Bacon cipher

Software Documentation

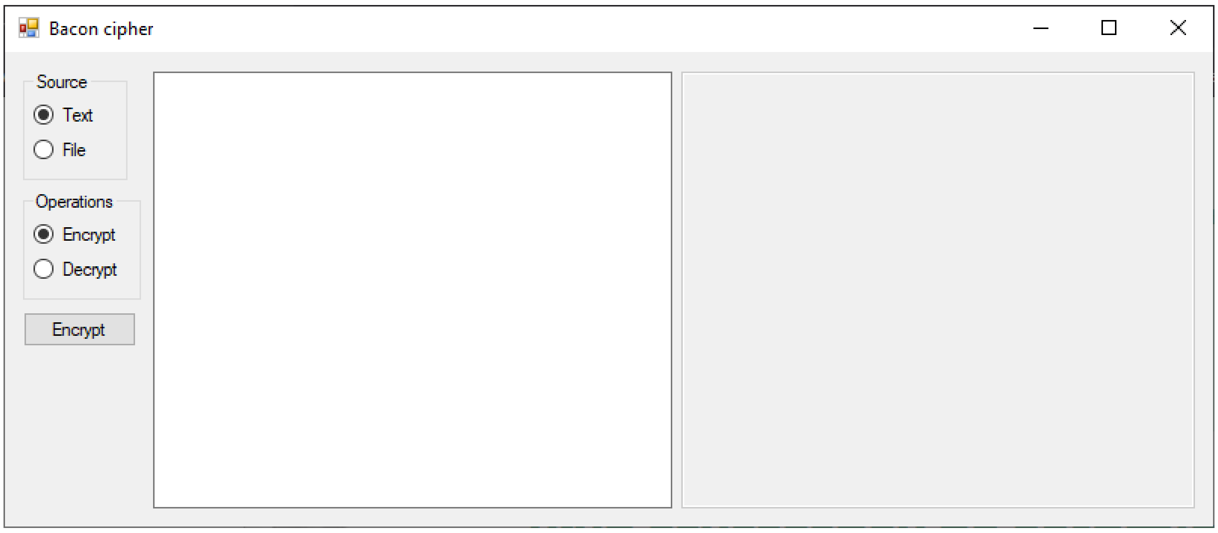
Author: matiwa

Table of contents

Introduction

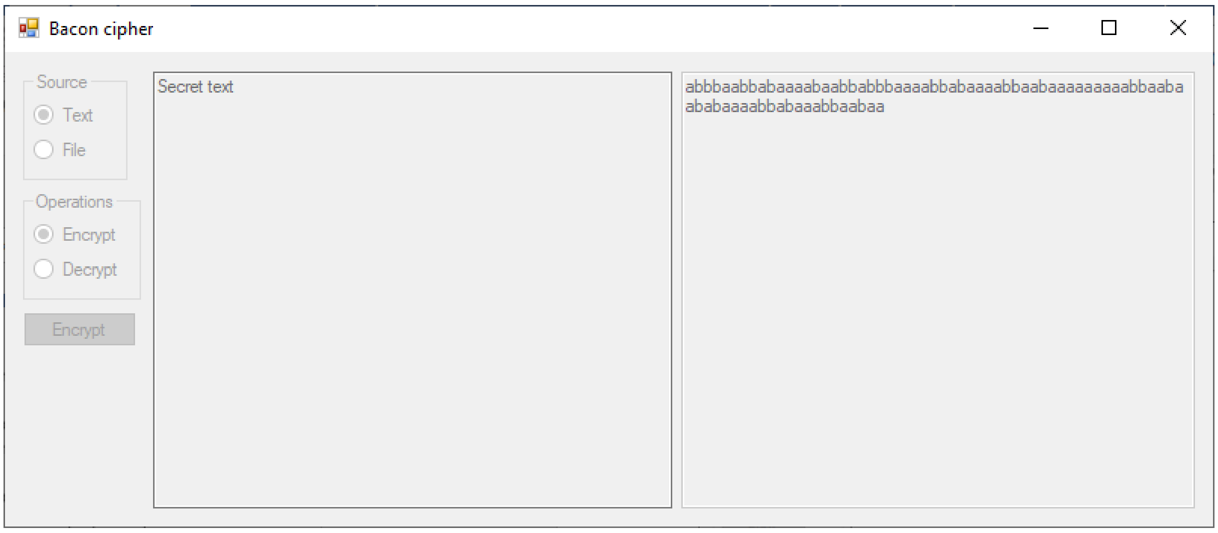
This software documentation includes: description of the application's operation, what is needed for use, algorithms used, interface description and source code description. This application is used to encoding and decoding Bacon cipher.

