КІТ-м120Б
Богомаз Олексій

Лабораторна робота №1

Тема: Вступ до стігонографії.

Мета роботи: Дослідити можливість «приховування» даних у зображеннях.

Завдання:

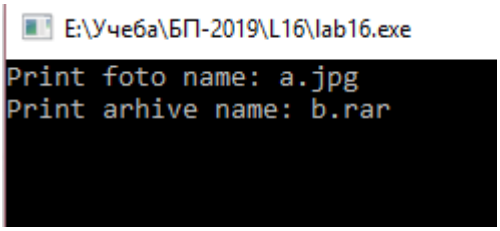
- Навести реалізацію технології Rar-Jpeg, та продемонструвати її роботу.
- Виконати скриття даних у зображення за допомогою методу найменьш значимих бітів (Less Significant Bits)
- Виконати аналіз скриття даних за допомогою методу стегоаналізу "атака хі квадрат"

Виконання роботи.

1) Rar-Jpeg

Код програми для Visual Studio:

```
// lab16.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается
выполнение программы.
//

#include <iostream>
#include <fstream>

int main()
{
	char namePf[80];
	char nameAr[80];

	std::cout << "Print foto name: " ;
	std::cin  >> namePf ;
	std::cout << "Print arhive name: ";
	std::cin  >> nameAr;

	std::ifstream if_a(namePf, std::ios_base::binary);
	std::ifstream if_b(nameAr, std::ios_base::binary);
	std::ofstream of_c("result.jpg", std::ios_base::binary);

	of_c << if_a.rdbuf() << if_b.rdbuf();
}
```



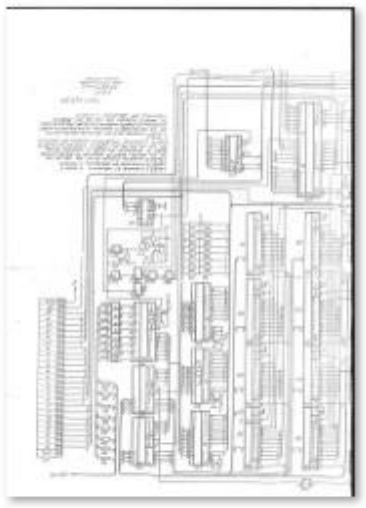


image2.jpg

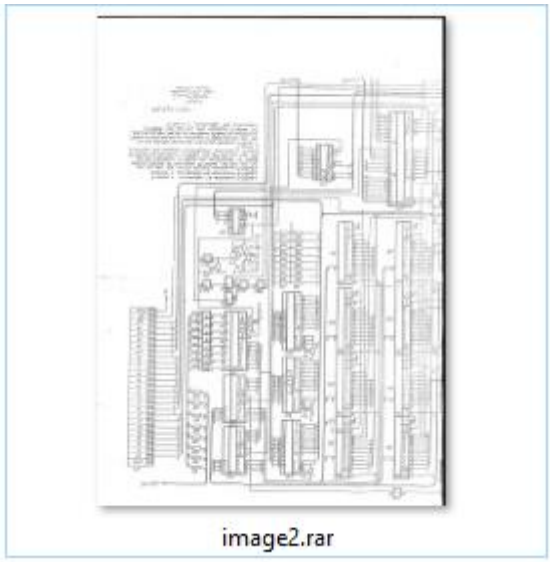


image2.rar

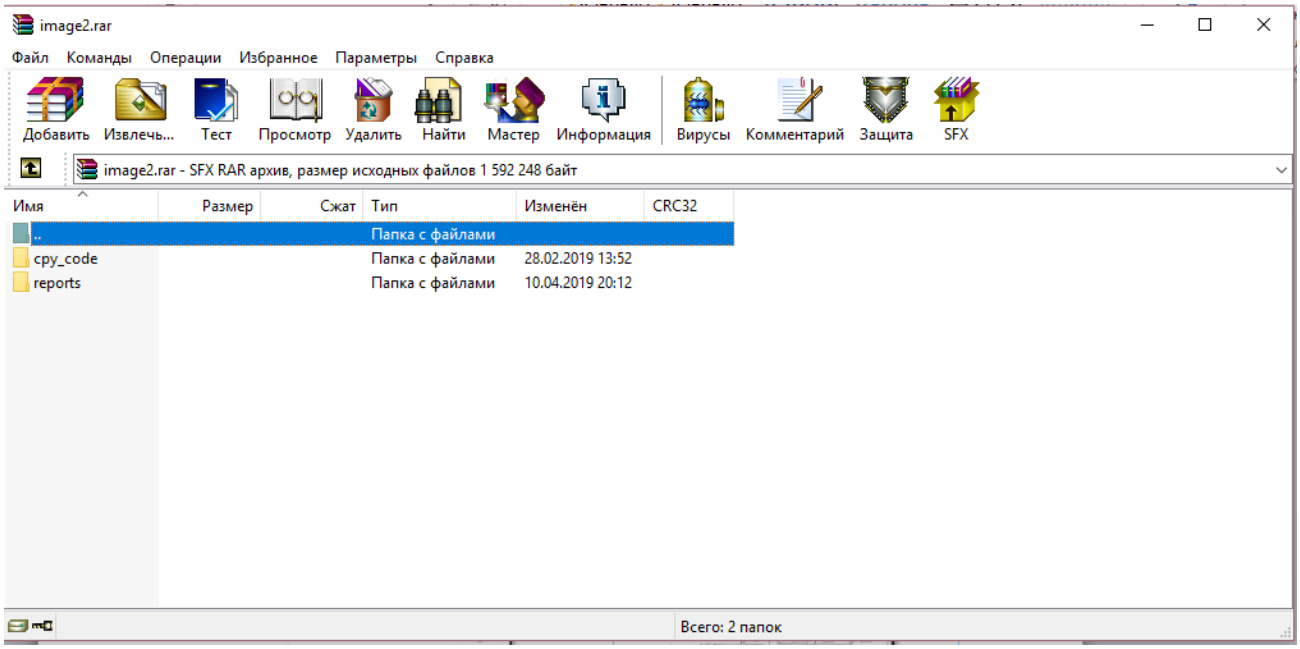


Рисунок 1 – Результат роботи

2) Less Significant Bits

Для виконання роботи використаємо на наш розсуд найбільш простий варіант побудови віконної програми - C#.

