**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Основы программирования»

Отчет по лабораторной работе №7

«Обработка символьных строк»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-12Б  Казицин Алексей |  | преподаватель каф. ИУ5  Козлов А.Д. |
|  |  |  |
| 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 используемых переменных и функций с указанием наименования, типа (int, float, и т.п.) и назначения в программе:**

**Int size** – кол-во слов блокнота;

**int len** – длина слова;

**string\* notestr** – шифровальный блокнот;

**int s** – вспомогательная переменная (нужна для расшифровки);

**unsigned int\* noteint** – слова шифровального блокнота в int; **string mssg** – сообщение для шифрования;

**string cipher** – зашифрованное сообщение;

**i, j, k** – счетчики;

**int\*\* p** – двумерный массив для записи индекса символа в шифре и количества его повторений;

**bool t** – флаг, проверка на то, встречался ли символ в начальной строке;

**bool flag** – флаг, проверка на то, есть ли введенный символ в сообщении или нет;

**char name** – символ, для которого выводится статистика;

**char ex** – вспомогательная переменная, которая определяет, выходим мы из программы или нет;

Текст программы

#include <iostream>

#include <iomanip>

#include <stdio.h>

#include <windows.h>

#include <string>

using namespace std;

typedef unsigned int ui;

int main(){

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int size, len;

cout << "Введите кол-во слов вашего блокнота.\n";

cin >> size;

string\* notestr = new string[size]; // шифровальный блокнот

ui\* noteint = new ui[size]; // слова шифровального блокнота в int

int s = 0;

cout << "Введите шифровальный блокнот:\n";

for (ui i = 0; i < size; i++)

cin >> notestr[i];

for (ui i = 0; i < size; i++){ // подсчет и запись слов

len = notestr[i].length();

noteint[i] = 0;

for (ui j = 0; j < len; j++)

noteint[i] += notestr[i][j];

noteint[i] %= 256;

}

for (ui i = 0; i < size; i++) // чистка памяти

notestr[i] = "\0";

delete[] notestr;

cout << "\nВведите сообщение, которое хотите зашифровать:\n\n";

cin.ignore();

string mssg; // сообщение для шифрования

getline(cin, mssg);

len = mssg.length();

cout << "\n";

cout << "Закодированное сообщение:" << endl;

string cipher(len, 'a'); // зашифрованное сообщение

for (ui i = 0; i < len; i++){// шифрование

cipher[i] = (char)(((int)mssg[i] + noteint[i % size]) % 256);

cout << cipher[i];

}

cout << endl;

cout << "Декодированное сообщение:" << endl;

for (ui i = 0; i < len; i++){ // процесс расшифровки

s = (int)cipher[i] - noteint[i % size] + 256;

cout << (char)s;

}

for (ui i = 0; i < size; i++) // чистка памяти

noteint[i] = 0;

delete[] noteint;

cout << "\n\n";

bool t = true; int m;

int\*\* p = new int\* [2]; // двумерный массив для записи индекса символа в шифре и количестве его повторений

for (int i = 0; i < 2; i++) // [0][ ] - для индекса символа, [1][ ] - для количества повторений

p[i] = new int[len];

for (ui i = 0; i < len; i++) // обнуляем строку количества повторений символов шифра

p[1][i] = 0;

char ex, name;

do{

bool flag = false;

cout << "Введите символ, статистику которого вы хотите увидеть: ";

cin >> name;

for (ui i = 0; i < len; i++){ // вывод статистики

s = 0; m = 0; t = true;

if (name == mssg[i]){

flag = true;

for (ui j = 0; j < i; j++){ // проверяем, встречался ли раньше символ в начальном сообщении

if (mssg[i] == mssg[j]){

t = false;

break;

}

}

if (t){ // если не встречался - продолжаем, иначе статистика по нему уже собрана

cout << "Символ '" << mssg[i] << "' встречается - ";

for (ui j = i; j < len; j++) // собираем индексы символа для их проверки в шифре

if (mssg[j] == mssg[i])

p[0][s++] = j;

for (ui j = 0; j < s; j++){ // работаем с отобранными индексами

t = true;

for (ui k = 0; k < j; k++) // проверяем, шифровался ли он таким символом раньше

if (cipher[p[0][j]] == cipher[p[0][k]]){ // если встречался ранее, количество повторений ++

p[1][k]++;

t = false;

break;

