МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

кафедра програмних засобів

Звіт

з лабораторної роботи № 1

з дисципліни «Безпека програм та даних»

на тему:

«Базові шифри. Частотний криптоаналіз»

Виконав

Ст. гр. КНТ-137 В.В. Козлов

Прийняв

Ст. викладач О.І. Качан

1. Мета роботи

Ознайомитися з базовими шифрами. Розглянути методику частотного криптоаналізу.

1. Завдання на лабораторну роботу

Розробити програми шифрування та дешифрування одним з наступних шифрів на вибір викладача:

* шифр простої заміни;
* квадрат Полібія;
* шифр Тритемія;
* шифр Віжинера;
* шифр перестановки.

Виконати частотний криптоаналіз отриманих зашифрованих текстів.

1. Результати виконання роботи

Протягом виконання цієї лабораторної роботи було розроблено програму на С# з графічним інтерфейсом, яка надає можливість зашифровування та розшифровування даних за такими шифрами: шифр Цезаря, квадрат Полібія, шифр Тритемія, шифр Віжинера, шифр перестановки. Також реалізовано метод частотного криптоаналізу для розшифровування зашифрованих даних. Надано можливість завантаження даних з файлу та збереження отриманих результатів у файл.

Реалізовані шифри та метод частотного криптоаналізу було виділено в окремий файл BaseCiphers.cs з класом BaseCiphers.

* 1. Опис файлу BaseCiphers.cs

Файл BaseCiphers.cs складається з одного класу – BaseCiphers. Клас BaseCiphers – це клас, який реалізує всі зазначені шифри та метод частотного криптоаналізу.

Поля класу BaseCiphers наведено у таблиці 3.1.

Таблиця 3.1 – Поля класу BaseCiphers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва поля | Призначення | Тип даних | Модифікатор доступу |
| 1 | EnglishAlphabet | Зберігає всі літери англійського алфавіту, що використовуються | string | private |
| 2 | Numbers | Зберігає цифри, що використовуються | string | private |
| 3 | Signs | Зберігає знаки, що використовуються | string | private |
| 4 | FullAlphabet | Весь список символів, що використовуються | string | private |
| 5 | FrequencyDictionary | Таблиця частот символів, що використовуються | Dictionary<char, double> | private |

Методи класу BaseCiphers наведено у таблиці 3.2.

Таблиця 3.2 – Методи класу BaseCiphers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва методу | Призначення | Тип даних | Модифікатор доступу |
| 1 | CaesarsCipher | Реалізація шифру Цезаря | string | public |
| 2 | CaesarsCipherUniversal | Реалізація шифру Цезаря універсальна | string | public |
| 3 | GetApproximateMaximumSCRT | Отримання значення для шифру Полібія | int | private |
| 4 | PolybiusSquareCipher | Реалізація шифру квадрат Полібія | string | public |
| 5 | TritemiusCipher | Реалізація шифру Тритемія | string | public |
| 6 | GetRepeatKey | Збільшення ключа до заданої довжини | string | private |
| 7 | VigenereCipher | Реалізація шифру Віжинера | string | public |
| 8 | XORCipher | Реалізація шифру XOR | string | public |
| 9 | RearrangementCipherByKey | Реалізація шифру перестановки | string | public |
| 10 | DictionaryToList | Перетворення типів даних | List<Tuple<char, double>> | private |
| 11 | SortListByFrequency | Сортування за частотою | List<Tuple<char, double>> | private |
| 12 | SimpleFrequencyAnalysis | Реалізація методу частотного криптоаналізу | string | public |

* 1. Лістинг файлу BaseCiphers.cs

using System;

using System.Collections.Generic;

using System.Linq;

namespace Lab1

{

class BaseCiphers

{

static private readonly string EnglishAlphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

static private readonly string Numbers = "0123456789";

static private readonly string Signs = " !\"#$%&'()\*+,-./:;<=>?@[\\]^\_`{|}~";

static private readonly string FullAlphabet = Numbers + EnglishAlphabet + EnglishAlphabet.ToLower() + Signs;

static private readonly Dictionary<char, double> FrequencyDictionary = new Dictionary<char, double>

{

{ 'a', 0.0856}, { 'b', 0.0139}, { 'c', 0.0279}, { 'd', 0.0378},

{ 'e', 0.1304}, { 'f', 0.0289}, { 'g', 0.0199}, { 'h', 0.0528},

{ 'i', 0.0627}, { 'j', 0.0013}, { 'k', 0.0042}, { 'l', 0.0339},

{ 'm', 0.0249}, { 'n', 0.0707}, { 'o', 0.0797}, { 'p', 0.0199},

{ 'q', 0.0012}, { 'r', 0.0677}, { 's', 0.0607}, { 't', 0.1045},

{ 'u', 0.0249}, { 'v', 0.0092}, { 'w', 0.0149}, { 'x', 0.0017},

{ 'y', 0.0199}, { 'z', 0.0008}

};

static public string CaesarsCipher(string data, int shift, bool encode = true)

