Vigenère's cipher

Software Documentation

Author: matiwa

Table of contents

Table of contents……………………………………………………………………………….2

Introduction…………………………..………………………………………………………...3

Describing of the application’s operation…………………………………………………...3

What is needed for use?..5

Algorithm used……………………………………………………………………………….5

Interface description…………………………………………………………………………8

Source code description…………………………………………………………………..….8

List of drawings……………………………………………………………………………13

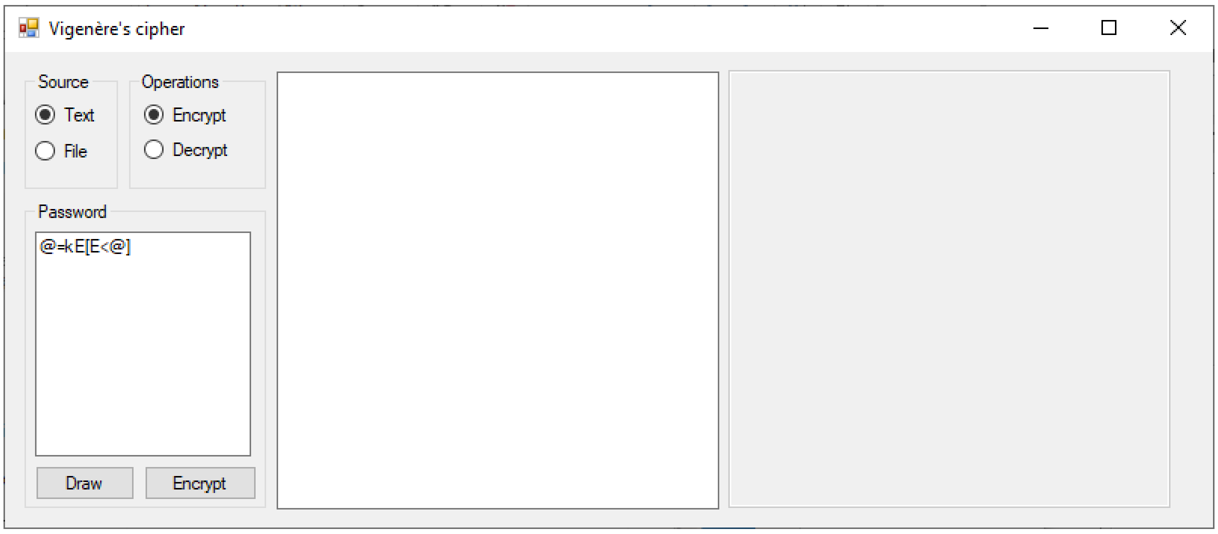
List of listings………………………………………………………………………………13

Bibliography..………………………………………………………………………………13

Introduction

This software documentation includes: description of the application's operation, what is needed for use, algorithms used, interface description and source code description. This application is used to encoding and decoding Vigenère's cipher.

