МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тихоокеанский государственный университет»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Лабораторная работа №2

По предмету «Защита информации»

«Методы замены»

Выполнил:

студент группы ПО(б) –81

Пшеничный Д. О.

Проверил:

Резак Е. В.

Хабаровск – 2021г.

**Цель:**

Открытый текст - фамилия, ключ - имя.

Произвести шифрование и расшифровывание фамилии с помощью шифра Вижинера и одной из формул Бофорта (расчет вручную, в любой прикладной программе, разработать программу на любом языке программирования).

**Теория:**

Общая формула моноалфавитной замены выглядит в виде

Yi = k1Xi + k2(mod N),

где Yi – i-й символ aлфавитa; k1 и k2 – константы; Xi – i-й символ открытого текста (номер буквы в алфавите); N – длина используемого алфавита.

Шифр, задаваемый фоpмулой

Yi = Xi + ki(mod N),

где ki – i-ая буква ключа, в качестве которого используются слово или фраза, называется шифpом Вижинера

**Ручной расчет:**

**Шифр Вижинера**

Исходный текст: “pshenichny danil olegovich”

Ключ: key

Зашифрованный текст: “zweorfmlkhdakrfvdlvidyzfml”

|  |  |  |  |
| --- | --- | --- | --- |
| Открытый текст | Ключ | Преобразование | Шифр |
| P | K |  | Z |
| S | E |  | W |
| H | Y |  | E |
| E | K |  | O |
| N | E |  | R |
| I | Y |  | F |
| C | K |  | M |
| H | E |  | L |
| N | Y |  | K |
| Y | K |  | H |
|  | E |  | d |
| D | Y |  | A |
| A | K |  | K |
| N | E |  | R |
| I | Y |  | F |
| L | K |  | V |
|  | E |  | D |
| O | Y |  | L |
| L | K |  | V |
| E | E |  | I |
| G | Y |  | D |
| O | K |  | Y |
| V | E |  | Z |
| I | Y |  | F |
| C | K |  | M |
| H | E |  | L |

**Шифр Бофорта**

Исходный текст: “pshenichny danil olegovich”

Ключ: key

Зашифрованный текст: “”

|  |  |  |  |
| --- | --- | --- | --- |
| Открытый текст | Ключ | Преобразование | Шифр |
| P | K |  | Z |
| S | E |  | W |
| H | Y |  | E |
| E | K |  | O |
| N | E |  | R |
| I | Y |  | F |
| C | K |  | M |
| H | E |  | L |
| N | Y |  | K |
| Y | K |  | H |
|  | E |  | d |
| D | Y |  | A |
| A | K |  | K |
| N | E |  | R |
| I | Y |  | F |
| L | K |  | V |
|  | E |  | D |
| O | Y |  | L |
| L | K |  | V |
| E | E |  | I |
| G | Y |  | D |
| O | K |  | Y |
| V | E |  | Z |
| I | Y |  | F |
| C | K |  | M |
| H | E |  | L |

**Код программы:**

using System;

namespace Cipher

{

public class TranspositionCipher

{

private int[] key = null;

public void SetKey(string[] \_key)

{

key = new int[\_key.Length];

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

key[i] = Convert.ToInt32(\_key[i]);

}

public void SetKey(string \_key)

{

SetKey(\_key.Split(' '));

}

public string Encrypt(string input)

{

for (int i = 0; i < input.Length % key.Length; i++)

input += input[i];

string result = "";

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

{

char[] transposition = new char[key.Length];

for (int j = 0; j < key.Length; j++)

transposition[j] = input[i + key[j] - 1];

for (int j = 0; j < key.Length; j++)

result += transposition[j];

}

return result;

}

public string Decrypt(string input)

{

string result = "";

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

{

char[] transposition = new char[key.Length];

for (int j = 0; j < key.Length; j++)

transposition[key[j] - 1] = input[i + j];

for (int j = 0; j < key.Length; j++)

result += transposition[j];

}

return result;

}

}

public class TranspositionCipherTwo

{

private int[] key;

private int[] keyTwo;

public void SetKey(string[] \_key)

{

key = new int[\_key.Length];

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

key[i] = Convert.ToInt32(\_key[i]);

}

public void SetKeyTwo(string[] \_key)

{

keyTwo = new int[\_key.Length];

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

keyTwo[i] = Convert.ToInt32(\_key[i]);

}

public string ExtendString(string value, int length)

{

while(value.Length < length)

