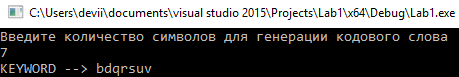
**Садов Д.В. ИВБО 7-14.**

**Лабораторная работа №1 (листинг по каждой лабораторной будет в конце каждой лабы)**

В данной лабораторной работе нужно было заменить исходный алфавит (я взял английский),

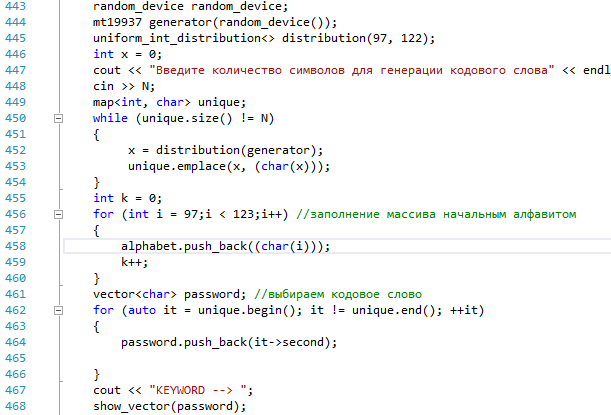
выбрав кодовое слово и записав его в начало алфавита. Затем нужно было зашифровать исходный текст новым алфавитом. Наконец, расшифровать этот текст используя частотный анализ.

1. При запуске программы пользователю предлагается ввести количество символов в кодовом слове.

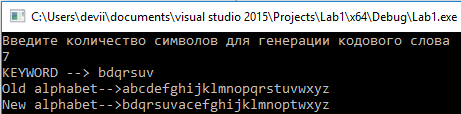


Ключевое слово формируется рандомно, исключая повторение символов

В программе это реализовано таким образом:



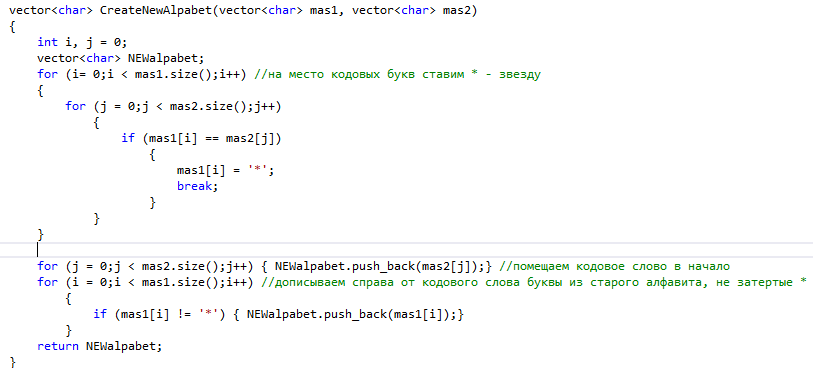
1. После генерации ключевого слова, оно помещается в начало и формируется новый алфавит. Пользователю выводится на экран для наглядности старый и новый.



В программе формирование нового алфавита реализуется в методе

CreateNewAlpabet

В программе это выглядит так:

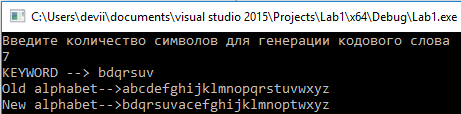


Я передаю в метод CreateNewAlpabet() исходный алфавит и ключевое слово.

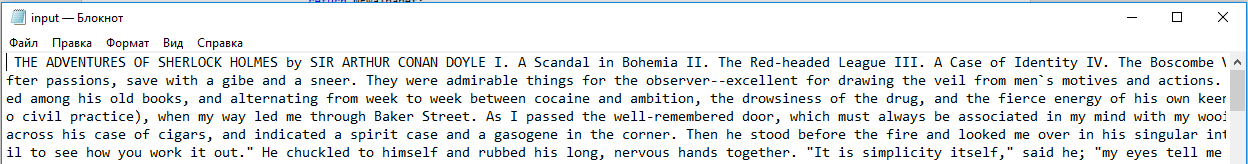
На место букв совпадающих в исходном алфавите с ключевым словом я ставлю \* (звездочки). В созданный новый алфавит (NEWalpabet) вначало помещается ключевое слово, а затем записываем по порядку символы из старого алфавита. Новый алфавит сформирован.

