Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Защита информации в сети Internet»

на тему «Стеганография»

Выполнили  
студенты группы 18ВВ1:

Ручкин М.А.

Коновалова Д.А.

Дубинин А.В.

Приняли:

к.т.н., доцент Дубравин А.В.

к.т.н., доцент Карамышева Н.С.

2020

**Цель работы:** изучить методы стеганографии.

**Задания:** Разработать программу, выполняющую внедрение, извлечение и проверку наличия некоторых данных в файле с изображением (формат файла с изображением — BMP, не содержащий палитру 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.

**Описание программы**

В программе выполняется одна из трёх операций (внедрение, извлечение или проверка) в зависимости от передаваемых параметров из командной строки.

При внедрении открывается изображение. Всё его содержимое заносится в динамическую память. Определяется место в изображении откуда начинаются цвета пикселей. Открывается текстовый файл, его содержимое помещается в динамическую память. Определяется размер текста. Далее в динамическую память, хранящую изображение, в младший бит байта записывается бит информации. Сначала записывается метка, потом размер текстового файла и его содержимое. По завершению формирования нового изображения, происходит удалении информации в оригинальном изображении и запись новых данных.

При проверке в динамическую память помещаются 100 байт изображения. Определяется смещение и с этой позиции восстанавливается метка из файла. Затем полученная метка сравнивается с оригинальной. Если они одинаковы, то выводится соответствующее сообщение.

Когда выполняется извлечение, то также открывается изображение и его содержимое помещается в динамическую память. Определяется смещение, проверяется метка, восстанавливается размер текстового файла. После этого под текст выделяется динамическая память и он восстанавливается из изображения. Далее создаётся текстовый файл, в который записывается восстановленный текст.

**Листинг**:

Файл Header.h

#pragma once

#include <Windows.h>

#include <iostream>

using namespace std;

//Прототипы функций

void Offset(unsigned int \*, const char \*);

void Check(const wchar\_t \* fileName);

void Retrieve(const wchar\_t \* imageName, const wchar\_t \*textName);

void Embed(const wchar\_t \* imageName, const wchar\_t \*textName);

//Константы

const char cMask = 0x01;//Для выдёргивания из текста бита

const char cMask\_2 = 0xFE;//Для обнуления младшего бита байта изображения

const char cTag[5] = "MARK";

Файл Functional.cpp

#include "Header.h"

#include <Windows.h>

#include <iostream>

using namespace std;

void Offset(unsigned int \* pOffset, const char \* Buffer)

{

\*pOffset = 0; //Получение смещения

\*pOffset = \*pOffset | Buffer[13];

\*pOffset = \*pOffset << 8;

\*pOffset = \*pOffset | Buffer[12];

\*pOffset = \*pOffset << 8;

\*pOffset = \*pOffset | Buffer[11];

\*pOffset = \*pOffset << 8;

\*pOffset = \*pOffset | Buffer[10];

}

void Check(const wchar\_t \* fileName) //Проверка на наличие секретного сообщения

{

HANDLE hFile;

char \* Buff;

DWORD dwBytes;

//Открываем изображение

if (INVALID\_HANDLE\_VALUE == (hFile = CreateFile(fileName, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL)))

cout << "Can't open file\n";

//Читаем информацию о файле и 40 байт растра

Buff = new char[100];

ReadFile(hFile, Buff, 100, &dwBytes, NULL);//Берём 100 первых байт

unsigned int uiOffset;

Offset(&uiOffset, Buff);

//Формирование метки из изображения

//от смещения — 40 байт

unsigned char cTag\_2[5] = { 0 };

//Получение значений цветов и формирование новых

char cImageByte;

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 8; j++)

{

char cImageBit = 0;

cImageByte = Buff[uiOffset];

cImageBit = cImageByte & cMask;

cImageBit = cImageBit << 7;

cTag\_2[i] = cTag\_2[i] >> 1;

cTag\_2[i] = cTag\_2[i] | cImageBit;

uiOffset++;

