**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 ИУ5-13Б |  | преподаватель каф. ИУ5 |
| Поляков Данила |  | Аксёнова М.В. |
| Подпись и дата |  | Подпись и дата: |

Москва, 2021 г.

**Постановка задачи**

Задание.

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**

**Описание алгоритма**

Алгоритм кодирует слова при помощи шифра Цезаря и кодового блокнота по принципу того, что каждое кодовое слово – сумма кодов всех букв одного слова в кодовом блокноте. После каждая буква кодируется n-m кодовым словом непосредственно шифром Цезаря. Для каждой буквы определяется сдвиг посредством кодового слова. Так же записывается статистика сколько раз какая буква была закодирована определённой.

**Описание используемых переменных**

void encription(char \*file\_name, char \*key\_file\_name, bool flag) – универсальная функция кодирования/декодирования.

char \*file\_name – имя файла для шифрования/дешифрования в директории программы.

char \*key\_file\_name – имя файла с ключом

bool flag – флаг выбора задачи – 1(true) – шифровка, 0(false) - дешифровка

**Текст программы**

**Листинг кода программы:**

**main.cpp**

#include "functions.h"

int main() {

encription("example.txt", "key.txt", true); // зашифровка

encription("encrypted.txt", "key.txt", false); // расшифровка

return 0;

}

**functions.h**

#ifndef LAB7\_HEADER\_H

#define LAB7\_HEADER\_H

#include <fstream>

#include <iostream>

void encription(char \*file\_name,char \*key\_file\_name, bool);

#endif //LAB7\_HEADER\_H

functions.cpp

#include "functions.h"

void encription(char \*file\_name, char \*key\_file\_name, bool flag) {

std::fstream original\_file;

original\_file.open(file\_name);

original\_file.seekg(std::ios\_base::beg, std::ios\_base::end);

long int file\_len = original\_file.tellg();

original\_file.seekg(0);

char \*text = new char[file\_len];

original\_file.read(text, file\_len + 1);

original\_file.close();

std::fstream key\_file;

key\_file.open(key\_file\_name);

key\_file.seekg(std::ios\_base::beg, std::ios\_base::end);

long int key\_len = key\_file.tellg();

key\_file.seekg(0);

char \*key = new char[key\_len];

int count = 0;

int space = 0;

for (int i = 0; i < key\_len; ++i) {

char tmp = key\_file.peek();

if (!std::ispunct(tmp) && !std::isspace(tmp)) {

key[count] += key\_file.get();

space = 0;

} else {

if (space != 0) {

--count;

} else {

++space;

}

++count;

key\_file.get();

}

}

std::fstream encryption;

encryption.open("encrypted.txt", std::ios::out);

long int i = 0;

int stat[256][256];

int k = flag ? 1 : -1;

for (; i < file\_len; ++i) {

stat[text[i]][k \* key[i % count]]++;

text[i] += k \* key[i % count];

}

std::cout << "По какому символу хотите получить статистику: " << std::endl;

char ch;

std::cin >> ch;

int sim = (int) ch;

std::cout << "Символ " << ch << " был закодирован: " << std::endl;

for (int i = 33; i < 127; i++) { //только читаемые

if (stat[sim][i] != 0) {

std::cout << "Символом " << " " << char(i) << " " << stat[sim][i] << std::endl;

}

}

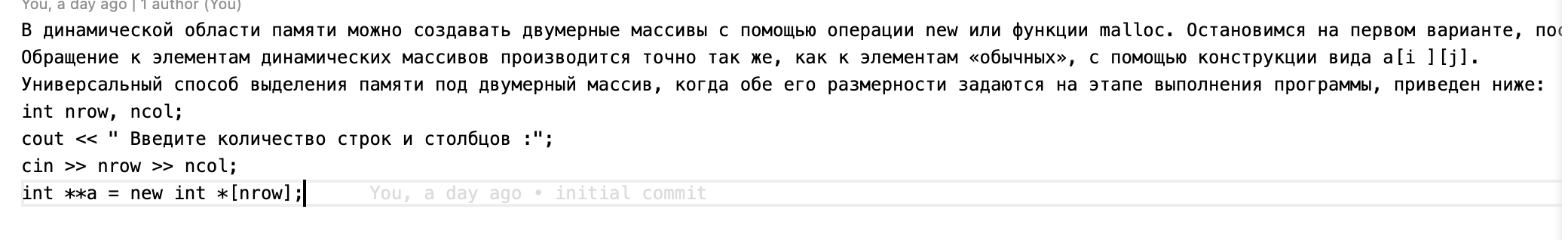
encryption.write(text, file\_len);

encryption.close();

