Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по курсу «Защита информации в сети Internet»

по лабораторной работе №4

на тему «Стеганография»

Выполнили  
студенты группы 18ВВ1:

Гросс Д.А

Репин И.А

Щербакова К.А

Приняли:  
к.т.н., доцент Дубравин А.В.

Пенза 2020

**Цель работы:** Изучить методы стеганографии.

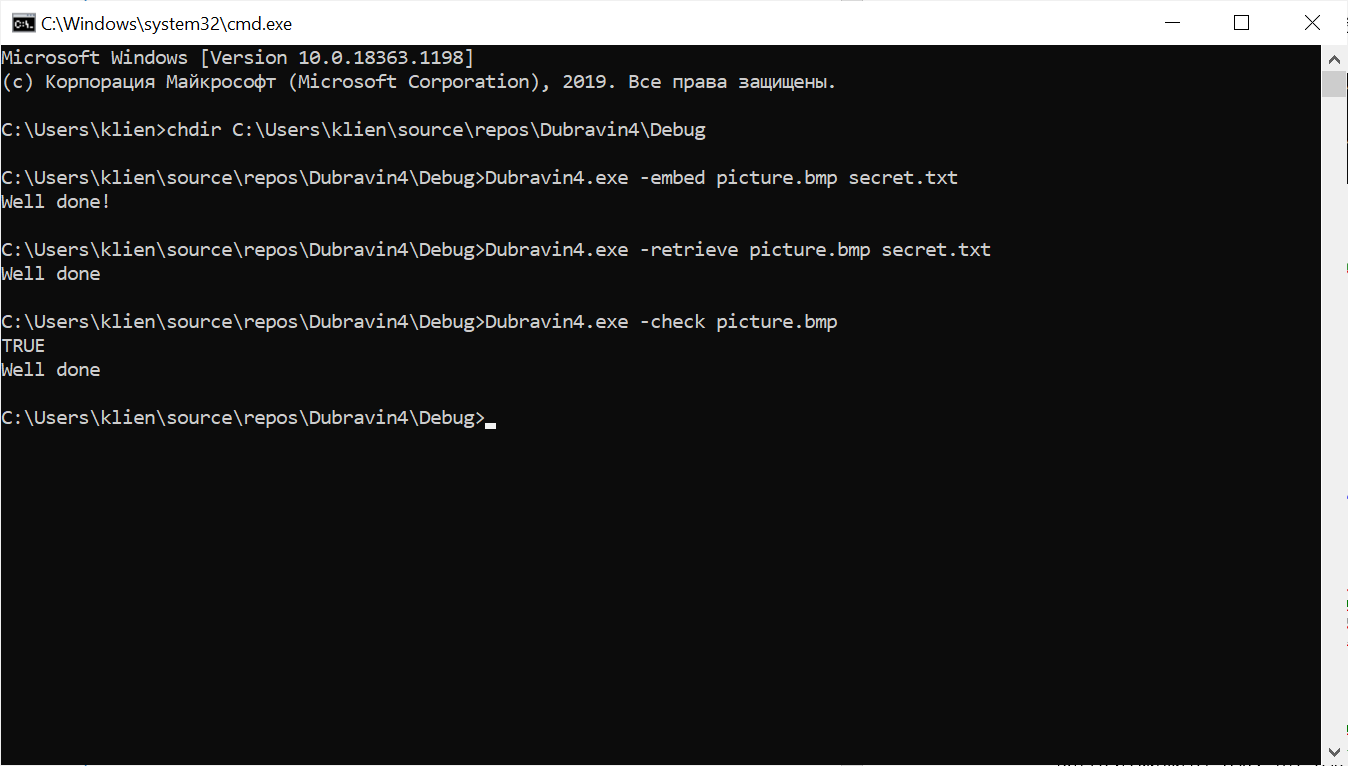
**Задание:**

Разработать программу, выполняющую внедрение, извлечение и проверку наличия некоторых данных в файле с изображением (формат файла с изображением — BMP, не содержащий плитру и имеющий значение biBitCount равное 24).

Тип операции (внедрение, извлечение и проверка), а также имена файлов, участвующих в каждой операции следует передавать в программу через командную строку.

**Результат работы программы:**

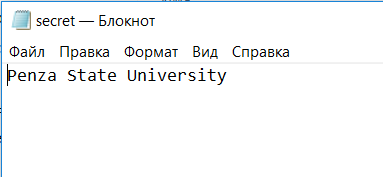
На рисунке ниже представлен варианты работы программы.



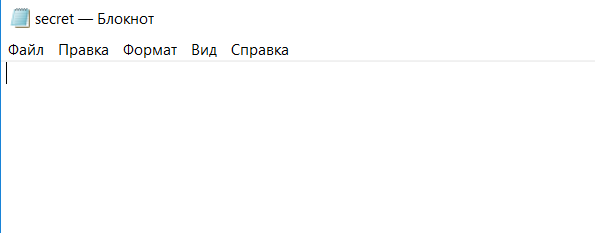
**Рисунок 1Результат работы программы**

Dubravin4.exe -embed picture.bmp secret.txt – данная операция помещает секретное содержимое secret.txt в picture.bmp.

