Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Лабораторная работа № 2

**«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.**

// l2.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <fstream>

#include <iostream>

#include <Windows.h>

#include <conio.h>

#include <vector>

#include <math.h>

#define BLOCK\_SIZE 5

int key[] = { 12, 13, 30, 5, 27, 6, 11, 25, 3, 21, 22, 2, 23, 0, 8, 4, 18, 19, 10, 1, 14, 29, 9, 28, 20, 17, 26, 31, 7, 16, 15, 24 };

const unsigned int SIZE\_BLOCK = (sizeof(key) / sizeof(int));

using namespace std;

typedef unsigned int Block;

vector<char> vec1;

int getBit(Block block, int numBit)

{

return ((block & (1 << numBit)) != 0);

}

int setBit(Block block, int numBit)

{

return (block | (1 << numBit));

}

int unsetBit(Block block, int numBit)

{

return (block & ~(1 << numBit));

}

int cryptBlock(Block block)

{

Block tempBlock = 0;

for (int i = 0; i < SIZE\_BLOCK; i++)

{

int tempBit = getBit(block, key[i]);

if (tempBit == 1)

tempBlock = setBit(tempBlock, i);

}

return tempBlock;

}

int decryptBlock(Block block)

{

Block tempBlock = 0;

for (int i = 0; i < SIZE\_BLOCK; i++)

{

int tempBit = getBit(block, i);

if (tempBit == 1)

tempBlock = setBit(tempBlock, key[i]);

}

return tempBlock;

}

void printBlock(Block block, bool debug = false)

{

if (debug)

{

printf("NumBit\t");

for (int i = SIZE\_BLOCK; i > 0; i--)

{

printf("%3d", i);

}

printf("\n");

}

printf("Block\t");

for (int i = SIZE\_BLOCK - 1; i >= 0; i--)

{

printf("%3d", getBit(block, i));

}

printf("\n");

}

Block performBlock(char buffer[BLOCK\_SIZE]) {

Block block = 0;

block |= buffer[0] << 24;

block |= (buffer[1] << 16) & 0x00ffffff;

block |= buffer[2] << 8 & 0x0000ffff;

block |= buffer[3] & 0x000000ff;

return block;

}

char\* performBufferFromBlock(Block block, char buffer[BLOCK\_SIZE]) {

buffer[0] = (block >> 24) & 255;

buffer[1] = (block >> 16) & 255;

buffer[2] = (block >> 8) & 255;

buffer[3] = (block) & 255;

return buffer;

}

void push\_back\_buffer(char buffer[BLOCK\_SIZE]) {

vec1.clear();

for (int i = 0; i < strlen(buffer); i++)

{

vec1.push\_back(buffer[i]);

}

}

void encrypt() {

ifstream read("Source.txt", ios::binary);

ofstream write("Encrypted.txt", ios::binary | ios::trunc);

char buffer[BLOCK\_SIZE];

Block block;

while (read.get(buffer, BLOCK\_SIZE))

{

block = performBlock(buffer);

block = cryptBlock(block);

performBufferFromBlock(block,buffer);

write.write(buffer, BLOCK\_SIZE);

}

write.close();

read.close();

}

void decrypt() {

ifstream read("Encrypted.txt", ios::binary);

ofstream write("Decrypted.txt", ios::binary | ios::trunc);

char buffer[BLOCK\_SIZE];

Block block;

while (read.getline(buffer, BLOCK\_SIZE,'\0')) {

block = performBlock(buffer);

block = decryptBlock(block);

if (!strcmp(buffer, " ")) {

continue;

}

performBufferFromBlock(block, buffer);

push\_back\_buffer(buffer);

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

write << vec1[i];

}

}

write.close();

read.close();

}

void menu() {

char menu\_word;

cout << "1) Зашифровать исходный файл.\n";

cout << "2) Расшифровать исходный файл.\n";

cout << "3) Сравнить исходный и результирующий файлы.\n";

cout << "ESC) Выход.\n";

switch ((menu\_word = \_getch()))

{

case '1':

cout << "Шифрование исходного текста...\n";

encrypt();

cout << "Выполнено.\n";

system("pause");

break;

case '2':

cout << "Расшифровка текста...\n";

decrypt();

cout << "Выполнено.\n";

system("pause");

break;

case '3':

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

system("FC Source.txt Decrypted.txt");

cout << "Выполнено.\n";

system("pause");

break;

case 27:

cout << "Завершение процессов и закрытие программы....\n";

exit(0);

break;

default:

break;