Describing of the application’s operation



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

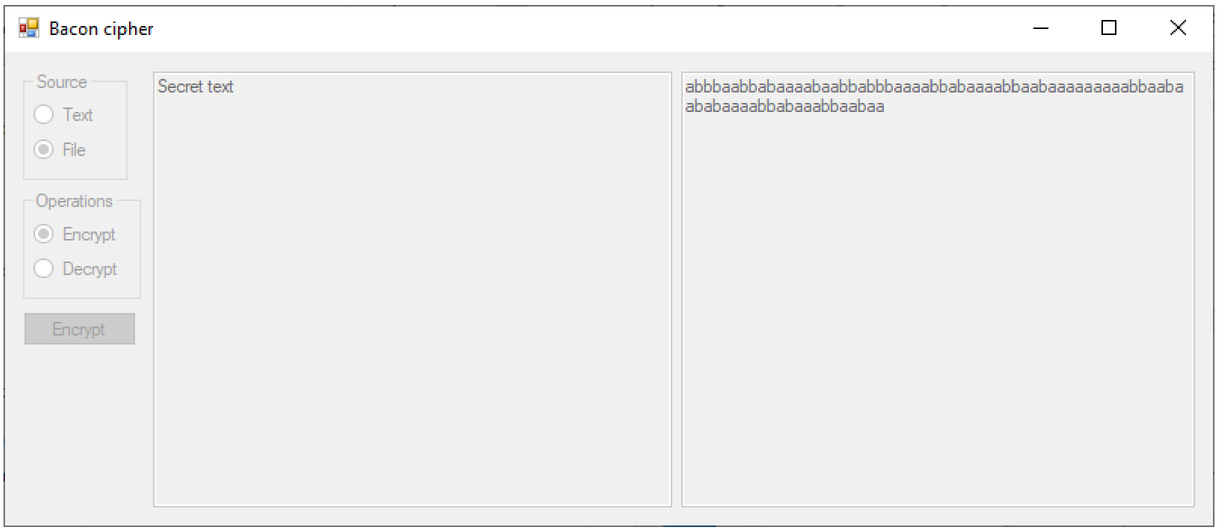
The user has to select the source of the text and the operation to be performed. If he selects a file, the text box is inactive. If it selects text, the text box is active. When selecting a text source, it is saved to the file as a backup. Selecting a file means that the text will be loaded and printed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



Secret text

```
abbbaabbabaaaabaabbabbbaaaabbabaaaabbaabaaaaaaaaaabbaaba
ababaaaabbabaaabbaabaa
```

Drawing 2: Text coding example [own study]


Bacon cipher

Source: Text [✓ File]

