

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

ОТЧЕТ

по лабораторной работе №1
по курсу «Защита информации»
на тему «Разработка программы шифрования/дешифрирования текста
с использованием перестановочного шифра»

Выполнили:
студенты группы 15BB2
Артёмов И.В.
Жданов Д.М.

Принял:
Дубравин А.В.

Цель работы

Разработать программу шифрования/дешифрования текста с использованием перестановочного шифра.

Лабораторное задание

Разработать программу выполняющую шифрование и расшифровывание произвольного текстового файла с использованием перестановочного шифра используя в качестве ключа последовательность, соответствующую номеру варианта. Выполнить проверку путём двоичного сравнения исходного файла и файла, полученного после расшифровывания.

Вариант №2. Последовательность: 3 9 8 1 6 10 5 2 7 4

Пояснительный текст к программе

Произвольный текстовый файл — test0.txt — открывается объектом ifstream a. Создаётся файл test1.txt для хранения зашифрованной информации. После завершения чтения test0.txt и записывания в test1.txt файлы закрываются.

Для чтения открывается test1.txt и создаётся файл test2.txt для хранения расшифрованных данных.

Листинг

Файл Zl1.cpp

```
#include "stdafx.h"
#include <fstream>
#include <locale>

using namespace std;

int main()
{
    char s[11];
    int i = 0;
    setlocale(LC_ALL, "Russian");
    ifstream a("test0.txt");
    ofstream b("test1.txt");

    s[10] = '\\0';

    while (!a.eof())
    {
        i = 0;
        for (int t = 0; t < 10; t++)
            s[t] = ' ';

        do
        {
            s[i] = a.get();
            i++;
        } while (a.peek() != '\\n' && i != 10 && !a.eof());
        b << s[2] << s[8] << s[7] << s[0] << s[5] << s[9] << s[4]
<< s[1] << s[6] << s[3]; //2870594163
        if (a.peek() == '\\n')
        {
            b << endl;
            a.get();
        }
    }
    a.close();
    b.close();

    a.open("test1.txt");
    b.open("test2.txt");

    while (!a.eof())
    {
        i = 0;
```

```

    for (int t = 0; t < 10; t++)
        s[t] = ' ';

    do
    {
        s[i] = a.get();
        i++;
    } while (a.peek() != '\n' && i != 10 && !a.eof());
    b << s[3] << s[7] << s[0] << s[9] << s[6] << s[4] << s[8]
<< s[2] << s[1] << s[5]; //обратно - 3709648215
    if (a.peek() == '\n')
    {
        b << endl;
        a.get();
    }
}
a.close();
b.close();
return 0;
}

```

Результаты работы программы

В результате шифрования и расшифровывания получились два файла большего размера, чем оригинальный, но равные друг другу по размеру.

	test0	txt	16 233
	test1	txt	16 468
	test2	txt	16 468

Рисунок 1 — Файл test0.txt оригинальный, test1.txt и test2.txt — созданные программой

Увеличение размера файлов связано с дополнительными символами пробела, которые добавляются в текст при недостаточной информации из текста, если строка не кратна 10 символам.

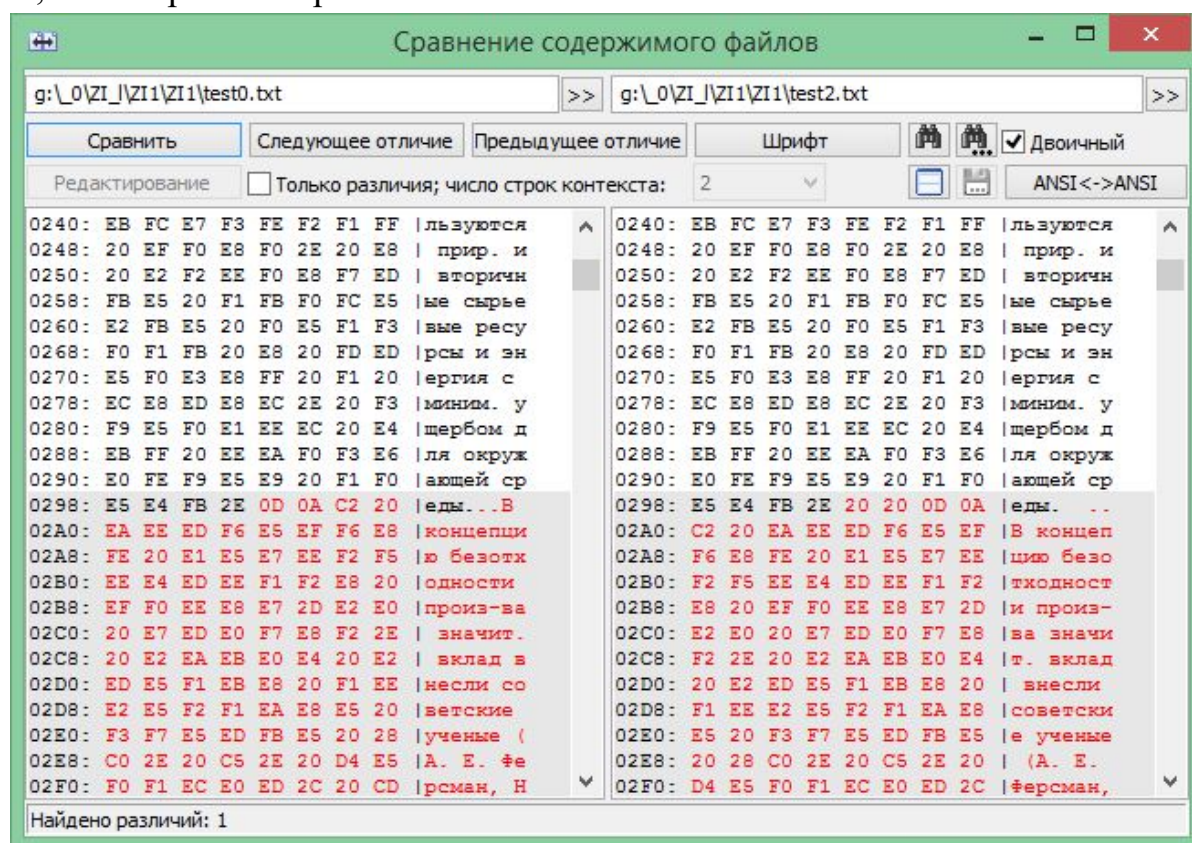


Рисунок 2 — Сравнение файлов

На рисунке 2 показано, что при добавлении новых пробелов в расшифрованный текст вызывает смещение информации.

Выводы

В ходе выполнения данной лабораторной работы была написана программа шифрования/дешифрования текста с использованием перестановочного шифра.

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

ОТЧЕТ

по лабораторной работе №2
по курсу «Защита информации»
на тему «Разработка программы шифрования/дешифрирования
двоичных файлов с использованием перестановочного шифра»

Выполнили:
студенты группы 15ВВ2
Артёмов И.В.
Жданов Д.М.

Принял:
Дубравин А.В.

