Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №1**

По дисциплине «Криптографические методы защиты информации»

Тема: «Основные принципы криптографии»

**Выполнил:**

Студент 2 курса

Группы ИИ-23

Романюк А. П.

**Проверил:**

Хацкевич А. С.

Брест 2024

**Цель:** научиться применять алгоритмы шифрования и сжатия информации.

**Ход работы:**

Шифрование: Гомофоническая замена

#include <iostream>

#include <string>

#include <list>

#include <vector>

#include <map>

using namespace std;

pair<pair<map<char, vector<int>>, map<char, vector<int>>>, vector<int>> encrypt\_key(string open\_text, string fake\_text) {

vector<int> nums;

while (open\_text.size() > fake\_text.size()) {

fake\_text += " ";

}

int index = 0;

for (int i = 0; i < 1000; i++) {

nums.push\_back(i);

}

map<char, vector<int>> key; // Исправлено на map<char, vector<int>>

map<char, vector<int>> fake\_key;

vector<int> text;

for (int i = 0; i < open\_text.size(); i++) {

index = rand() % nums.size();

key[open\_text[i]].push\_back(nums[index]);

fake\_key[fake\_text[i]].push\_back(nums[index]);

text.push\_back(nums[index]);

nums.erase(nums.begin() + index);

}

return make\_pair(make\_pair(key, fake\_key), text);

}

string decrypt\_key(const map<char, vector<int>>& key, const vector<int>& encrypt\_text) {

string open\_text = "";

for (int i = 0; i < encrypt\_text.size(); i++) {

for (auto it = key.begin(); it != key.end(); ++it) {

for (int j = 0; j < it->second.size(); ++j) {

if (encrypt\_text[i] == it->second[j]) {

open\_text += it->first;

break;

}

}

}

}

return open\_text;

}

int main()

{

string open\_text, fake\_text;

cout << "enter text to encrypt\n";

getline(cin, open\_text);

cout << "enter fake text\n";

getline(cin, fake\_text);

pair<pair<map<char, vector<int>>, map<char, vector<int>>>, vector<int>> crypt\_pair = encrypt\_key(open\_text,fake\_text);

map<char, vector<int>> key = crypt\_pair.first.first;

map<char, vector<int>> fake\_key = crypt\_pair.first.second;

vector<int> encrypt\_text = crypt\_pair.second;

// Вывод encrypt\_text

cout << "Encrypted text:" << endl;

for (int i = 0; i < encrypt\_text.size(); ++i) {

cout << encrypt\_text[i] << " ";

}

cout << endl;

// Вывод key

cout << "Key:" << endl;

for (auto it = key.begin(); it != key.end(); ++it) {

cout << it->first << ": ";

for (int i = 0; i < it->second.size(); ++i) {

cout << it->second[i] << " ";

}

cout << endl;

}

// Вывод fake\_key

cout << "Fake\_key:" << endl;

for (auto it = fake\_key.begin(); it != fake\_key.end(); ++it) {

cout << it->first << ": ";

for (int i = 0; i < it->second.size(); ++i) {

cout << it->second[i] << " ";

}

cout << endl;

}

string decrypted\_text = decrypt\_key(key, encrypt\_text);

string fake\_decrypted\_text = decrypt\_key(fake\_key, encrypt\_text);

cout << "Decrypted text: " << decrypted\_text << endl;

cout << "Fake decrypted text: " << fake\_decrypted\_text << endl;

}

Сжатие: LZW

#include <iostream>

#include <unordered\_map>

#include <string>

#include <vector>

// Функция сжатия текста с использованием алгоритма LZW

std::vector<int> compress(const std::string& input) {

std::unordered\_map<std::string, int> dictionary;

for (int i = 0; i < 256; ++i) {

dictionary[std::string(1, i)] = i;

}

std::string current;

std::vector<int> result;

for (char c : input) {

std::string combined = current + c;

if (dictionary.find(combined) != dictionary.end()) {

current = combined;

}

else {

result.push\_back(dictionary[current]);

std::cout << current <<":"<<dictionary[current] << std::endl;

dictionary[combined] = dictionary.size();

current = std::string(1, c);

}

}

if (!current.empty()) {

result.push\_back(dictionary[current]);

}

return result;

}

// Функция декомпрессии текста с использованием алгоритма LZW

std::string decompress(const std::vector<int>& compressed) {

std::unordered\_map<int, std::string> dictionary;

for (int i = 0; i < 256; ++i) {

dictionary[i] = std::string(1, i);

}

std::string previous(1, compressed[0]);

std::string result = previous;

std::string current;

for (int i = 1; i < compressed.size(); ++i) {

int code = compressed[i];

if (dictionary.find(code) != dictionary.end()) {

current = dictionary[code];

}

else if (code == dictionary.size()) {

current = previous + previous[0];

}

else {

throw std::invalid\_argument("Некорректные данные сжатия.");

}

result += current;

dictionary[dictionary.size()] = previous + current[0];

previous = current;

}

return result;

}

int main() {

std::string original\_text = "hi im ho ho ho";

// Сжатие текста

std::vector<int> compressed\_text = compress(original\_text);

std::cout << "Сжатый текст: ";

for (int code : compressed\_text) {

std::cout << code << " ";

}

std::cout << std::endl;

// Декомпрессия текста

std::string decompressed\_text = decompress(compressed\_text);

std::cout << "Декомпрессированный текст: " << decompressed\_text << std::endl;

return 0;

}

**Вывод:** в ходе лабораторной работы я научился шифровать и сжимать информацию.