{

string result = "";

int charactersQuantityInAlphabet = FullAlphabet.Length;

shift %= charactersQuantityInAlphabet;

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

{

char currentLetter = data[i];

int encodeLetterIndex = FullAlphabet.IndexOf(currentLetter);

if (encodeLetterIndex >= 0)

{

int codeIndex = (charactersQuantityInAlphabet + (encodeLetterIndex + (encode ? 1 : -1) \* shift) % charactersQuantityInAlphabet) % charactersQuantityInAlphabet;

result += FullAlphabet[codeIndex];

}

else result += currentLetter;

}

return result;

}

static public string CaesarsCipherUniversal(string data, int shift, bool encode = true)

{

string result = "";

foreach(char ch in data)

result += (char)(ch + (encode ? 1 : -1) \* shift);

return result;

}

static private int GetApproximateMaximumSCRT(int number)

{

number = number > 0 ? number : -number;

int result = 1;

for (; result \* result < number; ++result) ;

return result;

}

static public string PolybiusSquareCipher(string data, string key, bool encode = true)

{

string result = "";

string alphabet = FullAlphabet;

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

{

if (alphabet.IndexOf(key[i]) >= 0)

{

alphabet = alphabet.Replace(key[i].ToString(), "");

++i;

}

else key = key.Remove(i, 1);

}

alphabet = key + alphabet;

int charactersQuantityInAlphabet = alphabet.Length;

int shift = GetApproximateMaximumSCRT(charactersQuantityInAlphabet);

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

{

char currentLetter = data[i];

int encodeLetterIndex = alphabet.IndexOf(currentLetter);

if (encodeLetterIndex >= 0)

{

int codeIndex;

if (encode) codeIndex = (encodeLetterIndex + shift) >= charactersQuantityInAlphabet ? encodeLetterIndex % shift : encodeLetterIndex + shift;

else

{

if ((encodeLetterIndex - shift) < 0)

{

codeIndex = encodeLetterIndex;

while (codeIndex + shift < charactersQuantityInAlphabet) codeIndex += shift;

}

else codeIndex = encodeLetterIndex - shift;

}

result += alphabet[codeIndex];

}

else result += currentLetter;

}

return result;

}

static public string TritemiusCipher(string data, int key, bool encode = true)

{

string result = "";

int charactersQuantityInAlphabet = FullAlphabet.Length;

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

{

char currentLetter = data[i];

int encodeLetterIndex = FullAlphabet.IndexOf(currentLetter);

if (encodeLetterIndex >= 0)

{

int codeIndex = (charactersQuantityInAlphabet + (encodeLetterIndex + (encode ? -1 : 1) \* i + (encode ? -1 : 1) \* key) % charactersQuantityInAlphabet) % charactersQuantityInAlphabet;

result += FullAlphabet[codeIndex];

}

else result += currentLetter;

}

return result;

}

static private string GetRepeatKey(string key, int size)

{

string repeatKey = key;

while (repeatKey.Length < size)

repeatKey += key;

repeatKey = repeatKey.Substring(0, size);

return repeatKey;

}

static public string VigenereCipher(string data, string key, bool encode = true)

{

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

if (FullAlphabet.IndexOf(key[i]) >= 0) ++i;

else key = key.Remove(i, 1);

if (key.Length == 0) return data;

string repeatKey = GetRepeatKey(key, data.Length);

string result = "";

int charactersQuantityInAlphabet = FullAlphabet.Length;

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

{

char currentLetter = data[i];

int encodeLetterIndex = FullAlphabet.IndexOf(currentLetter);

int codeIndex = FullAlphabet.IndexOf(repeatKey[i]);

if(encodeLetterIndex >= 0 && codeIndex >= 0)

{

result += FullAlphabet[(charactersQuantityInAlphabet + encodeLetterIndex + (encode ? 1 : -1) \* codeIndex) % charactersQuantityInAlphabet];

}

else result += currentLetter;

}

return result;

}

static public string XORCipher(string data, string key)

{

string repeatKey = GetRepeatKey(key, data.Length);

string result = "";

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

{

result += (char)(data[i] ^ repeatKey[i]);

}

return result;

}

static public string RearrangementCipherByKey(string data, string key, bool encode = true)

{

if (key.Length == 0) return data;

string result = "";

string sortKey = "";