Цель работы

Разработать программу шифрования/дешифрования двоичных файлов с использованием перестановочного шифра.

Лабораторное задание

Разработать программу выполняющую шифрование и расшифровывание произвольного двоичного файла с использованием перестановочного шифра используя в качестве ключа последовательность, соответствующую номеру варианта. Выполнить проверку путём двоичного сравнения исходного файла и файла, полученного после расшифровывания.

Вариант №2. Размер блока: 4 байта

Последовательность: 2 28 13 15 20 12 5 24 25 27 1 3 14 9 8 21 22 4 26 23 19
30 18 6 11 17 31 16 10 7 29 0

Описание программы

В качестве блока данных размером 4 байта в программе выступает массив типа `char` на 4 элемента. Самый левый бит в элементе 0 принимается за 31, а самый правый бит в элементе 3 принимается за нулевой.

В процессе шифрования отдельный бит из исходного массива записывается во временную переменную с помощью маски и побитовой операции «и» (&). Далее бит смещается побитовыми операциями сдвига (<< или >>) на нужную позицию. После он заносится в результирующий массив побитовой операцией «или».

Листинг

Файл Zl2.cpp

```
#include <iostream>
#include <locale>
#include "FB.h"

using namespace std;

int main(int argc, char* argv[])
{
    setlocale(LC_ALL, "Russian");

    if (argc == 4)
    {
        FB * block = new FB();
        block->shifr(argv[1], argv[2]);
        block->deshifr(argv[2], argv[3]);
        delete block;
    }
    else
    {
        cout << "Для работы с программой необходимо указать\n";
        cout << "три параметра: имя файла, который будет  
зашифрован,\n";
        cout << "имя файла, в котором будет храниться зашифрованная  
информация,\n";
        cout << "и имя файла, где будет расшифрованная  
информация.";
    }

    return 0;
}
```

Файл FB.h

```
#include <fstream>

using namespace std;

class FB
{
public:
    FB();
    ~FB();
    //биты по блокам fb[0] fb[1] fb[2] fb[3] fb-fourbytes
    //   7   6   5   4   3   2   1   0   |fb[3]
    //  15  14  13  12  11  10   9   8   |fb[2]
    //  23  22  21  20  19  18  17  16   |fb[1]
    //  31  30  29  28  27  26  25  24   |fb[0]
    // и их маски
    //  80  40  20  10   8   4   2   1
    void shifr(const char* a, const char* b); //шифрование
    void deshifr(const char* a, const char* b); //дешифрование
private:
    char fourbytes[4]; //для хранения четырёх байтов
    char result[4]; //готовый блок
    void cleanFB(void); //очистка fourbytes
    void cleanRS(void);
    void formShifr(void); //формирование четырёх зашифрованных
байтов
    void form1sByte(char n); //n - номер байта
    void form2sByte(char n);
    void form3sByte(char n);
    void form4sByte(char n);
    void formDeshifr(void);
    void form1dByte(char n);
    void form2dByte(char n);
    void form3dByte(char n);
    void form4dByte(char n);
    void perenos(char maska, char fb, bool naprav, char kolvo,
char rs); //маска; блок, из которого читаем; направление смещения
(1-влево);
    //количество битов, на которое сдвигаем; блок, в который
записываем
};
```

Файл FB.cpp

```
#include "FB.h"
#include <iostream>

FB::FB()
{
    cleanFB();
};

FB::~FB()
{
    result[1] = 0;
};

void FB::shifr(const char* a, const char* b)
{
    ifstream f1(a);
    ofstream f2(b);
    int i = 0;

    if (f1.is_open() && f2.is_open())
    {
        while (!f1.eof())
        {
            i = 0;
            cleanFB();
            while (f1.peek() != EOF && i != 4)
            {
                fourbytes[i] = f1.get();
                i++;
            }
            cleanRS();
            formShifr();
            f2 << result[0] << result[1] << result[2] <<
result[3];
        }
    }
    else
    {
        if (!f1.is_open())
        {
            cout << "Проблемы с первым файлом.";
            exit(0);
        }
        else
        {
            cout << "Проблемы со вторым файлом.";
            exit(0);
        }
    }

    f1.close();
    f2.close();
};
```

```

void FB::deshifr(const char* a, const char* b)
{
    ifstream f1(a);
    ofstream f2(b);
    int i = 0;

    if (f1.is_open() && f2.is_open())
    {
        while (!f1.eof())
        {
            i = 0;
            cleanFB();
            while (f1.peek() != EOF && i != 4)
            {
                fourbytes[i] = f1.get();
                i++;
            }
            cleanRS();
            formDeshifr();
            f2 << result[0] << result[1] << result[2] <<
result[3];
        }
    }
    else
    {
        if (!f1.is_open())
        {
            cout << "Проблемы с первым файлом.";
            exit(0);
        }
        else
        {
            cout << "Проблемы со вторым файлом.";
            exit(0);
        }
    }

    f1.close();
    f2.close();
};

void FB::cleanFB(void)
{
    for (int t = 0; t < 4; t++)
        fourbytes[t] = 0;
}

void FB::cleanRS(void)
{
    for (int t = 0; t < 4; t++)
        result[t] = 0;
};

void FB::formShifr(void)

```

```

{
    form1sByte(0);
    form2sByte(1);
    form3sByte(2);
    form4sByte(3);
};

void FB::form1sByte(char n)
{
    //второй бит на 31ую позицию
    perenos(0x04, 3, 1, 5, n);
    //28ой бит на 30ую
    perenos(0x10, 0, 1, 2, n);
    //13ый на 29ую
    perenos(0x20, 2, 1, 0, n);
    //15ый на 28ую
    perenos(0x80, 2, 0, 3, n);
    //20ый на 27ую
    perenos(0x10, 1, 0, 1, n);
    //12ый на 26ую
    perenos(0x10, 2, 0, 2, n);
    //пятый на 25ую
    perenos(0x20, 3, 0, 4, n);
    //24ый на 24ую
    perenos(0x01, 0, 1, 0, n);
};

void FB::form2sByte(char n)
{
    //25ый на 23ью
    perenos(0x02, 0, 1, 6, n);
    //27ой на 22ую
    perenos(0x08, 0, 1, 3, n);
    //первый на 21ую
    perenos(0x02, 3, 1, 4, n);
    //третий на 20ую
    perenos(0x08, 3, 1, 1, n);
    //14ый на 19ую
    perenos(0x40, 2, 0, 3, n);
    //девятый на 18ую
    perenos(0x02, 2, 1, 1, n);
    //восьмой на 17ую
    perenos(0x01, 2, 1, 1, n);
    //21ый на 16ую
    perenos(0x20, 1, 0, 5, n);
};

void FB::form3sByte(char n)
{
    //22ой на 15ую
    perenos(0x40, 1, 1, 1, n);
    //четвёртый на 14ую
    perenos(0x10, 3, 1, 2, n);
    //26ой на 13ую
    perenos(0x04, 0, 1, 3, n);
    //23 на 12
    perenos(0x80, 1, 0, 3, n);
};

