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:

Арсентьева У.С.  
Горбунова В.В.

Приняли:

к.т.н., доцент Дубравин А.В.

к.т.н., доцент Карамышева Н.С.

2020

**Цель работы:** разработать программу, выполняющую внедрение и извлечение некоторых данных из файла с изображением.

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

|  |  |
| --- | --- |
| **Командная строка** | **Описание действия** |
| steg.exe -embed secret.txt picture.bmp | Помещает «секретное» содержимое файла secret.txt в файл picture.bmp |
| steg.exe -retrieve picture.bmp secret.txt | Извлекает «секретное» содержимое из файла picture.bmp и помещает его в файл secret.txt |

**Листинг программы:**

#include <stdio.h>

#include <tchar.h>

#include <windows.h>

#include <iostream>

#include <sstream>

#include <fstream>

#include <vector>

using namespace std;

//Возвращает N-ый бит 1-байтового числа A, начиная с начала

bool GetBit (unsigned char A, int N)

{

return (A >> 8 - N - 1) & 1u;

}

//Вставляет в значение A указанный бит Bit в позицию N (с начала)

void ChangeBit (unsigned char & A, bool Bit, int N)

{

if (Bit == 0) A &= ~(1 << 8 - N - 1);

if (Bit == 1) A |= (1 << 8 - N - 1);

}

//Определяет количество символов в тексте

int LengthFile (string FileName)

{

ifstream file (FileName, ios::binary | ios::ate);

int Length = file.tellg();

file.close();

return Length;

}

//Возвращает поток ввода на файл FileName; Завершает работу приложения в случае, если файл с указанным именем не найден

ifstream OpenFile (string FileName)

{

ifstream File (FileName, ios::binary);

if (!File.is\_open())

{

cout << "Файл не был открыт." << endl;

system ("pause");

exit(0);

}

return File;

}

//Возвращает текст из файла FileName и его размер Size

vector <unsigned char> GetTextInFile (string FileName, int & Size)

{

ifstream File = OpenFile (FileName);

vector <unsigned char> Text (Size, 0);

//Символы, считанные из файла, загружаются в массив

char C;

for (int i = 0; File.get(C); i ++)

Text[i] = unsigned char (C);

File.close();

return Text;

}

//Помещает текст в файл

void OutputTextInFile (vector <unsigned char> Text, string FileName)

{

ofstream File (FileName);

if (!File.is\_open())

{

cout << "Файл не был открыт." << endl;

system ("pause");

exit(0);

}

//Символы, считанные из файла, загружаются в массив

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

File << Text[i];

File.close();

}

//Помещает секретное сообщение из файла SecretText (txt) в файл PictureFileName (bmp)

void PutCryptMessage (string SecretText, LPTSTR PictureFileName)

{

int Size = LengthFile (SecretText);

vector <unsigned char> Crypt = GetTextInFile(SecretText, Size);

//Объявим нужные структуры

BITMAPFILEHEADER bfh;

BITMAPINFOHEADER bih;

unsigned char Palette [1024];

unsigned long RW;

//Откроем файл для чтения

HANDLE InputFile = CreateFile (PictureFileName, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, 0, NULL);

if (InputFile == INVALID\_HANDLE\_VALUE)

return;

//Загрузим заголовки и палитру

ReadFile (InputFile, &bfh, sizeof(bfh), &RW, NULL);

ReadFile (InputFile, &bih, sizeof(bih), &RW, NULL);

ReadFile (InputFile, Palette, sizeof(Palette), &RW, NULL);

int Height = bih.biHeight;

int Width = bih.biWidth;

//Объявим растр как одномерный массив точек

vector <RGBTRIPLE> Points (Height \* Width);

//Загрузим растр в массив

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

{

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

{

ReadFile (InputFile, &Points[Height \* i + j], sizeof(Points[Height \* i + j]), &RW, NULL);

}

//Обеспечим выравнивание строки

ReadFile (InputFile, Palette, (sizeof(RGBTRIPLE) \* Width) % 4, &RW, NULL);

}

//Закроем дескриптор

CloseHandle (InputFile);

//Заменим последние биты оттенка голубого в растре

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

{

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

{

ChangeBit (Points[4 \* i + j].rgbtBlue, GetBit(Crypt[i], 2 \* j), 6);

ChangeBit (Points[4 \* i + j].rgbtBlue, GetBit(Crypt[i], 2 \* j + 1), 7);

}

}

//Откроем данный файл для записи

HANDLE OutputFile = CreateFile (PictureFileName, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, 0, NULL);

if (OutputFile == INVALID\_HANDLE\_VALUE)

return;

//Запишем заголовки и палитру в файл-вывода

WriteFile (OutputFile, &bfh, sizeof(bfh), &RW, NULL);

WriteFile (OutputFile, &bih, sizeof(bih), &RW, NULL);

WriteFile (OutputFile, Palette, sizeof(Palette), &RW, NULL);

//Загрузим растр в файл вывода

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

{

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

{

WriteFile (OutputFile, &Points[Height \* i + j], sizeof(Points[Height \* i + j]), &RW, NULL);