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

{

if (key.IndexOf(key[i]) != i) key = key.Remove(i, 1);

else

{

int positionToInsert = 0;

for (; positionToInsert < i; ++positionToInsert)

if (sortKey[positionToInsert] > key[i]) break;

sortKey = sortKey.Insert(positionToInsert, key[i].ToString());

++i;

}

}

int charactersQuantityInKey = key.Length;

int shift = 0;

while (shift < data.Length)

{

if (encode)

{

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

{

if (shift + key.IndexOf(sortKey[i]) < data.Length)

result += data[shift + key.IndexOf(sortKey[i])];

}

}

else

{

if (shift + key.Length > data.Length)

{

for (int j = data.Length - shift; j < key.Length; ++j)

sortKey = sortKey.Remove(sortKey.IndexOf(key[j]), 1);

}

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

result += data[shift + sortKey.IndexOf(key[i])];

}

shift += key.Length;

}

return result;

}

static private List<Tuple<char, double>> DictionaryToList(Dictionary<char, double> dict)

{

List<Tuple<char, double>> list = new List<Tuple<char, double>>();

foreach (char key in dict.Keys)

list.Add(new Tuple<char, double>(key, dict[key]));

return list;

}

static private List<Tuple<char, double>> SortListByFrequency(List<Tuple<char, double>> list)

{

for (int i = 0; i < list.Count; ++i)

for (int j = list.Count - 1; j > i; --j)

if (list[j].Item2 > list[j - 1].Item2)

{

Tuple<char, double> temp = list[j];

list[j] = list[j - 1];

list[j - 1] = temp;

}

return list;

}

static public string SimpleFrequencyAnalysis(string data)

{

string resData = "";

if (data.Length < 1 || data == null) return "";

int encodedCharactersQuantityInAlphabet = 0;

Dictionary<char, double> DataCharFrequency = new Dictionary<char, double>();

foreach (char ch in data)

{

if (DataCharFrequency.ContainsKey(ch))

++DataCharFrequency[ch];

else DataCharFrequency[ch] = 1;

++encodedCharactersQuantityInAlphabet;

}

foreach (char key in DataCharFrequency.Keys.ToList())

DataCharFrequency[key] = DataCharFrequency[key] / encodedCharactersQuantityInAlphabet;

List<Tuple<char, double>> sortedDataCharFrequency = SortListByFrequency(DictionaryToList(DataCharFrequency));

List<Tuple<char, double>> sortedCharFriquency = SortListByFrequency(DictionaryToList(FrequencyDictionary));

Dictionary<char, char> encoder = new Dictionary<char, char>();

for (int i = 0; i < (sortedDataCharFrequency.Count > sortedCharFriquency.Count ? sortedCharFriquency.Count : sortedDataCharFrequency.Count); ++i)

encoder[sortedDataCharFrequency[i].Item1] = sortedCharFriquency[i].Item1;

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

if (encoder.ContainsKey(data[i])) resData += encoder[data[i]];

else resData += data[i];

return resData;

}

}

}

* 1. Графічний інтерфейс

Взаємодія користувача з програмою, має бути реалізована завдяки графічному користувацькому інтерфейсу. Графічний користувацький інтерфейс реалізовано за допомогою WPF (Windows Presentation Foundation). WPF – це основна система платформи .Net, що реалізує шаблон проектування MVVM (Model View View-Model) (рис. 3.1). В основі WPF лежить використання мови розмітки XAML, яка використовується для побудови графічного інтерфейсу.



Рисунок 3.1 – Графічне представлення шаблону MVVM

Графічний інтерфейс описано у файлі MainWindow.xaml, а код його взаємодії у файлі MainWindow.xaml.cs.

* + 1. Лістинг файлу MainWindow.xaml

<Window x:Class="Lab1.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Lab1"

mc:Ignorable="d"

Title="Base ciphers" Height="450" Width="550" Icon="/Resources/lock.ico">

<Window.Resources>

<SolidColorBrush x:Key="scbMainBackgroud">#D3D3D3</SolidColorBrush>

</Window.Resources>

<Border Background="{StaticResource scbMainBackgroud}">

<Border Margin="10" Background="White" CornerRadius="10">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<GroupBox Margin="10" Header="Method">

<ComboBox Name="cbMethod" Margin="10" ItemsSource="{Binding Methods, UpdateSourceTrigger=PropertyChanged}" SelectedItem="{Binding SelectedMethod, UpdateSourceTrigger=PropertyChanged}"/>

</GroupBox>

<GroupBox Grid.Column="1" Margin="10" Header="Key">

<TextBox Margin="10" Text="{Binding Key, UpdateSourceTrigger=PropertyChanged}"/>

</GroupBox>

</Grid>

<GroupBox Grid.Row="1" Margin="10" Header="Working with text">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1.5\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1.5\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Margin="15 10 10 0" Text="Source text"/>

