Министерство образования Российской Федерации  
Пензенский государственный университет  
Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4  
по курсу «Методы защиты КС»

Выполнили студенты группы 17ВВ2:

Левина Е.В.

Кудряшов Я.В.

Приняли:

Карамышева Н.С.

Дубравин А.В.

Пенза, 2020

**Цель** **работы:** Стеганография.

**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

**Код программы:**

#define \_SILENCE\_EXPERIMENTAL\_FILESYSTEM\_DEPRECATION\_WARNING

#include <iostream>

#include <fstream>

#include <experimental/filesystem>

#include <cstdlib>

#include <string>

#include <cstring>

#include <ctime>

using namespace std;

struct SupportingInfo

{

bool startStopRecording = false;

const char SEText = -2;

bool isEndText = true;

int arrKey[8] = { 2, 4, 0, 7, 1, 5, 6, 3 };

};

struct BitMapHeader

{

int size; // Размер файла

int offsetBits; // Смещение изображения от начала файла

short bitCount; // Глубина цвета

int sizeImage; // Размер изображения (байт)

};

short getTypeValueShort(const char arr[]); // Получить значение типа short

int getTypeValueInteger(const char arr[]); // Получить значение типа int

char getCharacter(const char arr[], const int SIZE); // Получить символ из массива BMP файла

bool isBitMapFile(string directory, BitMapHeader& bmh); // Это BMP файл

int getfileSizeTXT(string directory); // Получить размер файла

void WriteToFile(const char inputData[], const int SIZE, string directory); // Записать в файл строку

void WriteToFile(char symbol, string directory); // Записать в файл символа

void DeleteFileContents(string directory); // Очистить содержимое файла

void steganographyLSB(char inputData[], const int SIZE, char symbol); // Алгоритм шифрования

char characterEncryption(char symbol, const int key[]); // Зашифровать символ

char characterDecryption(char symbol, const int key[]); // Расшифровать символ

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

{

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

{

if (!strcmp(argv[i], "/?"))

{

cout << "\n[-embed] [-retrieve] [disk1:][path1]file\_name1 [disk2:][path2]file\_name2\n[-check] [disk1:][path1]file\_name1\n"

<< "\n -embed Puts \"secret\" content file\_name1.txt to file\_name2.bmp"

<< "\n -retrieve Extracts the \"secret\" content from file\_name1.bmp and puts it in file\_name2.txt"

<< "\n -check Checks for \"secret\" content in file\_name1.bmp\n";

}

else if (!strcmp(argv[i], "-embed"))

{

if ((argv[i + 1] != NULL && argv[i + 2] != NULL) && (strstr(argv[i + 1], ".txt") && strstr(argv[i + 2], ".bmp")))

{

BitMapHeader bmh;

if (isBitMapFile((string)argv[i + 2], bmh) && (bmh.bitCount == 24) && ((bmh.sizeImage / 4) >= (getfileSizeTXT((string)argv[i + 1]) + 2)))

{

ifstream finTXT;

finTXT.open(argv[i + 1], ios\_base::binary);

if (!finTXT.is\_open())

{

EXIT\_FAILURE;

}

ifstream finBMP;

finBMP.open(argv[i + 2], ios\_base::binary);

if (!finBMP.is\_open())

{

EXIT\_FAILURE;

}

//////////////////////////////////////////////////////////////////////////////////////////

char\* arrBMH = new char[bmh.offsetBits];

finBMP.read(arrBMH, bmh.offsetBits);

DeleteFileContents("C:\\Users\\Gesser\\Desktop\\steg test.bmp");

WriteToFile(arrBMH, bmh.offsetBits, "C:\\Users\\Gesser\\Desktop\\steg test.bmp");

delete[] arrBMH;

SupportingInfo si;