```

```

        //19 на 11
        perenos(0x08, 1, 1, 0, n);
        //30 10
        perenos(0x40, 0, 0, 4, n);
        //18 9
        perenos(0x04, 1, 0, 1, n);
        //6 8
        perenos(0x40, 3, 0, 6, n);
};

void FB::form4sByte(char n)
{
    //11 7
    perenos(0x08, 2, 1, 4, n);
    //17 6
    perenos(0x02, 1, 1, 5, n);
    //31 5
    perenos(0x80, 0, 0, 2, n);
    //16 4
    perenos(0x01, 1, 1, 4, n);
    //10 3
    perenos(0x04, 2, 1, 1, n);
    //7 2
    perenos(0x80, 3, 0, 5, n);
    //29 1
    perenos(0x20, 0, 0, 4, n);
    //0 0
    perenos(0x01, 3, 1, 0, n);
};

void FB::perenos(char maska, char fb, bool naprav, char kolvo,
char rs)
{
    unsigned char temp=0;

    temp = fourbytes[fb] & maska;
    if (naprav)
        temp = temp << kolvo;
    else
        temp = temp >> kolvo;
    result[rs] = result[rs] | temp;
};

void FB::formDeshifr(void)
{
    form1dByte(0);
    form2dByte(1);
    form3dByte(2);
    form4dByte(3);
};

void FB::form1dByte(char n)
{
    //5 31
    perenos(0x20, 3, 1, 2, n);
    //10 30
    perenos(0x04, 2, 1, 4, n);
};

```

```

    //1 29
    perenos(0x02, 3, 1, 4, n);
    //30 28
    perenos(0x40, 0, 0, 2, n);
    //22 27
    perenos(0x40, 1, 0, 3, n);
    //13 26
    perenos(0x20, 2, 0, 3, n);
    //23 25
    perenos(0x80, 1, 0, 6, n);
    //24 24
    perenos(0x01, 0, 1, 0, n);
};
void FB::form2dByte(char n)
{
    //12 23
    perenos(0x10, 2, 1, 3, n);
    //15 22
    perenos(0x80, 2, 0, 1, n);
    //16 21
    perenos(0x01, 1, 1, 5, n);
    //27 20
    perenos(0x08, 0, 1, 1, n);
    //11 19
    perenos(0x08, 2, 1, 0, n);
    //9 18
    perenos(0x02, 2, 1, 1, n);
    //6 17
    perenos(0x40, 3, 0, 5, n);
    //4 16
    perenos(0x10, 3, 0, 4, n);
};
void FB::form3dByte(char n)
{
    //28 15
    perenos(0x10, 0, 1, 3, n);
    //19 14
    perenos(0x08, 1, 1, 3, n);
    //29 13
    perenos(0x20, 0, 1, 0, n);
    //26 12
    perenos(0x04, 0, 1, 2, n);
    //7 11
    perenos(0x80, 3, 0, 4, n);
    //3 10
    perenos(0x08, 3, 0, 1, n);
    //18 9
    perenos(0x04, 1, 0, 1, n);
    //17 8
    perenos(0x02, 1, 0, 1, n);
};
void FB::form4dByte(char n)
{
    //2 7
    perenos(0x04, 3, 1, 5, n);

```

```
//8 6
perenos(0x01, 2, 1, 6, n);
//25 5
perenos(0x02, 0, 1, 4, n);
//14 4
perenos(0x40, 2, 0, 2, n);
//20 3
perenos(0x10, 1, 0, 1, n);
//31 2
perenos(0x80, 0, 0, 5, n);
//21 1
perenos(0x20, 1, 0, 4, n);
//0 0
perenos(0x01, 3, 1, 0, n);
};
```

Результаты работы программы

В результате шифрования и расшифровывания получились два файла: _a1.txt и _a2.txt.

	_a	txt	1 319
	_a1	txt	1 339
	_a2	txt	1 319

Рисунок 1 — Файл _a.txt оригинальный, _a1.txt и _a2.txt — созданные программой

Двоичное сравнение оригинального файла _a.txt и созданного _a2.txt различий в файлах не выявило.

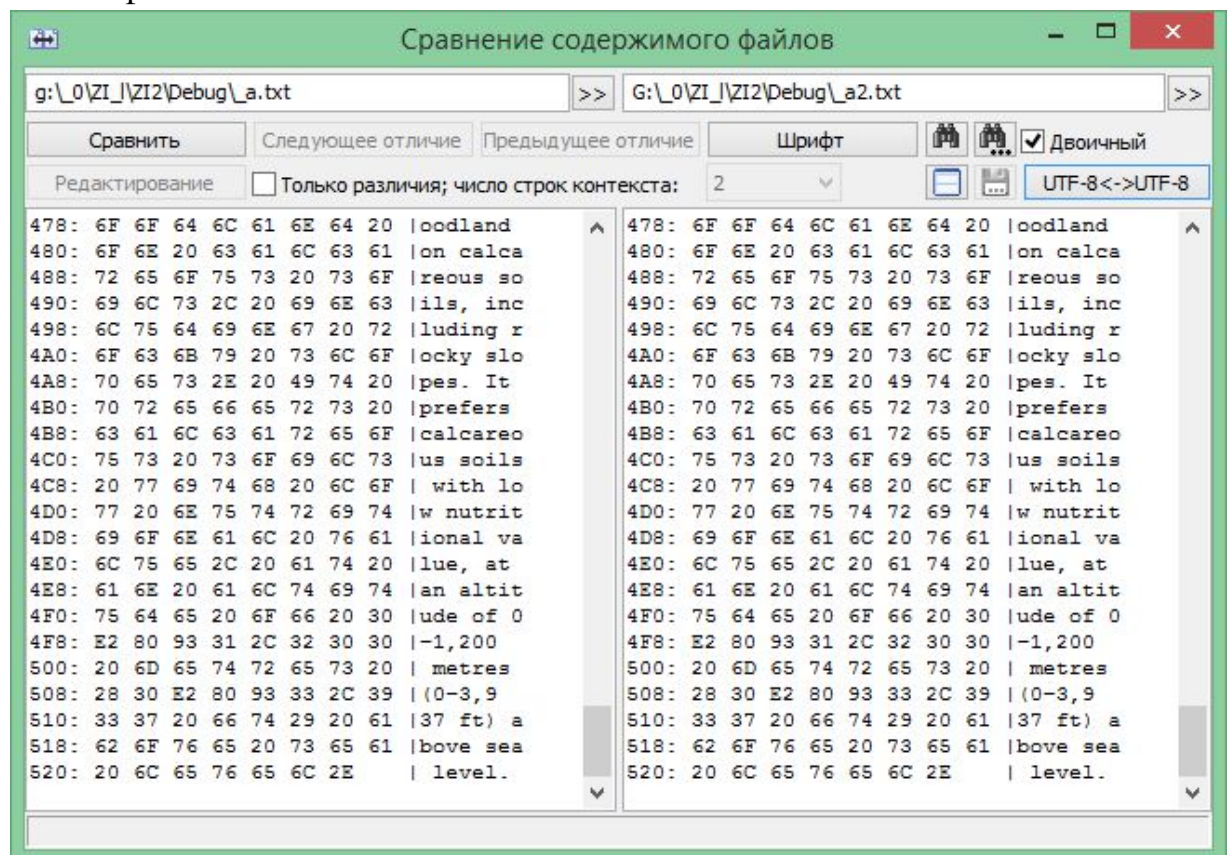


Рисунок 2 — Сравнение файлов

Выводы

В ходе выполнения данной лабораторной работы была написана программа шифрования/дешифрования двоичных файлов с использованием перестановочного шифра.