Код програми Шифрування C# :

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace lab8
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
```

```
        [STAThread]
        static void Main()
        {
            //try
            //{
                Application.EnableVisualStyles();
                Application.SetCompatibleTextRenderingDefault(false);
                Application.Run(new Form1());


        }
    }
}
```

Form1.cs

```
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;
using System.Collections;

namespace lab8
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private BitArray ByteToBit(byte src)
        {
            BitArray bitArray = new BitArray(8);
            bool st = false;
            for (int i = 0; i < 8; i++)
            {
                if ((src >> i & 1) == 1)
                {
                    st = true;
                }
                else st = false;
                bitArray[i] = st;
            }
            return bitArray;
        }

        private byte BitToByte(BitArray scr)
        {
            byte num = 0;
            for (int i = 0; i < scr.Count; i++)
                if (scr[i] == true)
                    num += (byte)Math.Pow(2, i);
            return num;
        }

        /*Проверяет, зашифрован ли файл,  возвращает true, если символ в первом пикселе
равен / иначе false */
```

```
        private bool isEncryption(Bitmap scr)
        {
            byte[] rez = new byte[1];
            Color color = scr.GetPixel(0, 0);
            BitArray colorArray = ByteToBit(color.R); //получаем байт цвета и преобразуем в
массив бит
            BitArray  messageArray  =  ByteToBit(color.R);  ;//инициализируем  результирующий
массив бит
            messageArray[0] = colorArray[0];
            messageArray[1] = colorArray[1];

            colorArray = ByteToBit(color.G);//получаем байт цвета и преобразуем в массив бит
            messageArray[2] = colorArray[0];
            messageArray[3] = colorArray[1];
            messageArray[4] = colorArray[2];

            colorArray = ByteToBit(color.B);//получаем байт цвета и преобразуем в массив бит
            messageArray[5] = colorArray[0];
            messageArray[6] = colorArray[1];
            messageArray[7] = colorArray[2];
            rez[0] = BitToByte(messageArray); //получаем байт символа, записанного в 1 пикселе
            string m = Encoding.GetEncoding(1251).GetString(rez);
            if (m == "/")
            {
                return true;
            }
            else return false;
        }

        /*Записыает количество символов для шифрования в первые биты картинки */
        private void WriteCountText(int count, Bitmap src)
        {
            byte[] CountSymbols = Encoding.GetEncoding(1251).GetBytes(count.ToString());
            for (int i = 0; i < 3; i++)
            {
                BitArray bitCount = ByteToBit(CountSymbols[i]); //биты количества символов
                Color pColor = src.GetPixel(0, i + 1); //1, 2, 3 пикселы
                BitArray bitsCurColor = ByteToBit(pColor.R); //бит цветов текущего пикселя
                bitsCurColor[0] = bitCount[0];
                bitsCurColor[1] = bitCount[1];
                byte nR = BitToByte(bitsCurColor); //новый бит цвета пиксея

                bitsCurColor = ByteToBit(pColor.G);//бит бит цветов текущего пикселя
                bitsCurColor[0] = bitCount[2];
                bitsCurColor[1] = bitCount[3];
                bitsCurColor[2] = bitCount[4];
                byte nG = BitToByte(bitsCurColor);//новый цвет пиксея

                bitsCurColor = ByteToBit(pColor.B);//бит бит цветов текущего пикселя
                bitsCurColor[0] = bitCount[5];
                bitsCurColor[1] = bitCount[6];
                bitsCurColor[2] = bitCount[7];
                byte nB = BitToByte(bitsCurColor);//новый цвет пиксея

                Color nColor = Color.FromArgb(nR, nG, nB); //новый цвет из полученных битов
                src.SetPixel(0, i + 1, nColor); //записали полученный цвет в картинку
            }
        }

        /*Читает количество символов для дешифрования из первых бит картинки*/
```

```
        private int ReadCountText(Bitmap src)
        {
            byte[] rez = new byte[3]; //массив на 3 элемента, т.е. максимум 999 символов
шифруется
            for (int i = 0; i < 3; i++)
            {
                Color color = src.GetPixel(0, i + 1); //цвет 1, 2, 3 пикселей
                BitArray colorArray = ByteToBit(color.R); //биты цвета
                BitArray bitCount = ByteToBit(color.R); ; //инициализация результирующего
массива бит
                bitCount[0] = colorArray[0];
                bitCount[1] = colorArray[1];

                colorArray = ByteToBit(color.G);
                bitCount[2] = colorArray[0];
                bitCount[3] = colorArray[1];
                bitCount[4] = colorArray[2];

                colorArray = ByteToBit(color.B);
                bitCount[5] = colorArray[0];
                bitCount[6] = colorArray[1];
                bitCount[7] = colorArray[2];
                rez[i] = BitToByte(bitCount);
            }
            string m = Encoding.GetEncoding(1251).GetString(rez);
            return Convert.ToInt32(m, 10);
        }

        private void pictureBox1_Click(object sender, EventArgs e)
        {

        }
        /* Открыть файл для шифрования */
        private void button1_Click(object sender, EventArgs e)
        {
            string FilePic;
            string FileText;
            OpenFileDialog dPic = new OpenFileDialog();
            dPic.Filter = "Файлы изображений (*.bmp)|*.bmp|Все файлы (*.*)|*.*";
            if (dPic.ShowDialog() == DialogResult.OK)
            {
                FilePic = dPic.FileName;
            }
            else
            {
                FilePic = "";
                return;
            }