//si.startStep = bmh.offsetBits;

const int N = 4;

char arrBuffer[4];

char symbolBuffer;

finBMP.read(arrBuffer, N);

steganographyLSB(arrBuffer, N, characterEncryption(si.SEText, si.arrKey)); // Записать квадрад (начало текста)

WriteToFile(arrBuffer, N, "C:\\Users\\Gesser\\Desktop\\steg test.bmp");

do

{

finBMP.read(arrBuffer, N);

symbolBuffer = finTXT.get();

if (!finTXT.eof())

{

steganographyLSB(arrBuffer, N, characterEncryption(symbolBuffer, si.arrKey));

//si.finishStep = (int)finBMP.tellg();

}

else if (si.isEndText)

{

steganographyLSB(arrBuffer, N, characterEncryption(si.SEText, si.arrKey)); // Записать квадрад (конец текста)

si.isEndText = false;

}

WriteToFile(arrBuffer, N, "C:\\Users\\Gesser\\Desktop\\steg test.bmp");

} while (!finBMP.eof());

//////////////////////////////////////////////////////////////////////////////////////////

finBMP.close();

finTXT.close();

}

else

{

cout << "\nErrors occurred while performing the current operation...\n";

}

}

else

{

cout << "\n-embed [disk1:][path1]file\_name1 [disk2:][path2]file\_name2\n";

}

}

else if (!strcmp(argv[i], "-retrieve"))

{

if ((argv[i + 1] != NULL && argv[i + 2] != NULL) && (strstr(argv[i + 1], ".bmp") && strstr(argv[i + 2], ".txt")))

{

BitMapHeader bmh;

if (isBitMapFile((string)argv[i + 1], bmh) && (bmh.bitCount == 24))

{

ifstream finBMP;

finBMP.open(argv[i + 1], ios\_base::binary);

if (!finBMP.is\_open())

{

EXIT\_FAILURE;

}

//////////////////////////////////////////////////////////////////////////////////////////

char\* arrBMH = new char[bmh.offsetBits];

finBMP.read(arrBMH, bmh.offsetBits);

delete[] arrBMH;

DeleteFileContents(argv[i + 2]);

SupportingInfo si;

const int N = 4;

char arrBuffer[4];

char symbolBuffer;

do

{

finBMP.read(arrBuffer, N);

symbolBuffer = characterDecryption(getCharacter(arrBuffer, N), si.arrKey);

if (symbolBuffer == -2)

{

si.startStopRecording = !si.startStopRecording;

if (!si.startStopRecording)

break;

}

else if (si.startStopRecording)

{

WriteToFile(symbolBuffer, argv[i + 2]);

}

} while (!finBMP.eof());

//////////////////////////////////////////////////////////////////////////////////////////

finBMP.close();

}

else

{

cout << "\n-embed [disk1:][path1]file\_name1 [disk2:][path2]file\_name2\n";

}

}

else

{

cout << "\n-retrieve [disk1:][path1]file\_name1 [disk2:][path2]file\_name2\n";

}

}

else if (!strcmp(argv[i], "-check"))

{

if (argv[i + 1] != NULL && strstr(argv[i + 1], ".bmp"))

{

BitMapHeader bmh;

if (isBitMapFile((string)argv[i + 1], bmh) && (bmh.bitCount == 24))

{

namespace fs = std::experimental::filesystem;

int fileSizeBMP;

try {

fileSizeBMP = (int)fs::file\_size(argv[i + 1]);

if (bmh.size != fileSizeBMP)

cout << "\nThe file contains the presence of \"hidden\" the contents\n";

else

cout << "\nThe file is missing the presence of \"hidden\" the contents\n";

}

catch (fs::filesystem\_error& ex) {

cout << ex.what() << '\n';

}

}

}

else

{

cout << "\n-check [disk1:][path1]file\_name1\n";

}

}

}

return 0;

}

short getTypeValueShort(const char arr[])

{

int value = 0;

for (int i = 1; i > 0; i--)

{

if (arr[i] != NULL)