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

ОТЧЕТ

по лабораторной работе №3
по курсу «Защита информации»
на тему «Генераторы псевдослучайных последовательностей»

Выполнили:
студенты группы 15ВВ2
Артёмов И.В.
Жданов Д.М.

Принял:
Дубравин А.В.

Цель работы

Изучить методы генерации случайных чисел.

Лабораторное задание

Разработать генератор случайных чисел, используя метод в соответствии с номером варианта. Исследовать качество генератора, путём оценки распределения генерируемых чисел.

Вариант №2. Метод серединных произведений. Разрядность при вычислении 16 бит, разрядность результата 8 бит.

Описание программы

Начальное значение переменных, участвующих в произведении, получается вызовом функции `time(NULL)`. В первую переменную записываются младшие биты полученного значения, а во вторую — значение с побитовым сдвигом.

После выполнения произведения, значение сдвигается на четыре бита вправо и записывается в одну из переменных чередуясь. Середина значения произведения является очередным случайным числом.

Получив новое значения результата, определяется какому из интервалов оно принадлежит.

Листинг

Файл Zl3.cpp

```
#include <ctime>
#include <fstream>

using namespace std;

int main()
{
    time_t timeVar;
    timeVar = time(NULL); //Текущее время
    unsigned short int usiTemp; //для хранения результата
произведения
    unsigned char ucResult, ucVar1, ucVar2;
    ucVar1 = timeVar;

    timeVar = timeVar >> 7;
    ucVar2 = timeVar;

    bool whichVar = 0;
    ofstream ol("test.txt");

    int graf[10];
    for (int j = 0; j < 10; j++)
        graf[j] = 0;

    for (int i = 0; i < 100; i++)
    {
        usiTemp = ucVar1 * ucVar2; //произведение
        usiTemp = usiTemp >> 4;

        whichVar = !whichVar; //записывать поочерёдно в переменные
результат
        if (whichVar)
            ucVar1 = usiTemp;
        else
            ucVar2 = usiTemp;

        ucResult = usiTemp; //разрядность результата 8 бит

        ol << int(ucResult) << '\n';
        if (ucResult < 25)
            graf[0]++;
        else if (ucResult < 50)
            graf[1]++;
        else if (ucResult < 75)
```

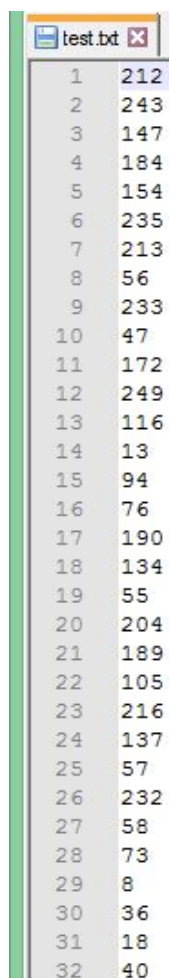
```

        graf[2]++;
    else if (ucResult < 100)
        graf[3]++;
    else if (ucResult < 125)
        graf[4]++;
    else if (ucResult < 150)
        graf[5]++;
    else if (ucResult < 175)
        graf[6]++;
    else if (ucResult < 200)
        graf[7]++;
    else if (ucResult < 225)
        graf[8]++;
    else
        graf[9]++;
}
o1.close();
ofstream o2("test1.txt");
for (int i = 0; i < 10; i++)
{
    o2 << graf[i] << '\n';
}
o2.close();
return 0;
}

```

Результаты работы программы

В результате генерации случайных чисел получаются два файла: test.txt и test1.txt. В первом содержится псевдослучайная последовательность цифр, а во втором — количество цифр, принадлежащих одному из десяти интервалов, что показывает качество генерации.



1	212
2	243
3	147
4	184
5	154
6	235
7	213
8	56
9	233
10	47
11	172
12	249
13	116
14	13
15	94
16	76
17	190
18	134
19	55
20	204
21	189
22	105
23	216
24	137
25	57
26	232
27	58
28	73
29	8
30	36
31	18
32	40

Рисунок 1 — Фрагмент псевдослучайной последовательности из файла test.txt, сформированного программой

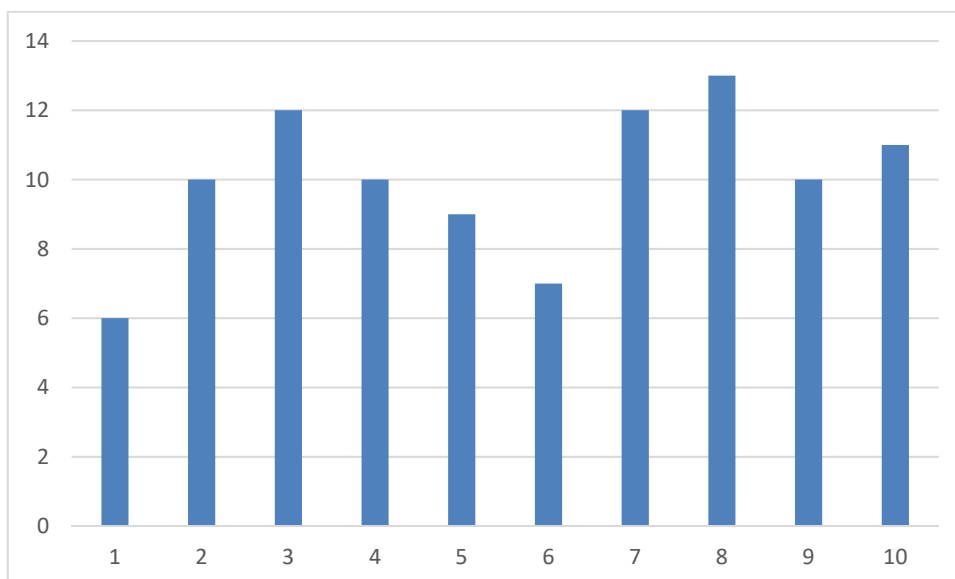


Рисунок 2 — Распределение псевдослучайных чисел по интервалам

Выводы

В ходе выполнения данной были изучены методы генерации случайных чисел.

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

ОТЧЕТ

по лабораторной работе №4
по курсу «Защита информации»
на тему «Стеганография»

Выполнили:
студенты группы 15ВВ2
Артёмов И.В.
Жданов Д.М.

Принял:
Дубравин А.В.

