# ЛАБОРАТОРНА РОБОТА 1

**ШИФРИ ЗАМІНИ**

**Мета роботи:** вивчити теоретичні основи побудови шифрів заміни, провести зашифрування відкритого і розшифрування шифрованого повідомлення. Усвідомити сильні і слабкі сторони шифрів заміни.

## **Короткі теоретичні відомості**

Шифри заміни (підстановки) – це алгоритми шифрування, при якому кожен знак вихідного тексту заміняється шифропозначенням – одним, або декількома знаками деякого набору символів (алфавіту).

Шифри заміни поділяються на моноалфавітні та поліалфавітні. У моноалфавітних шифрах підстановки літера (або символ) у початковому тексті завжди змінюється на ту саму літеру (або символ) у зашифрованому тексті незалежно від її позиції в тексті. Очевидні слабкі сторони шифрів моноалфавітної підстаноівки – у шифрованому тексті зберігаються всі частотні характеристики відкритого тексту, усі сполучення і повторення.

Найпростіший моноалфавітний шифр підстановки — адитивний шифр, його іноді називають шифром зсуву, а іноді — шифром Цезаря, але термін “адитивний шифр” краще показує його математичний сенс. Зашифрування текстових повідомлень за допомогою адитивних шифрів підстановки здійснюється за допомогою формули:

*довжина алфавіту***; *М* –** *відкритий текст***; С-** *шифртекст*;

***K –*** *ключ (ціле число в* ***)***

Розшифрування здійснюється відповідно за формулою:

Зашифрування текстових повідомлень за допомогою *мультипліктивних* шифрів заміни здійснюється за формулою:

а розшифрування відповідно за формулою:

**Хід роботи**

1. За допомогою афінного алгоритму шифрування та даних ключів (таблиця 1) зашифрувати своє прізвище. Вміст файлу AffineCipher.cs наведений у лістингу

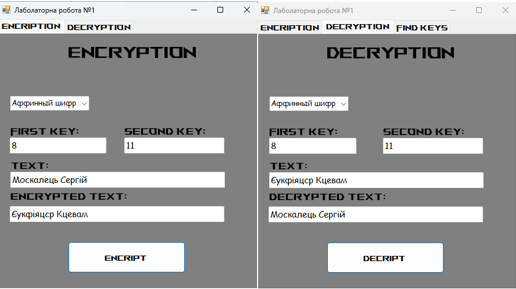


Рисунок 1.1 – Зашифроване повідомлення, розшифроване повідомлення

Лістинг 1 – вміст файлу AffineCipher.cs

public class AffineCipher

{

static string UkrainianAlphabet = "абвгґдеєжзиіїйклмнопрстуфхцчшщьюя";

// Функція для шифрування тексту

public static string Encrypt(string text, int a, int b)

{

string result = "";

foreach (char ch in text)

{

if (UkrainianAlphabet.Contains(Char.ToLower(ch)))

{

int index = UkrainianAlphabet.IndexOf(Char.ToLower(ch));

int encryptedIndex = (a \* index + b) % UkrainianAlphabet.Length;

char encryptedChar = UkrainianAlphabet[encryptedIndex];

if (Char.IsUpper(ch))

result += Char.ToUpper(encryptedChar);

else

result += encryptedChar;

}

else

{

result += ch;

}

}

return result;

}

// Функція для розшифрування тексту

public static string Decrypt(string cipherText, int a, int b)

{

// Обернена для множення за модулем довжини алфавіту

int aInverse = 0;

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

{

if ((a \* i) % UkrainianAlphabet.Length == 1)

{

aInverse = i;

break;

}

}

string result = "";

foreach (char ch in cipherText)

{

if (UkrainianAlphabet.Contains(Char.ToLower(ch)))

{

int index = UkrainianAlphabet.IndexOf(Char.ToLower(ch));

int decryptedIndex = (aInverse \* (index - b + UkrainianAlphabet.Length)) % UkrainianAlphabet.Length;

char decryptedChar = UkrainianAlphabet[decryptedIndex];

if (Char.IsUpper(ch))

result += Char.ToUpper(decryptedChar);

else

result += decryptedChar;

}

else

{

result += ch;

}

}

return result;

}

}

1. Розшифруйте шифртекст, отриманий за допомогою афінного алгоритму шифрування (ключі шифру відомі).

