** **(**int**)**text**[**i**];**

**if** **(**temp **<** 0**)** temp **+=** 256**;**

int codesymbol **=** **(**char**)((**temp **+** keys**[**i **%** keyscount**])** **%** 256**);**

codtext**[**i**]** **=** codesymbol**;**

statistic**[**temp**][((**temp **+** keys**[**i **%** keyscount**])** **%** 256**)]** **+=** 1**;**

**}**

**return** codtext**;**

**}**

// Расшифровывает текст

char**\*** Decode**(**const char**\*** cipher\_file**,** const char**\*** keys\_file**)**

**{**

char**\*** codtext **=** ReadFile**(**cipher\_file**);**

int size **=** FileSize**(**cipher\_file**);**

int**\*** keys **=** GetKeyArray**(**keys\_file**);**

int keyscount **=** GetWordsCount**(**keys\_file**);**

int temp**;**

char**\*** decodtext **=** **new** char**[**size**];**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

**{**

temp **=** **(**int**)**codtext**[**i**];**

**if** **(**temp **<** 0**)** temp **+=** 256**;**

decodtext**[**i**]** **=** **(**char**)((**temp **-** keys**[**i **%** keyscount**])** **%** 256**);**

**}**

**return** decodtext**;**

**}**

int main**()**

**{**

system**(**"chcp 1251"**);**

int size **=** FileSize**(**"input.txt"**);** // Количество символов в тексте

int**\*\*** statistic **=** CreateMatrix**(**256**);**

char**\*** codtext **=** Encode**(**"input.txt"**,** "key.txt"**,** statistic**);** // Кодирование текста

WriteFile**(**codtext**,** size**,** "shifr.txt"**);** // Запись в файл

char symbol **=** 'e'**;**

GetStatistic**(**statistic**,** symbol**);**

char**\*** decodtext **=** Decode**(**"shifr.txt"**,** "key.txt"**);** // Кодирование текста