            FileStream rFile;
            try
            {
                rFile = new FileStream(FilePic, FileMode.Open); //открываем поток
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return;
            }
```

```
            Bitmap bPic = new Bitmap(rFile);

            OpenFileDialog dText = new OpenFileDialog();
            dText.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*";
            if (dText.ShowDialog() == DialogResult.OK)
            {
                FileText = dText.FileName;
            }
            else
            {
                FileText = "";
                return;
            }

            FileStream rText;
            try
            {
                rText = new FileStream(FileText, FileMode.Open); //открываем поток
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return;
            }
            BinaryReader bText = new BinaryReader(rText, Encoding.ASCII);

            List<byte> bList = new List<byte>();
            while (bText.PeekChar() != -1)
            { //считали весь текстовый файл для шифрования в лист байт
                bList.Add(bText.ReadByte());
            }
            int CountText = bList.Count; // в CountText - количество в байтах текста, который
нужно закодировать
            bText.Close();
            rFile.Close();

            //проверяем, поместиться ли исходный текст в картинке
            if (CountText > ((bPic.Width * bPic.Height)) - 4)
            {
                MessageBox.Show("Выбранная картинка мала для размещения выбранного текста",
"Информация", MessageBoxButtons.OK);
                return;
            }

            //проверяем, может быть картинка уже зашифрована
            if (isEncryption(bPic))
            {
                MessageBox.Show("Файл уже зашифрован", "Информация", MessageBoxButtons.OK);
                return;
            }

            byte[] Symbol = Encoding.GetEncoding(1251).GetBytes("/");
            BitArray ArrBeginSymbol = ByteToBit(Symbol[0]);
            Color curColor = bPic.GetPixel(0, 0);
            BitArray tempArray = ByteToBit(curColor.R);
            tempArray[0] = ArrBeginSymbol[0];
            tempArray[1] = ArrBeginSymbol[1];
            byte nR = BitToByte(tempArray);
```

```
            tempArray = ByteToBit(curColor.G);
            tempArray[0] = ArrBeginSymbol[2];
            tempArray[1] = ArrBeginSymbol[3];
            tempArray[2] = ArrBeginSymbol[4];
            byte nG = BitToByte(tempArray);

            tempArray = ByteToBit(curColor.B);
            tempArray[0] = ArrBeginSymbol[5];
            tempArray[1] = ArrBeginSymbol[6];
            tempArray[2] = ArrBeginSymbol[7];
            byte nB = BitToByte(tempArray);

            Color nColor = Color.FromArgb(nR, nG, nB);
            bPic.SetPixel(0, 0, nColor);
            //то есть в первом пикселе будет символ /, который говорит о том, что картика
зашифрована

            WriteCountText(CountText, bPic); //записываем количество символов для шифрования

            int index = 0;
            bool st = false;
            for (int i = 4; i < bPic.Width; i++)
            {
                for (int j = 0; j < bPic.Height; j++)
                {
                    Color pixelColor = bPic.GetPixel(i, j);
                    if (index == bList.Count)
                    {
                        st = true;
                        break;
                    }
                    BitArray colorArray = ByteToBit(pixelColor.R);
                    BitArray messageArray = ByteToBit(bList[index]);
                    colorArray[0] = messageArray[0]; //меняем
                    colorArray[1] = messageArray[1]; // в нашем цвете биты
                    byte newR = BitToByte(colorArray);

                    colorArray = ByteToBit(pixelColor.G);
                    colorArray[0] = messageArray[2];
                    colorArray[1] = messageArray[3];
                    colorArray[2] = messageArray[4];
                    byte newG = BitToByte(colorArray);

                    colorArray = ByteToBit(pixelColor.B);
                    colorArray[0] = messageArray[5];
                    colorArray[1] = messageArray[6];
                    colorArray[2] = messageArray[7];
                    byte newB = BitToByte(colorArray);

                    Color newColor = Color.FromArgb(newR, newG, newB);
                    bPic.SetPixel(i, j, newColor);
                    index++;
                }
                if (st)
                {
                    break;
                }
            }
            pictureBox1.Image = bPic;
```

```
            String sFilePic;
            SaveFileDialog dSavePic = new SaveFileDialog();
            dSavePic.Filter = "Файлы изображений (*.bmp)|*.bmp|Все файлы (*.*)|*.*";
            if (dSavePic.ShowDialog() == DialogResult.OK)
            {
                sFilePic = dSavePic.FileName;
            }
            else
            {
                sFilePic = "";
                return;
            };

            FileStream wFile;
            try
            {
                wFile = new FileStream(sFilePic, FileMode.Create); //открываем поток на запись
результатов
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла на запись", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            bPic.Save(wFile, System.Drawing.Imaging.ImageFormat.Bmp);
            wFile.Close(); //закрываем поток
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string FilePic;
            OpenFileDialog dPic = new OpenFileDialog();
            dPic.Filter = "Файлы изображений (*.bmp)|*.bmp|Все файлы (*.*)|*.*";
            if (dPic.ShowDialog() == DialogResult.OK)
            {
                FilePic = dPic.FileName;
            }
            else
            {
                FilePic = "";
                return;
            }

            FileStream rFile;
            try
            {
                rFile = new FileStream(FilePic, FileMode.Open); //открываем поток
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return;
            }
            Bitmap bPic = new Bitmap(rFile);
            if (!isEncryption(bPic))
            {
```