}

}

char cTagTemp[5] = { cTag\_2[0], cTag\_2[1], cTag\_2[2], cTag\_2[3], cTag\_2[4] };//для сравнение с меткой

if (strcmp(cTag, cTagTemp))

{

printf("No secret text here.\n");

}

else

printf("Secret text.\n");

CloseHandle(hFile);

delete[] Buff;

}

void Retrieve(const wchar\_t \* imageName, const wchar\_t \*textName)

{

HANDLE hFile;

if (INVALID\_HANDLE\_VALUE == (hFile = CreateFile(imageName, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL)))

cout << "Can't open file\n";

DWORD dwBytes = 0, BuffSize = 0, dwBytes1 = 0;

BuffSize = GetFileSize(hFile, NULL);

char \* Buff = new char[BuffSize / sizeof(char)];

ReadFile(hFile, Buff, BuffSize, &dwBytes, NULL);

unsigned int uiOffset;

Offset(&uiOffset, Buff);

char cTagTemp[5], cImageByte;

unsigned char cTag\_2[5] = { 0 };

//Проверка метки

for (int i = 0; i < 5; i++)

cTag\_2[i] = 0;

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 8; j++)

{

char cImageBit = 0;

cImageByte = Buff[uiOffset];

cImageBit = cImageByte & cMask;

cImageBit = cImageBit << 7;

cTag\_2[i] = cTag\_2[i] >> 1;

cTag\_2[i] = cTag\_2[i] | cImageBit;

uiOffset++;

}

}

for (int i = 0; i < 5; i++)

cTagTemp[i] = cTag\_2[i];

if (!strcmp(cTag, cTagTemp))//Если метка есть узнаём размер текста и восстанавливаем его

{

//Считываем размер

unsigned char cTextSize[4] = { 0 };

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 8; j++)

{

char cImageBit = 0;

cImageByte = Buff[uiOffset];

cImageBit = cImageByte & cMask;

cImageBit = cImageBit << 7;

cTextSize[i] = cTextSize[i] >> 1;

cTextSize[i] = cTextSize[i] | cImageBit;

uiOffset++;

}

}

//Преобразование в int cTextSize

int iTextSize = 0;

for (int i = 3; i > -1; i--)

{

iTextSize = iTextSize << 8;

iTextSize = iTextSize | cTextSize[i];

}

//Отводим память под текст

char \* Buffer2 = new char[iTextSize];

//Считываем текст

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

{

unsigned char BufferTemp = 0;

for (int j = 0; j < 8; j++)

{

char cImageBit = 0;

cImageByte = Buff[uiOffset];

cImageBit = cImageByte & cMask;

cImageBit = cImageBit << 7;

BufferTemp = BufferTemp >> 1;

BufferTemp = BufferTemp | cImageBit;

uiOffset++;

}

Buffer2[i] = BufferTemp;

}

HANDLE hText;

//Запись в файл

if (INVALID\_HANDLE\_VALUE == (hText = CreateFile(textName, GENERIC\_WRITE, FILE\_SHARE\_WRITE, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL)))

cout << "Can't create file\n";

WriteFile(hText, Buffer2, iTextSize, &dwBytes1, NULL);

CloseHandle(hText);

delete[] Buffer2;

}

else

{

cout << "Nothing to retrieve.\n";

exit(2);

}

CloseHandle(hFile);

delete[] Buff;

}

void Embed(const wchar\_t \* imageName, const wchar\_t \*textName)

{

HANDLE hFile;

//HANDLE hFile1;

if (INVALID\_HANDLE\_VALUE == (hFile = CreateFile(imageName, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL)))

cout << "Can't open the image\n";

DWORD dwBytes = 0, BuffSize = 0, dwBytes\_1 = 0;

BuffSize = GetFileSize(hFile, NULL);

char \*Buff = new char[BuffSize / sizeof(char)];

ReadFile(hFile, Buff, BuffSize, &dwBytes, NULL);

//Открытие текста

HANDLE hText;

if (INVALID\_HANDLE\_VALUE == (hText = CreateFile(textName, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL)))

cout << "Can't open the text file\n";