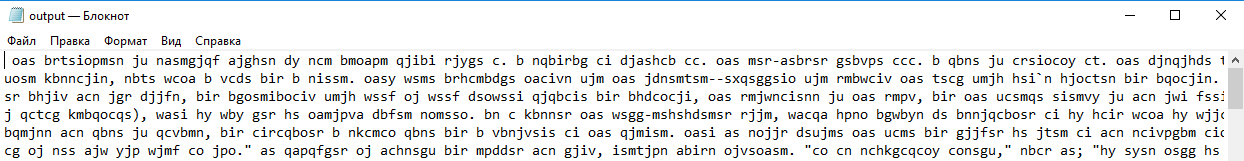
1. Далее методом FileEncoding шифруется считываемый текст из файла input.txt.

Каждый считанный символ исходного текста ищется в массиве начального алфавита, а затем по его порядковому номеру заменяется на символ из массива нового алфавита. Спец знаки и цифры остаются без шифрации. Исходный текст приводится к нижнему регистру.

Исходя из 

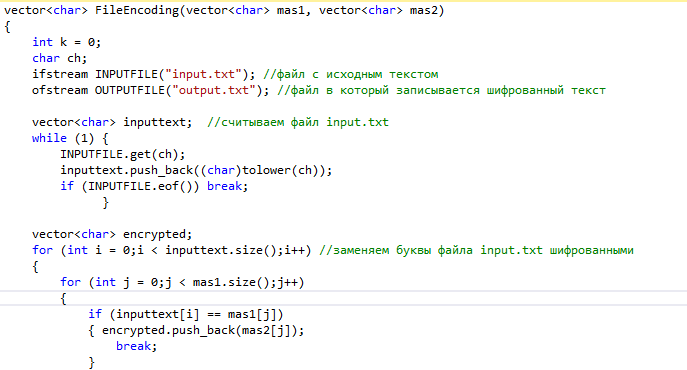
можно увидеть, что буквы из исходного текста (файл input.txt) меняются другими (a на b, b на d и так далее):



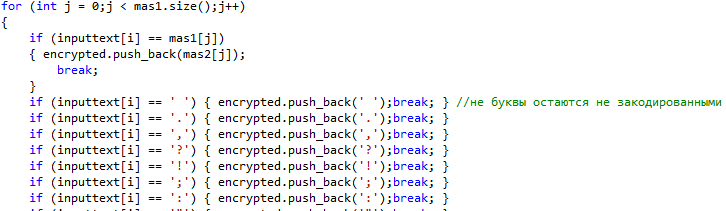


В программе это реализовано в методе FileEncoding. В него передается исходный и новый

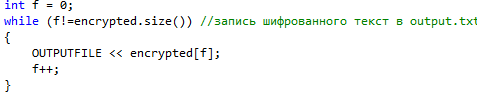
алфавиты. Затем происходит сопоставление считанного символа и в двойном цикле for происходит поиск его номера в новом алфавите.



Спец знаки и цифры не кодируются . Проверка на спец знаки осуществляется во втором цикле for



Далее записанный массив символов в векторе encrypted выводится в файл output.txt



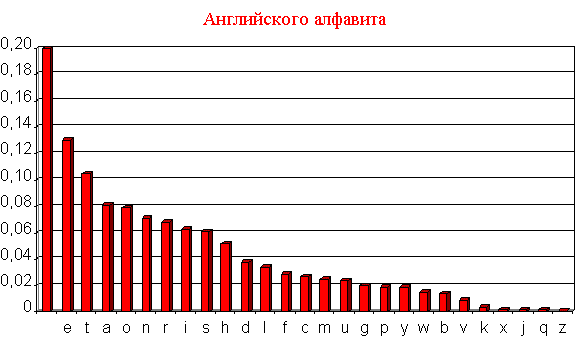
1. Итак, исходный текст зашифрован. Нужно его расшифровать методом частного анализа.

