Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №1

по курсу «Защита информации в сети Internet»

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

Вариант 2

Выполнили  
студенты группы 18ВВ1:

Коротов В.О.

Максимова Д.Р.

Приняли:

к.т.н., доцент Дубравин А.В.

к.т.н., доцент Карамышева Н.С.

**2020**

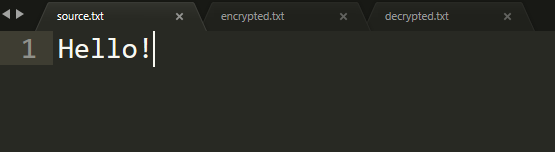
**Цель работы:** разработать программу шифрования/дешифрирования текста с использованием перестановочного шифра.

**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 фала, полученного после расшифровывания.

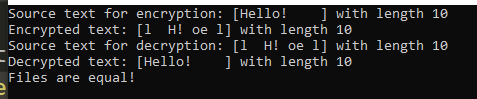
|  |  |
| --- | --- |
| Номер варианта | Последовательность перестановки |
| 2 | 3 9 8 1 6 10 5 2 7 4 |

**Описание работы программы:** имеется исходный файл source.txt, который шифруется по ключу. Результат шифрования записывается в файл encrypted.txt. Далее текст из этого файла расшифровывается и записывается в файл decrypted.txt. После этого сравниваем файл с исходным текстом source.txt и расшифрованный файл decrypted.txt. Результат сравнения выводим в консоль.

Файл с исходным текстом:



Результат работы программы:



Листинг:

#include "pch.h"

#include <iostream>

void check\_file(FILE \*file, const char \*filename)

{

if (!file)

{

printf("Can't open file %s!", filename);

exit(1);

}

}

char\* read\_file(const char\* filename)

{

FILE \*file = fopen(filename, "rb");

check\_file(file, filename);

fseek(file, 0, SEEK\_END);

size\_t size = ftell(file);

fseek(file, 0, SEEK\_SET);

size\_t newSize = size;

while (newSize % 10 != 0) newSize++;

char \*source\_text = (char\*)malloc(newSize \* sizeof(char) + 1);

fread(source\_text, 1, size, file);

for (int i = size; i < newSize; i++) source\_text[i] = ' ';

source\_text[newSize] = '\0';

fclose(file);

return source\_text;

}

char\* encrypt\_str(const char\* text)

{

const int key[10] = { 3, 9, 8, 1, 6, 10, 5, 2, 7, 4 };

size\_t size = strlen(text);

size\_t blocks = size / 10;

char\* encrypted\_text = (char \*)malloc(sizeof(char) \* size + 1);

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

{

for (int j = 0; j < 10; j++)

{

encrypted\_text[10 \* i + j] = text[10 \* i + key[j] - 1];

}

}

encrypted\_text[size] = '\0';

return encrypted\_text;

}

char\* decrypt\_str(const char\* text)

{

const int key[10] = { 3, 9, 8, 1, 6, 10, 5, 2, 7, 4 };

//const int key[10] = { 4, 8, 1, 10, 7, 5, 9, 3, 2, 6 };

size\_t size = strlen(text);

size\_t blocks = size / 10;

char\* decrypted\_text = (char \*)malloc(sizeof(char) \* size + 1);

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

{

for (int j = 0; j < 10; j++)

{

decrypted\_text[10 \* i + key[j] - 1] = text[10 \* i + j];

}

}

decrypted\_text[size] = '\0';

return decrypted\_text;

}

void encrypt\_file(const char\* source\_fname, const char\* output\_fname)

{

FILE \*fsource = fopen(source\_fname, "rb");

check\_file(fsource, source\_fname);

char \*source\_text = read\_file(source\_fname);

printf("Source text for encryption: [%s] with length %d\n", source\_text, strlen(source\_text));

char \*encrypted\_text = encrypt\_str(source\_text);

printf("Encrypted text: [%s] with length %d\n", encrypted\_text, strlen(encrypted\_text));

FILE \*foutput = fopen(output\_fname, "wb");

check\_file(foutput, output\_fname);

fwrite(encrypted\_text, 1, strlen(encrypted\_text), foutput);

fclose(fsource);

fclose(foutput);

free(source\_text);

free(encrypted\_text);

}

void decrypt\_file(const char\* source\_fname, const char\* output\_fname)

{

FILE \*fsource = fopen(source\_fname, "rb");

if (!fsource)

{

printf("Can't open file %s!\n", source\_fname);

exit(1);

}

char \*source\_text = read\_file(source\_fname);

printf("Source text for decryption: [%s] with length %d\n", source\_text, strlen(source\_text));

char \*decrypted\_text = decrypt\_str(source\_text);

printf("Decrypted text: [%s] with length %d\n", decrypted\_text, strlen(decrypted\_text));

FILE \*foutput = fopen(output\_fname, "wb");

if (!foutput)

{

printf("Can't open file %s!\n", output\_fname);

exit(1);

}

fwrite(decrypted\_text, 1, strlen(decrypted\_text), foutput);

fclose(fsource);

fclose(foutput);

free(source\_text);

free(decrypted\_text);

}

bool file\_equals(const char\* file1\_name, const char\* file2\_name)

{

bool files\_are\_equal = true;

FILE \*file1 = fopen(file1\_name, "rb");

check\_file(file1, file1\_name);

FILE \*file2 = fopen(file2\_name, "rb");

check\_file(file2, file2\_name);

int c1, c2;

while ((c1 = fgetc(file1)) != EOF && (c2 = fgetc(file2)) != EOF)

{

if (c1 != c2)

{

files\_are\_equal = false;

break;

}

}

fclose(file1);

fclose(file2);

return files\_are\_equal;

}

int main()

{

const char \*source\_fname = "source.txt";

const char \*encrypted\_fname = "encrypted.txt";

const char \*decrypted\_fname = "decrypted.txt";

encrypt\_file(source\_fname, encrypted\_fname);

decrypt\_file(encrypted\_fname, decrypted\_fname);

if (file\_equals(source\_fname, decrypted\_fname))

{

printf("Files are equal!\n");

}

else

{

printf("Files are not equal!\n");

}

return 0;

}

**Вывод:** разработали программу шифрования/дешифрирования текста с использованием перестановочного шифра.