|  |
| --- |
| Пензенский государственный университет  Факультет вычислительной техники  Кафедра «Вычислительная техника» |
| Отчет  по лабораторной работе №2  по дисциплине «Методы защиты компьютерных систем»  Вариант № 1 |
|  |
|  |
| Выполнили: студенты группы 19ВВ3:  Земляков В. Д.  Ерёмин А. А.  Проверил:  Дубравин А. В. |
| Пенза  2022 |

**Тема:** разработка программы шифрования/дешифрирования текста с использованием перестановочного шифра.

Шифрование перестановкой заключается в том, что символы шифруемого текста переставляются по определенному правилу в пределах некоторого блока этого текста. При достаточной длине блока, в пределах которого осуществляется перестановка, и сложном неповторяющемся порядке перестановки можно достигнуть приемлемой стойкости шифра. Ключом является последовательность номеров символов в блоке.

**Задание**

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

Последовательность перестановки: 3, 6, 5, 2, 4, 1, 7, 9, 8, 10

Ход работы

Листинг программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

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

void CopyArray(int\* source, int\* dest, int size)

{

for (size\_t i = 0; i < size; i++)

{

dest[i] = source[i];

}

}

class Encoder {

private:

int\* key;

int size\_key;

int size\_block;

void SetBit(int num, int index)

{

num |= 1 << index;

}

public:

Encoder()

{

this->size\_key = 32;

this->key = new int[this->size\_key];

CopyArray(key\_std, this->key, size\_key);

this->size\_block = 4;

}

Encoder(int\* key, int size\_key)

{

this->size\_key = size\_key;

this->key = new int[this->size\_key];

CopyArray(key, this->key, size\_key);

this->size\_block = 4;

}

void Encode(char\* source\_text, char\* encode\_text, int size\_source)

{

if (size\_source % this->size\_block!= 0)

{

memset(&source\_text[size\_source], '0', this->size\_block - size\_source);

source\_text[this->size\_block] = '\0';

}

int source\_num = \*(int\*)source\_text;

int enc\_num = 0;

for(int i = 0; i < size\_key; i++)

{

enc\_num |= ((source\_num >> this->key[i])&1) << i;

}

\*(int\*)encode\_text = enc\_num;

}

void Decode(char\* source\_text, char\* decode\_text)

{

int source\_num = \*(int\*)source\_text;

int dec\_num = 0;

for(int i = 0; i < size\_key; i++)

{

dec\_num |= ((source\_num >> i)&1) << this->key[i];

}

\*(int\*)decode\_text = dec\_num;

}

bool FileEncode(string text\_filename, string encode\_filename)

{

ifstream text\_file(text\_filename, ios::binary);

ofstream encode\_file(encode\_filename, ios::binary);

char\* text = new char[size\_block + 1];

char\* encode\_text = new char[size\_block + 1];

text[size\_block] = '\0';

encode\_text[size\_block] = '\0';

int count\_ch;

if (text\_file.is\_open() && encode\_file.is\_open())

{

cout << "Files is open. Start Encoding: " << text\_filename << "\n";

text\_file.seekg(0, text\_file.end);

count\_ch = text\_file.tellg();

text\_file.seekg(0, text\_file.beg);

encode\_file.write((char\*)&count\_ch, sizeof(int));

while (!text\_file.eof())

{

text\_file.read(text, size\_block);

if (text\_file.eof() && text\_file.gcount() == 0)

break;

Encode(text, encode\_text, text\_file.gcount());

encode\_file.write(encode\_text, size\_block);

}

text\_file.close();

encode\_file.close();

cout << "Success! Encode file: " << encode\_filename << "\n";

return 0;

}

else

{

cout << "Files opened error!\n";

return 1;

}

}

bool FileDecode(string encode\_filename, string decode\_filename)

{

ifstream encode\_file(encode\_filename, ios::binary);

ofstream decode\_file(decode\_filename, ios::binary);

char\* encode\_text = new char[size\_block + 1];

encode\_text[size\_block] = '\0';

char\* decode\_text = new char[size\_block + 1];

decode\_text[size\_block] = '\0';

int length\_file = 0;

if (encode\_file.is\_open() && decode\_file.is\_open())