Describing of the application’s operation



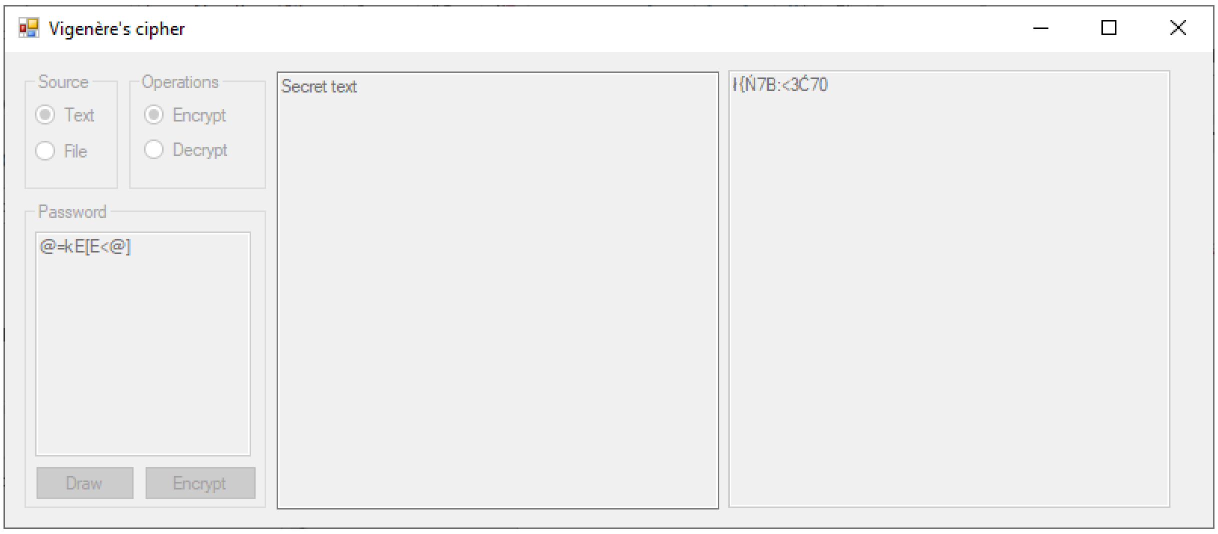
Source: [✓ Text] File

Operations: [✓ Encrypt] Decrypt

Password: @=kE[E<@]

Drawing 1: The beginning of the application's operation [own study]

The user has to select the source of the text and the operation to be performed. If he selects a file, the text box is inactive. If it selects text, the text box is active. When selecting a text source, it is saved to the file as a backup. Selecting a file means that the text will be loaded and printed in the text field. By default, the application generates a random key that can be changed by the user.


Vigenère's cipher

Source: [✓ Text] File

Operations: [✓ Encrypt] Decrypt

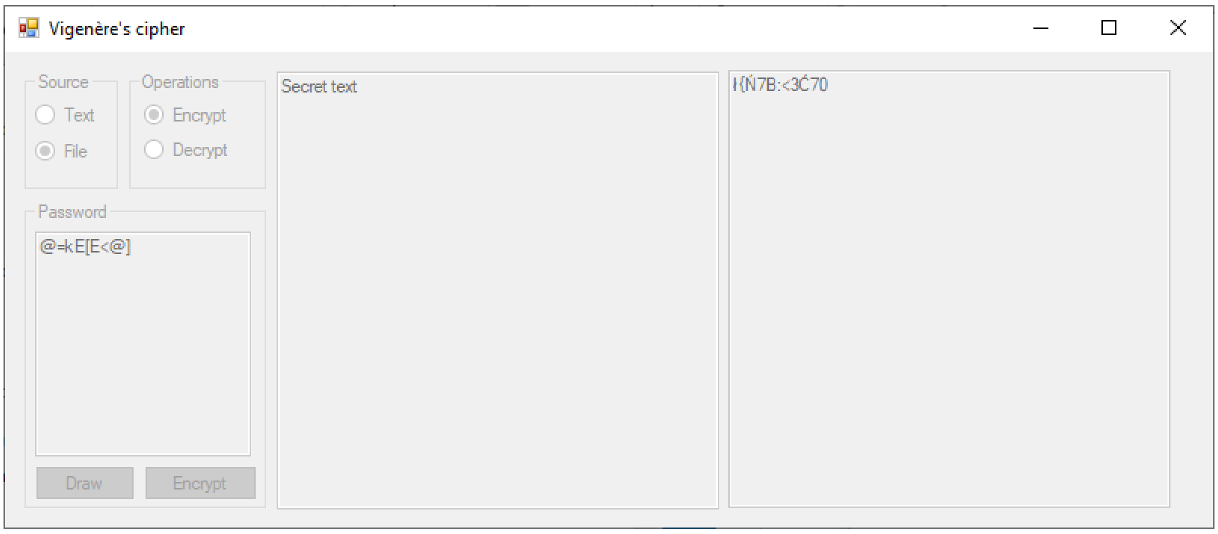
Password: @=kE[E<@]

Draw | Encrypt

Secret text

ł{Ń7B:<3Ć70

Drawing 2: Text coding example [own study]


