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-21

Кирилович А. А.

**Проверил:**

Хацкевич М. В.

Брест 2023

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

**Ход работы:**

Шифрование

#include "libs.hh"

void rotate\_matrix(std::vector<std::vector<int>>& matrix, std::vector<std::vector<int>>& rotatedMatrix, int angle) {

int n = matrix.size();

if (angle == 0 || angle % 360 == 0) {

rotatedMatrix = matrix;

return;

}

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

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

rotatedMatrix[i][j] = 0;

if (angle == 90)

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

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

rotatedMatrix[j][n-i-1] = matrix[i][j];

else if (angle == 180)

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

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

rotatedMatrix[n-i-1][n-j-1] = matrix[i][j];

else if (angle == 270)

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

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

rotatedMatrix[n-j-1][i] = matrix[i][j];

}

std::vector<std::vector<int>> make\_grid(int n) {

std::vector<int> indecis(n \* n);

for (int inx = 0; inx < n \* n; inx++) indecis[inx] = inx;

int index, i, j;

std::vector<int> grid\_indecis;

while(!indecis.empty()) {

index = (indecis.size() - indecis.size() % n) / n;

i = indecis[index] / n, j = indecis[index] % n;

int els[] = {

i \* n + j, j \* n + (n - i - 1),

(n - i - 1) \* n + (n - j - 1), (n - j - 1) \* n + i

};

grid\_indecis.push\_back(indecis[index]);

for (auto el : els) indecis.erase(std::remove(indecis.begin(), indecis.end(), el), indecis.end());

}

std::vector<std::vector<int>> grid\_matrix(n, std::vector<int>(n, 0));

for(int g : grid\_indecis) grid\_matrix[g / n][g % n] = 1;

return grid\_matrix;

}

std::string encode\_cardano(std::vector<std::vector<int>> &grid, std::string str) {

std::vector<std::vector<int>> buf\_grid = grid;

int n = grid.size();

std::vector<std::vector<std::string>> encode\_matrix(n, std::vector<std::string>(n));

while (str.length() < n \* n) str.append(" ");

int k = 0;

for (int side = 0; side <= 270; side += 90) {

rotate\_matrix(grid, buf\_grid, side);

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

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

if(buf\_grid[i][j] == 1)

encode\_matrix[i][j] = str[k++];

}

std::string result;

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

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

result += encode\_matrix[i][j];

return result;

}

std::string decode\_cardano(std::vector<std::vector<int>> &grid, std::string str) {

std::vector<std::vector<int>> buf\_grid = grid;

int n = grid.size();

std::vector<std::vector<std::string>> encode\_matrix(n, std::vector<std::string>(n));

int k = 0;

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

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

encode\_matrix[i][j] = str[k++];

std::string result;

for (int side = 0; side <= 270; side += 90) {

rotate\_matrix(grid, buf\_grid, side);

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

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

if(buf\_grid[i][j] == 1)

result += encode\_matrix[i][j];

}

return result;

}

Сжатие

#include "libs.hh"

struct Seg {

char Symbol;

double Left;

double Right;

};

std::vector<Seg> DefineSeg(std::string letters, std::vector<double> probabilities) {

int m = letters.length();

std::vector<Seg> seg(m);

double l = 0;

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

seg[i].Left = l;

seg[i].Right = l + probabilities[i];

seg[i].Symbol = letters[i];

l = seg[i].Right;

}

return seg;

}

double ArithmeticEncoding(std::string letters, std::vector<double> probabilities, std::string str) {

auto seg = DefineSeg(letters, probabilities);

double left = 0, right = 1;

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

int symb = find(letters.begin(), letters.end(), str[i]) - letters.begin();

double newRight = left + (right - left) \* seg[symb].Right;

double newLeft = left + (right - left) \* seg[symb].Left;

left = newLeft;

right = newRight;

}

return (left + right) / 2;

}

std::string ArithmeticDecoding(std::string letters, std::vector<double> probabilities, double code, int n)

{

auto seg = DefineSeg(letters, probabilities);

std::string str = "";

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

for (int j = 0; j < letters.length(); j++) {

if (code >= seg[j].Left && code < seg[j].Right) {

str += seg[j].Symbol;

code = (code - seg[j].Left) / (seg[j].Right - seg[j].Left);

break;

}

}

}

return str;

}

std::vector<double> FindProbability(std::string str, std::string letters) {

std::vector<double> probabilities;

for (char c : letters) {

auto match = std::count(str.begin(), str.end(), c);

probabilities.push\_back((double)match / str.length());

}

return probabilities;

}

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