Таблиця 1

|  |  |  |  |
| --- | --- | --- | --- |
| Варіант | К1 | К2 | Шифтекст |
| 3 | 8 | 11 | КУЛ |

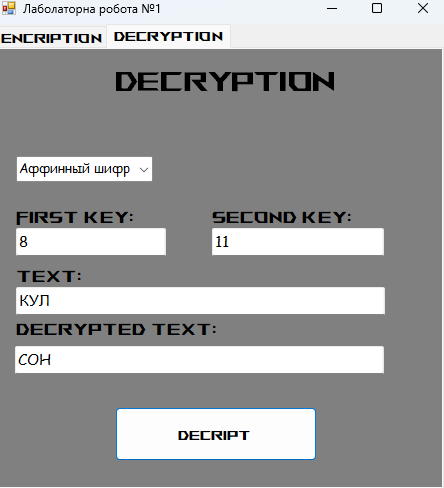


Рисунок 1.2 – розшифрований шифртекст

1. Відома пара відкритий текст-шифртекст (біграми). Відновіть ключі та запишіть формули, за якими відбувається шифрування і розшифрування. Функція пошуку ключів наведена в лістингу 2.

Таблиця 2

|  |  |  |
| --- | --- | --- |
| Варіант | Відкритий текст | Шифртекст |
| 3 | Ло | ВЖ |

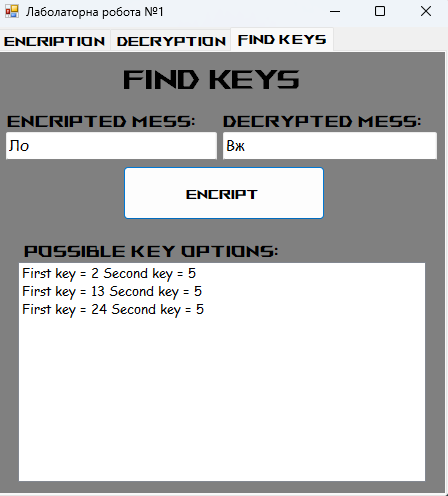


Рисунок 1.3 – пошук ключів

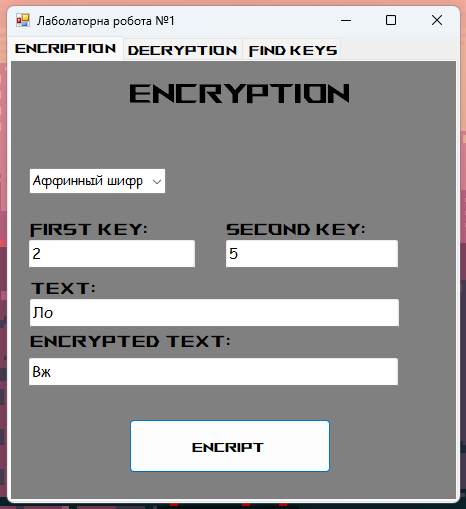


Рисунок 1.4 – перевірка ключів

Лістинг 2 – функція пошуку ключів

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

{

string enMess;

string decriptMess = decryptedMessBox.Text.ToLower();

string encriptMess = encriptedMessBox.Text.ToLower();

bool flag = false;

listBoxKeys.Items.Clear();

for (int i = 1; i < 34; i++)

{

for (int j = 0; j < 34; j++)

{

enMess = AffineCipher.Encrypt(encriptMess, i, j);

if (enMess == decriptMess.ToLower())

{

listBoxKeys.Items.Add("First key = " + i + " Second key = " + j);

break;

}

}

}

if (listBoxKeys.Items.Count == 0)

{

listBoxKeys.Items.Add("No keys were found for this encryption");

}

}

1. За допомогою шифру Віженера та ключового слова зашифруйте фразу з книги Л. Керрола « Аліса в країні чудес». Вміст файлу VigenèreCipher.cs наведений у лістингу 3.

Таблиця 3

|  |  |  |
| --- | --- | --- |
|  | **Фраза для шифрування** | Ключ |
| 3 | Чого варта книжка без малюнків та розмов | Шифр |