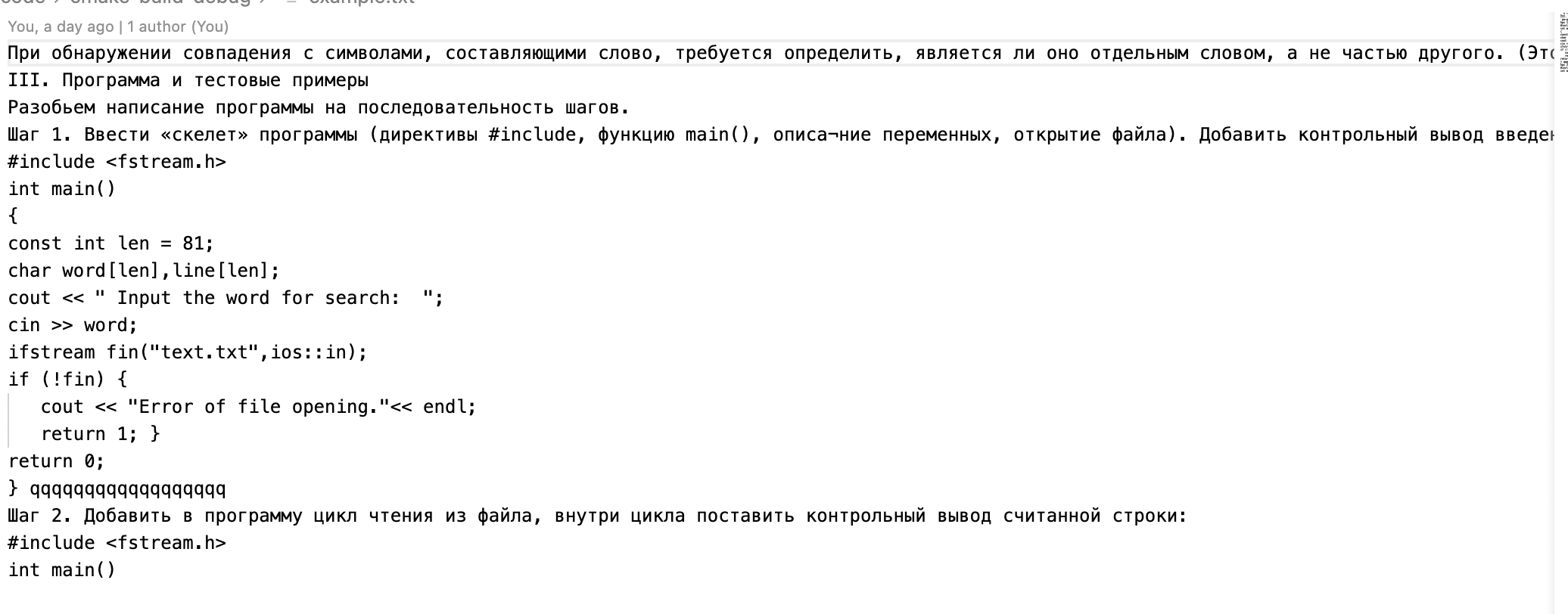
}

**Анализ результатов**

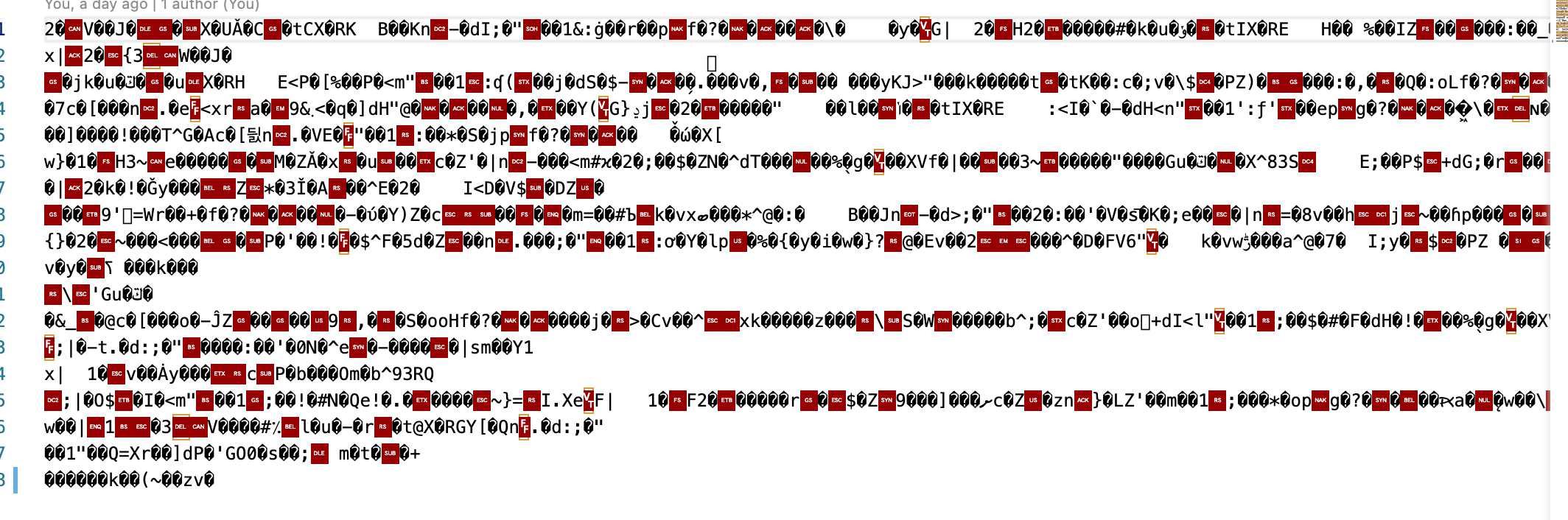
Кодовый блокнот