{

value += "\_";

}

return value;

}

public string Encrypt(string input)

{

char[,] intermediantResult = new char[key.Length,keyTwo.Length];

int rows = intermediantResult.GetUpperBound(0) + 1;

int columns = intermediantResult.Length / rows;

input = ExtendString(input, rows \* columns);

int cnt = 0;

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

{

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

{

intermediantResult[key[i] - 1, j] = input[cnt];

++cnt;

}

}

Console.WriteLine("Таблица:");

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

{

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

{

Console.Write(intermediantResult[i, j] + " ");

}

Console.WriteLine();

}

string result = "";

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

{

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

{

result += intermediantResult[j, keyTwo[i] - 1];

}

}

return result;

}

public string Decrypt(string input)

{

string result = "";

char[,] intermediantResult = new char[key.Length, keyTwo.Length];

int rows = intermediantResult.GetUpperBound(0) + 1;

int columns = intermediantResult.Length / rows;

int cnt = 0;

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

{

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

{

intermediantResult[j, keyTwo[i] - 1] = input[cnt];

++cnt;

}

}

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

{

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

{

result += intermediantResult[key[i] - 1, j];

}

}

return result;

}

}

public class CaesarCipher

{

//символы русской азбуки

const string alfabet = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

private string CodeEncode(string text, int k)

{

//добавляем в алфавит маленькие буквы

var fullAlfabet = alfabet + alfabet.ToLower();

var letterQty = fullAlfabet.Length;

var retVal = "";

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

{

var c = text[i];

var index = fullAlfabet.IndexOf(c);

if (index < 0)

{

//если символ не найден, то добавляем его в неизменном виде

retVal += c.ToString();

}

else

{

var codeIndex = (letterQty + index + k) % letterQty;

retVal += fullAlfabet[codeIndex];

}

}

return retVal;

}

//шифрование текста

public string Encrypt(string plainMessage, int key)

=> CodeEncode(plainMessage, key);

//дешифрование текста

public string Decrypt(string encryptedMessage, int key)

=> CodeEncode(encryptedMessage, -key);

}

class Program

{

static void Main(string[] args)

{

//OneKey();

//TwoKey();

Shiza();

}

static void OneKey()

{

var simpleCipher = new TranspositionCipher();

Console.Write("Введите текст: ");

var messageForSimpleCipher = Console.ReadLine();

Console.Write("Введите ключ: ");

var secretKeyForSimpleCipher = Console.ReadLine();

simpleCipher.SetKey(secretKeyForSimpleCipher);

var encryptedTextForSimpleCipher = simpleCipher.Encrypt(messageForSimpleCipher);

Console.WriteLine("Зашифрованное сообщение: {0}", encryptedTextForSimpleCipher);

Console.WriteLine("Расшифрованное сообщение: {0}", simpleCipher.Decrypt(encryptedTextForSimpleCipher));

Console.ReadLine();

}

static void TwoKey()

{

var simpleCipher = new TranspositionCipherTwo();

Console.Write("Введите текст: ");

var messageForSimpleCipher = Console.ReadLine();

Console.Write("Введите первый ключ: ");

var secretKey = Console.ReadLine();

Console.Write("Введите второй ключ: ");

var secretKeyTwo = Console.ReadLine();

simpleCipher.SetKey(secretKey.Split(' '));

simpleCipher.SetKeyTwo(secretKeyTwo.Split(' '));

var encryptedTextForSimpleCipher = simpleCipher.Encrypt(messageForSimpleCipher);

Console.WriteLine("Зашифрованное сообщение: {0}", encryptedTextForSimpleCipher);

Console.WriteLine("Расшифрованное сообщение: {0}", simpleCipher.Decrypt(encryptedTextForSimpleCipher));

Console.ReadLine();

}

static void Shiza()

{

var cipher = new CaesarCipher();

Console.Write("Введите текст: ");

var message = Console.ReadLine();

Console.Write("Введите ключ: ");

var secretKey = Convert.ToInt32(Console.ReadLine());

var encryptedText = cipher.Encrypt(message, secretKey);

Console.WriteLine("Зашифрованное сообщение: {0}", encryptedText);

Console.WriteLine("Расшифрованное сообщение: {0}", cipher.Decrypt(encryptedText, secretKey));

Console.ReadLine();

}

}

}

**Вывод:** в ходе выполнения лабораторной работы были изучены алгоритмы шифрования перестановкой.