Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Лабораторная работа № 1

**«Разработка программы шифрования/дешифрирования текста с**

**использованием перестановочного шифра.»**

## Отчёт по дисциплине

**«Методы защиты КС»**

Пенза 2020

**Выполнили с-ты гр. 17ВВ2:**

Живаев В.  
Безжонов Р.

**Приняли:**

к.т.н. Дубравин А.В.  
Кармышева Н.С.

**Цель работы.**

Научиться разрабатывать программы шифрования/дешифрования текста с использованием перестановочного шифра.

**Задание.**

Разработать программу, выполняющую шифрование ирасшифровывание произвольного текстового файла с использованиемперестановочного шифра используя в качестве ключа последовательность,соответствующую номеру варианта (табл. 1). Выполнить проверку путем двоичногосравнения исходного файла и фала, полученного после расшифровывания.

Таблица 1 - Вариант задания.

|  |  |
| --- | --- |
| № Варианта | Последовательность перестановки |
| 6 | 2 5 9 3 4 10 1 7 6 8 |

**Листинг.**

// CryptpoLab1.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "pch.h"

#include <iostream>

#include <vector>

#include<locale>

#include <Windows.h>

#include<fstream>

#include <conio.h>

using namespace std;

bool debug = false;

int key[10] = { 4, 1, 6, 2, 9, 3, 5, 7, 10, 8 };

int getFileSize(const wchar\_t\* toCryptFileName);

void encrypt(int key[], const wchar\_t\* toCryptFileName, const wchar\_t\* toDecryptFileName = L"Encrypt.txt");

void encrypt(int key[], const wchar\_t\* toCryptFileName, const wchar\_t\* toWriteFileName) {

char sourceChar;

ifstream read(toCryptFileName, ios::binary);

vector<char> vec1, vec2;

while (read.get(sourceChar))

{

vec1.push\_back(sourceChar);

}

read.close();

int check = 10 - vec1.size() % 10, part = 0, encryptionKeyPosition = 0;

//не все строки можно поделить на блоки по 10 элементов для шифрования

if (check != 10) {

for (int i = 0; i < check; i++) {

vec1.push\_back(' '); // дополняем пробелами

}

}

//непосредственно перестановка

for (int i = 0; i < vec1.size(); i++)

{

vec2.push\_back(vec1[part + key[encryptionKeyPosition] - 1]); //отнимаем единицу т.к. индексы идут с 0..9

if (encryptionKeyPosition > 8) { //>8, значит до 9

encryptionKeyPosition = 0;

part += 10;

}

else

{

encryptionKeyPosition++;

}

}

int msgSize = getFileSize(toCryptFileName);

ofstream write(toWriteFileName, ios::trunc | ios::binary);

write << msgSize;

for (int i = 0; i < (vec2.size()); i++)

{

write << vec2[i]; //запись результата в файл

}

write.close();

cout << "Выполнено.\n";

}

char\* cutString(char\* str, int targetSize) {

char\* result = 0;

int i = 0;

result = (char\*)malloc(sizeof(char\*)\*targetSize + 1);

memset(result, 0, sizeof(result));

for (i; i < targetSize; i++)

{

result[i] = str[i];

}

free(str);

result[i] = '\0';

return result;

}

void decrypt(int key[],

const wchar\_t\* toDecryptFileName = L"Encrypted.txt",

const wchar\_t\* toWriteFileName = L"Decrypted.txt",

const wchar\_t\* originalFile = L"Source.txt") {

char sourceChar;

vector<char> vec1, vec2;

int originFileSize = 0;

ifstream read(toDecryptFileName, ios::binary);

read >> originFileSize;

while (read.get(sourceChar))

{

vec1.push\_back(sourceChar);

}

read.close();

//доплняем пробелами, если нужно

int check = 10 - vec1.size() % 10, part = 0, encryptionKeyPosition = 0;

if (check != 10) {

for (int i = 0; i < check; i++) {

vec1.push\_back(' ');