}

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

menu();

while (true) {

system("cls");

menu();

}

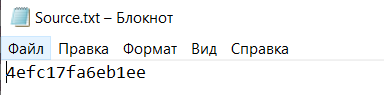
system("pause");

return 0;

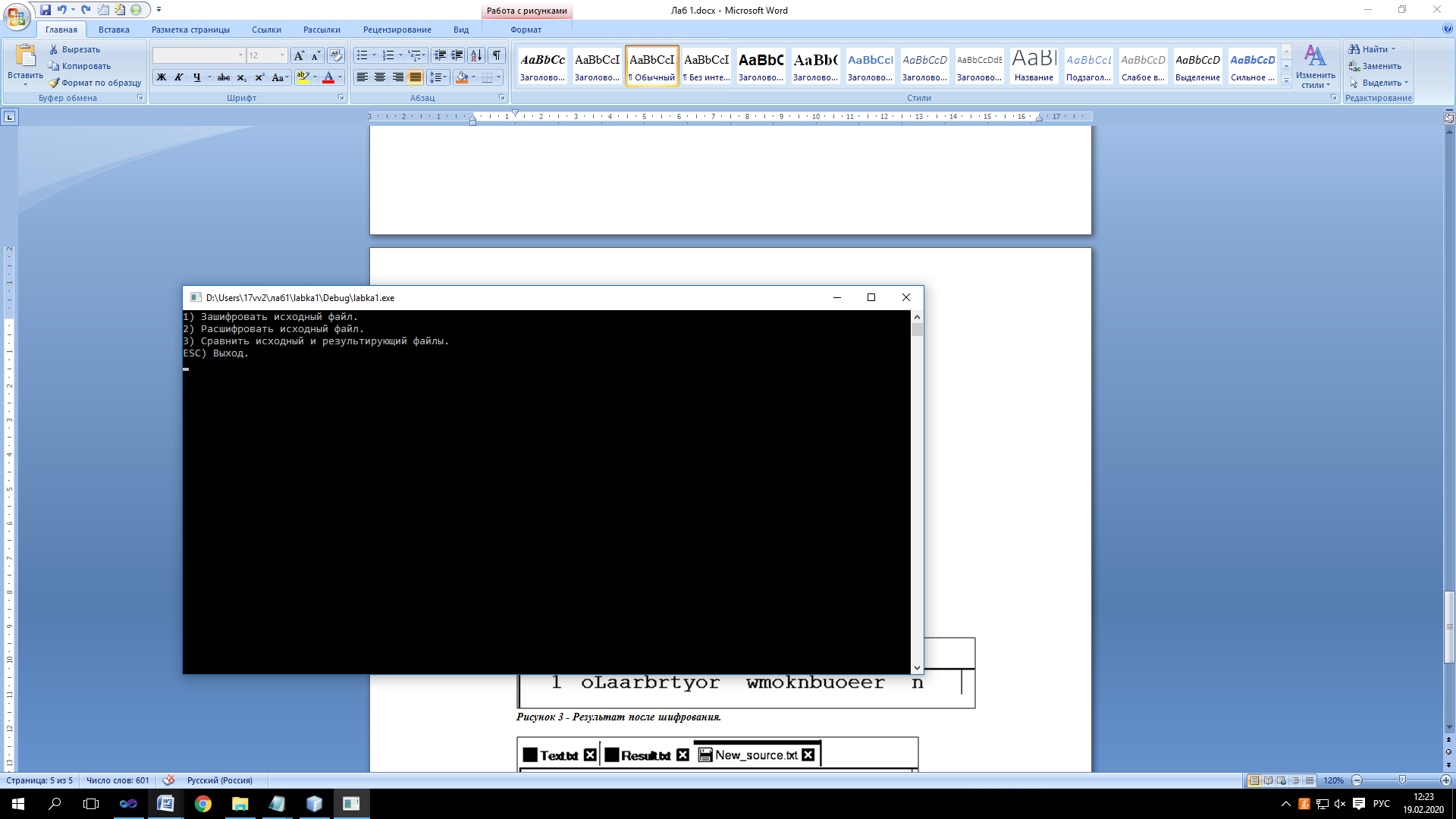
}

**Результат работы.**

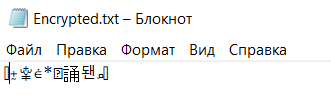
Результат работы программы показан на рисунках 1-5.



