Министерство науки и высшего образования

Пензенский Государственный Университет

ОТЧЕТ

по лабораторной работе №4

по курсу «Защита информации»

на тему «**Стеганография**»

Вариант 1

Выполнили студенты группы 18ВВ1:

Колобов К.А.

Амплеев Д.О.

Приняли:

Дубравин А.В.

2020

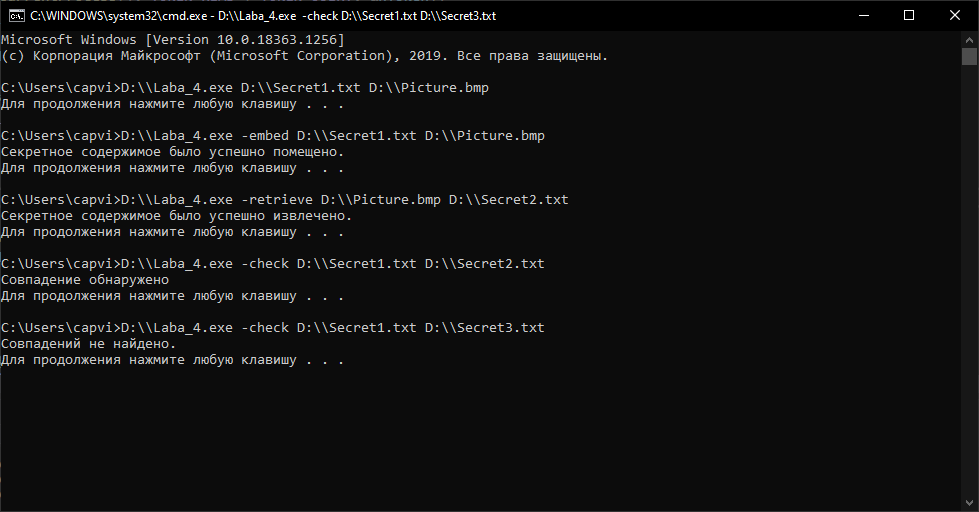
**Лабораторное задание:**

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..

**Описание работы программы:**

1. Программа переносит данные из .txt в .bmp.
2. Программа переносит данные из .bmp в .txt.
3. Программа сравнивает изначальный .txt и получившийся в результате переноса данных из .bmp.

**Результат работы программы:**



**Листинг:**

#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;

}

vector<unsigned char> readText(const string& filename) {

ifstream fin(filename, ios::binary);

vector<unsigned char> text((istreambuf\_iterator<char>(fin)),

istreambuf\_iterator<char>());

fin.close();

return text;

}

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

{

ofstream File(FileName);

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 = readText(SecretText);

//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);

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

}

writeText(Text, Output);

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

}

void CheckCryptMessage(string Input, string Output)

{

int input\_lenght = LengthFile(Input);

int output\_lenght = LengthFile(Output);

vector <unsigned char> input\_vector = readText(Input);

vector <unsigned char> output\_vector = readText(Output);

BOOL flag = TRUE;

if (input\_lenght > output\_lenght) flag = FALSE;

else

for (int i = 0; i < input\_lenght; i++)

{

if (!(input\_vector[i] == output\_vector[i]))

flag = FALSE;

}

if (flag)

std::cout << "Совпадение обнаружено" << std::endl;

else if(!flag)

std::cout << "Совпадений не найдено." << std::endl;

}

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]);

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

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

system("pause");

return 0;

}