WriteFile**(**decodtext**,** size**,** "output.txt"**);** // Запись в файл

// Освобождение памяти

**for** **(**int i **=** 1**;** i **<** size**;** i**++)**

**delete[]** statistic**[**i**];**

**delete[]** codtext**;**

**return** 0**;**

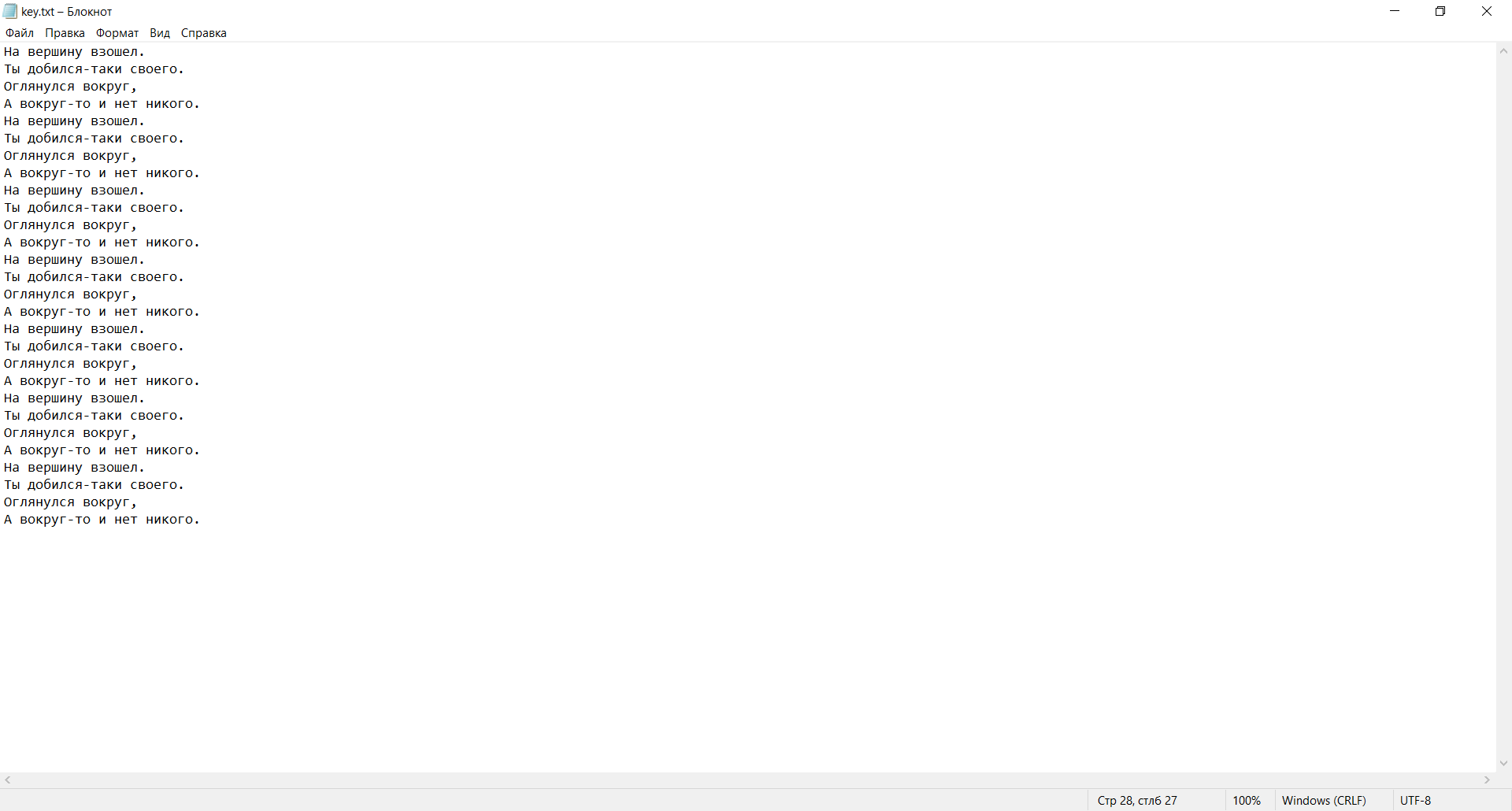
**}**

Анализ результатов

Имеем файл input.txt



Файл key.txt

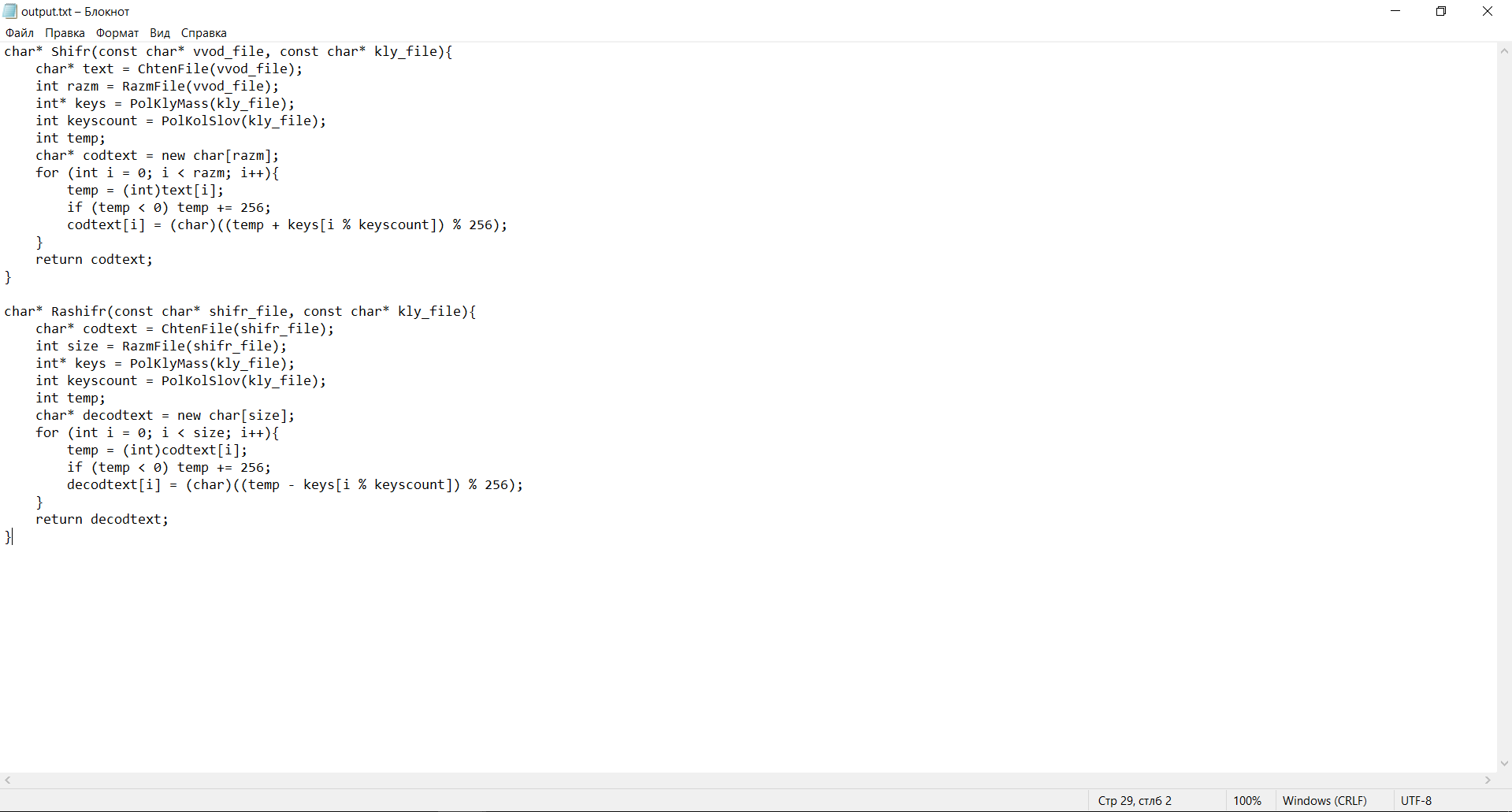


После выполнения программы мы получаем два файла:

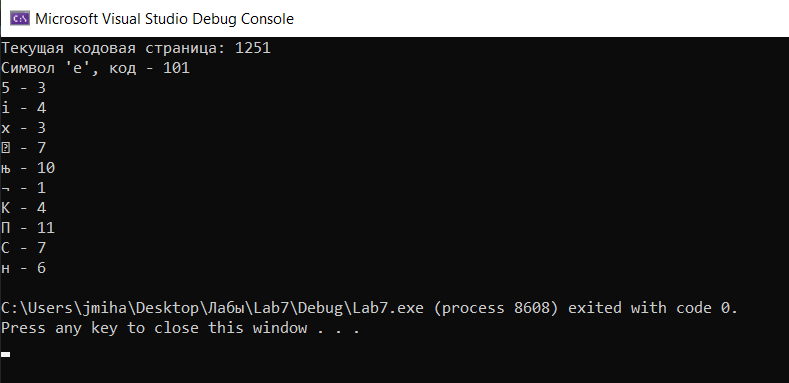
shifr.txt – зашифрованный текст



output.txt – расшифрованный текст



При запросе символа ‘e’ программа выводит статистику по этому символу



Вывод

Я научился

* Работать с файлами и динамическими массивами
* Кодировать и декодировать текст
* Собирать статистические данные
* Работать с кодировкой ANCI