}

if (t){ // если встретился впервые, записываем как новый

p[0][m] = p[0][j];

p[1][m++]++;

}

}

cout << s << " раз:\n";

for (ui j = 0; j < m; j++){ // вывод статистики

cout << "\t зашифрован в виде '" << cipher[p[0][j]] << "' и кодом " << (int)(unsigned char)cipher[p[0][j]] << " - " << p[1][j] << " раз;\n";

p[1][j] = 0;

}

}

}

}

if (flag == true){

cout << "Выйти? Введите 'y' если да, 'n' если нет: ";

cin >> ex;

} else {

cout << "Такого символа нет в строке. Введите другой." << endl;

}

} while (ex != 'y');

for (ui i = 0; i < len; i++){ // чистка памяти

cipher[i] = '\0';

mssg[i] = '\0';

p[0][i] = 0;

}

for (int i = 0; i < 2; i++)

delete[] p[i];

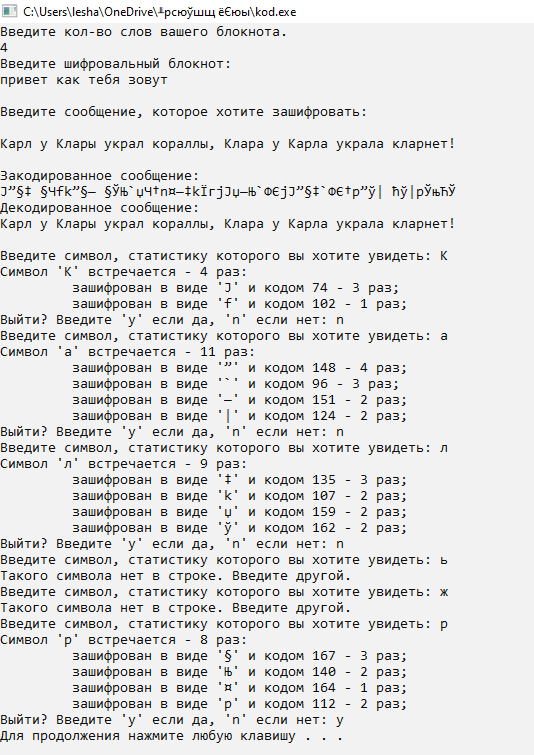
delete[] p;

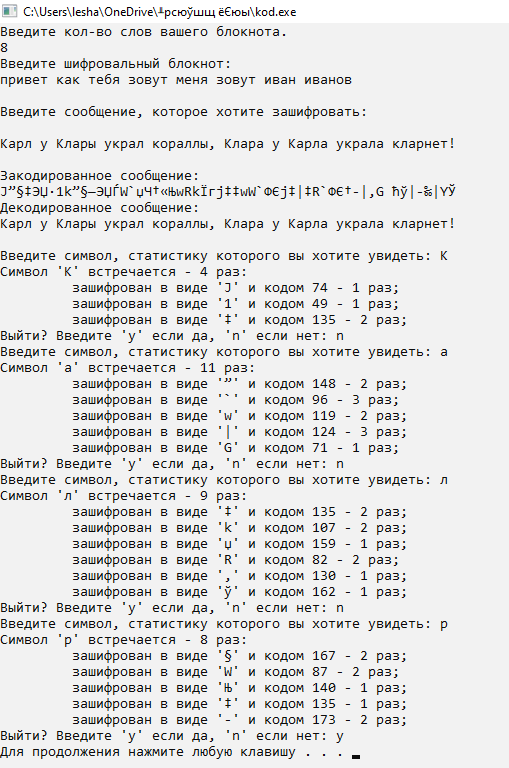
system("pause");

return 0;

}

Анализ результатов





Из этих двух тестов видно, что чем больше блокнот, тем лучше производится шифрование (меньше повторений для одних и тех же символов).