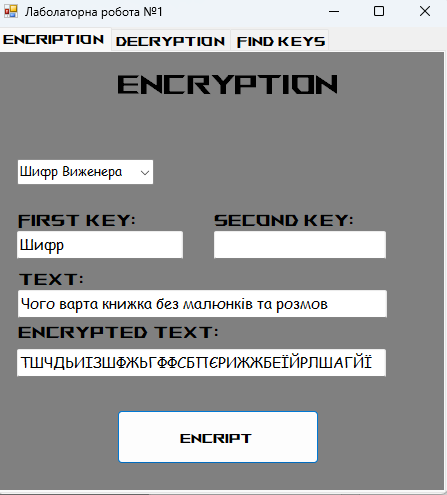


Рисунок 1.5 – Шифр Віженера зашифроване повідомлення

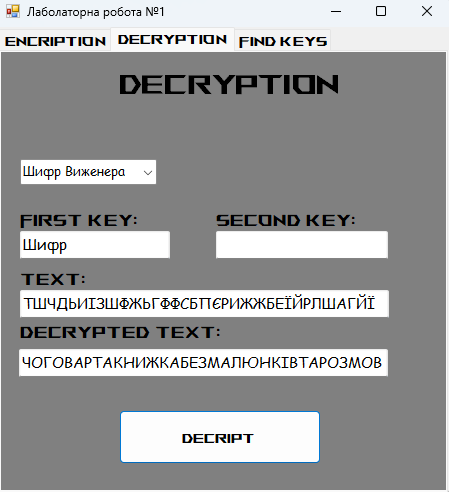


Рисунок 1.6 – Шифр Віженера розшифроване повідомлення

Лістинг 3 - Вміст файлу VigenèreCipher.cs

internal class VigenèreCipher

{

static string alphabet = "АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ";

public static string Decript(string criptMessage, string key)

{

int M, C, K;

string decript = null;

key = KeyModify(key, criptMessage);

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

{

C = alphabet.IndexOf(criptMessage[i]);

K = alphabet.IndexOf(key[i]);

M = (C - K + 33) % 33;

decript += alphabet[M];

}

return decript;

}

public static string Encript(string message, string key)

{

int C, P, K;

string cript = null;

key = KeyModify(key, message);

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

{

P = alphabet.IndexOf(message[i]);

K = alphabet.IndexOf(key[i]);

C = (P + K) % 33;

cript += alphabet[C];

}

return cript;

}

static string KeyModify(string key, string message)

{

while (key.Length != message.Length)

{

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

{

key += key[i];

if (message.Length == key.Length)

break;

}

}

return key;

}

}

1. Зашифруйте фразу з попереднього завдання за допомогою шифру Плейфера. Вміст файлу PleyfraCipher.cs наведений у лістингу 4.

Таблиця 4

|  |  |  |
| --- | --- | --- |
|  | **Фраза для шифрування** | Ключ |
| 3 | Чого варта книжка без малюнків та розмов | Шифр |



Рисунок 1.7 – Шифр Плейфера зашифроване повідомлення

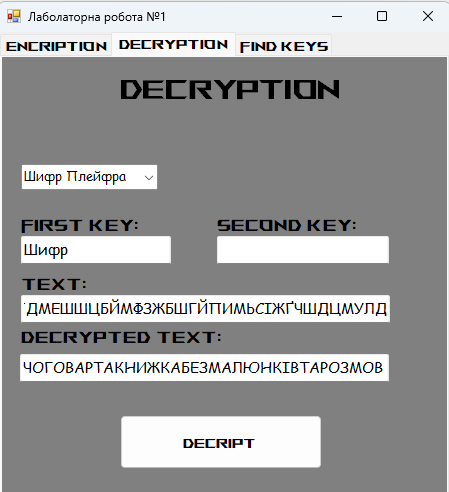


Рисунок 1.7 – Шифр Плейфера розшифроване повідомлення

Лістинг 4 – вміст файлу PleyfraCipher.cs.

internal class PleyfraCipher

{

static string alphabet = "АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ";

static char[,] tablePleyfra = new char[6, 6];

public static string DecriptPleyfraMessage(string encriptMessage, string key)

{

key = RemoveDuplicates(key);

alphabet = ModificateAplhabet(alphabet, key);

FillTable(tablePleyfra, alphabet);

string decriptMessage = null;

int rowIndex = -1, colIndex = -1,

rowIndex1, rowIndex2, colIndex1, colIndex2;

char firstEl, secondEl;

int num = encriptMessage.Length / 2;

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

{

firstEl = encriptMessage[0];

secondEl = encriptMessage[1];

FindElIndex(tablePleyfra, firstEl, ref rowIndex, ref colIndex);

rowIndex1 = rowIndex;

colIndex1 = colIndex;

FindElIndex(tablePleyfra, secondEl, ref rowIndex, ref colIndex);

rowIndex2 = rowIndex;

colIndex2 = colIndex;

encriptMessage = encriptMessage.Substring(2);

if ((rowIndex1 == rowIndex2) && (colIndex1 != colIndex2)) //

{

if (colIndex1 == 0)

colIndex1 = 6;

else if (colIndex2 == 0)

colIndex2 = 6;

decriptMessage += (tablePleyfra[rowIndex1, colIndex1 - 1].ToString() + tablePleyfra[rowIndex2, colIndex2 - 1]);