<TextBox Grid.Row="1" Margin="10 5 10 10" AcceptsReturn="True" Text="{Binding SourceText, UpdateSourceTrigger=PropertyChanged}"/>

<StackPanel Grid.Column="1" Grid.Row="1" HorizontalAlignment="Stretch" VerticalAlignment="Center">

<Button Height="32" Margin="10" Content="Encode" Command="{Binding EncodeCommand}"/>

<Button Height="32" Margin="10" Content="Decode" Command="{Binding DecodeCommand}"/>

</StackPanel>

<TextBlock Grid.Column="2" Margin="15 10 10 0" Text="Modified text"/>

<TextBox Grid.Column="2" Grid.Row="1" Margin="10 5 10 10" AcceptsReturn="True" Text="{Binding ModifiedText, UpdateSourceTrigger=PropertyChanged}"/>

</Grid>

</GroupBox>

<GroupBox Grid.Row="3" Margin="10" Header="Working with file">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Button Height="32" Margin="10" Content="Load from file" Command="{Binding LoadFromFileCommand}"/>

<Button Grid.Column="1" Height="32" Margin="10" Content="Save to file" Command="{Binding SaveToFileCommand}"/>

</Grid>

</GroupBox>

</Grid>

</Border>

</Border>

</Window>

* + 1. Лістинг файлу MainWindow.xaml.cs

using System.Windows;

namespace Lab1

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

DataContext = new MainWindowViewModel();

}

}

}

* + 1. Стилізація

Для стилізації програми було використано колір #D3D3D3, а в якості іконки програми зображення навісного замка (рис. 3.2).

C:\Users\Valeriy\Desktop\lock.png

Рисунок 3.2 – Іконка програми

* 1. Реалізація програми

Для реалізації шаблону MVVM було створено основний клас MainWindowViewModel, який зберігається в окремому файлі MainWindowViewModel.cs., що реалізує логіку програми та допоміжний клас RelayCommand, який зберігається в окремому файлі RelayCommand.cs.

* + 1. Лістинг файлу RelayCommand.cs

using System;

using System.Windows.Input;

namespace Lab1

{

class RelayCommand : ICommand

{

readonly Action<object> execute;

readonly Predicate<object> canExecute;

public RelayCommand(Action<object> execute, Predicate<object> canExecute = null)

{

this.execute = execute ?? throw new ArgumentNullException("execute");

this.canExecute = canExecute;

}

public event EventHandler CanExecuteChanged

{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

public bool CanExecute(object parameter)

{

return canExecute == null ? true : canExecute(parameter);

}

public void Execute(object parameter)

{

execute(parameter);

}

}

}

* + 1. Лістинг файлу MainWindowViewModel.cs

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.IO;

using System.Runtime.CompilerServices;

using System.Windows;

namespace Lab1

{

class MainWindowViewModel : INotifyPropertyChanged

{

static public List<string> Methods

{

get

{

return new List<string>() { "Caesars cipher", "Polybius square cipher", "Tritemius cipher", "Vigenere cipher", "Rearrangement cipher", "Simple frequency analysis" };

}

set

{

;

}

}

private string selectedMethod = Methods[0];

public string SelectedMethod

{

get { return selectedMethod; }

set

{

selectedMethod = value;

OnPropertyChanged("SelectedMethod");

}

}

private string key = "";

public string Key

{

get { return key; }

set

{

key = value;

OnPropertyChanged("Key");

}

}

private string sourceText = "";

public string SourceText

{

get { return sourceText; }

set

{

sourceText = value;

OnPropertyChanged("SourceText");

}

}

private string modifiedText = "";

public string ModifiedText

{

get { return modifiedText; }

set

{

modifiedText = value;

OnPropertyChanged("ModifiedText");

}

}

private RelayCommand encodeCommand;

public RelayCommand EncodeCommand

{

get { return encodeCommand; }

set

{

;

}

}

private RelayCommand decodeCommand;

public RelayCommand DecodeCommand

{

get { return decodeCommand; }

set

{

;

}

}

private RelayCommand loadFromFileCommand;

public RelayCommand LoadFromFileCommand

{

get { return loadFromFileCommand; }

set

{

;

}

}

private RelayCommand saveToFileCommand;

public RelayCommand SaveToFileCommand

{

get { return saveToFileCommand; }

set

{

;

}

}

public MainWindowViewModel()

{

encodeCommand = new RelayCommand(Encode);

decodeCommand = new RelayCommand(Decode);

loadFromFileCommand = new RelayCommand(LoadFromFile);

saveToFileCommand = new RelayCommand(SaveToFile);