Расшифровка реализована в методе Chastotniy\_analiz.

Он принимает на вход зашифрованный текст. Далее производится подсчет числа вхождения каждого символа в нашем тексте (спец знаки и цифры не считаются) и записывается в матрицу которая имеет 26 строк (26 символов в англ алфавите) и в каждой строке есть по три столбца. 1-ый столбец – количество вхождений символа в нашем тексте, 2-ой столбец символ который был считан и посчитано его вхождение, а 3ий столбец пока остается пустым. Далее происходит сортировка данной матрицы по первому столбцу в порядке убывания.

Далее руководствуясь имеющимися данными по частоте встречаемости английских букв

(я руководствовался этой диаграммой:)

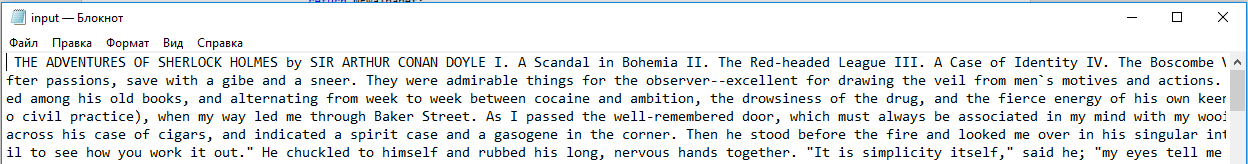


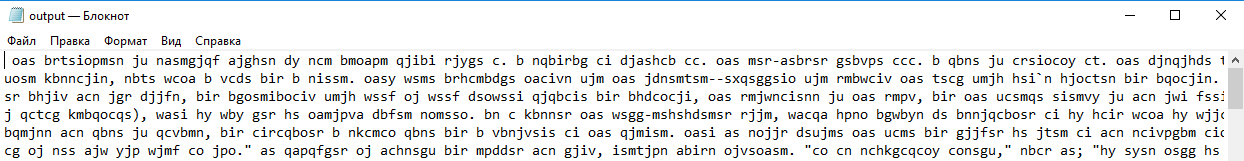
Записывается в третий столбец отсортированной матрицы в порядке убывания символы из диаграммы (т.е. в первой строке в третьем столбце будет записан символ e и т.д.)

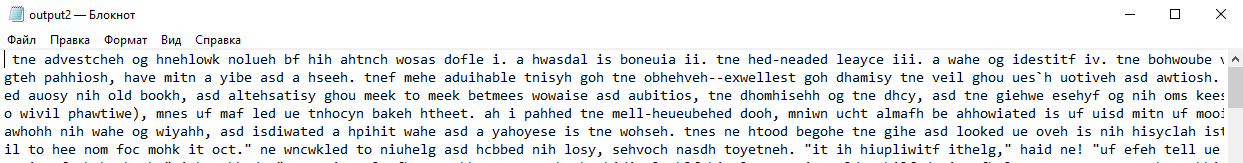
Затем используя такой же подход какой применялся в методе FileEncoding

производится считанный символ находится в матрице во втором столбце и заменяется символом записанным в третьем.

Вот так выглядят отрывки исходного текста, шифрованного текста и после дешифрации:







Как видно не все символы дешифрируются верно, но при большем количестве слов в тексте будет расти число расшифрованных верно букв.