```
                MessageBox.Show("В файле нет зашифрованной информации", "Информация",
MessageBoxButtons.OK);
                return;
            }

            int countSymbol = ReadCountText(bPic); //считали количество зашифрованных символов
            byte[] message = new byte[countSymbol];
            int index = 0;
            bool st = false;
            for (int i = 4; i < bPic.Width; i++)
            {
                for (int j = 0; j < bPic.Height; j++)
                {
                    Color pixelColor = bPic.GetPixel(i, j);
                    if (index == message.Length)
                    {
                        st = true;
                        break;
                    }
                    BitArray colorArray = ByteToBit(pixelColor.R);
                    BitArray messageArray = ByteToBit(pixelColor.R); ;
                    messageArray[0] = colorArray[0];
                    messageArray[1] = colorArray[1];

                    colorArray = ByteToBit(pixelColor.G);
                    messageArray[2] = colorArray[0];
                    messageArray[3] = colorArray[1];
                    messageArray[4] = colorArray[2];

                    colorArray = ByteToBit(pixelColor.B);
                    messageArray[5] = colorArray[0];
                    messageArray[6] = colorArray[1];
                    messageArray[7] = colorArray[2];
                    message[index] = BitToByte(messageArray);
                    index++;
                }
                if (st)
                {
                    break;
                }
            }
            string strMessage = Encoding.GetEncoding(1251).GetString(message);

            string sFileText;
            SaveFileDialog dSaveText = new SaveFileDialog();
            dSaveText.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*";
            if (dSaveText.ShowDialog() == DialogResult.OK)
            {
                sFileText = dSaveText.FileName;
            }
            else
            {
                sFileText = "";
                return;
            };