}

void Encode(object obj)

{

EncodeOrDecode(true);

}

void Decode(object obj)

{

EncodeOrDecode(false);

}

void EncodeOrDecode(bool encode = true)

{

switch(SelectedMethod)

{

case "Caesars cipher":

{

int shift;

if (Int32.TryParse(Key, out shift))

ModifiedText = BaseCiphers.CaesarsCipherUniversal(SourceText, shift, encode);

else

{

MessageBox.Show("The key must be an integer value!");

ModifiedText = "";

}

break;

}

case "Polybius square cipher":

{

ModifiedText = BaseCiphers.PolybiusSquareCipher(SourceText, Key, encode);

break;

}

case "Tritemius cipher":

{

int shift;

if (Int32.TryParse(Key, out shift))

ModifiedText = BaseCiphers.TritemiusCipher(SourceText, shift, encode);

else

{

MessageBox.Show("The key must be an integer value!");

ModifiedText = "";

}

break;

}

case "Vigenere cipher":

{

ModifiedText = BaseCiphers.VigenereCipher(SourceText, Key, encode);

break;

}

case "Rearrangement cipher":

{

ModifiedText = BaseCiphers.RearrangementCipherByKey(SourceText, Key, encode);

break;

}

case "Simple frequency analysis":

{

if (encode) ModifiedText = "";

else ModifiedText = BaseCiphers.SimpleFrequencyAnalysis(SourceText);

break;

}

}

}

void LoadFromFile(object obj)

{

OpenFileDialog dialog = new OpenFileDialog();

dialog.FileName = "Input";

dialog.DefaultExt = ".txt";

if (dialog.ShowDialog() == true)

{

try

{

using (StreamReader FileRead = new StreamReader(dialog.FileName))

SourceText = FileRead.ReadToEnd();

}

catch (Exception)

{

MessageBox.Show("Something went wrong while reading a file...");

}

}

}

void SaveToFile(object obj)

{

SaveFileDialog dialog = new SaveFileDialog();

dialog.FileName = "Output";

dialog.DefaultExt = ".txt";

if (dialog.ShowDialog() == true)

{

try

{

using (StreamWriter FileWriter = new StreamWriter(dialog.FileName))

FileWriter.Write(ModifiedText);

}

catch (Exception)

{

MessageBox.Show("Something went wrong while writing to file...");

}

}

}

#region INotifyPropertyChanged Members

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName]string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

#endregion

}

}

* 1. Робота з програмою

На рисунку 3.3 наведено вигляд графічного інтерфейсу користувача.

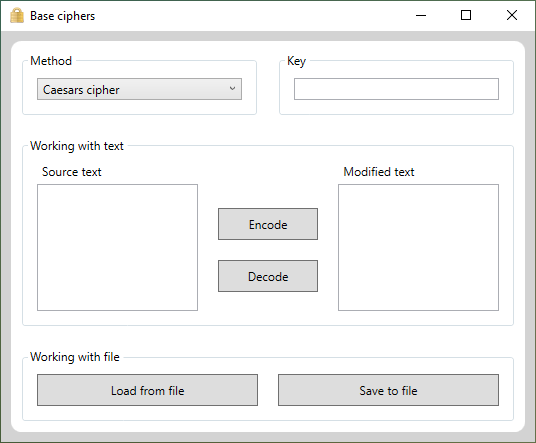


Рисунок 3.3 – Інтерфейс програми

Для зашифровування чи розшифровування інформації необхідно вибрати метод з розкривного списку, та ввести ключ. Далі необхідно власноруч ввести дані до поля Source text, або завантажити їх із файлу за допомогою кнопки Load from file. Далі необхідно натиснути кнопку Encode для зашифровування даних або кнопку Decode для розшифровування даних. Після виконаних дій зашифровані/розшифровані дані буде відображено в полі Modified text. За бажанням отримані дані можна зберегти у файл, натиснувши кнопку Save to file.

* + 1. **Зашифровування та розшифровування даних**

Нехай обране повідомлення «Valeriy Kozlov. First laboratory work», ключ для шифру Цезаря та Тритемія – 3, а для інших шифрів – «Success».

* + - 1. **Шифр Цезаря**

На рисунку 3.4 зображено результат зашифровування.

