МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Лабораторная работа

Шифрование Xor

РУКОВОДИТЕЛЬ:

Мартынов Д.С.

СТУДЕНТ:

группы 21-ИВТз-3

Морозов О.С.

Нижний Новгород 2022

Цель работы

Написать программу шифрования и дешифрования файла с использованием алгоритма Xor

Описание программы

1. Пользователь вводит в терминал команду в формате

./codec -e 23 file.txt

codec -- название файла программы

-e или -encrypt -- флаг шифрования

23 -- ключ

file.txt -- название файла для шифрования/дешифрования

Далее пользователь вводит строку для шифрования

Строка шифруется и пишется в файл file.txt

2. Пользователь вводит в терминал команду в формате

./codec -d 23 file.txt

-d или -decrypt -- флаг дешифрования

Строка читается из файла, дешифруется и отображается пользователю.

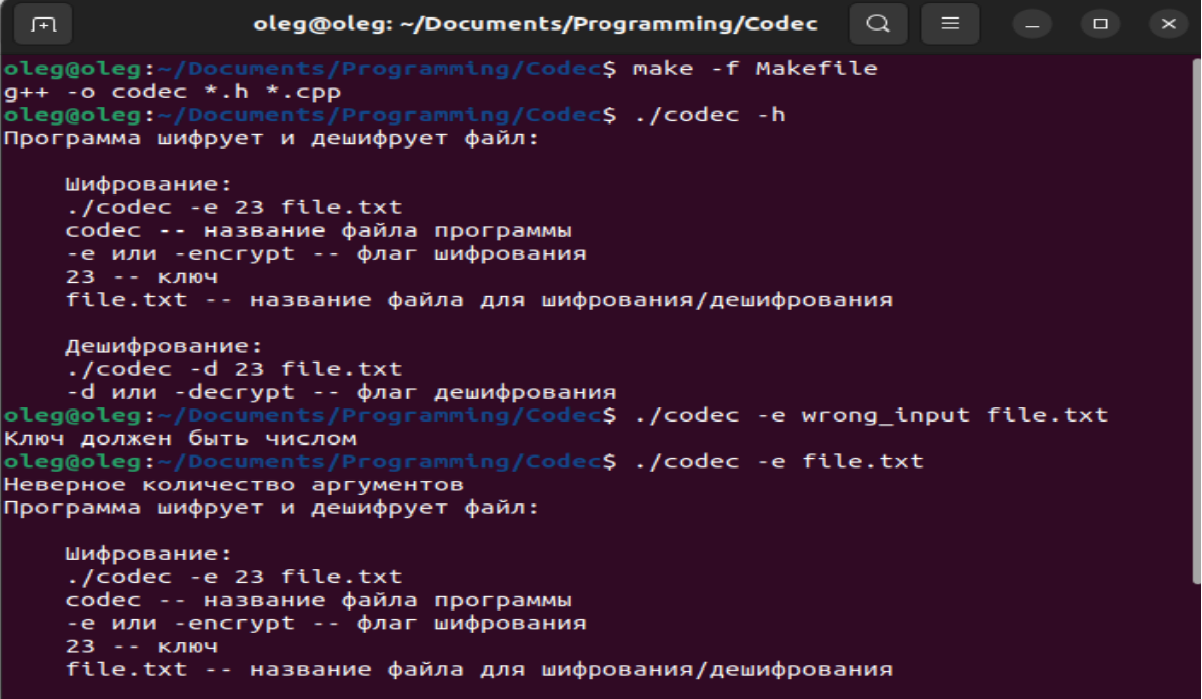
Если пользователь ввел тот же ключ, по которому выполнялось шифрование, то пользователю отображается верная строка, иначе неверная