//Считывание текста в буфер

DWORD dwBytes\_2 = 0, BuffSize\_2 = 0;

BuffSize\_2 = GetFileSize(hText, NULL);

char \*Buff\_2 = new char[BuffSize\_2 / sizeof(char)];

ReadFile(hText, Buff\_2, BuffSize\_2, &dwBytes\_2, NULL);

//Запись текста в изображение

//Получение значения смещения

unsigned int uiOffset = 0;

Offset(&uiOffset, Buff);

//Получение значений цветов и формирование новых

char cImageByte;

//Для обработки текста

char cTextByte = 0, cTextBit = 0;

unsigned int uiTextOffset = 0;

//Проверка размера

if (BuffSize < (BuffSize\_2 \* 8) + 88)

{

cout << "Text do not fit into the image.\n";

exit(3);

}

//Запись метки в изображение

for (int i = 0; i < 5; i++)

{

char temp = cTag[i], tempbit = 0;

for (char j = 0; j < 8; j++)

{

cImageByte = Buff[uiOffset];

//и запись в него бита текста

tempbit = temp & cMask;

temp = temp >> 1;

if (cImageByte % 2 == 0)

cImageByte = cImageByte | tempbit;

else

{

cImageByte = cImageByte & cMask\_2;

cImageByte = cImageByte | tempbit;

}

Buff[uiOffset] = cImageByte;

uiOffset++;

}

}

//запись размера текстового файла в изображение

//32 бита => 32 байта

for (char j = 0; j < 32; j++)

{

//int temp = BufSize2;

char tempbit = 0;

cImageByte = Buff[uiOffset];

//и запись в него бита текста

tempbit = BuffSize\_2 & cMask;

BuffSize\_2 = BuffSize\_2 >> 1;//изменение BufSize2

if (cImageByte % 2 == 0)

cImageByte = cImageByte | tempbit;

else

{

cImageByte = cImageByte & cMask\_2;

cImageByte = cImageByte | tempbit;

}

Buff[uiOffset] = cImageByte;

uiOffset++;

}

//восстановление BufSize2

BuffSize\_2 = GetFileSize(hText, NULL);

//Цикл записи текста в изображене

for (unsigned int i = 0; i < BuffSize\_2; i++)

{

cTextByte = Buff\_2[uiTextOffset];//один байт текста на восемь байт изображения

for (char j = 0; j < 8; j++)

{

//Получение байта изображения

cImageByte = Buff[uiOffset];

//и запись в него бита текста

cTextBit = cTextByte & cMask;

cTextByte = cTextByte >> 1;

if (cImageByte % 2 == 0)

cImageByte = cImageByte | cTextBit;

else

{

cImageByte = cImageByte & cMask\_2;//обнуляем младший бит для правильной записи

cImageByte = cImageByte | cTextBit;

}

Buff[uiOffset] = cImageByte;

uiOffset++;

}

uiTextOffset++;

}

//Конец цикла

CloseHandle(hFile);

if (INVALID\_HANDLE\_VALUE == (hFile = CreateFile(imageName, GENERIC\_WRITE, FILE\_SHARE\_WRITE, NULL, TRUNCATE\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL)))

cout << "Writing in the file failed.\n";

//Запись в файл

WriteFile(hFile, Buff, BuffSize, &dwBytes\_1, NULL);

//Закрытие HANDLE

CloseHandle(hText);

CloseHandle(hFile);

delete Buff;

delete Buff\_2;

}

Файл Main.cpp

#include "Header.h"

#include <iostream>

#include <conio.h>

using namespace std;

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

{

if (argc == 3)

{

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

Check(argv[2]);

else if (!wcscmp(argv[2], TEXT("-check")))

Check(argv[1]);

else

{

cout << "-check <\*.bmp>\n";

}

}

else if (argc == 4)

{

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

Embed(argv[2], argv[3]);

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

Retrieve(argv[2], argv[3]);

else

{

cout << "-embed <\*.bmp> <\*.txt>\n";

cout << "-retrieve <\*.bmp> <\*.txt>\n";