Vigenère's cipher

Source: Text [✓ File]

Operations: [✓ Encrypt] Decrypt

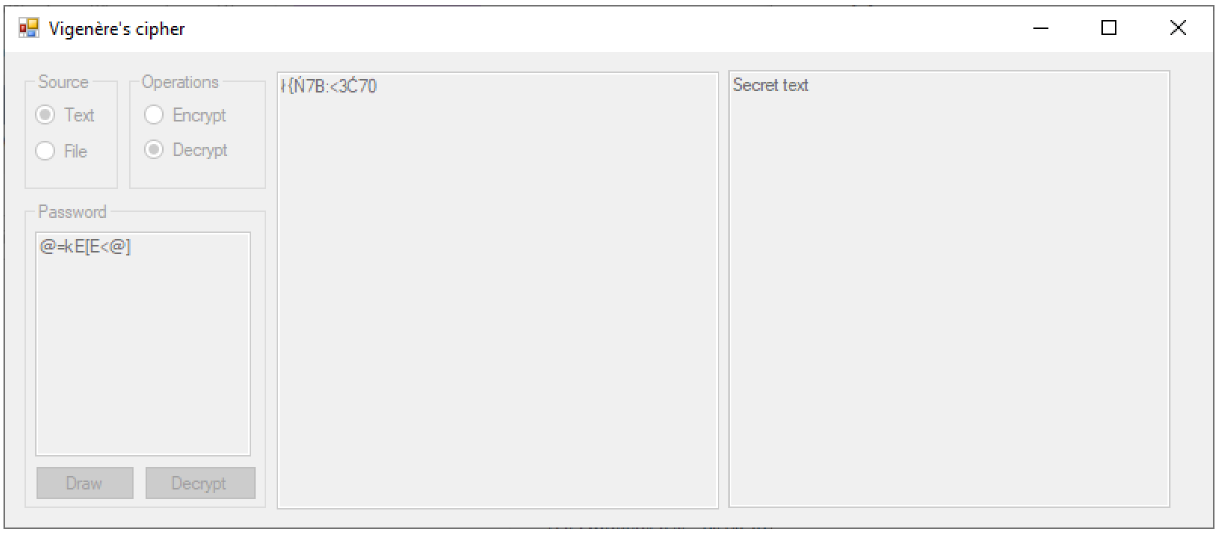
Password: @=kE[E<@]

Draw | Encrypt

Secret text

ł{Ń7B:<3Ć70

Drawing 3: An encoding example from a file [own study]


Vigenère's cipher

Source: [✓ Text] File

Operations: Encrypt [✓ Decrypt]

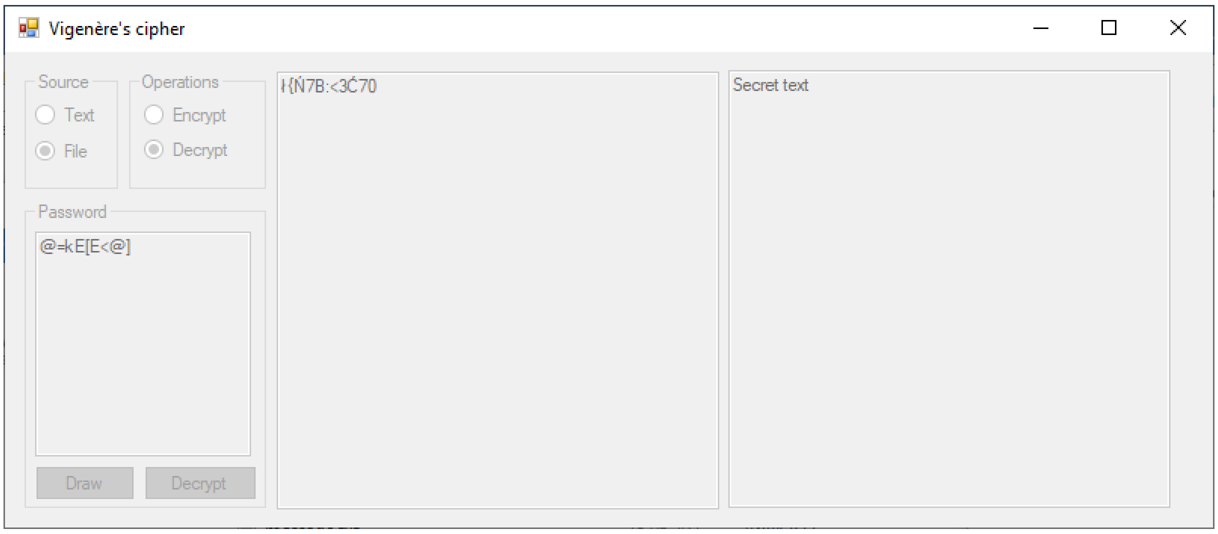
Password: @=kE[E<@]