Цель работы

Изучить методы стеганографии.

Лабораторное задание

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 удаление 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 ZI4.h

```
#include <Windows.h>
#include <iostream>

using namespace std;

//Прототипы функций
void GetOffset(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 cMask2 = 0xFE; //Для обнуления младшего бита байта
изображения
const char cTag[5] = "ZI04";
```

Файл ZI4_Functions.cpp

```
#include "ZI4.h"

void GetOffset(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)
{
    HANDLE hFile;
    char * Buffer;
    DWORD dwBytes;
    ///Проверка на наличие секретного сообщения
    //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
    Buffer = new char[100];
    ReadFile(hFile, Buffer, 100, &dwBytes, NULL); //Берём 100
    первых байт

    unsigned int uiOffset;
    GetOffset(&uiOffset, Buffer);

    //Формирование метки из изображения
    //от смещения - 40 байт
    unsigned char cTag2[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 = Buffer[uiOffset];
            cImageBit = cImageByte & cMask;
```

```

        cImageBit = cImageBit << 7;
        cTag2[i] = cTag2[i] >> 1;
        cTag2[i] = cTag2[i] | cImageBit;
        uiOffset++;
    }
}
char cTagTemp[5] = { cTag2[0], cTag2[1], cTag2[2], cTag2[3],
cTag2[4] }; //для сравнение с меткой
if (strcmp(cTag, cTagTemp))
{
    printf("No secret text here.\n");
}
else
    printf("Secret text.\n");

CloseHandle(hFile);
delete[] Buffer;
}

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, BufSize = 0, dwBytes1 = 0;
    BufSize = GetFileSize(hFile, NULL);
    char * Buffer = new char[BufSize / sizeof(char)];
    ReadFile(hFile, Buffer, BufSize, &dwBytes, NULL);

    unsigned int uiOffset;
    GetOffset(&uiOffset, Buffer);

    char cTagTemp[5], cImageByte;
    unsigned char cTag2[5] = { 0 };
    //Проверка метки
    for (int i = 0; i < 5; i++)
        cTag2[i] = 0; //Пока что обнуляем
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 8; j++)
        {
            char cImageBit = 0;
            cImageByte = Buffer[uiOffset];
            cImageBit = cImageByte & cMask;

```

```

        cImageBit = cImageBit << 7;
        cTag2[i] = cTag2[i] >> 1;
        cTag2[i] = cTag2[i] | cImageBit;
        uiOffset++;
    }
}

for (int i = 0; i < 5; i++)
    cTagTemp[i] = cTag2[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 = Buffer[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 = Buffer[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[] Buffer;
}

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, BufSize = 0, dwBytes1 = 0;
    BufSize = GetFileSize(hFile, NULL);
    char *Buffer = new char[BufSize / sizeof(char)];
    ReadFile(hFile, Buffer, BufSize, &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 dwBytes2 = 0, BufSize2 = 0;
BufSize2 = GetFileSize(hText, NULL);
char *Buffer2 = new char[BufSize2 / sizeof(char)];
ReadFile(hText, Buffer2, BufSize2, &dwBytes2, NULL);

////Запись текста в изображение
//Получение значения смещения
unsigned int uiOffset = 0;
GetOffset(&uiOffset, Buffer);

//Получение значений цветов и формирование новых
char cImageByte;
//Для обработки текста
char cTextByte = 0, cTextBit = 0;

unsigned int uiTextOffset = 0;
//Проверка размера
if (BufSize < (BufSize2 * 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 = Buffer[uiOffset];
        //и запись в него бита текста
        tempbit = temp & cMask;
        temp = temp >> 1;
        if (cImageByte % 2 == 0)
            cImageByte = cImageByte | tempbit;
        else
        {
            cImageByte = cImageByte & cMask2;
            cImageByte = cImageByte | tempbit;
        }
        Buffer[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 = Buffer[uiOffset];
        //и запись в него бита текста
        tempbit = BufSize2 & cMask;
        BufSize2 = BufSize2 >> 1; //изменение BufSize2
        if (cImageByte % 2 == 0)
            cImageByte = cImageByte | tempbit;
        else
        {
            cImageByte = cImageByte & cMask2;
            cImageByte = cImageByte | tempbit;
        }
        Buffer[uiOffset] = cImageByte;
        uiOffset++;
    }

    //восстановление BufSize2
    BufSize2 = GetFileSize(hText, NULL);

    //Цикл записи текста в изображение
    for (unsigned int i = 0; i < BufSize2; i++)
    {
        cTextByte = Buffer2[uiTextOffset]; //один байт текста на
        восемь байт изображения
        for (char j = 0; j < 8; j++)
        {
            //Получение байта изображения
            cImageByte = Buffer[uiOffset];
            //и запись в него бита текста
            cTextBit = cTextByte & cMask;
            cTextByte = cTextByte >> 1;
            if (cImageByte % 2 == 0)
                cImageByte = cImageByte | cTextBit;
            else
            {
                cImageByte = cImageByte & cMask2; //обнуляем
                младший бит для правильной записи
                cImageByte = cImageByte | cTextBit;
            }
            Buffer[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, Buffer, BufSize, &dwBytes1, NULL);

//Закрытие HANDLE
CloseHandle(hText);
CloseHandle(hFile);
delete[] Buffer;
delete[] Buffer2;
}

```

Файл ZI4.cpp

```
#include "ZI4.h"

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 << "-retrive <*.bmp> <*.txt>\n";
        }
    }
    else
    {
        cout << "-check <*.bmp>\n";
        cout << "-embed <*.bmp> <*.txt>\n";
        cout << "-retrieve <*.bmp> <*.txt>\n";
    }

    return 0;
}
```

Результаты работы программы

Операция проверки:

```
G:\_0\ZI_1\ZI4\Debug>ZI4.exe -check a.bmp  
No secret text here.
```

Рисунок 1 — Операция проверки на изображении без текста

```
G:\_0\ZI_1\ZI4\Debug>ZI4.exe -check a.bmp  
Secret text.
```

Рисунок 2 — Операция проверки изображения после записи в него текста

Операция внедрения записывает в младший бит байтов изображения бит текста. Визуального различия между изображениями не заметно.

