Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

по курсу «Защита информации в сети Internet»

на тему «Разработка программы шифрования/дешифрирования двоичных файлов с использованием перестановочного шифра»

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.

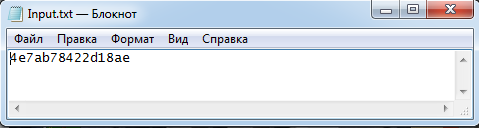
2020

**Цель работы:** разработать программу шифрования/дешифрирования двоичных файлов с использованием перестановочного шифра.

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

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

**Исходный двоичный файл:**



**Рисунок 1**

**Листинг программы:**

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

using namespace std;

//Значения 16-ой системы счисления

char Hex[16] = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f' };

//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 индекс массива A размера Size, значение которого равно Value (Используется при дешифровании)

template <class Type>

int FindIndex(Type A[], int Size, Type Value)

{

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

if (A[i] == Value)

return i;

return -1;

}

//Определяет количество нулевых байт в конце массива (Используется при дешифровании)

int NZeroBytes(vector <bool> Bits)

{

int Size = Bits.size() / 4;

int ZeroBytes = 0;

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

{

int ZeroBits = 0;

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

if (Bits[4 \* i + j] == 0)

ZeroBits++;

if (ZeroBits == 4) ZeroBytes++;

else break;

}

return ZeroBytes;

}

//Определяет количество символов в файле FileName

int LengthFile(string FileName)

{

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

int Length = file.tellg();

file.close();

return Length;

}

//Возвращает текст из файла FileName размера Size

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

{

ifstream File = OpenFile(FileName);

vector <char> Text(Size);

//Символы, считанные из файла, загружаются в массив

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

File.get(Text[i]);

File.close();

return Text;

}

//Возвращает массив битов текста Text. Добавляет ZeroBytes нулевых битов

vector <bool> TextToBitsArray(vector <char> Text, int ZeroBits)

{

int Size = Text.size(); //Находит количество символов в исходном тексте

vector <bool> Bits(Size \* 4 + ZeroBits); //Инициализирует массив битов

for (int i = 0; i < Size; i++) //Получает биты для каждого символа текста и заносит их в массив

{

int C = FindIndex(Hex, 16, Text[i]); //Переводит число из 16-ой в 10-ую систему счисления

for (int j = 4; j > 0; j--) //Переводит число в 2-ую сс, вычисляя остаток от деления на 2

{

Bits[4 \* i + j - 1] = C % 2;

C /= 2;

}

}

//Нулевые биты, добавляемые в массив для срабатывания ключа:

for (int i = 4 \* Size; i < 4 \* Size + ZeroBits; i++)

Bits[i] = 0;

return Bits;

}

//Выводит элементы вектора Vector шаблонного типа Type в поток вывода Stream (Используется для вывода массива битов и строк в файл или на экран)

template <class Type>

ostream & operator << (ostream & Stream, vector <Type> Vector)

{

int Size = Vector.size();

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

Stream << Vector[i];

return Stream;

}

//Переводит массив битов Bits размера Size байт обратно в текст, переводя каждые 4 бита в отдельный символ

vector <char> BitsArrayToText(vector <bool> Bits, int Size)

{

vector <char> Text(Size, 0);

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

{

for (int j = 0; j < 4; j++) //Переводит число из 2-ой системы счисления в 10-ую

Text[i] += Bits[4 \* i + j] \* pow(2.0, 3 - j);

Text[i] = Hex[Text[i]]; //Переводит число из 10-ой сс в 16 сс

}

return Text;

}

//Шифрование/Дешифрование текста файла InputFile ключом Key размера Size. Результат шифрования выодится в файл OutputFile.

//Если Mod = Coding - выполняется шифрование,

//Если Mod = Decode - выполняется дешифрование

void CodingFile(string InputFile, string OutputFile, int Key[], int Size, string Mode)