Draw | Decrypt

ł{Ń7B:<3Ć70

Secret text

Drawing 4: Text decoding example [own study]



Vigenère's cipher

Source: Text [✓ File]

Operations: Encrypt [✓ Decrypt]

Password: @=kE[E<@]

```
ł{Ń7B:<3Ĉ70
```

Secret text

Drawing 5: An encoding example from a file [own study]

This is the correct use of the program. There may be issues that have not been detected by the developer.

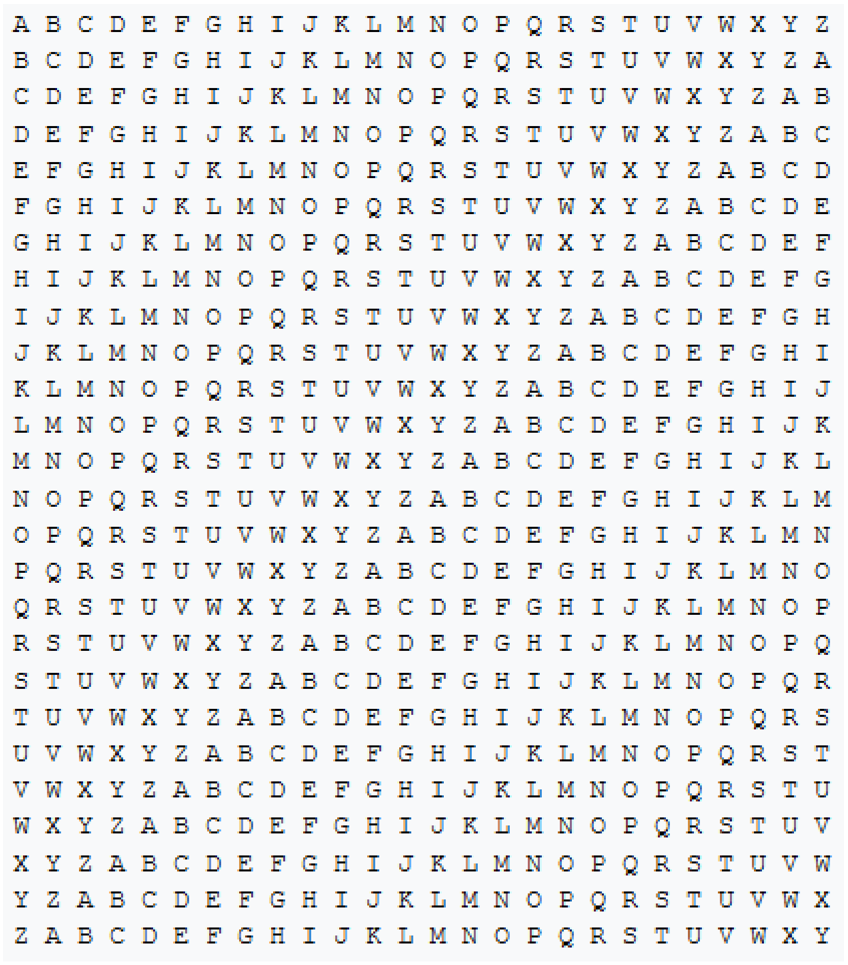
What is needed for use?