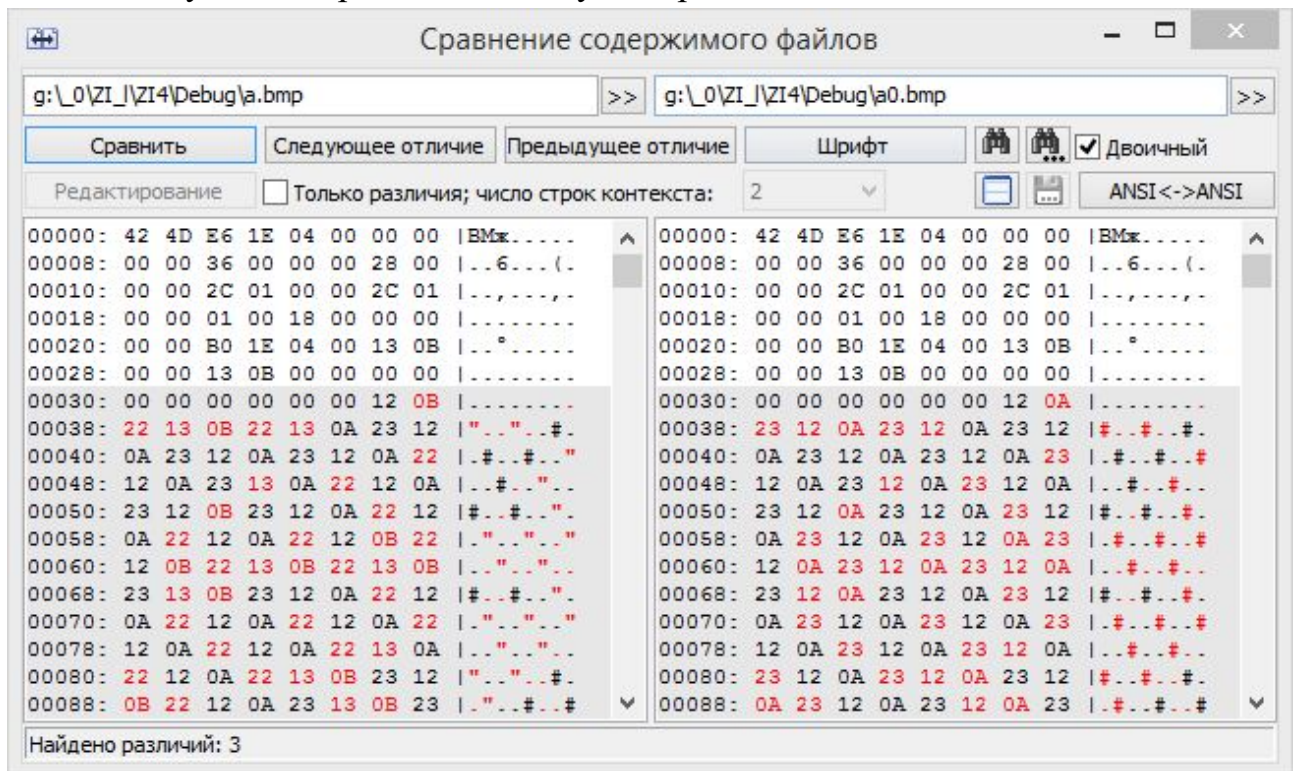


Рисунок 3 — Двоичное сравнение оригинального изображения(a0.bmp) и нового

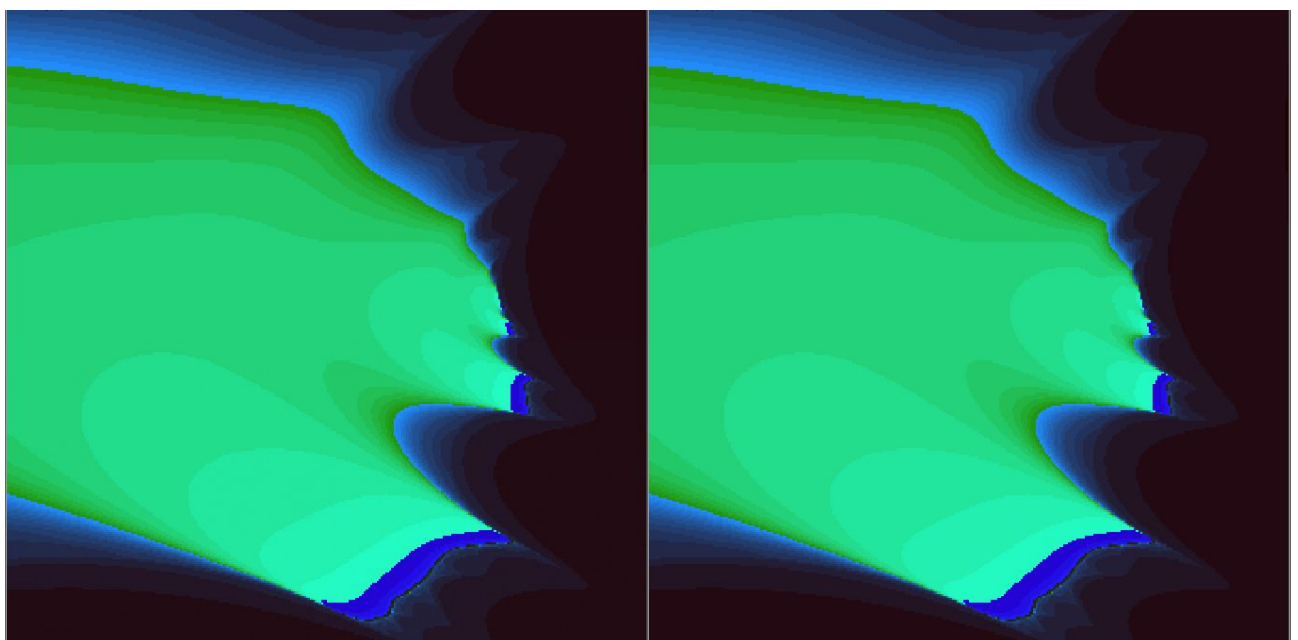


Рисунок 4 — Слева новое изображение, справа оригинальное

Операция извлечения:

```
G:\_0\ZI_1\ZI4\Debug>ZI4.exe -retrieve a.bmp b.txt  
Nothing to retrieve.
```