{

cout << "Files is open. Start Decoding " << encode\_filename << "\n";

encode\_file.read((char\*)&length\_file, sizeof(int));

while (!encode\_file.eof())

{

encode\_file.read(encode\_text, size\_block);

if (encode\_file.eof() && encode\_file.gcount() == 0)

break;

Decode(encode\_text, decode\_text);

int length\_encode = encode\_file.tellg();

length\_encode -= 4;

if (length\_encode > length\_file)

{

decode\_text[length\_file % size\_block] = '\0';

decode\_file.write(decode\_text, length\_file % size\_block);

break;

}

decode\_file.write(decode\_text, size\_block);

}

encode\_file.close();

decode\_file.close();

cout << "Success! Decode file: " << decode\_filename << "\n";

return 0;

}

else

{

cout << "Files opened error!\n";

return 1;

}

}

};

int main()

{

Encoder enc = Encoder();

enc.FileEncode("text.txt", "enc\_text.txt");

enc.FileDecode("enc\_text.txt", "dec\_text.txt");

enc.FileEncode("img.png", "enc\_img.png");

enc.FileDecode("enc\_img.png", "dec\_img.png");

}

Результат работы программы

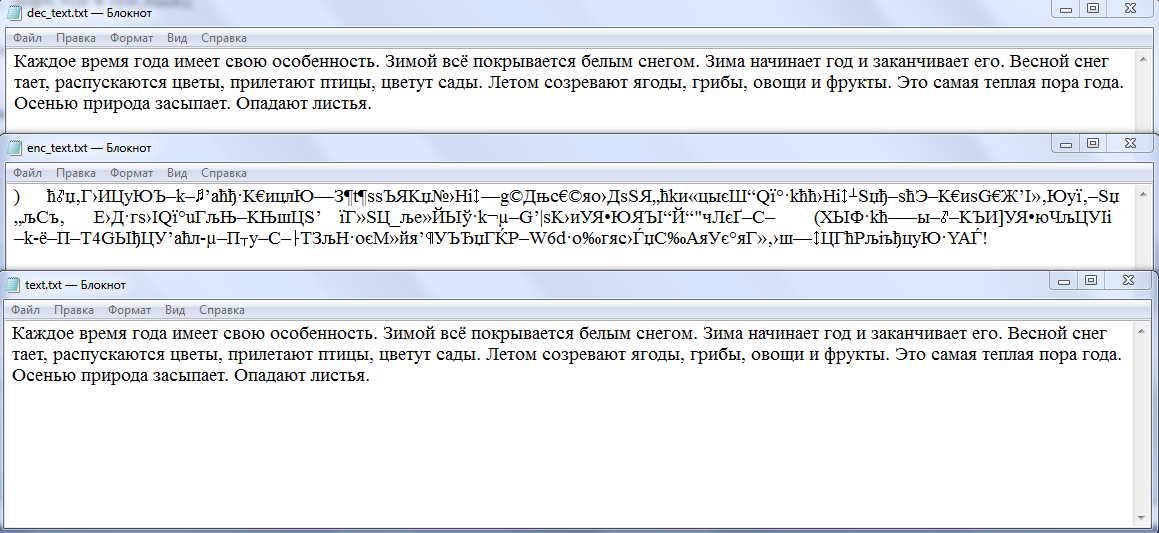


рисунок 1. Сравнение текстовых файлов

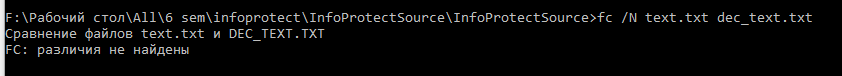


рисунок 2. Сравнение текстовых файлов.

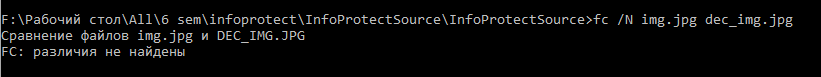


рисунок 3. Сравнение изображений.

**Вывод**

В ходе работы познакомились с концепцией перестановочных шифров, написали функции перестановки порядка символов в файле, с помощью которых реализовали программу шифрования/дешифрирования текста с использованием данного шифра. Итоговый и исходный файлы были сопоставлены между собой и оказались одинаковыми.