**Листинг программы** (файлы input.txt,output.txt, output2.txt есть на гитхабе):

#include <iostream>

#include <fstream>

#include <time.h>

#include <stdlib.h>

#include <string>

#include <vector>

#include <ctime>

#include <map>

#include <math.h>

#include <random>

#include <algorithm>

using namespace std;

void show\_vector(vector<char>&a)

{

for (vector<char>::iterator it = a.begin(); it != a.end(); ++it)

cout << \*it;

}

class HelloWorld

{

public:

vector<char> CreateNewAlpabet(vector<char> mas1, vector<char> mas2)

{

int i, j = 0;

vector<char> NEWalpabet;

for (i= 0;i < mas1.size();i++) //на место кодовых букв ставим \* - звезду

{

for (j = 0;j < mas2.size();j++)

{

if (mas1[i] == mas2[j])

{

mas1[i] = '\*';

break;

}

}

}

for (j = 0;j < mas2.size();j++) { NEWalpabet.push\_back(mas2[j]);} //помещаем кодовое слово в начало

for (i = 0;i < mas1.size();i++) //дописываем справа от кодового слова буквы из старого алфавита, не затертые \*

{

if (mas1[i] != '\*') { NEWalpabet.push\_back(mas1[i]);}

}

return NEWalpabet;

}

vector<char> FileEncoding(vector<char> mas1, vector<char> mas2)

{

int k = 0;

char ch;

ifstream INPUTFILE("input.txt"); //файл с исходным текстом

ofstream OUTPUTFILE("output.txt"); //файл в который записывается шифрованный текст

vector<char> inputtext; //считываем файл input.txt

while (1) {

INPUTFILE.get(ch);

inputtext.push\_back((char)tolower(ch));

if (INPUTFILE.eof()) break;

}

vector<char> encrypted;

for (int i = 0;i < inputtext.size();i++) //заменяем буквы файла input.txt шифрованными

{

for (int j = 0;j < mas1.size();j++)

{

if (inputtext[i] == mas1[j])

{ encrypted.push\_back(mas2[j]);

break;

}

if (inputtext[i] == ' ') { encrypted.push\_back(' ');break; } //не буквы остаются не закодированными

if (inputtext[i] == '.') { encrypted.push\_back('.');break; }

if (inputtext[i] == ',') { encrypted.push\_back(',');break; }

if (inputtext[i] == '?') { encrypted.push\_back('?');break; }

if (inputtext[i] == '!') { encrypted.push\_back('!');break; }

if (inputtext[i] == ';') { encrypted.push\_back(';');break; }

if (inputtext[i] == ':') { encrypted.push\_back(':');break; }

if (inputtext[i] == '"') { encrypted.push\_back('"');break; }

if (inputtext[i] == '(') { encrypted.push\_back('(');break; }

if (inputtext[i] == ')') { encrypted.push\_back(')');break; }

if (inputtext[i] == '`') { encrypted.push\_back('`');break; }

if (inputtext[i] == '-') { encrypted.push\_back('-');break; }

if (inputtext[i] == '0') { encrypted.push\_back('0');break; }

if (inputtext[i] == '1') { encrypted.push\_back('1');break; }

if (inputtext[i] == '2') { encrypted.push\_back('2');break; }

if (inputtext[i] == '3') { encrypted.push\_back('3');break; }

if (inputtext[i] == '4') { encrypted.push\_back('4');break; }

if (inputtext[i] == '5') { encrypted.push\_back('5');break; }

if (inputtext[i] == '6') { encrypted.push\_back('6');break; }

if (inputtext[i] == '7') { encrypted.push\_back('7');break; }

if (inputtext[i] == '8') { encrypted.push\_back('8');break; }

if (inputtext[i] == '9') { encrypted.push\_back('9');break; }

}

}

int f = 0;

while (f!=encrypted.size()) //запись шифрованного текст в output.txt

{

OUTPUTFILE << encrypted[f];

f++;

}

INPUTFILE.close();

OUTPUTFILE.close();

return encrypted;

}

void Chastotniy\_analiz(vector<char> text)