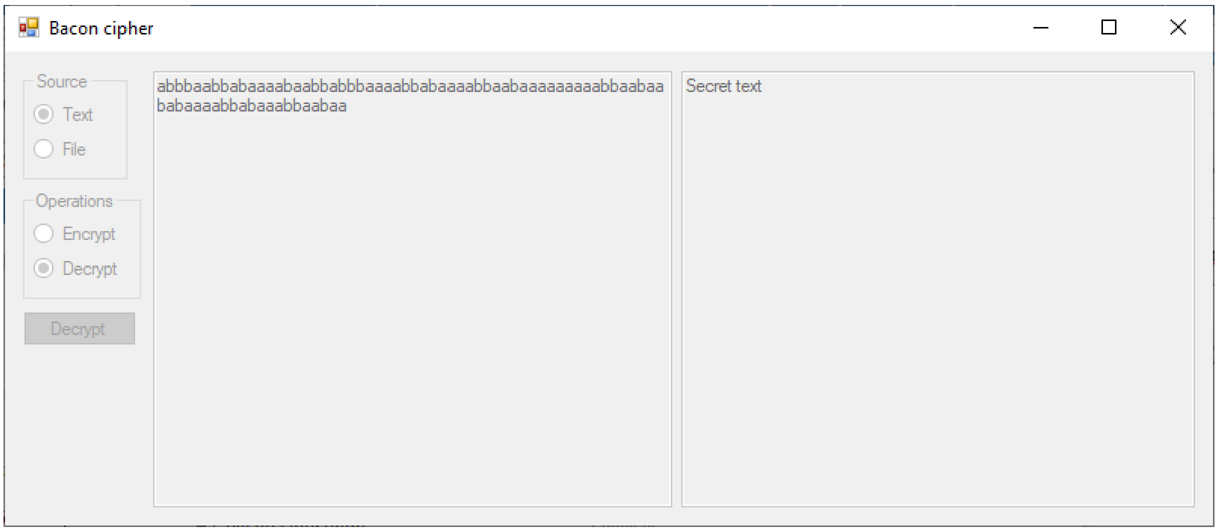
Operations: [✓ Encrypt] Decrypt

Encrypt

Secret text

abbbaabbabaaaabaabbabbbaaaabbabaaaabbaabaaaaaaaaaabbaaba
ababaaaabbabaaabbaabaa

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


Bacon cipher

Source: [✓ Text] File

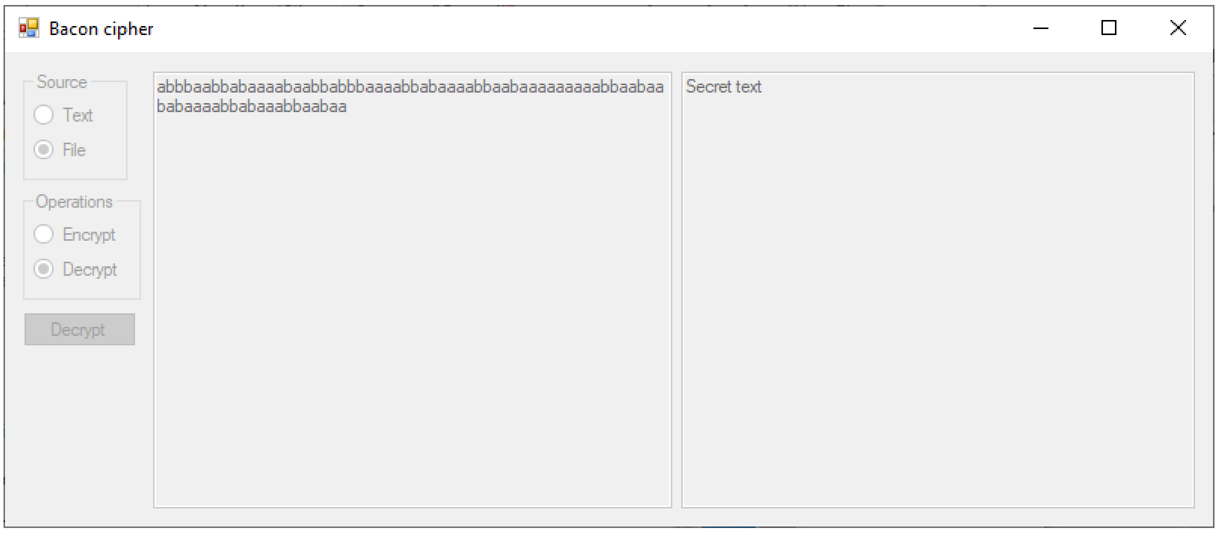
Operations: Encrypt [✓ Decrypt]

Decrypt

abbbaabbabaaaabaabbabbbaaaabbabaaaabbaabaaaaaaaaaabbaabaa
babaaaabbabaaabbaabaa

Secret text

Drawing 4: Text decoding example [own study]


Bacon cipher

Source: Text [✓ File]

Operations: Encrypt [✓ Decrypt]

Decrypt

abbbaabbabaaaabaabbabbbaaaabbabaaaabbaabaaaaaaaaaabbaabaa
babaaaabbabaaabbaabaa

Secret text

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

This is the correct use of the program. However, the user may make a mistake by entering a number with a comma or period. The program has no security, so its operation will be suspended.

What is needed for use?

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

Algorithm used

The Bacon cipher is a cipher in which the cipher text contains five-letter strings consisting of the letters a and b. The encryption and decryption follows the following scheme:

A = aaaaa

B = aaaab

C = aaaba

D = aaabb

E = aabaa

F = aabab

G = aabba

H = aabbb

I / J = abaaa

K = abaab

L = ababa

M = ababb

N = abbaa

O = abbab

P = abbba

Q = abbbb

R = baaaa

S = baaab

T = baaba

U / V = baabb

W = babaa

X = babab

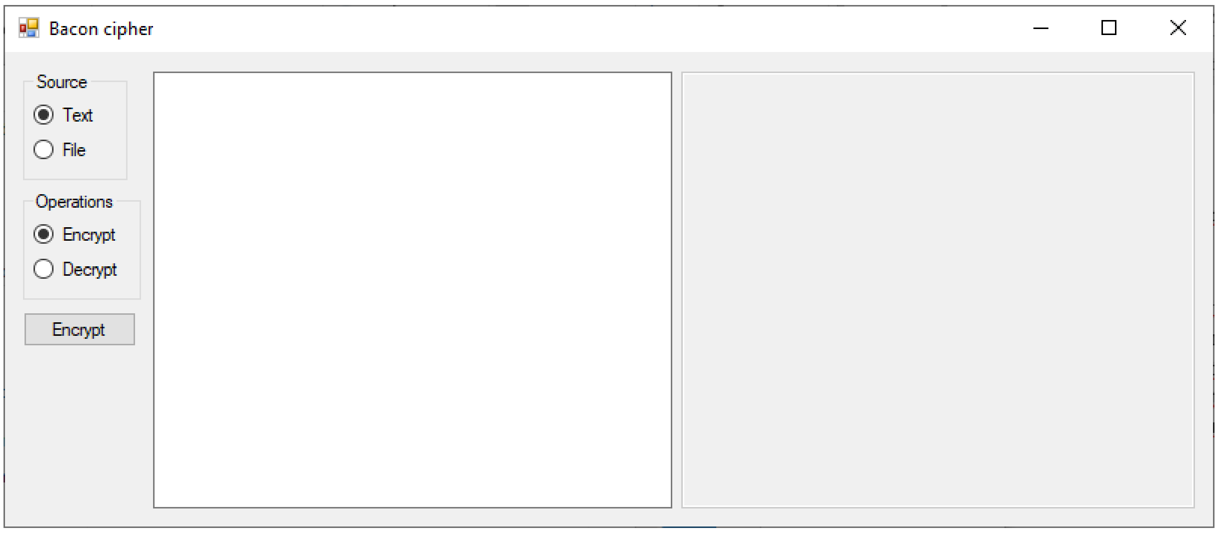
Y = babba

Z = babbb

Example:

- "Wikipedia" → "babaa abaaa abaab abaaa abbba aabaa aaabb abaaa aaaaa"
- "Szyfr Bacona" → "baaab babbb babba aabab baaaa aaaab aaaaa aaaba abbab abbaa aaaaa" lub "baaabbabbbbabbaaababbbaaaa aaaabaaaaaaaabaabbababbaaaaaaa"[1]

Interface description



Source
● Text
○ File

Operations
● Encrypt
○ Decrypt

Drawing 6: Graphical interface [own study]

The interface is typical for a windowing application. There are essential components: button, text boxes, and radio buttons. The 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 Bacon_cipher
{
    public partial class Form1 : Form
    {
        FileStream file = new FileStream("message.txt",
FileMode.OpenOrCreate, FileAccess.ReadWrite);
        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
        string[] bacon;//Bacon ciphertext declaration
        string message = "";

        public Form1()
        {
            InitializeComponent();
            bacon = new string[chars.Length];
            //generating a ciphertext table according to Bacon's
scheme
            for (int i = 0; i < chars.Length; i++)
            {
                string b = "";
                int j = i;
                while (j > 0)
                {
                    if(j%2==0) b = "a"+b;
                    else if (j % 2 == 1) b = "b" + b;
                    j /= 2;
                }
                while (b.Length < 7) b = "a" + b;
                bacon[i] = b;
            }
        }

        //for decryption, the algorithm does not work if the message
length is not divided by seven!!!
        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)
            {
                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;
            }
            //message encryption
            if (rBE.Checked == true)
            {
                char[] messagechar = message.ToCharArray();
                for (int i = 0; i < messagechar.Length; i++)
                {
                    for(int j = 0; j < chars.Length; j++)
                    {
                        if (messagechar[i] == chars[j])
tBMessage.Text += bacon[j];
                    }
                }
            }
            //message decryption (each encrypted character has 7
bits)
            else if (rBD.Checked == true)
            {
                char[] messagechar = message.ToCharArray();
                string bufor = "";
                for (int i = 0; i < messagechar.Length; i++)
                {
                    bufor += messagechar[i].ToString();
                    if (i % 7 == 6)
                    //each encrypted character has 7 bits, so I
check the character against the Bacon table
                    {
                        for (int j = 0; j < bacon.Length; j++)
                        {
                            if (bufor == bacon[j]) tBMessage.Text +=
chars[j].ToString();
                        }
                        bufor = "";
                    }
                }
            }
            //protection against the next operation
            gBOperation.Enabled = false;
            gBSource.Enabled = false;
            btngo.Enabled = false;
            tBText.Enabled = false;
        }

        //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

List of listings

Bibliography

[1] https://pl.wikipedia.org/wiki/Szyfr_Bacona