}

else if ((colIndex1 == colIndex2) && (rowIndex1 != rowIndex2)) //

{

if (rowIndex1 == 0)

rowIndex1 = 6;

else if (rowIndex2 == 0)

rowIndex2 = 6;

decriptMessage += (tablePleyfra[rowIndex1 - 1, colIndex1].ToString() + tablePleyfra[rowIndex2 - 1, colIndex2]);

}

else if ((rowIndex1 != rowIndex2) && (colIndex2 != colIndex1)) //

{

decriptMessage += (tablePleyfra[rowIndex1, colIndex2].ToString() + tablePleyfra[rowIndex2, colIndex1]);

}

}

return decriptMessage;

}

static void ShowTable(char[,] table)

{

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

{

for (int j = 0; j < table.GetLength(1); j++)

{

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

}

Console.WriteLine();

}

}

public static string EncriptPleyfra(string message, string key)

{

key = RemoveDuplicates(key);

alphabet = ModificateAplhabet(alphabet, key);

FillTable(tablePleyfra, alphabet);

message = CheckForDuplicates(message);

if (message.Length % 2 != 0)

message += "Ь";

int rowIndex = -1, colIndex = -1,

rowIndex1, rowIndex2, colIndex1, colIndex2;

char firstEl, secondEl;

string encriptMessage = null;

int num = message.Length / 2;

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

{

firstEl = message[0];

secondEl = message[1];

FindElIndex(tablePleyfra, firstEl, ref rowIndex, ref colIndex);

rowIndex1 = rowIndex;

colIndex1 = colIndex;

FindElIndex(tablePleyfra, secondEl, ref rowIndex, ref colIndex);

rowIndex2 = rowIndex;

colIndex2 = colIndex;

message = message.Substring(2);

if ((rowIndex1 == rowIndex2) && (colIndex1 != colIndex2)) // переход по стовбцям

{

if (colIndex1 == 5)

colIndex1 = -1;

else if (colIndex2 == 5)

colIndex2 = -1;

encriptMessage += (tablePleyfra[rowIndex1, colIndex1 + 1].ToString() + tablePleyfra[rowIndex2, colIndex2 + 1]);

}

else if ((colIndex1 == colIndex2) && (rowIndex1 != rowIndex2)) // переход по строкам

{

if (rowIndex1 == 5)

rowIndex1 = -1;

else if (rowIndex2 == 5)

rowIndex2 = -1;

encriptMessage += (tablePleyfra[rowIndex1 + 1, colIndex1].ToString() + tablePleyfra[rowIndex2 + 1, colIndex2]);

}

else if ((rowIndex1 != rowIndex2) && (colIndex2 != colIndex1)) // переход по стовбцям

{

encriptMessage += (tablePleyfra[rowIndex1, colIndex2].ToString() + tablePleyfra[rowIndex2, colIndex1]);

}

}

return encriptMessage;

}

static void FillTable(char[,] tablePleyfra, string alphabet)

{

int count = 0;

char[] symbol = { '@', '#', '$' };

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 6; j++)

{

if (count <= 32)

{

tablePleyfra[i, j] = alphabet[count];

}

else

{

tablePleyfra[i, j] = symbol[count - 33];

}

count++;

}

}

}

static void FindElIndex(char[,] Array, char el, ref int rowIndex, ref int colIndex)

{

bool found = false;

for (int i = 0; i < Array.GetLength(0); i++)

{

for (int j = 0; j < Array.GetLength(1); j++)

{

//

if (Array[i, j] == el)

{

rowIndex = i;

colIndex = j;

found = true;

break;

}

}

if (found)

{

break;

}

}

}

static string CheckForDuplicates(string str)

{

for (int i = 0; i < str.Length - 1; i++)

{

char currentChar = str[i];

if (currentChar == str[i + 1])

{

str = str.Insert(i + 1, "Ь"); //

}

}

return str; //

}

static string ModificateAplhabet(string alphabet, string key)

{

int i;

StringBuilder sb = new StringBuilder(alphabet);

foreach (char c in key)

{

i = sb.ToString().IndexOf(c);

if (i != -1)

{

sb.Remove(i, 1);

}

}

return key + sb.ToString();

}

static string RemoveDuplicates(string str)

{

StringBuilder builder = new StringBuilder();

foreach (char c in str)

{

//

if (builder.ToString().IndexOf(c) == -1)

{

builder.Append(c);

}

}

return builder.ToString();

}

}

**Висновок**

На даній лаболаторній роботі ми вивчили теоретичні основи побудови шифрів заміни, провесли зашифрування відкритого і розшифрування шифрованого повідомлення. Усвідомили сильні і слабкі сторони шифрів заміни.