{

int k = 0;

int tempmas[26][3] = { 0 };

for (int i = 0; i < text.size(); i++) //находим количество каждой буквы алфавита в тексте

{

if (text[i] == 'a') {

++tempmas[0][0];

tempmas[0][1] = 97;

}

if (text[i] == 'b') {

++tempmas[1][0];

tempmas[1][1] = 98;

}

if (text[i] == 'c') {

++tempmas[2][0];

tempmas[2][1] = 99;

}

if (text[i] == 'd') {

++tempmas[3][0];

tempmas[3][1] = 100;

}

if (text[i] == 'e') {

++tempmas[4][0];

tempmas[4][1] = 101;

}

if (text[i] == 'f') {

++tempmas[5][0];

tempmas[5][1] = 102;

}

if (text[i] == 'g') {

++tempmas[6][0];

tempmas[6][1] = 103;

}

if (text[i] == 'h') {

++tempmas[7][0];

tempmas[7][1] = 104;

}

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

++tempmas[8][0];

tempmas[8][1] = 105;

}

if (text[i] == 'j') {

++tempmas[9][0];

tempmas[9][1] = 106;

}

if (text[i] == 'k') {

++tempmas[10][0];

tempmas[10][1] = 107;

}

if (text[i] == 'l') {

++tempmas[11][0];

tempmas[11][1] = 108;

}

if (text[i] == 'm') {

++tempmas[12][0];

tempmas[12][1] = 109;

}

if (text[i] == 'n') {

++tempmas[13][0];

tempmas[13][1] = 110;

}

if (text[i] == 'o') {

++tempmas[14][0];

tempmas[14][1] = 111;

}

if (text[i] == 'p') {

++tempmas[15][0];

tempmas[15][1] = 112;

}

if (text[i] == 'q') {

++tempmas[16][0];

tempmas[16][1] = 113;

}

if (text[i] == 'r') {

++tempmas[17][0];

tempmas[17][1] = 114;

}

if (text[i] == 's') {

++tempmas[18][0];

tempmas[18][1] = 115;

}

if (text[i] == 't') {

++tempmas[19][0];

tempmas[19][1] = 116;

}

if (text[i] == 'u') {

++tempmas[20][0];

tempmas[20][1] = 117;

}

if (text[i] == 'v') {

++tempmas[21][0];

tempmas[21][1] = 118;

}

if (text[i] == 'w') {

++tempmas[22][0];

tempmas[22][1] = 119;

}

if (text[i] == 'x') {

++tempmas[23][0];

tempmas[23][1] = 120;

}

if (text[i] == 'y') {

++tempmas[24][0];

tempmas[24][1] = 121;

}

if (text[i] == 'z') {

++tempmas[25][0];

tempmas[25][1] = 122;

}

}

//сортировка по убыванию

size\_t imax;

for (size\_t i = 0; i < 26; ++i)

{

imax = i;

for (size\_t j = i + 1; j < 26; ++j)

{

if (tempmas[j][0] > tempmas[imax][0])

imax = j;

}

if (imax != i) {

for (size\_t c = 0; c < 3; ++c)

swap(tempmas[imax][c], tempmas[i][c]);

}

}

//шаблонная таблица встречаемости символов, заполняем ей третий стоблбец, чтобы сопоставить с отсортированным на предыдущем шаге массивом

tempmas[0][2] = 101;

tempmas[1][2] = 116;

tempmas[2][2] = 97;

tempmas[3][2] = 111;

tempmas[4][2] = 105;

tempmas[5][2] = 110;

tempmas[6][2] = 115;

tempmas[7][2] = 104;

tempmas[8][2] = 104;

tempmas[9][2] = 100;

tempmas[10][2] = 108;

tempmas[11][2] = 99;

tempmas[12][2] = 117;

tempmas[13][2] = 109;

tempmas[14][2] = 119;

tempmas[15][2] = 102;

tempmas[16][2] = 103;

tempmas[17][2] = 121;

tempmas[18][2] = 112;

tempmas[19][2] = 98;

tempmas[20][2] = 118;

tempmas[21][2] = 107;

tempmas[22][2] = 120;

tempmas[23][2] = 106;

tempmas[24][2] = 113;

tempmas[25][2] = 122;

vector<char> decrypted;

ofstream OUTPUTFILE2("output2.txt"); //файл в который будет дешифрироваться текст

int i;