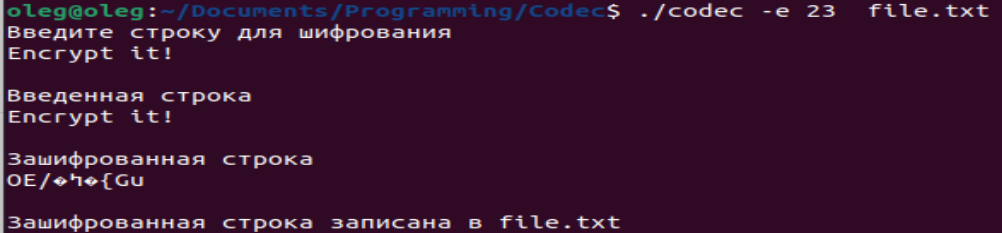
Файлы программы

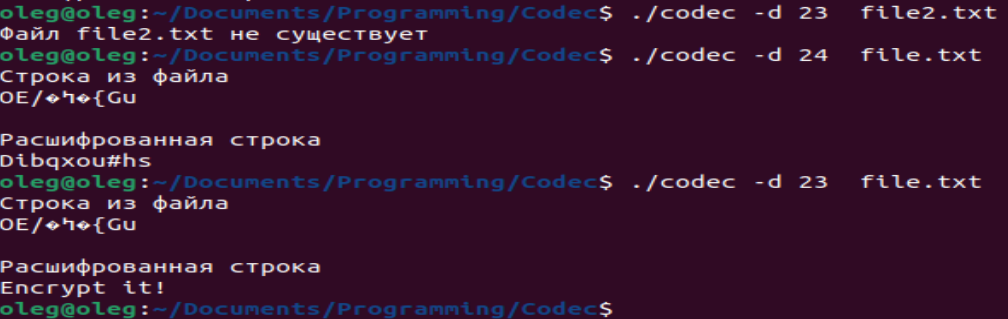
IOManager\_h.h – заголовочный файл менеджера ввода/вывода

IOManager\_c.cpp – файл реализации функций IOManager\_h.h

Codec.cpp – главная логика программы

Скриншоты





Codec.cpp

/\* Codec.cpp - Листинг программы шифрования/дешифрования

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Имя файла : Codec.cpp

\* Резюме : Пример программы, написанной на языке программирования C++

\* Описание : Разрабатываемая программа шифрует и дешифрует файл

\* Автор : Morozov Oleg

\* Платформа : OS Linux Ubuntu

\*/

#define A 8253729

#define B 2396403

#define C 32768

#include "IOManager\_h.h"

#include <iostream>

#include <string>

#include <cstdlib>

using std::cout;

using std::cin;

using std::string;

/\* Генератор случайных чисел \*/

unsigned int Random(unsigned int seed){ return (A \* seed + B) % C;}

/\* Шифрование/дешифрование \*/

string ModifySequence(string str, int key, int seed) {

if (str.size() == 0) { throw 1; }

string modified\_str(str);

int seed2 = seed;

for (int i = 0; i < str.size(); i++) {

modified\_str[i] = str[i]^key + Random(seed2);

++seed2;

}

return modified\_str;

}

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

if (argc < 2 || argc == 3 || argc > 4) {

cout << "Неверное количество аргументов\n";

cout << IOManager::GetManual();

exit(1);

}

string commandType = argv[1]; // Тип команды

if (commandType == "-h" || commandType == "-help"){ // Вызов справки

cout << IOManager::GetManual();

exit(0);

}

string keyString = argv[2]; // Ключ

string filename = argv[3]; // Имя файла

string string\_to\_modify; // Строка для шифрования/дешифрования

string modified\_string; // Измененная строка

int seed = 0; // Сид для генератора случайных чисел

int key; // Ключ

try{

/\* ШИФРОВАНИЕ \*/

if (commandType == "-e" || commandType == "-encrypt"){

if (!IOManager::IsDigit(keyString)) {

cout << "Ключ должен быть числом\n";

exit(1);

}

key = std::stoi(keyString);

cout << "Введите строку для шифрования\n";

getline(cin, string\_to\_modify);

modified\_string = ModifySequence(string\_to\_modify, key, seed);

cout << "\nВведенная строка\n" << string\_to\_modify << "\n\n";

cout << "Зашифрованная строка\n" << modified\_string << "\n\n";

if (!IOManager::FileExists(filename)) {

IOManager::CreateFile(filename);