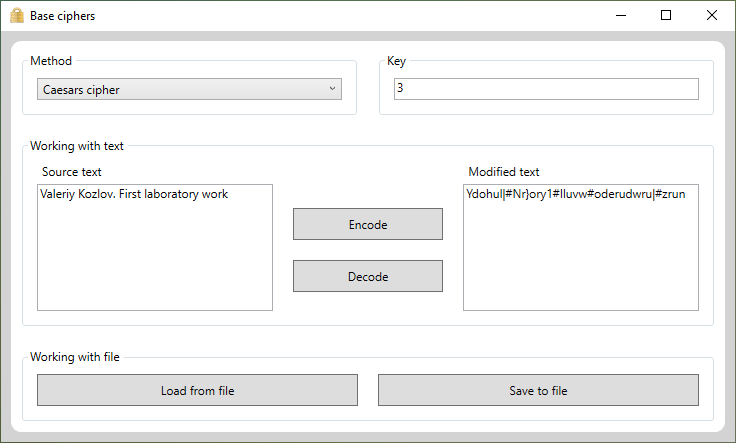


Рисунок 3.4 – Результат зашифровування шифром Цезаря

На рисунку 3.5 зображено результат розшифровування.

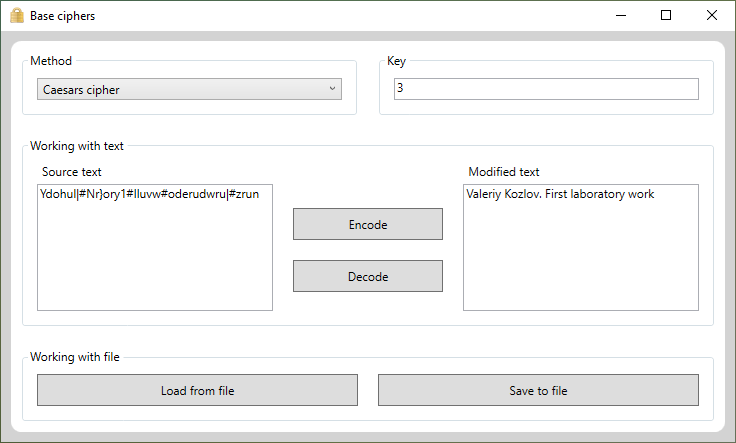


Рисунок 3.5 – Результат розшифровування шифром Цезаря

* + - 1. **Шифр квадрат Полібія**

На рисунку 3.6 зображено результат зашифровування.

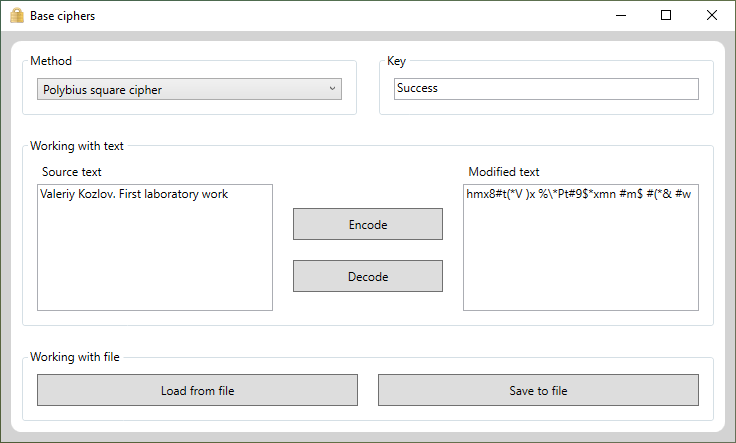


Рисунок 3.6 – Результат зашифровування шифром квадрат Полібія

На рисунку 3.7 зображено результат розшифровування.

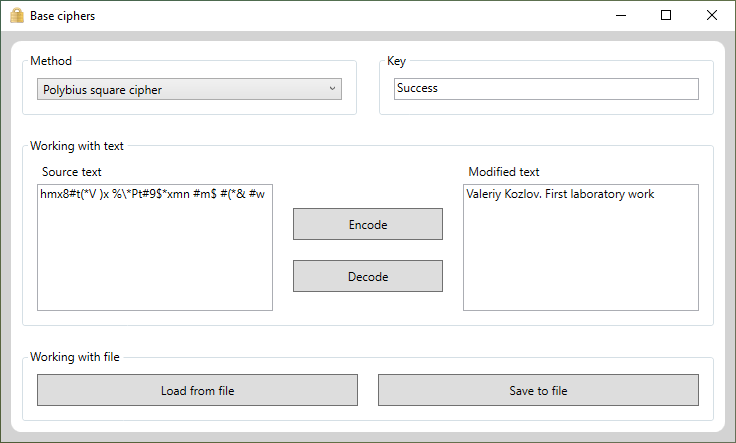


Рисунок 3.7 – Результат розшифровування шифром квадрат Полібія

* + - 1. **Шифр Тритемія**

На рисунку 3.8 зображено результат зашифровування.