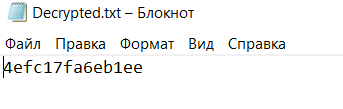
**Рисунок 1 - Исходный файл.**



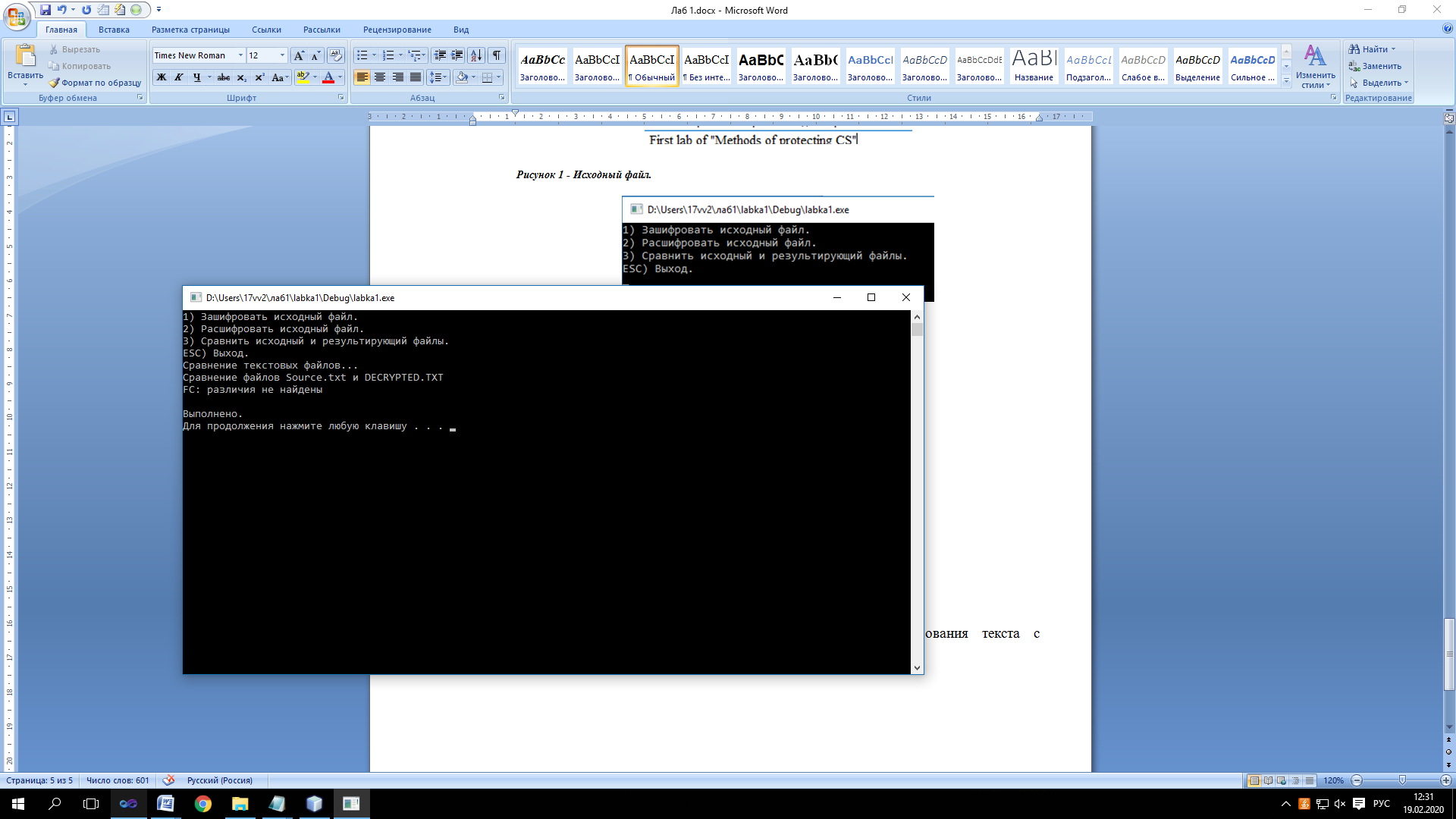
**Рисунок 2 - Меню программы.**



**Рисунок 3 - Результат после шифрования.**



**Рисунок 4 - Результат после дешифрования.**



**Рисунок 5 - Результат проверки файлов.**

Исходный текст файла совпал с результатом дешифрования.

**Вывод.**

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