            FileStream wFile;
            try
            {
```

```
                wFile = new FileStream(sFileText, FileMode.Create); //открываем поток на
запись результатов
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла на запись", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            StreamWriter wText = new StreamWriter(wFile, Encoding.Default);
            wText.Write(strMessage);
            MessageBox.Show("Текст записан в файл", "Информация", MessageBoxButtons.OK);
            wText.Close();
            wFile.Close(); //закрываем поток
        }
    }
}
```

Результат:

Вікно програми(рис.2):

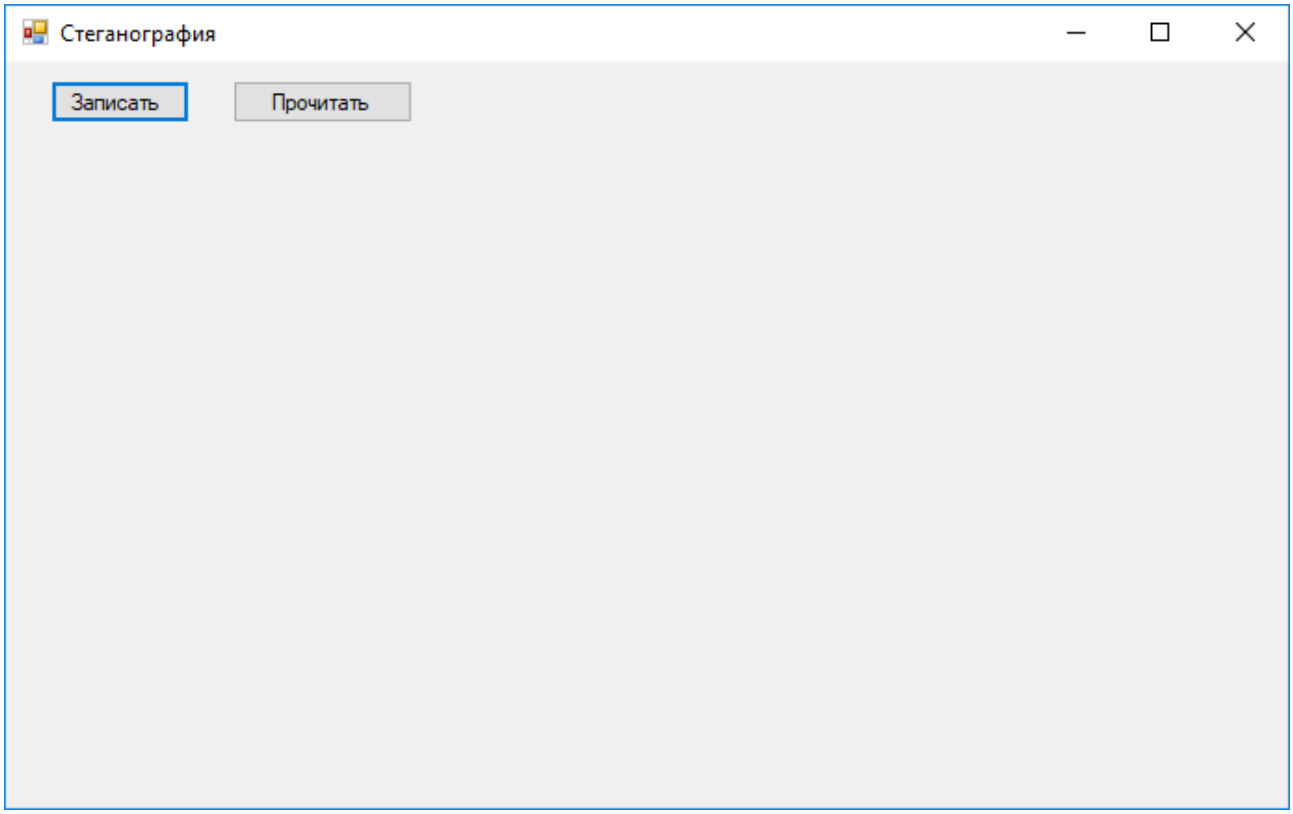


Рисунок 2

Шифруємо(рис.3-рис.4):

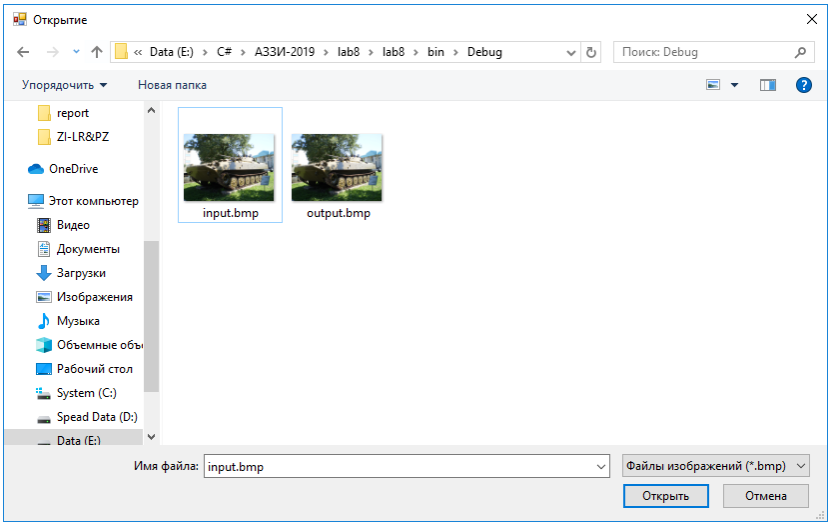


Рисунок 3

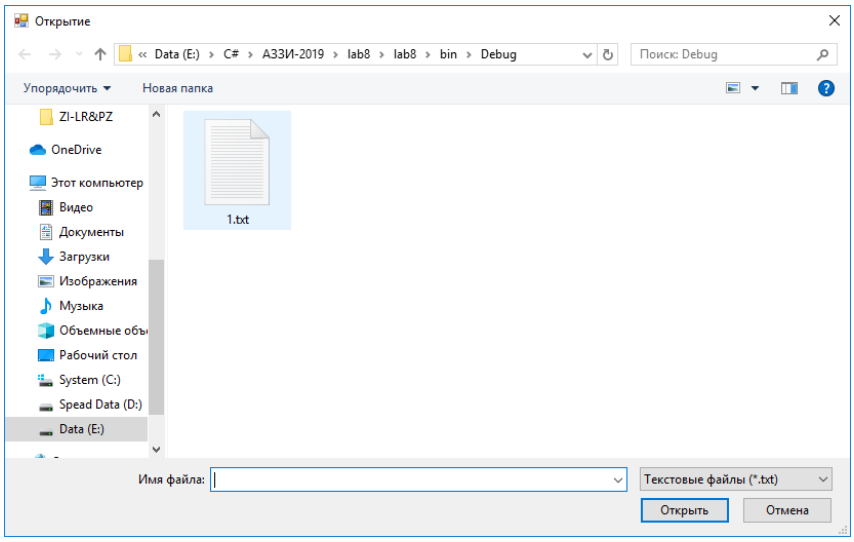


Рисунок 4

Після успішного шифрування отримуємо(рис.5):

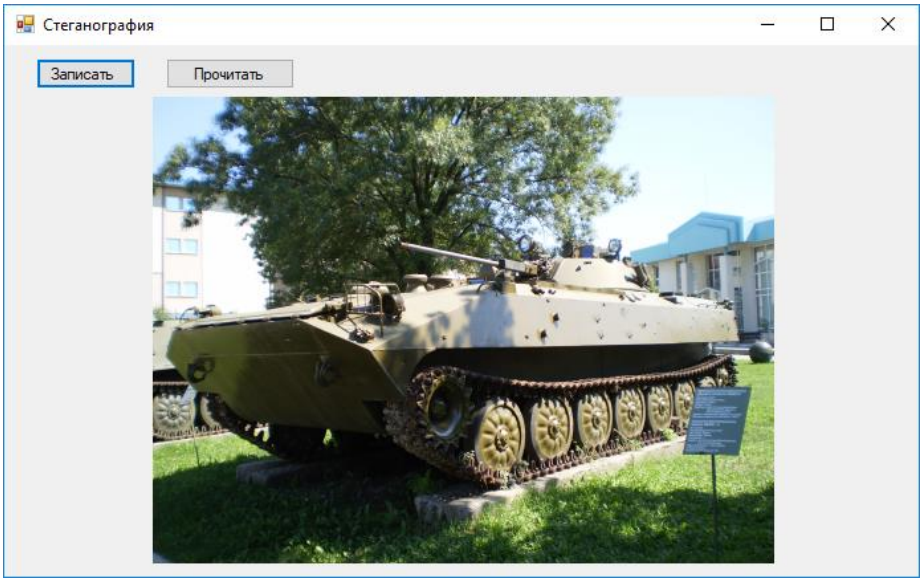


Рисунок 5

Вікно дешифрування(рис.6):

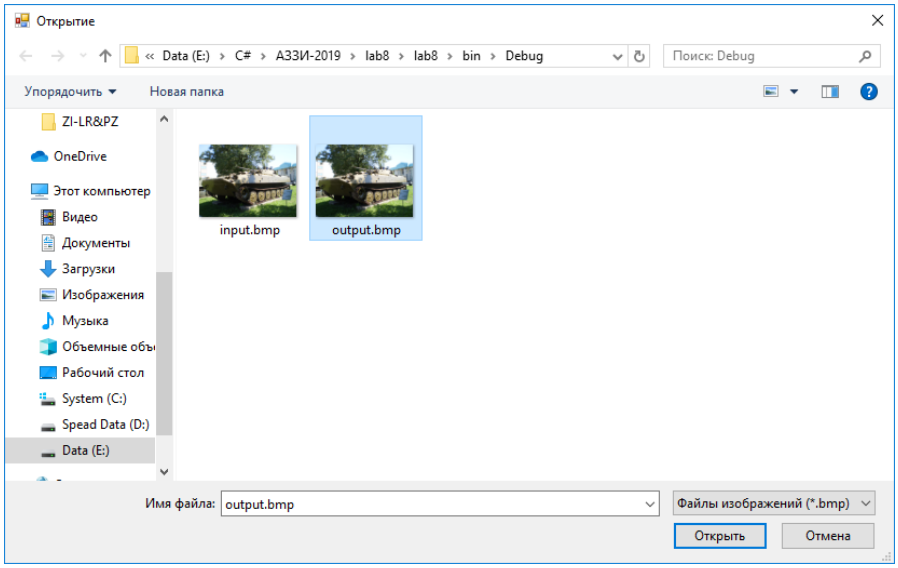


Рисунок 6

Після успішної дешифрації отримуємо(рис.7):

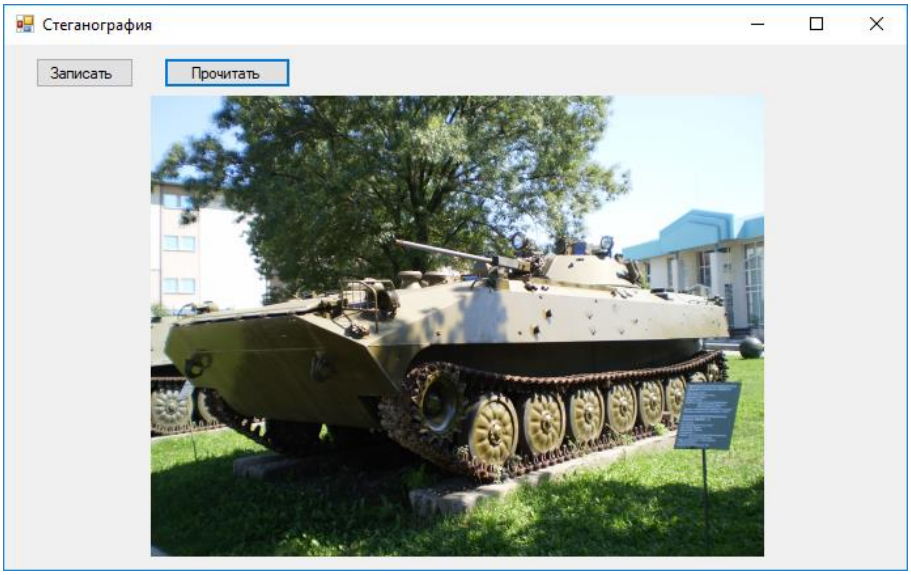


Рисунок 7

Дешифрована програма повністю функціональна й готова до повторного циклу(рис.8-рис.9).

Рисунок 8 Вхідний файл

Рисунок 9 Вихідний файл

3) атака хі квадрат

Для виконання роботи використаємо на наш розсуд найбільш простий варіант побудови віконної програми - C#.

Код програми Шифрування C# :

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace lab8
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            //try
            //{
                Application.EnableVisualStyles();
                Application.SetCompatibleTextRenderingDefault(false);
                Application.Run(new Form1());


        }
    }
}
```