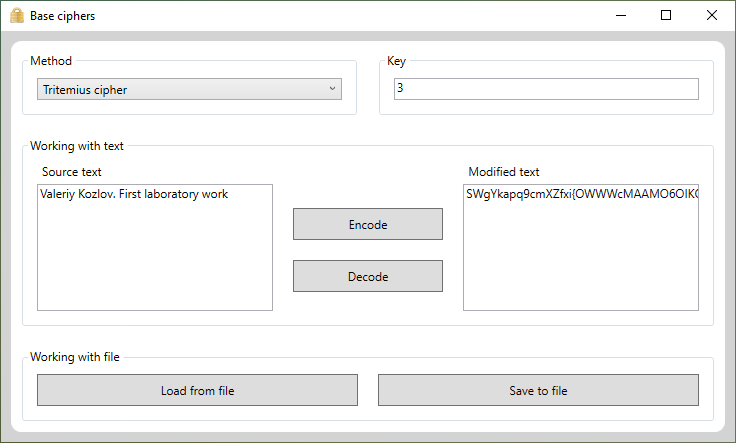


Рисунок 3.8 – Результат зашифровування шифром Тритемія

На рисунку 3.9 зображено результат розшифровування.

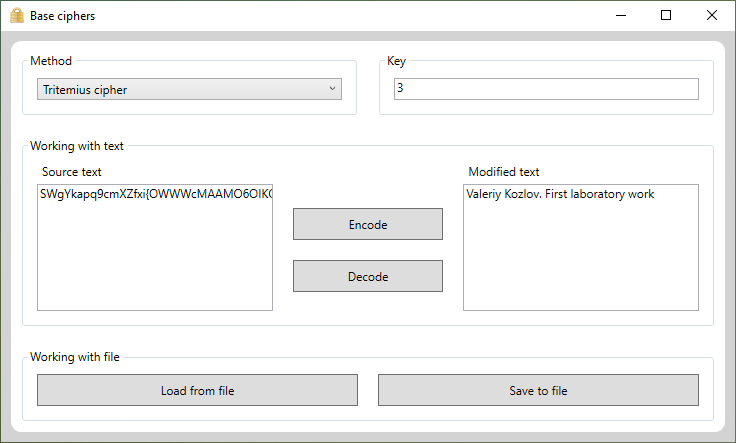


Рисунок 3.9 – Результат розшифровування шифром Тритемія

* + - 1. **Шифр Віжинера**

На рисунку 3.10 зображено результат зашифровування.

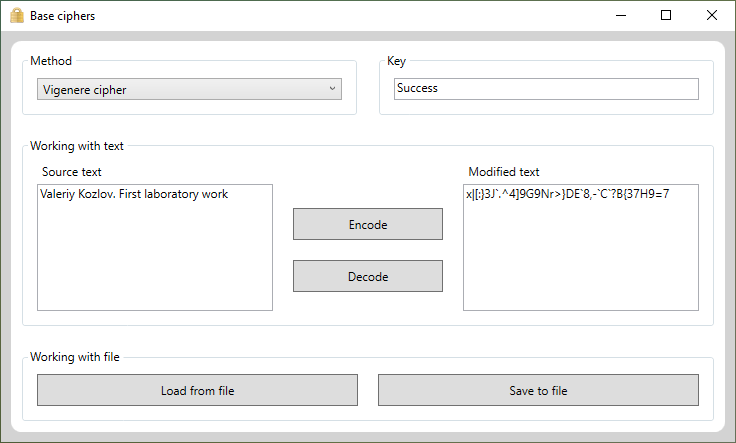


Рисунок 3.10 – Результат зашифровування шифром Віжинера

На рисунку 3.11 зображено результат розшифровування.

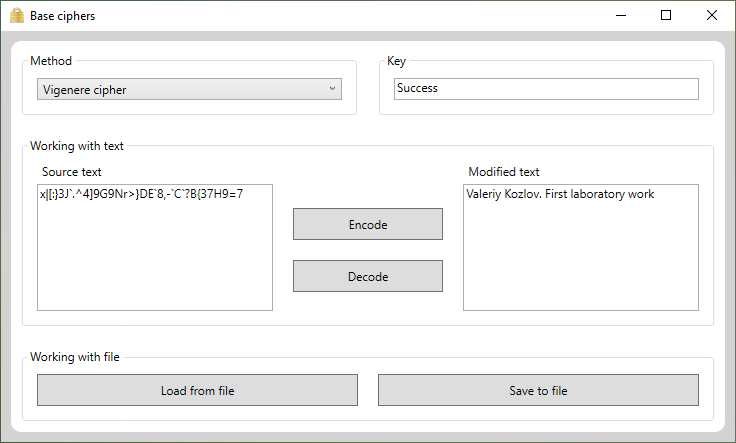


Рисунок 3.11 – Результат розшифровування шифром Віжинера

* + - 1. **Шифр перестановки**

На рисунку 3.12 зображено результат зашифровування.