}

}

else

{

cout << "-check <\*.bmp>\n";

cout << "-embed <\*.bmp> <\*.txt>\n";

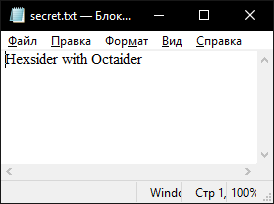
cout << "-retrieve <\*.bmp> <\*.txt>\n";

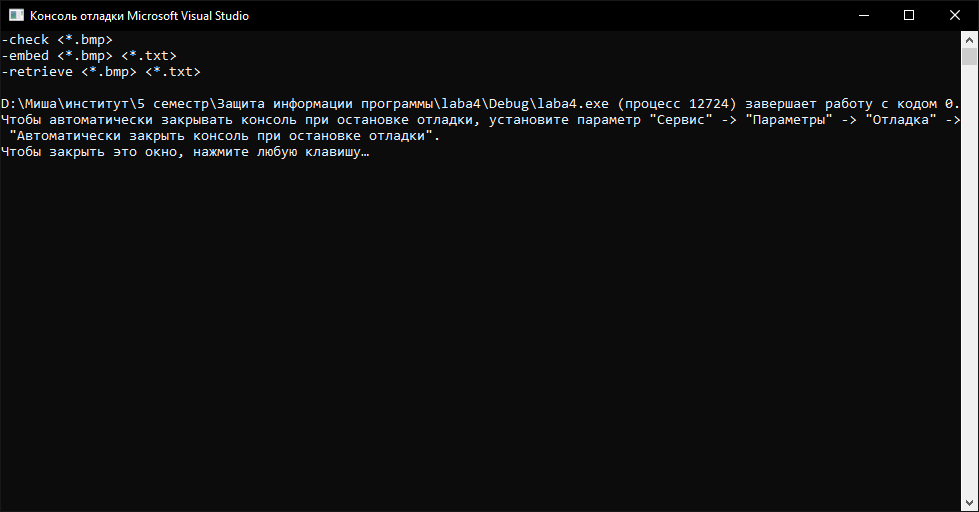
}

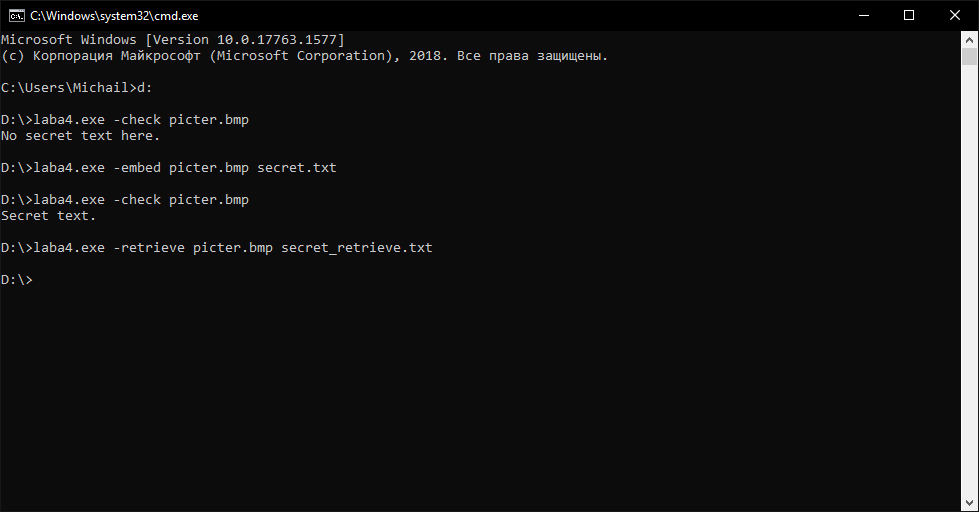
return 0;

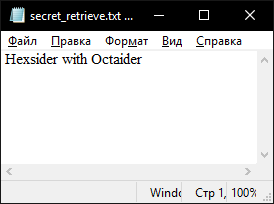
}

**Вывод программы**:









**Вывод:** в ходе лабораторной работы изучили методы стеганографии.