}

}

//сразу расширяем второй, результирующий вектор до размеров первого

//т.к. будем совершать перестановку и обращаться по индексу.

vec2.resize(vec1.size());

for (int i = 0; i < vec2.size(); i++)

{

vec2[part + key[encryptionKeyPosition] - 1] = vec1[i];

if (encryptionKeyPosition > 8) {

encryptionKeyPosition = 0;

part += 10;

}

else

{

encryptionKeyPosition++;

}

}

int fileSize = getFileSize(toDecryptFileName);

ofstream write(toWriteFileName, ios::binary);

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

{

write << vec2[i];

}

write.close();

cout << "Выполнено.\n";

if (debug) {

char\* command = 0;

wchar\_t\* fc = NULL;

fc = (wchar\_t\*)malloc(sizeof(wchar\_t) \* 2);

memset(fc, 0, sizeof(fc));

lstrcatW(fc, L"FC ");

lstrcatW(fc, toWriteFileName);

lstrcatW(fc, L" ");

lstrcatW(fc, originalFile);

lstrcatW(fc, L"\0");

command = (char\*)malloc(sizeof(char) \* lstrlenW(fc));

memset(command, 0, sizeof(command));

wcstombs(command, fc, lstrlenW(fc));

strcat(command, "\0");

command = cutString(command, lstrlenW(fc));

cout << "Сравнение текстовых файлов...\n";

system(command);

cout << "Выполнено.\n";

system("pause");

free(command);

}

}

int getFileSize(const wchar\_t\* toCryptFileName) {

char sourceChar;

int size = 0;

ifstream read(toCryptFileName, ios::binary);

while (read.get(sourceChar))

{

size++;

}

read.close();

return size;

}

bool isDebugEnabled(int args, wchar\_t \*argv[]) {

for (int i = 0; i < args; i++) {

if ((lstrcmpW(argv[i], L"-d") == 0 || lstrcmpW(argv[i], L"--debug") == 0)

&& ((bool)argv[i + 1] == true || (bool)argv[i + 1] == false))

{

if (lstrcmpW(argv[i + 1], L"false") == 0) return false;

if (lstrcmpW(argv[i + 1], L"true") == 0) return true;

}

}

return false;

}

bool isHelpRequested(int args, wchar\_t \*argv[]) {

for (int i = 0; i < args; i++) {

if (args == 2 && lstrcmpW(argv[i], L"-h") == 0) {

printf("Мануал:\n\t\t -h Вызов man\n\t\t -p указание пути к файлу или имени файла, \nесли он находится в текущей директории\n\t\t -d Режим Debug (Выведет подробный diff файлов)");

printf("\n\t\t --encrypt filePathToEncrypt filePathToWrite вызов операции шифрования, если не указывать имена фалов, то будут использованы стандартные\n\t\t --decrypt filePathToDecrypt filePathToWrite originalFileToCompare вызов операции расшифровки");

exit(0);

}

}

}

int validateArg(wchar\_t\* arg, const wchar\_t\* argName) {

if (arg == NULL) {

wcout << argName << L" аргумент отсутствует проверьте мануал ключ -h" << endl;

exit(-666);

}

return 1;

}

void isEncryptCalled(int argc, wchar\_t \*argv[]) {

for (int i = 0; i < argc; i++) {

if (lstrcmpW(argv[i], L"--encrypt") == 0) {

if (argc < 4) {

printf("Возможно формат команды неверен, проверьте мануал. Ключ \"-h\"\n");

exit(-1);

}

validateArg(argv[i + 1], L"originalFileName");

validateArg(argv[i + 2], L"toEncryptFileName");

encrypt(key, argv[i + 1], argv[i + 2]);