Рисунок 5 — Извлечение до записи текста в изображение

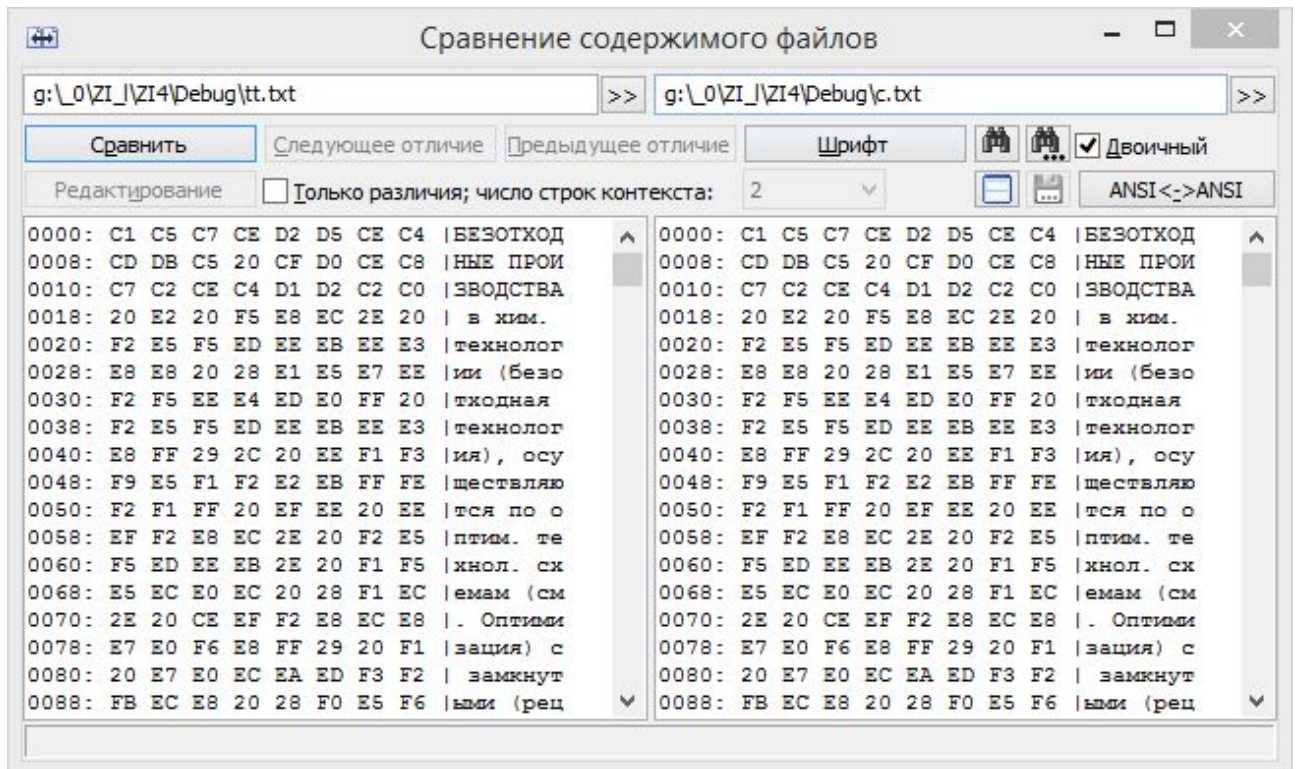


Рисунок 6 — Двоичное сравнение оригинального текста(слева) и восстановленного

```
G:\_0\ZI_1\ZI4\Debug>ZI4.exe  
-check <*.bmp>  
-embed <*.bmp> <*.txt>  
-retrieve <*.bmp> <*.txt>
```

Рисунок 7 — Запуск программы без параметров

Выводы

В ходе выполнения данной лабораторной работе были изучены методы стеганографии.

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

ОТЧЕТ

по лабораторной работе №5
по курсу «Защита информации»
на тему «Атрибуты безопасности в ОС Windows»

Выполнили:
студенты группы 15BB2
Артёмов И.В.
Жданов Д.М.

Принял:
Дубравин А.В.

Цель работы

Изучить методы работы с атрибутами безопасности в ОС Windows.

Лабораторное задание

Разработать программу, выполняющую создание файла с атрибутами безопасности для группы пользователей «Все» и правами на чтение и исполнение, а для текущего пользователя следует запретить чтение и исполнение.

Описание программы

Программа выполняет получение идентификаторов группы «Все» и текущего пользователя. Далее с их помощью создаётся список прав доступа посредством заполнения массивов структур EXPLICIT_ACCESS. Создаётся дескриптор безопасности путём выделения по нему памяти и инициализации. Список прав доступа заносится в дескриптор безопасности и далее дескриптор можно использовать при создании объектов ядра, указав его в поле lpSecurityDescriptor структуры SECURITY_ATTRIBUTES.

Листинг

Файл Zl5.cpp

```
#include <tchar.h>
#include <Windows.h>
#include <AclAPI.h>

int _tmain(int argc, TCHAR * argv[], TCHAR * envp[])
{
    PSID pEveryoneSID = NULL;
    SID_IDENTIFIER_AUTHORITY SIDAAuthWorld =
SECURITY_WORLD_SID_AUTHORITY;
    AllocateAndInitializeSid(&SIDAuthWorld, 1,
SECURITY_WORLD_RID, 0, 0, 0, 0, 0, 0, &pEveryoneSID);

    EXPLICIT_ACCESS ea[2];
    ZeroMemory(&ea, 2 * sizeof(EXPLICIT_ACCESS));
    ea[0].grfAccessPermissions = GENERIC_EXECUTE | GENERIC_READ;
    ea[0].grfAccessMode = SET_ACCESS;
    ea[0].grfInheritance = NO_INHERITANCE;
    ea[0].Trustee.TrusteeForm = TRUSTEE_IS_SID;
    ea[0].Trustee.TrusteeType = TRUSTEE_IS_WELL_KNOWN_GROUP;
    ea[0].Trustee.ptstrName = (LPTSTR)pEveryoneSID;

    HANDLE hToken = NULL;
    DWORD dwBufferSize = 0;
    PTOKEN_USER pTokenUser = NULL;

    if (!OpenProcessToken(GetCurrentProcess(), TOKEN_QUERY,
&hToken)) {
        _tprintf(_T("OpenProcessToken failed.\n"),
            GetLastError());
        return -1;
    }

    (void)GetTokenInformation(hToken, TokenUser, NULL, 0,
&dwBufferSize);

    pTokenUser = (PTOKEN_USER)malloc(dwBufferSize);
    memset(pTokenUser, 0, dwBufferSize);

    if (!GetTokenInformation(hToken, TokenUser, pTokenUser,
dwBufferSize,
        &dwBufferSize)) {
        _tprintf(_T("GetTokenInformation failed.\n"),
            GetLastError());
        return -2;
    }
}
```

```

}

CloseHandle(hToken);

ea[1].grfAccessPermissions = GENERIC_EXECUTE | GENERIC_READ;
ea[1].grfAccessMode = DENY_ACCESS;
ea[1].grfInheritance = NO_INHERITANCE;
ea[1].Trustee.TrusteeForm = TRUSTEE_IS_SID;
ea[1].Trustee.TrusteeType = TRUSTEE_IS_USER;
ea[1].Trustee.ptstrName = (LPTSTR)pTokenUser->User.Sid;

DWORD dwRes;
PACL pACL = NULL;

dwRes = SetEntriesInAcl(2, ea, NULL, &pACL);
if (ERROR_SUCCESS != dwRes)
{
    _tprintf(_T("SetEntriesInAcl Error %u\n"),
GetLastError());
    FreeSid(pEveryoneSID);
    return 1;
}

PSECURITY_DESCRIPTOR pSD = NULL;
pSD = (PSECURITY_DESCRIPTOR)LocalAlloc(LPTR,
    SECURITY_DESCRIPTOR_MIN_LENGTH);
if (NULL == pSD)
{
    _tprintf(_T("pSD 1 Error %u\n"), GetLastError());
    FreeSid(pEveryoneSID);
    LocalFree(pACL);
    return 2;
}

if (!InitializeSecurityDescriptor(pSD,
SECURITY_DESCRIPTOR_REVISION))
{
    _tprintf(_T("pSD 2 Error %u\n"), GetLastError());
    FreeSid(pEveryoneSID);
    LocalFree(pACL);
    LocalFree(pSD);
    return 3;
}

if (!SetSecurityDescriptorDacl(pSD, TRUE, pACL, FALSE))
{
    _tprintf(_T("Dacl Error %u\n"), GetLastError());