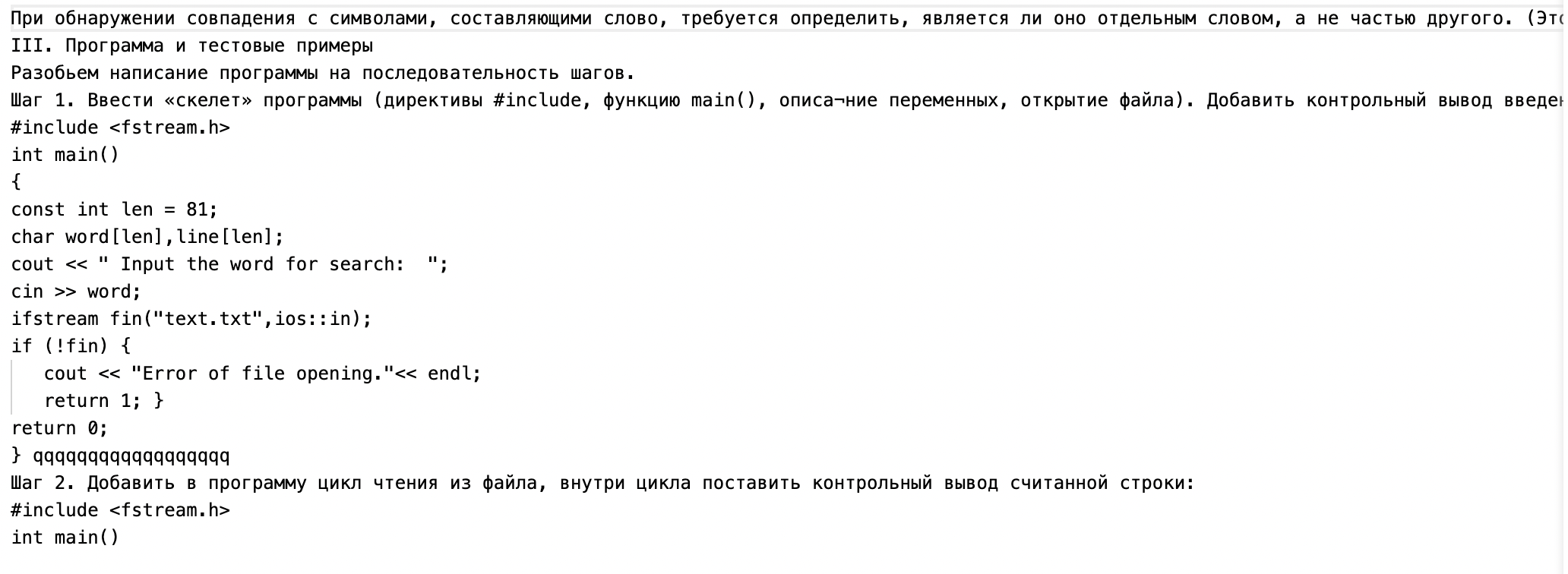


Файл для шифрования



Файл после зашифровки

Расшифрованный текст



Статистика