Начальный файл с секретным сообщением:

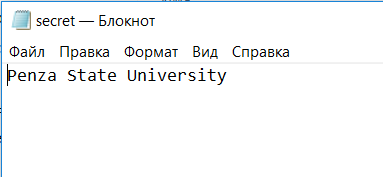


Секретное сообщение извлечено из текстового файла:



Dubravin4.exe -retrieve picture.bmp secret.txt – эта операция извлекает секретное сообщение из изображения в текстовый файл.

Секретное сообщение извлечено из изображения:



Dubravin4 -check picture.bmp – операция проверяет содержание кодового слова в изображении.

**Листинг программы:**

#include <iostream>

#include <Windows.h>

#include <conio.h>

#include <fstream>

#include <string.h>

using namespace std;

char \*secret = new char[6]{ 's', 'e', 'c', 'r', 'e', 't' }; //тайное слово

void write\_secret(const wchar\_t \*picture, const wchar\_t \*text) {

char \*buffer\_picture;

char \*buffer\_text;

ifstream BMPFile(picture, ios::binary); //открытие бинарного файла

size\_t size\_picture = BMPFile.seekg(0, ios::end).tellg(); //указатель в конец, вычисляем размер файла

BMPFile.seekg(0, ios::beg); //возвращаем указатель на начало

buffer\_picture = new char[size\_picture]; //выделяем память для буффера изображения

BMPFile.read(buffer\_picture, size\_picture); //записываем файл в буффер

BMPFile.close();

ifstream TXTFile(text, ios::binary); //открытие бинарного файла

size\_t size\_text = TXTFile.seekg(0, ios::end).tellg(); //указатель в конец, вычисляем размер файла

TXTFile.seekg(0, ios::beg); //возвращаем указатель на начало

buffer\_text = new char[size\_text]; //выделяем память для буффера тескта

TXTFile.read(buffer\_text, size\_text); //записываем файл в буффер

TXTFile.close();

int bit;

int mask = 0;

int num\_bit = 31;

int width\_picture = 0;

for (int i = 21; i > 17; i--) {

for (bit = 7; bit > -1; bit--) {

mask = ((buffer\_picture[i] >> bit) & 1);

if (mask == 1) {

width\_picture += pow(2, num\_bit);

}

num\_bit--;

}

}

num\_bit = 31;

int high\_picture = 0;

for (int i = 25; i > 21; i--) {

for (bit = 7; bit > -1; bit--) {

mask = ((buffer\_picture[i] >> bit) & 1);

if (mask == 1) {

high\_picture += pow(2, num\_bit);

}

num\_bit--;

}

}

int size\_for\_secret = (high\_picture \* width\_picture \* 3 - sizeof(size\_t) - sizeof(secret)) / 4;

if (size\_text < size\_for\_secret) {

bit = 15;

mask = 0;

for (int i = 54; i < 54 + sizeof(size\_t) \* 4; i++) {

buffer\_picture[i] = buffer\_picture[i] & 0xFC;

for (int j = 1; j > -1; j--) {

mask = ((size\_text >> bit) & 1) << j;

buffer\_picture[i] = buffer\_picture[i] | mask;

bit--;

}

}

bit = 7;

int num\_text = 0; //отвечает за номер бита который будем записывать

size\_text = 54 + sizeof(size\_t) \* 4 + (size\_text \* 4); //записываем с 62 до 62+ размер картинки умноженный на 4

for (int i = 54 + sizeof(size\_t) \* 4; i < size\_text; i++) {

buffer\_picture[i] = buffer\_picture[i] & 0xFC;

for (int j = 1; j > -1; j--) { //то же самое что и с размером

mask = ((buffer\_text[num\_text] >> bit) & 1) << j;

buffer\_picture[i] = buffer\_picture[i] | mask;

bit--;

}

if (bit == -1) { //если мы записали все 8 бит

num\_text++; //то передвигаем указатель на следующий

bit = 7;

}

}

num\_text = 0;

bit = 7;

int size = size\_text + (6 \* 4); //записываем секретное сообщение

for (int i = size\_text; i < size; i++) { //начинаем после записи текстового файла

buffer\_picture[i] = buffer\_picture[i] & 0xFC;

for (int j = 1; j > -1; j--) {

mask = ((secret[num\_text] >> bit) & 1) << j;

buffer\_picture[i] = buffer\_picture[i] | mask;

bit--;

}

if (bit == -1) {

num\_text++;

bit = 7;

}

}

ofstream BMPprintFile(picture, ios\_base::trunc | ios::binary);

for (int i = 0; i < size\_picture; i++) {

BMPprintFile << buffer\_picture[i]; //поэлементно перезаписываем картинку

}

ofstream TXTFilePrint(text, ios::binary);

TXTFilePrint.open(text);