The application does not require installation. It only needs the Windows operating system.

Algorithm used

Vigenère's algorithm - one of the classic encryption algorithms. It belongs to the group of the so-called polyalphabetic substitution ciphers. This cipher was incorrectly assigned to the creator of the more complicated cipher Blaise de Vigenère.

The operation of the Vigenère cipher is based on the following table:



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Drawing 6: Vigenère's table [1]

Each of the lines of the table corresponds to the Caesar cipher, where the first line has an offset of 0, the second line is 1, and so on.

A keyword is needed to encrypt some text. The keyword is secret and tells you which row (or column) to use at any given time.

Suppose we want to encrypt a simple text, e.g .:

THIS IS A VERY SECRET TEXT

For this purpose, we will use a keyword known only to us, e.g. SECRET

At the beginning, we notice that the keyword used is too short to encrypt the entire text, so be sure to use multiple of it. It will look like this:

THIS IS A VERY SECRET TEXT
TA JNET AJNETA JNETA JNETA

Then we perform the encryption as follows: the letter of the ciphertext corresponds to the letter from the table located at the intersection of the line, determined by the letter of the plaintext, and the column determined by the letter of the keyword, e.g. "T" and "T" in turn gives "M", "O" "And" A "gives" O "etc. As a result, we get the ciphertext:

MO SRWM BJEHSO CNNGY CROLT

It doesn't matter if the plaintext letter designates the line and the keyword determines the column, or vice versa, the effect of the encryption will always be the same.

Decryption is similar. We take the next letters of the ciphertext and the corresponding letters of the keyword (similar to encryption). We choose the column corresponding to the letter of the keyword. Then, in this column, we look for the letter of the ciphertext. The line number corresponding to the letter found is the letter number of the plaintext. For example, in the 'T' column, the letter "M" is in the 'T' line, in the 'A' column, the letter "O" is in the 'O' line, etc.

There is, however, a simpler, especially for implementation purposes, method of decryption. It requires a password "flip" operation as below:

K2 (i) = [26 - K (i)] mod 26

where K (i) - the next letter of the keyword, numbered A = 0, B = 1 etc., and K2 (i) - the next letter of the "inverted" password. 26 is the number of letters of the Latin alphabet.

The effect of such a transformation for the entry "SECRET" will be the word "HARNW".

Then, the encryption operation should be performed on the ciphertext with the received password. The result, as you can see, will be the open form of the text.

The original Vigenère cipher used an auto key, the first letter of the key was determined and the subsequent letters were subsequent letters of the plaintext.

Let our cipher letter be "N":

```
      plaintext: THIS IS A VERY SECRET TEXT
            key: NT OJES TBARDZ OTAJN YTEKS
ciphertext: GH XNWL UBRUCN HTJWL RXOCL
```

The decryption process is as follows: the first letter of the ciphertext is decrypted with a fixed letter ("N"), subsequent letters with the letters that have just been deciphered:

G N → T

H T → O

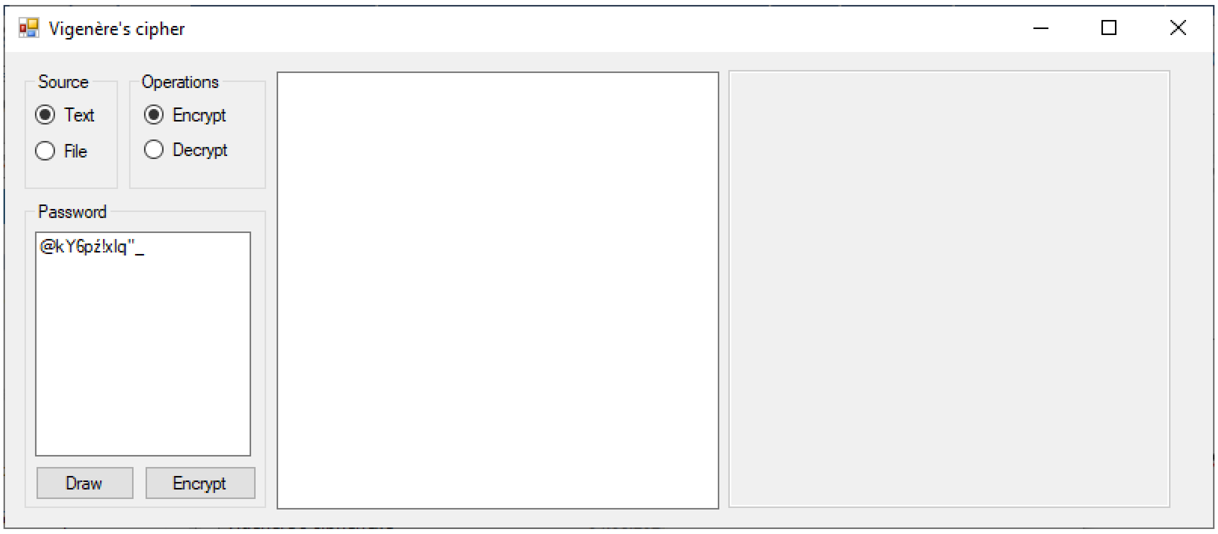
X O → J

...

```
       ciphertext: G H X N W L ...
            key: N T O J E S ...
decrypted text: T O J E S T ...
```

This cipher, due to the fact that the length of the key was equal to the length of the plaintext, was based on statistical analysis.[1]

Interface description



Source: [✓ Text] File

Operations: [✓ Encrypt] Decrypt

Password: @kY6pź!xlq"_

Drawing 7: Graphical interface [own study]

The interface is typical for a windowing application. There are essential components: buttons, text boxes, and radio buttons. The Draw button randomizes the number that is the key and is entered into the text box. The second button starts the encryption and decryption procedure according to the option that the user selects from the radio buttons. The text field for the message is active when selecting the text option button, and inactive for the file option.

Source code description

The project was made in the C# programming language, in the Visual Studio Community 2019 programming environment. All work was done on the Windows 10 operating system. The application's source code looks like this.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace Vigenère_s_cipher
```

```
{
    public partial class Form1 : Form
    {
        FileStream file = new FileStream("message.txt",
FileMode.OpenOrCreate, FileAccess.ReadWrite);
        static char[] chars = new char[]{'
','!','"','#','$','%','&','\'','(',')','*','+',',','-','.','/','0',

'1','2','3','4','5','6','7','8','9',':',';','<','=','>','?','@','A',
'Ą','B','C','Ć','D','E',

'Ę','F','G','H','I','J','K','L','Ł','M','N','Ń','O','Ó','P','Q','R',
'S','Ś','T','U','V','W',

'X','Y','Z','Ź','Ż','[','/',']','^','_','`','a','ą','b','c','ć','d',
'e','ę','f','g','h','i',

'j','k','l','ł','m','n','ń','o','ó','p','q','r','s','ś','t','u','v',
'w','x','y','z','ź','ż',
            '{','|','}','~'}; //ASCII code combined with Polish
diacritized letters

        static char[,] cryptogram = new char[chars.Length,
chars.Length];
        static char[] password;
        static char[] passwordjob;
        static int[] p;
        static int[] m;
        static string message = "";
        static char[] messagechar;

        public Form1()
        {
            InitializeComponent();
            //create cryptogram
            for (int i = 0; i < chars.Length; i++)
            {
                for (int j = 0; j < chars.Length; j++)
                {
                    if (j + i < chars.Length)
                        cryptogram[i, j] = chars[j + i];
                    else
                        cryptogram[i, j] = chars[j + i -
chars.Length];
                }
            }
            //draw a key
            Random r = new Random();
            for (int i = 0; i < r.Next(1, chars.Length); i++)
```