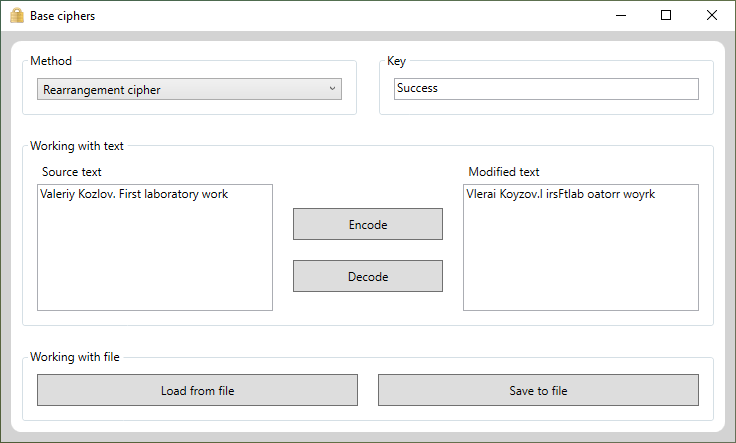


Рисунок 3.12 – Результат зашифровування шифром перестановки

На рисунку 3.13 зображено результат розшифровування.

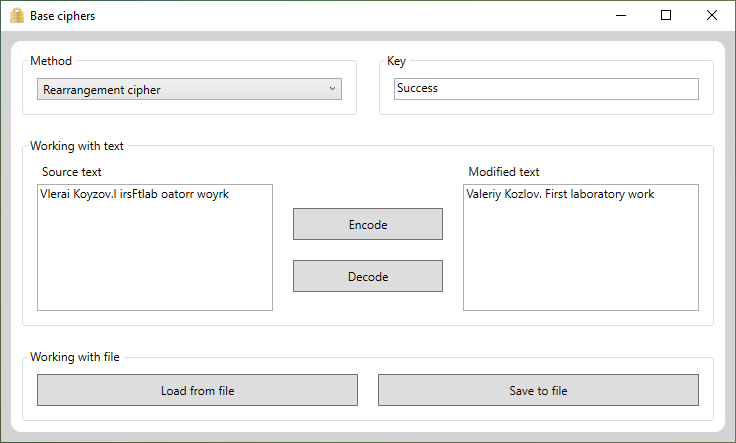


Рисунок 3.13 – Результат розшифровування шифром перестановки

* + - 1. **Частотний криптоаналіз**

На рисунку 3.14 зображено результат розшифровування.

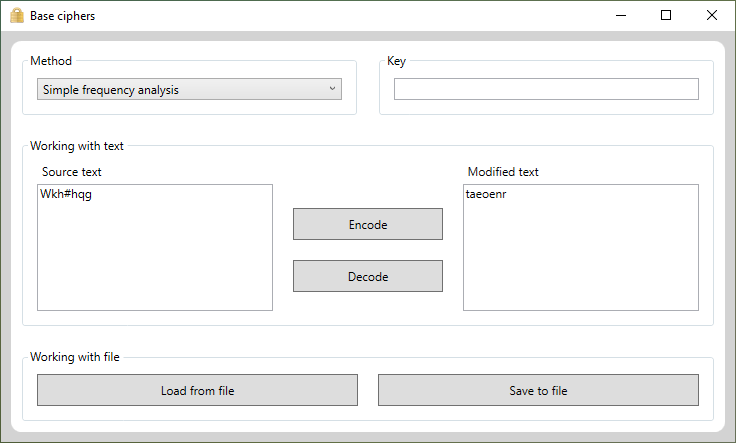


Рисунок 3.13 – Результат розшифровування частотний криптоаналіз

При частотному криптоаналізі розшифровувався текст «The end», зашифрований шифром Цезаря з ключом 3. Результат криптоаналізу – 4 з 7 символів вибрано вірно. Для більш точних результатів, кількість символів повідомлення має бути значно більшою.

Висновки

Протягом виконання цієї лабораторної роботи ознайомився з такими базовими шифрами: шифр простої заміни, шифр Цезаря, квадрат Полібія, шифр Тритемія, шифр Віжинера, шифр перестановки, транспонування, зсуву, маршрутної перестановки. Розглянув методику проведення частотного криптоаналізу.

Розробив програму на С# з графічним інтерфейсом, яка надає можливість зашифровування та розшифровування даних за такими шифрами: шифр Цезаря, квадрат Полібія, шифр Тритемія, шифр Віжинера, шифр перестановки. Також реалізував метод частотного криптоаналізу для розшифровування зашифрованих даних.