}

//Обеспечим выравнивание строки

WriteFile (OutputFile, Palette, (sizeof(RGBTRIPLE) \* Width) % 4, &RW, NULL);

}

//Закроем дескриптор

CloseHandle (OutputFile);

cout << "Секретное содержимое было успешно помещено.\n";

}

//"Достает" секретное сообщение и возвращает его

void TakeCryptMessage (LPTSTR InputFileName, string Output)

{

//Объявим нужные структуры

BITMAPFILEHEADER bfh;

BITMAPINFOHEADER bih;

unsigned char Palette [1024];

unsigned long RW;

//Откроем файл для чтения

HANDLE InputFile = CreateFile (InputFileName, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, 0, NULL);

if (InputFile == INVALID\_HANDLE\_VALUE)

return;

//Загрузим заголовки и палитру

ReadFile (InputFile, &bfh, sizeof(bfh), &RW, NULL);

ReadFile (InputFile, &bih, sizeof(bih), &RW, NULL);

ReadFile (InputFile, Palette, sizeof(Palette), &RW, NULL);

int Height = bih.biHeight;

int Width = bih.biWidth;

//Объявим растр как одномерный массив точек

vector <RGBTRIPLE> Points (Height \* Width);

//Загрузим растр в массив

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

{

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

{

ReadFile (InputFile, &Points[Height \* i + j], sizeof(Points[Height \* i + j]), &RW, NULL);

}

// Обеспечим выравнивание строки

ReadFile (InputFile, Palette, (sizeof(RGBTRIPLE) \* Width) % 4, &RW, NULL);

}

//Закроем дескриптор

CloseHandle (InputFile);

vector <unsigned char> Text (Height \* Width / 4);

//Заменим последние биты оттенка голубого в растре

for (int i = 0; i < Height \* Width / 4; i ++)

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

{

ChangeBit (Text[i], GetBit(Points[4 \* i + j].rgbtBlue, 6), 2 \* j);

ChangeBit (Text[i], GetBit(Points[4 \* i + j].rgbtBlue, 7), 2 \* j + 1);

}

OutputTextInFile (Text, Output);

cout << "Секретное содержимое было успешно извлечено.\n";

}

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

{

setlocale (LC\_ALL, "Russian");

/\*

\* argv[0] - Расположение программы

\* argv[1] - Тип операции (-embed (внедрение), -retrieve (извлечение), -check (проверка))

\* argv[2] - Первый файл

\* argv[3] - Второй файл

\*/

//argv[2] - "Secret.txt", argv[3] - "Picture.bmp".

if (string(argv[1]) == "-embed")

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

//argv[2] - "Picture.bmp", argv[3] - "Secret.txt".

if (string(argv[1]) == "-retrieve")

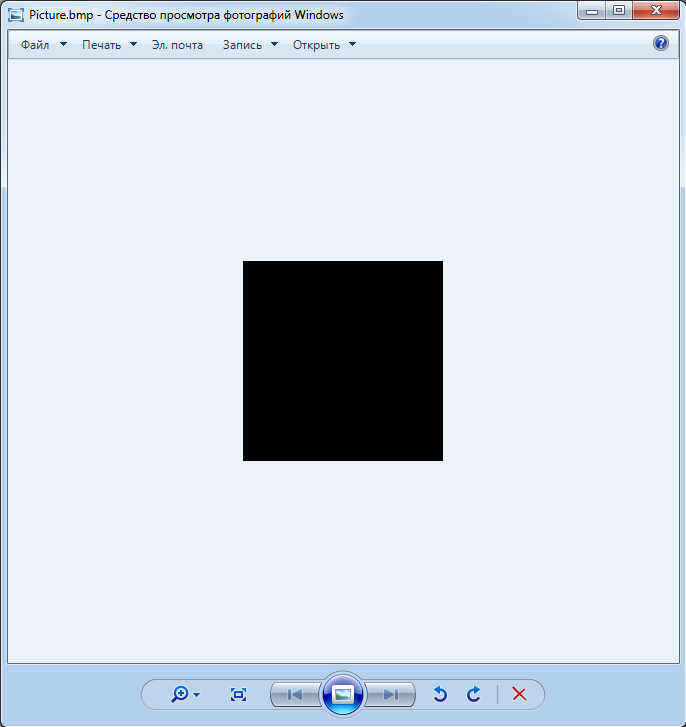
TakeCryptMessage (argv[2], argv[3]);

system ("pause");

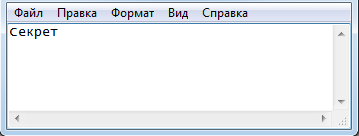
return 0;

}

**Результаты работы программы:**

****

**Рисунок 1 – Исходное изображение**

****

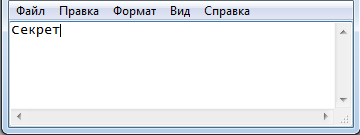
**Рисунок 2 – Исходный текстовый файл**

****

**Рисунок 3 –Помещение секретного сообщения**

****

**Рисунок 4 – Извлечение секретного сообщения**

****

**Рисунок 1 – Результат извлечения секретного сообщения**

**Вывод:** Была разработана программа, выполняющая внедрение и извлечение некоторых данных из файла с изображением.