{

value += (unsigned char)arr[i];

value <<= 8;

}

}

return value += (unsigned char)arr[0];

}

int getTypeValueInteger(const char arr[])

{

int value = 0;

for (int i = 3; i > 0; i--)

{

if (arr[i] != NULL)

{

value |= (unsigned char)arr[i];

value <<= 8;

}

}

return value |= (unsigned char)arr[0];

}

char getCharacter(const char arr[], const int SIZE)

{

char symbol = NULL;

for (int i = 0; i < (SIZE - 1); i++)

{

symbol = (symbol + (bool)((1 << 1) & arr[i])) << 1;

symbol = (symbol + (bool)((1 << 0) & arr[i])) << 1;

}

symbol = (symbol + (bool)((1 << 1) & arr[SIZE - 1])) << 1;

symbol = (symbol + (bool)((1 << 0) & arr[SIZE - 1]));

return symbol;

}

bool isBitMapFile(string directory, BitMapHeader& bmh)

{

ifstream fin;

fin.open(directory, ios\_base::binary);

if (!fin.is\_open())

{

EXIT\_FAILURE;

}

fin.seekg(0, fin.end);

int fileLengthBMP = (int)fin.tellg();

fin.seekg(0, fin.beg);

char arrType[2];

char arrSize[4];

fin.read(arrType, 2);

fin.read(arrSize, 4);

bmh.size = getTypeValueInteger(arrSize);

while (fin.tellg() < 38)

{

fin.get();

switch (fin.tellg())

{

case 10:

{

char arrOffsetBits[4];

fin.read(arrOffsetBits, 4);

bmh.offsetBits = getTypeValueInteger(arrOffsetBits);

}

break;

case 28:

{

char arrBitCount[2];

fin.read(arrBitCount, 2);

bmh.bitCount = getTypeValueShort(arrBitCount);

}

break;

case 34:

{

char arrSizeImage[4];

fin.read(arrSizeImage, 4);

bmh.sizeImage = getTypeValueInteger(arrSizeImage);

}

break;

}

}

fin.close();

namespace fs = std::experimental::filesystem;

int fileSizeBMP;

try {

fileSizeBMP = (int)fs::file\_size(directory);

}

catch (fs::filesystem\_error& ex) {

cout << ex.what() << '\n';

}

//return ((arrType[0] == 'B' && arrType[1] == 'M') && (getTypeValueInteger(arrSize) == fileLengthBMP));

return ((arrType[0] == 'B' && arrType[1] == 'M') && (fileSizeBMP == fileLengthBMP));

}

int getfileSizeTXT(string directory)

{

ifstream fin;

fin.open(directory, ios\_base::binary);

if (!fin.is\_open())

{

EXIT\_FAILURE;

}

fin.seekg(0, fin.end);

int fileLengthTXT = (int)fin.tellg();

fin.close();

return fileLengthTXT;

}

void WriteToFile(const char inputData[], const int SIZE, string directory)

{

ofstream fout;

fout.open(directory, ios\_base::binary | ios\_base::app);

if (!fout.is\_open())

{

EXIT\_FAILURE;

}

fout.write(inputData, SIZE);

fout.close();

}

void WriteToFile(char symbol, string directory)

{

ofstream fout;

fout.open(directory, ios\_base::binary | ios\_base::app);

if (!fout.is\_open())

{

EXIT\_FAILURE;

}

fout << symbol;

fout.close();

}

void DeleteFileContents(string directory)

{

ofstream fout;

fout.open(directory, ios\_base::binary | ios\_base::trunc);

if (!fout.is\_open())

{

EXIT\_FAILURE;

}

fout.close();

}

void steganographyLSB(char inputData[], const int SIZE, char symbol)

{

int index = 7;

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

{

inputData[i] >>= 2;

inputData[i] <<= 1;

inputData[i] = (inputData[i] + (bool)((1 << index) & symbol)) << 1;

index--;

inputData[i] += (bool)((1 << index) & symbol);

index--;