{

ofstream Output(OutputFile, ios::trunc); //Открывает файл для вывода в него зашифрованного текста, ios::trunc удаляет содержимое файла, если он не пуст

int Length = LengthFile(InputFile); //Определяет количество символов в исходном файле

int NZeros = 4 \* (Size - Length % Size); //Определяет число нулевых бит, добавляемых в массив битов для срабатывания ключа

if (Length % Size == 0) NZeros = 0;

vector <char> Text = GetTextInFile(InputFile, Length); //Загружает текст из исхдного файла

vector <bool> Bits = TextToBitsArray(Text, NZeros); //Получает массив битов исходного файла

cout << Text << endl; //Выводит исходный текст на экран

cout << Bits << endl; //Выводим массив битов на экран

if (Mode == "Coding") //Шифрование файла

{

vector <bool> Encrypt(4 \* Length + NZeros); //Массив битов зашифрованного файла

for (int i = 0; i < (Length \* 4 + NZeros) / (8 \* Size); i++) //Шифрование битов

for (int j = 0; j < Size \* 8; j++)

Encrypt[Size \* 8 \* i + j] = Bits[Size \* 8 \* i + Key[j]];

Output << BitsArrayToText(Encrypt, (Length \* 4 + NZeros) / 4); //Вывод полученного текста после шифрования битов в файл

}

if (Mode == "Decode") //Дешифрование файла

{

vector <bool> Decrypt(4 \* Length); //Массив битов дешифрованного файла

for (int i = 0; i < Length / (2 \* Size); i++) //Дешифрование битов

for (int j = 0; j < Size \* 8; j++)

Decrypt[Size \* 8 \* i + j] = Bits[Size \* 8 \* i + FindIndex(Key, 8 \* Size, j)];

//Вывод полученного текста после дешифрования битов в файл, отбрасывая нулевые байты в конце массива

Output << BitsArrayToText(Decrypt, Length - 1 \* NZeroBytes(Decrypt));

}

Output.close();

}

//Выполняет сравнение файлов по содержимому

bool CompareFiles(string FileName1, string FileName2)

{

ifstream File1 = OpenFile(FileName1);

ifstream File2 = OpenFile(FileName2);

char ch1, ch2;

bool Result = true;

while (File1.get(ch1) && File2.get(ch2))

{

if (ch1 != ch2)

{

Result = false;

break;

}

}

File1.close();

File2.close();

return Result;

}

/\*

Input.txt - Исходный текстовый файл

Encrypt.txt - Шифрованный текстовый файл

Decrypt.txt - Дешифрованный текстовый файл

\*/

void main()

{

setlocale(LC\_ALL, "Russian");

const int N = 32; //Число значений в перестановочном ключе

int Key[N] = { 12, 13, 30, 5, 27, 6, 11, 25, 3, 21, 22, 2, 23, 0, 8, 4, 18, 19, 10, 1, 14, 29, 9, 28, 20, 17, 26, 31, 7, 16, 15, 24 };//Перестановочный ключ

cout << "Исходный текстовый файл:" << endl;

CodingFile("Input.txt", "Encrypt.txt", Key, N / 8, "Coding"); //Шифрование текста

cout << endl << "Зашифрованный текстовый файл:" << endl;

CodingFile("Encrypt.txt", "Decrypt.txt", Key, N / 8, "Decode"); //Расшифрование текста

bool Compare = CompareFiles("Input.txt", "Decrypt.txt"); //Двоичное сравнение исхдного и расшифрованного файлов

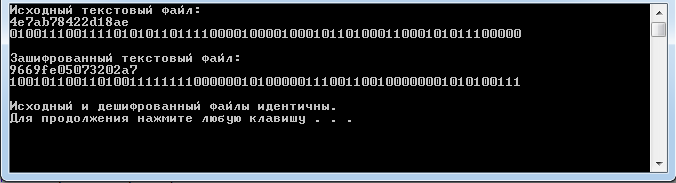
if (Compare) cout << endl << "Исходный и дешифрованный файлы идентичны." << endl;

else cout << endl << "Исходный и дешифрованный файлы различны." << endl;

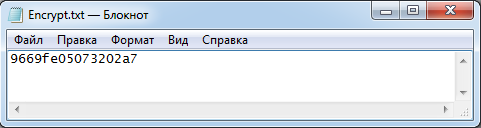
system("pause");

}

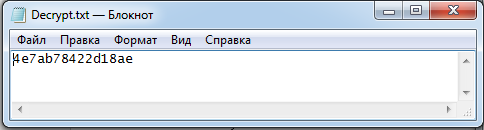
**Результаты работы программы:**



**Рисунок 2 – Результат отладки программы**

****

**Рисунок 3 – Зашифрованный текст**

****

**Рисунок 4 – Дешифрованный текст**

**Вывод:** Была разработана программа, выполняющая шифрование, дешифрование двоичного файла с использованием перестановочного шифра.