# Практическая работа № 1

**Шифры перестановки**

**Цели:**

1. Рассмотреть шифр двойной перестановки и выполнить его программную реализацию.
2. Познакомиться с технологией Windows Forms.

**Приложение 1.1. Программная реализация шифра двойной перестановки**

***Задание:*** Разработать приложение с пользовательским интерфейсом, которое позволит осуществлять шифрование и дешифрование текста методом двойной перестановки..

***Константы:*** отсутствуют.

***Переменные:***

1. text – для ввода и вывода текста (тип string)

***Исходные данные:***

1) text – для ввода и вывода текста

***Результат:*** зашифрованное и расшифрованное сообщение

***Код приложения:***

***Листинг 1 – Код приложения MainForm***

namespace DoubleShiftApplication

{

public partial class MainForm : Form

{

DoubleShiftCode code = new DoubleShiftCode();

public MainForm()

{

InitializeComponent();

}

private void encryptButton\_Click(object sender, EventArgs e)

{

string text = sourseTextBox.Text;

if (string.IsNullOrEmpty(text))

{

MessageBox.Show("Пустое исходное сообщение");

return;

}

if (text.Length != 20)

{

MessageBox.Show("Сообщение больше или меньше 20 символов");

return;

}

encryptedTextBox.Text = code.Encode(text);

}

private void decryptButton\_Click(object sender, EventArgs e)

{

string entext = encryptedTextBox.Text;

if (string.IsNullOrEmpty(encryptedTextBox.Text))

{

MessageBox.Show("Пустое исходное сообщение");

return;

}

if (encryptedTextBox.Text.Length != 20)

{

MessageBox.Show("Сообщение больше или меньше 20 символов");

return;

}

decryptedTextBox.Text = code.Decode(encryptedTextBox.Text);

}

}

}

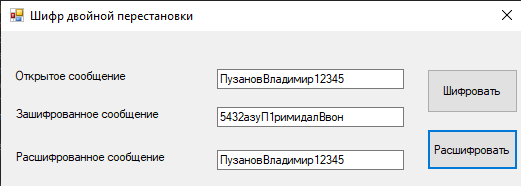


Рисунок 1 – Результат работы приложения MainForm

**Приложение DoubleShiftCode**

***Листинг 2 – Код приложения DoubleShiftCode***

namespace DoubleShiftApplication

{

/// <summary>

/// Класс для выполнения двойной перестановки шифрования и дешифрования текста.

/// </summary>

class DoubleShiftCode

{

private char[][] table;

private int nRows;

private int nCols;

private int[] shiftRow;

private int[] shiftCol;

private Dictionary<int, int> rows;

private Dictionary<int, int> cols;

/// <summary>

/// Рандомизирует перестановку строк и столбцов

/// </summary>

public DoubleShiftCode()

{

nRows = 5;

nCols = 4;

table = new char[nRows][];

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

{

table[i] = new char[nCols];

}

List<int> index = new List<int>(nRows);

Random rnd = new Random(100);

shiftRow = new int[nRows];

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

{

index.Add(i);

}

int m;

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

{

m = rnd.Next(index.Count);

shiftRow[i] = index[m];

index.RemoveAt(m);

}

rows = new Dictionary<int, int>(nRows);

for (int i = 0; i < shiftRow.Length; i++)

{

rows.Add(shiftRow[i], i);

}

index.Clear();

shiftCol = new int[nCols];

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

{

index.Add(i);

}

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

{

m = rnd.Next(index.Count);

shiftCol[i] = index[m];

index.RemoveAt(m);

}

cols = new Dictionary<int, int>(nCols);

for (int i = 0; i < shiftCol.Length; i++)

{

cols.Add(shiftCol[i], i);

}

}

/// <summary>

/// Шифрует входной текст, используя двойную перестановку.

/// </summary>

/// <param name="text">Текст, который необходимо зашифровать.</param>

/// <returns>Зашифрованный текст.</returns>

public string Encode(string text)

{

StringBuilder builder = new StringBuilder();

if (string.IsNullOrEmpty(text))

{

builder.Append("Не задана строка для шифрования.");

return builder.ToString();

}

if (text.Length != nRows \* nCols)

{

builder.Append("Размер сообщения должен быть равен 20.");

return builder.ToString();

}

int k = 0;

int size = nRows \* nCols;

int iRow;

int iCol;

for (int i = 0; i < text.Length; i++)

{

iRow = k / nCols;

iCol = k - iRow \* nCols;

table[iRow][iCol] = text[i];

k++;

}

if (k > 0)

{

builder.Append(TableEncoded());

ClearTable();

}

return builder.ToString();

}

/// <summary>

/// Реализация метода, который переводит таблицу шифрования в строку.

/// </summary>

/// <returns>Зашифрованный текст на основе текущей таблицы.</returns>

public string TableEncoded()

{

StringBuilder builder = new StringBuilder();

char c;

int m;

int n;

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

{

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

{

m = shiftRow[i];

n = shiftCol[j];

c = table[m][n];

builder.Append(c);

}

}

return builder.ToString();

}

/// <summary>

/// Очистка таблицы.

/// </summary>

public void ClearTable()

{

for (int i = 0; i < table.Length; i++)

{

for (int j = 0; j < table[i].Length; j++)

{

table[i][j] = '\0';

}

}

}

/// <summary>

///реализация метода, который будет осуществлять дешифрование сообщения.

/// </summary>

public string Decode(string text)

{

StringBuilder builder = new StringBuilder();

int k = 0;

int size = nRows \* nCols;

int iRow;

int iCol;

for (int i = 0; i < text.Length; i++)

{

iRow = k / nCols;

iCol = k - iRow \* nCols;

table[iRow][iCol] = text[i];

k++;

}

if (k > 0)

{

builder.Append(TableDecode());

ClearTable();

}

return builder.ToString();

}

/// <summary>

///реализация метод TableDecoded(),

///который переводит таблицу дешифрования в строку

/// </summary>

private string TableDecode()

{

StringBuilder builder = new StringBuilder();

char c;

int m;

int n;

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

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

{

m = rows[i];

n = cols[j];

c = table[m][n];

builder.Append(c);

}

return builder.ToString();

}

}

}

Выполнил студент Пузанов В. Е., ФИТУ 010304-КМСб-о22

Проверил ст. преподаватель каф. ПМ Черноиван Д.Н.