}

}

char characterEncryption(char symbol, const int key[])

{

char symbolEncryption = NULL;

for (int i = 0; i < 7; i++)

{

symbolEncryption = (symbolEncryption + (bool)((1 << key[i]) & symbol)) << 1;

}

symbolEncryption = symbolEncryption + (bool)((1 << key[7]) & symbol);

return symbolEncryption;

}

char characterDecryption(char symbol, const int key[])

{

char symbolDecryption = NULL;

for (int i = 7; i >= 0; i--)

{

int index = 0;

for (; key[index] != i; index++);

if (!i)

symbolDecryption = (symbolDecryption + (bool)((1 << (7 - index)) & symbol));

else

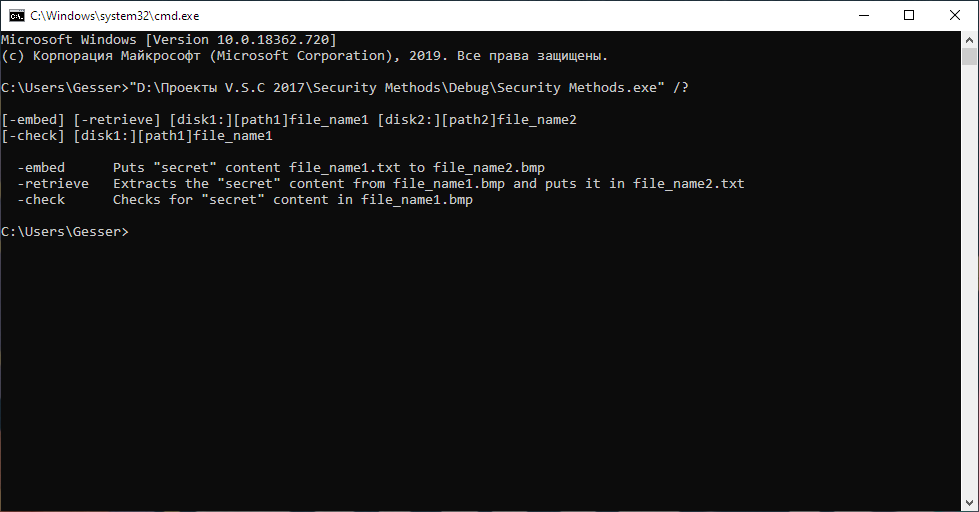
symbolDecryption = (symbolDecryption + (bool)((1 << (7 - index)) & symbol)) << 1;

}

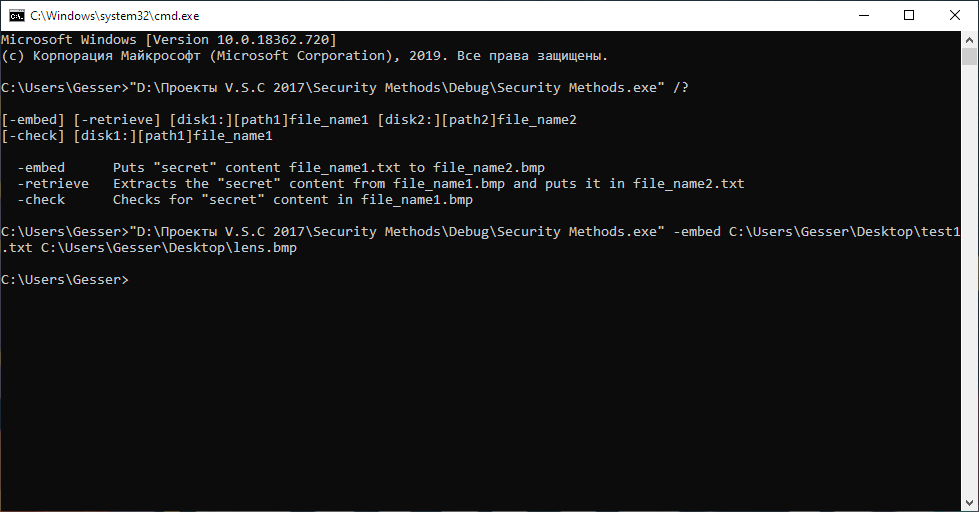
return symbolDecryption;