```
                tBPassword.Text += Convert.ToString(chars[r.Next(0,
chars.Length)]);
        }

        private void Btngo_Click(object sender, EventArgs e)
        {
            //loading the message from the text field and saving it
to a file
            if (rBText.Checked == true)
            {
                if (tBText.Text == "")
                {
                    Random r = new Random();
                    for (int i = 0; i < r.Next(1, chars.Length);
i++)
                        tBText.Text +=
Convert.ToString(chars[r.Next(0, chars.Length)]);
                }
                message = tBText.Text;
                StreamWriter sw = new StreamWriter(file);
                sw.WriteLine(message);
                sw.Close();
            }
            //loading a message from a file and saving it to the
text field
            else if (rBFile.Checked==true)
            {
                StreamReader sr = new StreamReader(file);
                message = sr.ReadLine();
                sr.Close();
                tBText.Text = message;
            }
            messagechar = message.ToCharArray();
            //the key cannot be empty!!!
            if (tBPassword.Text == "")
            {
                //draw a key
                Random r = new Random();
                for (int i = 0; i < r.Next(1, chars.Length); i++)
                    tBPassword.Text +=
Convert.ToString(chars[r.Next(0, chars.Length)]);
            }
            passwordjob = tBPassword.Text.ToCharArray();
            password = new char[message.Length];
            //the length of the password must be equal to or greater
than the message!!!
            if (passwordjob.Length < message.Length)
            {
                int x = 0;
                for(int i = 0; i < message.Length; i++)
                {
                    password[i] = passwordjob[x];
                    if (i < passwordjob.Length - 2) x++;
                    else if (i >= passwordjob.Length - 2 && x <
passwordjob.Length - 1) x++;
                    else if (i >= passwordjob.Length - 2 && x >=
passwordjob.Length - 1) x = 0;
                }
            }
            else for (int i = 0; i < message.Length; i++)
password[i] = passwordjob[i];
            p = new int[message.Length];
            //the beginning of the algorithm
            m = new int[message.Length];
            for (int i = 0; i < messagechar.Length; i++)
            {
                for (int j = 0; j < chars.Length; j++)
                {
                    if (messagechar[i] == chars[j]) m[i] = j;
                    if (password[i] == chars[j])
                    {
                        p[i] = j;
                        //operation of "reversing" the password if
there is decryption
                        if (rBD.Checked==true) p[i] = (chars.Length
- p[i]) % chars.Length;
                    }
                }
            }
            for (int i = 0; i < messagechar.Length; i++)
tBMessage.Text += Convert.ToString(cryptogram[m[i], p[i]]);
            //protection against the next operation
            gBSource.Enabled = false;
            gBOperation.Enabled = false;
            gBPassword.Enabled = false;
            tBText.Enabled = false;
        }

        //draw a key
        private void Btndraw_Click(object sender, EventArgs e)
        {
            tBPassword.Text = "";
            Random r = new Random();
            for (int i = 0; i < r.Next(1, chars.Length); i++)
                tBPassword.Text += Convert.ToString(chars[r.Next(0,
chars.Length)]);
        }
        //display the selected operation as the button text
        private void Operation(object sender, EventArgs e)
        {
            if (rBE.Checked == true) btngo.Text = rBE.Text;
            else if (rBD.Checked == true) btngo.Text = rBD.Text;
        }

        //select a message input method
        private void Selectsource(object sender, EventArgs e)
        {
            if (rBText.Checked == true) tBText.Enabled = true;
            else if (rBFile.Checked == true) tBText.Enabled = false;
        }
    }
}
```

Listing 1: Source code [own study]

List of drawings

Drawing 1: The beginning of the application's operation [own study]…………………………...3

Drawing 2: Text coding example [own study]……………………………………………………4

Drawing 3: An encoding example from a file [own study]………………………………………4

Drawing 4: Text decoding example [own study]…………………………………………………4

Drawing 5: An encoding example from a file [own study]………………………………………5

Drawing 6: Vigenère's table [1]…………………………………………………………………..6

Drawing 7: Graphical interface [own study]……………………………………………..……8

List of listings

Listing 1: Source code [own study]………………………………………………………………8

Bibliography

[1] https://pl.wikipedia.org/wiki/Szyfr_Vigen%C3%A8re%E2%80%99a