```

```

        FreeSid(pEveryoneSID);
        LocalFree(pACL);
        LocalFree(pSD);
        return 4;
    }

    SECURITY_ATTRIBUTES sa;
    sa.nLength = sizeof(SECURITY_ATTRIBUTES);
    sa.lpSecurityDescriptor = pSD;
    sa.bInheritHandle = FALSE;

    HANDLE hFile;
    if (INVALID_HANDLE_VALUE == (hFile =
CreateFile(_T("D://_0/_fill10Vzvv.txt"),
        GENERIC_WRITE, FILE_SHARE_WRITE,
        &sa, CREATE_ALWAYS,
        FILE_ATTRIBUTE_NORMAL, NULL)))
    {
        CloseHandle(hFile);
        FreeSid(pEveryoneSID);
        LocalFree(pACL);
        LocalFree(pSD);
        return 5;
    }
    TCHAR * tcArray = _T("fill10.txt");
    DWORD dwBytesWritten;
    WriteFile(hFile, tcArray, _tcslen(tcArray)*sizeof(TCHAR),
&dwBytesWritten, NULL);

    CloseHandle(hFile);
    FreeSid(pEveryoneSID);
    LocalFree(pACL);
    LocalFree(pSD);
    free(pTokenUser);
    return 0;
}

```

Результаты работы программы

Программа производит создание текстового файла с заданными атрибутами безопасности — для группы пользователей «Все» и правами на чтение и выполнение, а для текущего пользователя следует запретить чтение и выполнение.

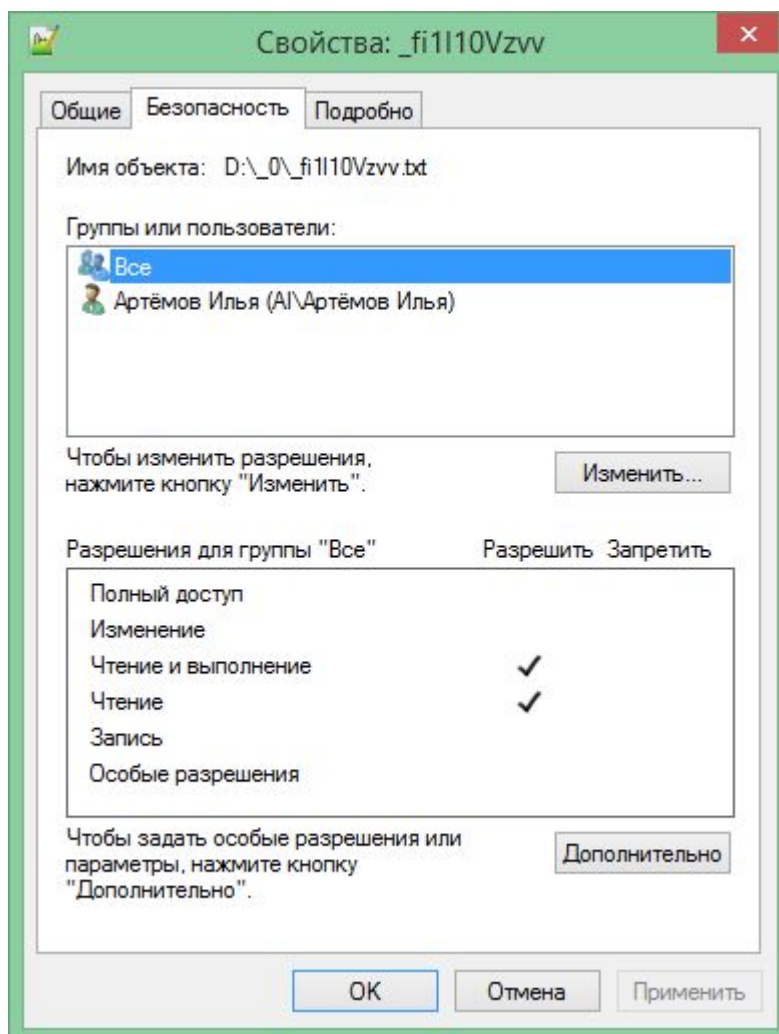


Рисунок 1 — Атрибуты безопасности для группы «Все»

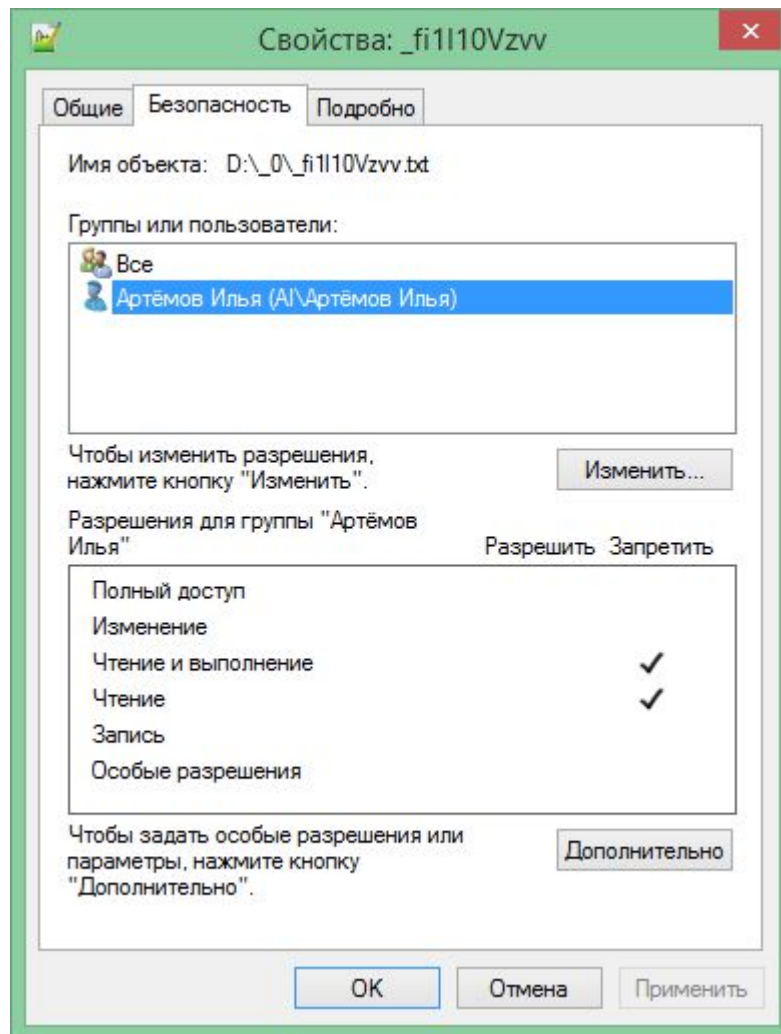


Рисунок 2 — Атрибуты безопасности для текущего пользователя

Выводы

В ходе выполнения данной лабораторной работы были изучены методы работы с атрибутами безопасности в ОС Windows.