}

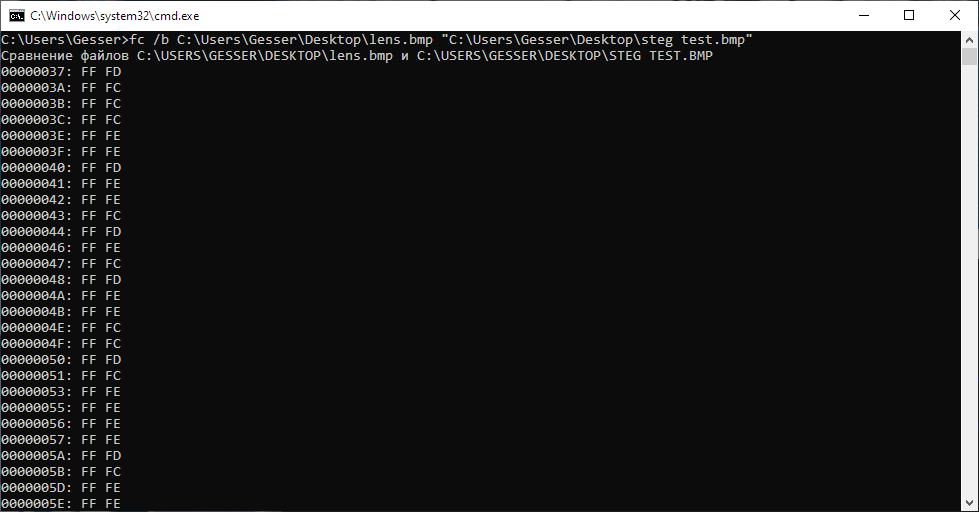
**Результат работы программы:**



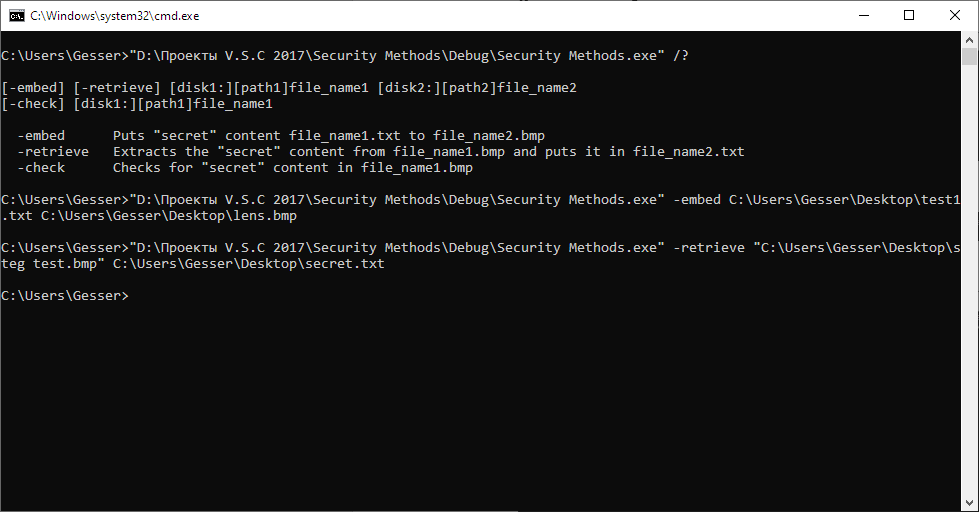
**Figure 1 Меню программы**

****

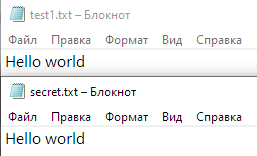
**Figure 2 Команда -embed**



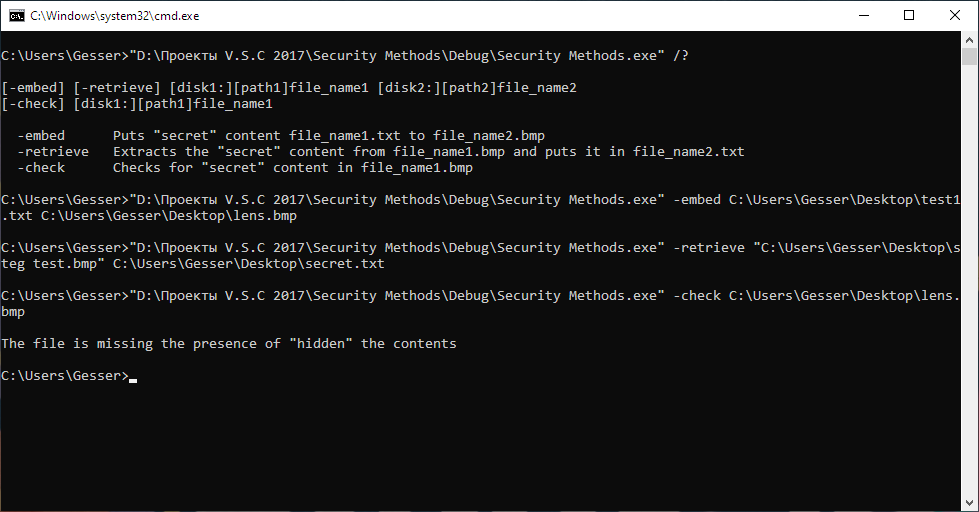
**Figure 3 Результат работы команды -embed**



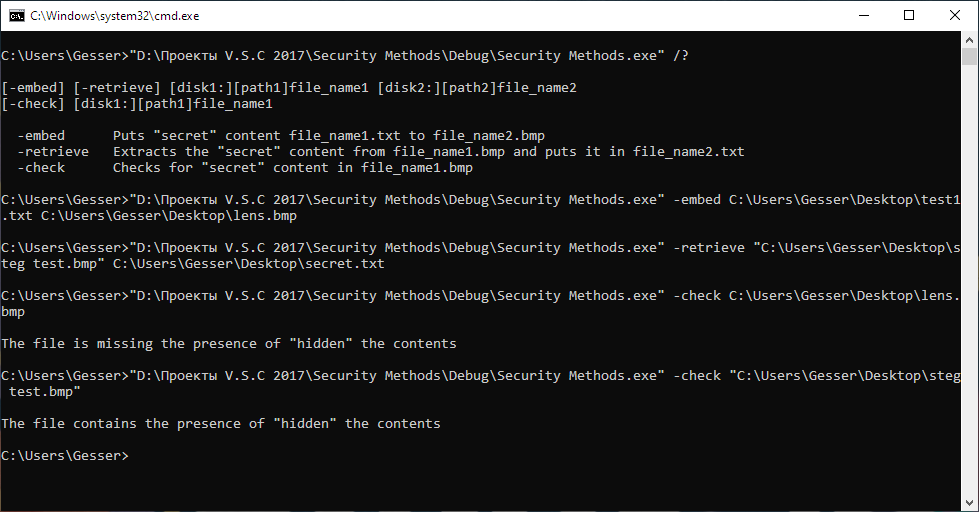
**Figure 4 Команда -retrieve**



**Figure 5 Результат работы команды -retrieve**



**Figure 6 Результат работы команды -check (без Стеганографии)**



**Figure 7 Результат работы команды -check (Стеганографии)**

**Вывод**

При выполнении данной лабораторной работы мы научились стеганографии