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

{

for (int j = 0; j < 26; j++)

{

if (text[i] == (char)tempmas[j][1]) {

decrypted.push\_back((char)tempmas[j][2]);

break;

}

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

decrypted.push\_back(' ');

break;

}

if (text[i] == '.') {

decrypted.push\_back('.');

break;

}

if (text[i] == ',') {

decrypted.push\_back(',');

break;

}

if (text[i] == '?') {

decrypted.push\_back('?');

break;

}

if (text[i] == '!') {

decrypted.push\_back('!');

break;

}

if (text[i] == ';') {

decrypted.push\_back('!');

break;

}

if (text[i] == ':') {

decrypted.push\_back(':');

break;

}

if (text[i] == '"') {

decrypted.push\_back('"');

break;

}

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

decrypted.push\_back('(');

break;

}

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

decrypted.push\_back(')');

break;

}

if (text[i] == '`') {

decrypted.push\_back('`');

break;

}

if (text[i] == '-') {

decrypted.push\_back('-');

break;

}

if (text[i] == '0') {

decrypted.push\_back('0');

break;

}

if (text[i] == '1') {

decrypted.push\_back('1');

break;

}

if (text[i] == '2') {

decrypted.push\_back('2');

break;

}

if (text[i] == '3') {

decrypted.push\_back('3');

break;

}

if (text[i] == '4') {

decrypted.push\_back('4');

break;

}

if (text[i] == '5') {

decrypted.push\_back('5');

break;

}

if (text[i] == '6') {

decrypted.push\_back('6');

break;

}

if (text[i] == '7') {

decrypted.push\_back('7');

break;

}

if (text[i] == '8') {

decrypted.push\_back('8');

break;

}

if (text[i] == '9') {

decrypted.push\_back('9');

break;

}

}

}

int f = 0;

while (f != decrypted.size()) //запись шифрованного текст в output.txt

{

OUTPUTFILE2 << decrypted[f];

f++;

}

OUTPUTFILE2.close();

}

};

int main(int args, const char \* argv[])

{

int N = 0;

vector<char> alphabet;

vector<char> NEWalphabet;

setlocale(LC\_ALL, "rus");

HelloWorld A;

srand(time(0));

random\_device random\_device;

mt19937 generator(random\_device());

uniform\_int\_distribution<> distribution(97, 122);

int x = 0;

cout << "Введите количество символов для генерации кодового слова" << endl;

cin >> N;

map<int, char> unique;

while (unique.size() != N)

{

x = distribution(generator);

unique.emplace(x, (char(x)));

}

int k = 0;

for (int i = 97;i < 123;i++) //заполнение массива начальным алфавитом

{

alphabet.push\_back((char(i)));

k++;

}

vector<char> password; //выбираем кодовое слово

for (auto it = unique.begin(); it != unique.end(); ++it)

{

password.push\_back(it->second);

}

cout << "KEYWORD --> ";

show\_vector(password);

cout << endl;

cout << "Old alphabet-->";

show\_vector(alphabet);

NEWalphabet = A.CreateNewAlpabet(alphabet, password); //создание нового алфавита

cout << endl;

cout << "New alphabet-->";

show\_vector(NEWalphabet);

vector<char> encryptedtext=A.FileEncoding(alphabet,NEWalphabet); // шифруем текст новым алфавитом и записываем в output.txt

A.Chastotniy\_analiz(encryptedtext); //дешифрируем методом частотного анализа

cout << endl;

system("pause");

return 0;

}