TXTFilePrint.close();

}

else {

cout << ("Text file is bigger than max size") << endl;

}

}

void read\_secret(const wchar\_t \*picture, const wchar\_t \*text) {

char \*buffer\_picture;

char \*buffer\_text;

ifstream BMPFile(picture, ios::binary); //чтение из файла

size\_t size\_picture = BMPFile.seekg(0, ios::end).tellg(); //указатель на конец,что бы узнать размер

BMPFile.seekg(0, ios::beg); //возвращаем указатель на начало

buffer\_picture = new char[size\_picture]; //выделяем память для буффера изображения

BMPFile.read(buffer\_picture, size\_picture); //записываем файл в буффер

BMPFile.close();

int size = 0;

int mask = 0;

int bit = 15;

for (int i = 54; i < 54 + sizeof(size\_t) \* 4; i++) { //получаем размер текста

for (int j = 1; j > -1; j--) {

mask = (buffer\_picture[i] >> j) & 1;

mask = mask << bit;

size = size | mask;

bit--;

}

buffer\_picture[i] = buffer\_picture[i] & 0xFC;

}

buffer\_text = new char[size]; //приравниваем размер буффера тхт к полученному размеру

bit = 7;

int num\_text = 0;

int mask\_text = 0;

int size\_res = size \* 4;

size\_res = 54 + sizeof(size\_t) \* 4 + size\_res;

for (int i = 54 + sizeof(size\_t) \* 4; i < size\_res; i++) { //получаем текстовое сообщение из картинки

for (int j = 1; j > -1; j--) {

mask = (buffer\_picture[i] >> j) & 1;

mask = mask << bit;

mask\_text = mask\_text | mask;

bit--;

}

buffer\_picture[i] = buffer\_picture[i] & 0xFC;

if (bit == -1) {

buffer\_text[num\_text] = mask\_text;

mask\_text = 0;

num\_text++;

bit = 7;

}

}

for (int i = size\_res; i < size\_res + 6 \* 4; i++) {

buffer\_picture[i] = buffer\_picture[i] & 0xFC;

}

ofstream BMPprintFile(picture, ios\_base::trunc | ios::binary);

for (int i = 0; i < size\_picture; i++) {

BMPprintFile << buffer\_picture[i]; //поэлементно перезаписываем картинку

}

ofstream TXTprint(text, ios::binary);

int sz = sizeof(buffer\_text);

for (int i = 0; i < size; i++) { //записываем полученное сообщение в txt

TXTprint << buffer\_text[i];

}

}

void check\_secret(const wchar\_t \*picture) {

char \*bufferbmp;

char \*buffertxt;

ifstream BMPFile(picture, ios::binary); //чтение из файла

size\_t size1 = BMPFile.seekg(0, ios::end).tellg(); //указатель в конец,что бы узнать размер

BMPFile.seekg(0, ios::beg); //возрващаем указатель на начало

bufferbmp = new char[size1]; //выделяем память для буффера изображения

BMPFile.read(bufferbmp, size1); //записываем файл в буффер

BMPFile.close();

int size = 0;

int mask = 0;

int bit = 15;

for (int i = 54; i < 54 + sizeof(size\_t) \* 4; i++) { //получаем размер текста

for (int j = 1; j > -1; j--) {

mask = (bufferbmp[i] >> j) & 1;

mask = mask << bit;

size = size | mask;

bit--;

}

}

int txtnext = 0;

bit = 7;

int masktxt = 0;

buffertxt = new char[sizeof(secret)];

size = size \* 4;

size = size + 54 + sizeof(size\_t) \* 4;

int sizee = size + (8 \* 4);

for (int i = size; i < sizee; i++) { //тайное сообщение находится после 62 байта плюс размер картинки умноженный на 4

for (int j = 1; j > -1; j--) {

mask = (bufferbmp[i] >> j) & 1;

if (mask == 1) {

masktxt = masktxt | (1 << bit);

}

else {

mask = 1;

mask = mask << bit;

mask = ~mask;

masktxt = masktxt & mask;

}

bit--;

}

if (bit == -1) {

buffertxt[txtnext] = masktxt;

txtnext++;

bit = 7;

}

}

bool check = TRUE;

for (int i = 0; i < 6; i++) {

if (buffertxt[i] != secret[i]) {

check = FALSE;

}

}

if (check == FALSE) {

cout << ("TRUE") << endl;

}

else {

cout << ("FALSE") << endl;

}

}

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

{

if (!wcscmp(argv[1], TEXT(L"-embed"))) {

write\_secret(argv[2], argv[3]);

cout << ("Well done!") << endl;

}

if (!wcscmp(argv[1], TEXT(L"-retrieve"))) {

read\_secret(argv[2], argv[3]);

cout << ("Well done") << endl;

}

if (!wcscmp(argv[1], TEXT(L"-check"))) {

check\_secret(argv[2]);

cout << ("Well done") << endl;

}

}

**Вывод:** Мы изучили методы стеганографии.