}

IOManager::WriteFile(filename, modified\_string);

cout << "Зашифрованная строка записана в " << filename << "\n";

}

/\* ДЕШИФРОВАНИЕ \*/

if (commandType == "-d" || commandType == "-decrypt") {

if (!IOManager::IsDigit(keyString)) {

cout << "Ключ должен быть числом\n";

exit(1);

}

key = std::stoi(keyString);

if (!IOManager::FileExists(filename)) {

cout << "Файл " << filename << " не существует\n";

exit(1);

}

else if (IOManager::FileIsEmpty(filename)) {

cout << "Файл " << filename << " пуст\n";

exit(1);

}

string string\_from\_file = IOManager::ReadFile(filename);

cout << "Строка из файла\n" << string\_from\_file << "\n\n";

modified\_string = ModifySequence(string\_from\_file, key, seed);

cout << "Расшифрованная строка\n" << modified\_string << "\n";

}

}catch(std::runtime\_error er){

cout << "Неизвестная ошибка\n";

exit(1);

}

return 0;

}

IOManager\_h.h

#ifndef IOMANAGER\_H

#define IOMANAGER\_H

#include <iostream>

#include <string>

#include <cstring>

#include <fstream>

#include <vector>

#include <limits>

using std::cout;

using std::cin;

using std::string;

using std::ofstream;

using std::ifstream;

using std::vector;

class IOManager {

public:

static string GetManual(); // Справка

static bool IsDigit(string); // Проверка на цифру

static bool FileExists(string); // Проверка на существование файла

static bool FileIsEmpty(string); // Проверка файла на пустоту

static void CreateFile(string); // Создание файла

static void WriteFile(string, string); // Запись в файл

static string ReadFile(string); // Чтение файйла в строку

};

#endif

IOManager\_c.cpp

#ifndef IOMANAGER\_C

#define IOMANAGER\_C

#include "IOManager\_h.h"

string IOManager::GetManual() {

string manual = "Программа шифрует и дешифрует файл:\n\n\

Шифрование:\n\

./codec -e 23 file.txt\n\

codec -- название файла программы\n\

-e или -encrypt -- флаг шифрования\n\

23 -- ключ\n\

file.txt -- название файла для шифрования/дешифрования\n\n\

Дешифрование:\n\

./codec -d 23 file.txt\n\

-d или -decrypt -- флаг дешифрования\n\n\

-h или -help -- справка\n";

return manual;

}

bool IOManager::IsDigit(string str) {

if (str.empty()) { return false; }

bool isDigit = true;

for (char s : str) {

if (!isdigit(s)) {

isDigit = false;

break;

}

}

return isDigit;

}

bool IOManager::FileExists(string fileName) {

bool isExist = false;

std::ifstream fin(fileName.c\_str());

if (fin.is\_open())isExist = true;

fin.close();

return isExist;

}

bool IOManager::FileIsEmpty(string fileName) {

ifstream fin(fileName);

if (!fin.is\_open()) { throw std::runtime\_error("File opening error\n"); }

int symCount = 0;

char c;

while (!fin.eof()) {

fin >> c;

++symCount;

}

fin.close();

return (symCount == 1 || symCount == 0) ? true : false;

}

void IOManager::CreateFile(string fileName) {

ofstream fout(fileName, std::ios\_base::out);

if (!fout.is\_open()) { throw std::runtime\_error("File opening error\n"); }

fout.close();

}

void IOManager::WriteFile(string fileName, string str) {

ofstream fout(fileName, std::ios\_base::out | std::ios\_base::trunc);

if (!fout.is\_open()) { throw std::runtime\_error("File openning error\n"); }

fout << str;

fout.close();

}

string IOManager::ReadFile(string fileName) {

ifstream fin(fileName, std::ios\_base::in);

if (!fin.is\_open()) { throw std::runtime\_error("File opening error\n"); }

int i = 0;

string line;

while (!fin.eof()) {

getline(fin, line);

}

fin.close();

return line;

}

#endif