Form1.cs

```
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;
using System.Collections;
using MathNet.Numerics;

namespace lab8
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private BitArray ByteToBit(byte src)
        {
            BitArray bitArray = new BitArray(8);
            bool st = false;
            for (int i = 0; i < 8; i++)
            {
                if ((src >> i & 1) == 1)
                {
                    st = true;
                }
                else st = false;
                bitArray[i] = st;
            }
            return bitArray;
        }

        private byte BitToByte(BitArray scr)
        {
            byte num = 0;
            for (int i = 0; i < scr.Count; i++)
                if (scr[i] == true)
                    num += (byte)Math.Pow(2, i);
            return num;
        }

        private void pictureBox1_Click(object sender, EventArgs e)
        {

        }
        /* Открыть файл для шифрования */
        private void button1_Click(object sender, EventArgs e)
        {
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string FilePic;
            OpenFileDialog dPic = new OpenFileDialog();
            dPic.Filter = "Файлы изображений (*.bmp)|*.bmp|Все файлы (*.*)|*.*";
            if (dPic.ShowDialog() == DialogResult.OK)
            {
                FilePic = dPic.FileName;
            }
            else
            {
                FilePic = "";
                return;
            }

            FileStream rFile;
            try
            {
                rFile = new FileStream(FilePic, FileMode.Open); //открываем поток
            }
            catch (IOException)
            {
```

```
                MessageBox.Show("Ошибка открытия файла", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return;
            }
            Bitmap bPic = new Bitmap(rFile);
            float[] GColors = new float[256];
            float[] nFact = new float[128];
            float[] nOzid = new float[128];
            float[] x = new float[bPic.Height+1];
            double[] p = new double[bPic.Height + 1];
            double k = 128;
            //double x = 0;

            for (int i = 0; i < bPic.Height; i++)
            {
                for(int j = 0; j < GColors.Length; j++)
                {
                    GColors[j] = 0.001f;
                }
                for (int j = 0; j < bPic.Width; j++)
                {
                    Color pixelColor = bPic.GetPixel(j, i);
                    GColors[pixelColor.R]++;
                    //GColors[pixelColor.G]++;
                    //GColors[pixelColor.B]++;
                }
                for(int j = 0; j < 128; j++)
                {
                    nFact[j] = 0;
                    nOzid[j] = 0;
                }
                x[i] = 0;
                for (int j = 0; j < 128; j++)
                {
                    nFact[j] = GColors[2 * j];
                    nOzid[j] = (GColors[2 * j] + GColors[(2 * j)+1])/2;
                    x[i] += (nFact[j]-nOzid[j])*(nFact[j] - nOzid[j])/ nOzid[j];
                }
                p[i] = 1 - (1 / (Math.Pow(2, (k - 1) / 2) *
SpecialFunctions.GammaLowerIncomplete((k - 1) / 2, x[i])));
            }

            //for (int j = 0; j < bPic.Height; j++)
            //{
            //    double k = 128;
            //    //double step2 = Math.Pow(2, k);
            //    //double gama = SpecialFunctions.GammaLowerIncomplete(k, x[j]);
            //    //double sg = Math.Pow(2, k) * SpecialFunctions.GammaLowerIncomplete(k,
x[j]);
            //    //double sg1 = 1 / (Math.Pow(2, k) *
SpecialFunctions.GammaLowerIncomplete(k, x[j]));
            //    p[j] = 1 - (1 / (Math.Pow(2, (k-1)/2) *
SpecialFunctions.GammaLowerIncomplete((k - 1) / 2, x[j])));
            //}
            string writePath = @".\X2.txt";
            for (int i = 0; i < p.Length; i++)
            {
                using (StreamWriter sw = new StreamWriter(writePath, true,
System.Text.Encoding.Default))
                {
                    sw.WriteLine(p[i].ToString());
                }
            }
        }
    }
}
```