}

}

}

void isDecryptCalled(int argc, wchar\_t \*argv[]) {

for (int i = 0; i < argc; i++) {

if (lstrcmpW(argv[i], L"--decrypt") == 0) {

if (argc < 4) {

printf("Возможно формат команды неверен, проверьте мануал. Ключ \"-h\"\n");

exit(-1);

}

validateArg(argv[i + 1], L"toDecryptFileName");

validateArg(argv[i + 2], L"encryptedFileName");

if (debug) {

validateArg(argv[i + 3], L"originalFileName");

}

decrypt(key, argv[i + 1], argv[i + 2], argv[i + 3]);

}

}

}

void printAllArgs(int argc, wchar\_t \*argv[]) {

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

{

wcout << argv[i] << endl;

}

}

int wmain(int argc, wchar\_t \*argv[], wchar\_t \*envp[])

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

int c, key[10] = { 4, 1, 6, 2, 9, 3, 5, 7, 10, 8 };

debug = isDebugEnabled(argc, argv);

isHelpRequested(argc, argv);

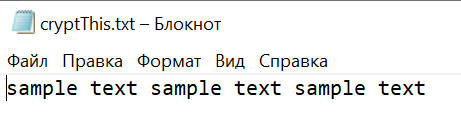
isEncryptCalled(argc, argv);

isDecryptCalled(argc, argv);

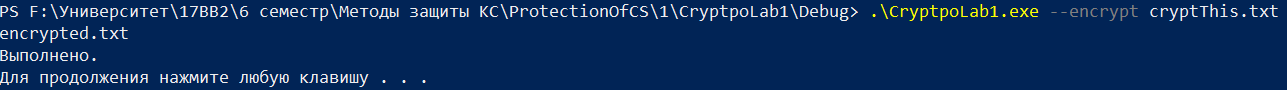
}

**Результат работы.**

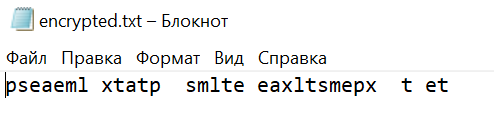
Результат работы программы показан на рисунках 1-5.



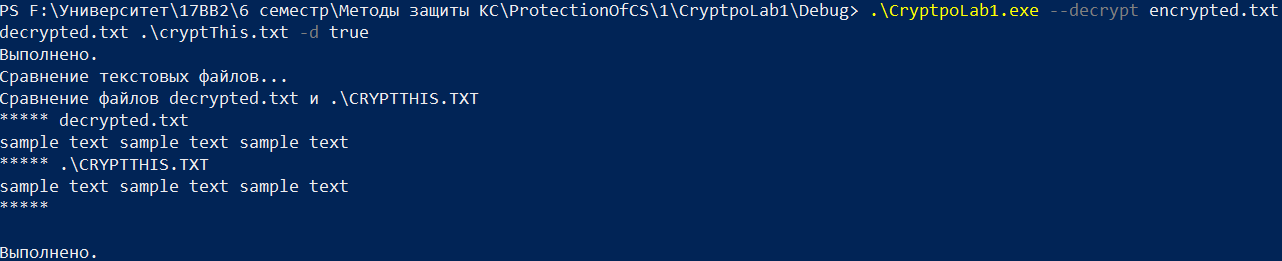
**Рисунок 1 - Исходный файл.**



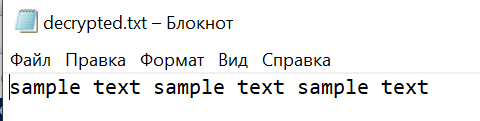
**Рисунок 2 – шифрование через аргументы командной строки.**



**Рисунок 3 - Результат после шифрования.**



**Рисунок 4 - Результат после шифрования.**



**Рисунок 5 - Результат после дешифрования.**

Исходный текст файла совпал с результатом дешифрования.

**Вывод.**

Научились разрабатывать программы шифрования/дешифрования текста с использованием перестановочного шифра.