Результат хи квадратів рядків «чистого» зображення:

6,35899 19,89659 50,14728 35,876 38,39762 30,86046
6,602352 22,7258 47,79624 34,1584 39,11563 31,42785
6,882231 20,92413 40,65534 39,22456 39,51297 35,85347
10,4114 23,55863 35,58175 33,49566 33,15423 27,66432
11,72546 17,76337 40,56166 30,39234 34,73637 35,40471
10,08114 22,54622 37,25591 39,38005 32,27465 40,32709
7,488267 23,76629 42,29782 33,45901 42,82469 21,88405
9,154365 21,39973 39,11872 37,36061 37,02657 33,45871
9,017269 30,29913 43,30504 30,44549 33,55157 38,63847
9,071028 22,88251 31,76235 46,98542 31,07365 29,50723
8,712497 20,89263 38,13852 42,52323 36,77075 34,01437
15,49204 23,06738 43,4772 43,03728 40,29977 37,79895
13,59591 28,21039 42,72965 40,55154 31,96975 34,22886
4,368248 26,11608 46,1685 42,98789 29,42743 34,93753
9,805822 24,96853 39,67209 41,52327 40,52412 35,13897
10,2925 32,25324 42,10828 49,71091 29,42583 34,39285
13,31227 26,54238 37,84917 34,71738 31,58653 31,63271
15,37749 28,89034 34,03489 50,14543 24,40223 27,59285
14,04251 30,60033 42,60714 38,8946 29,93193 31,5464
14,48361 29,36459 43,84552 40,0382 27,65165 37,40307
15,53108 24,73317 40,53698 40,97203 37,03685 36,13053
18,71885 27,01699 36,6781 54,97368 44,57619 34,56627
21,12788 25,06766 54,29446 37,65459 31,82221 39,44002
17,07851 29,417 47,8241 36,67244 30,14185 38,92864
17,51955 29,90853 42,26297 31,65229 36,91786 41,75237
18,39029 36,64503 49,2329 49,43009 30,23227 27,90025
16,26345 32,45295 29,29431 44,93486 29,77549 37,61999
18,51371 33,11042 34,69809 38,26728 45,34048 43,30811
14,57256 32,52485 35,50072 44,6308 41,00381 34,60318
16,22443 35,32042 38,30441 45,11111 34,94019 37,09918
18,76369 29,54469 39,9416 35,04748 31,41444 29,98084
19,08683 29,23939 42,60372 35,84376 39,17629 34,37227
17,81503 32,25732 44,97622 43,15501 40,35654 36,21127
18,43302 36,53647 34,10729 43,98745 33,79017 34,19172
19,52293 37,91182 40,58337 49,17834 29,86265 38,72752
24,00557 34,74085 34,54359 38,50854 45,31366 39,57607
17,21871 38,0374 38,2799 51,80996 39,48125 38,19257
15,77192 33,18106 33,84617 44,39981 31,87279 40,71127
30,20258 39,00436 40,04724 43,5591 37,39197 35,10137
19,97088 40,74645 33,51714 37,65248 35,68297 43,9899
18,53673 32,1642 50,21315 31,50911 35,03404 41,04857
27,31954 35,88916 46,96571 39,74249 37,69159 36,80381
20,78723 34,77683 46,38053 33,35456 32,86285 34,79934
17,30277 30,35462 48,75936 39,88101 43,92839 39,58604
22,58024 28,86841 49,75933 42,43529 37,13186 38,65533
16,61148 42,37957 40,7524 39,89148 39,15253 47,12251
20,69745 33,35374 43,55427 39,16735 34,37307 46,93114
19,95212 27,9877 31,97704 38,49815 29,1833 43,71571
21,99485 36,30834 37,4986 38,01804 37,01498 45,9427
21,10829 30,37259 31,8292 44,00561 37,48056 47,12938
22,10979 35,60266 41,44631 45,63758 34,29092 36,88138
22,91933 31,29341 42,21203 41,12879 34,94159 47,52624
23,64973 36,29037 33,86383 42,59253 22,92594 44,7867
28,77957 39,86333 30,93434 44,80273 30,72241 41,92608
25,51016 39,49258 40,10492 44,29272 36,91498 38,89398
21,07578 31,30599 29,16931 48,75478 30,68996 29,98594
25,72509 31,51244 40,06855 44,0441 29,29756 45,022
25,63928 25,75222 37,91342 44,4375 27,01113 40,46099
24,617 32,28563 32,68763 40,20526 34,83069 39,50034
26,89802 21,55489 43,81496 46,71464 35,16977 44,89994
28,15198 33,00792 37,18126 35,16461 26,06658 32,92783
22,40869 33,01551 30,46997 35,31532 40,49747 42,14116
24,35066 39,56182 43,61493 45,45911 40,71859 36,39652
26,45418 39,41772 41,94183 49,50489 28,91859 34,19759
21,77562 28,46305 34,7701 40,59302 25,64052 30,93514
20,91942 38,5104 35,42057 39,42157 36,66591 41,78849
24,84826 31,92656 41,80043 43,18629 31,03383 33,13213
15,8903 35,41357 34,17306 41,72626 32,35425 37,35632
24,00463 30,17971 35,25652 54,97544 25,62327 39,44782
25,71889 35,73144 39,25831 45,83641 32,11322 35,94606
26,1856 45,44111 40,39367 37,8209 34,21517 31,72992

30,71639	36,53402	34,77979	34,42636		
37,14606	32,3359	31,08962			

Результат хи квадратів рядків зображення з вкладеною інформацією:

42,47453	76,11854	42,77676	60,69299	66,89694	56,75117
54,4408	59,01086	60,87002	70,41791	78,4477	64,64355
65,59309	47,03923	50,41034	51,1603	55,3736	58,34174
83,59988	59,89692	47,26809	57,5417	66,04953	40,61874
54,27842	69,57568	67,1585	71,68217	67,65064	54,84371
39,26973	52,89431	79,25841	51,2576	58,70349	46,66391
20,99938	72,52607	61,53725	44,42337	61,56452	35,7288
26,85778	69,69469	52,86876	48,71173	56,93673	36,99012
36,38498	72,81008	57,97093	56,97649	67,83335	39,83581
23,66479	47,86933	60,78809	33,75999	55,08116	51,83533
25,47545	32,37724	67,25588	38,53487	79,62293	54,70419
66,75601	44,32936	74,50051	47,62765	64,07375	44,39528
64,36694	44,9172	58,5841	57,9001	47,24936	50,10118
47,26804	43,49032	66,61465	67,92023	60,53736	47,46539
60,92719	51,27681	66,17934	53,4127	77,36625	34,69413
45,21439	40,41265	70,12425	48,58256	59,61318	65,3563
37,83136	63,75618	69,8591	56,12619	52,75139	41,67408
58,70046	56,65478	56,38532	47,95124	48,74036	50,24082
66,99435	44,72974	66,93418	77,38607	62,10619	46,48997
79,36099	79,04656	68,41708	66,57689	49,94729	39,42767
48,95057	53,22759	76,45185	67,00589	50,73326	68,04929
36,50655	41,05281	63,28655	61,40928	56,17179	45,02218
38,85399	45,53221	62,71111	63,59022	57,20036	38,88076
42,12949	52,359	56,49332	81,16699	43,18262	43,57238
53,5116	50,19253	56,70868	63,90564	72,1612	53,25527
31,38219	52,92986	65,16451	64,13677	50,41838	59,35398
35,31183	52,62099	53,27174	58,18629	72,81551	45,49922
39,10818	60,63887	57,58268	72,73596	60,27036	40,64647
39,11304	52,78805	53,44176	68,02479	43,37765	39,25702
41,38117	56,22968	65,99594	64,86551	67,22698	47,10947
42,24396	35,70281	56,01403	86,286	48,93403	41,29514
63,681	60,37237	50,76433	77,00802	51,26209	33,66933
51,08801	74,96816	73,34885	65,7515	50,22253	54,4841
51,73277	62,2673	72,10496	81,87862	50,82164	45,1784
51,1414	67,151	58,67938	83,57922	52,40303	57,65304
57,33867	64,79539	55,19647	63,84426	58,04147	52,78015
50,60395	60,89849	59,40428	64,14471	55,28946	52,79571
49,15891	81,91236	52,16621	74,10134	45,81351	43,41679
41,64053	66,12454	49,74672	59,39745	55,43174	44,21023
41,14334	45,72314	50,55278	97,5839	54,56239	53,00783
44,04578	40,58092	56,24935	88,20016	44,36292	50,81821
52,32316	57,7103	49,73215	60,13615	47,46068	54,52293
54,56229	52,79189	45,44344	70,64928	49,55946	54,87702
67,04174	51,24915	44,64343	44,25771	47,08215	43,30005
69,25237	49,93408	66,18188	51,54741	58,31717	48,81794
76,57677	66,08736	56,30234	80,87861	46,78362	39,86977
73,47035	51,72562	59,33621	83,3726	59,65697	44,82701
60,10399	56,57706	61,08214	79,6158	42,82506	64,1391
55,56709	55,14745	64,86573	48,14482	72,28587	62,75515
45,84217	58,02719	68,92293	78,08871	70,3337	61,05027
47,46931	49,50146	55,39114	60,68181	52,21385	40,56471
66,22114	48,85485	62,49053	55,39999	53,39967	40,97718
72,61024	43,12928	46,46656	58,92553	56,67577	43,01525
71,98864	40,23577	56,75907	58,02767	48,86791	40,80488
57,97213	50,50664	63,76674	64,86452	49,20096	51,69314
61,64906	56,13168	62,67404	80,99428	41,51067	41,83514
56,12095	56,55489	72,65446	68,81754	53,76774	63,91231
42,14645	68,54079	73,54232	63,25925	64,15732	46,67363
34,0197	45,32204	45,29131	54,67317	62,60519	39,73546
37,13741	54,7087	42,60632	50,13046	44,36513	65,55203
41,22444	67,01881	65,87994	52,29858	61,013	42,94699
53,22012	56,81571	51,549	73,91283	64,74513	50,60548
51,18861	56,90122	51,97898	66,34473	71,99778	38,02135
67,28111	47,66026	53,95005	51,48182	82,03628	46,38654
39,31367	61,69001	64,96548	67,83221	60,50557	51,4328
52,52551	60,51334	62,00703	68,61536	60,01382	45,15878

40,08523	56,87395	49,91731	58,49981	66,43913	42,06428
54,788	42,75156	40,30049	54,54104	88,27955	41,33057
46,4454	48,72091	69,75895	51,67847	85,28542	
64,78365	36,00398	48,47874	66,93414	54,97636	
55,35313	57,9824	73,39046	67,06931	88,69685	
70,75896	71,91811	61,84441	56,87725	48,84365	
47,10146	61,57063	55,72882	41,12015	69,3123	

Висновок: у ході лабораторної роботи розроблено програму приховання повідомлення у графічному файлі за допомогою стеганографічних перетворень з використанням мови C#.