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;

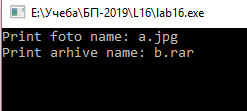
std::ifstream if\_a(namePf, std::ios\_base::binary);

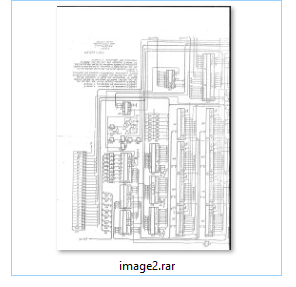
std::ifstream if\_b(nameAr, std::ios\_base::binary);

std::ofstream of\_c("result.jpg", std::ios\_base::binary);

of\_c << if\_a.rdbuf() << if\_b.rdbuf();

}



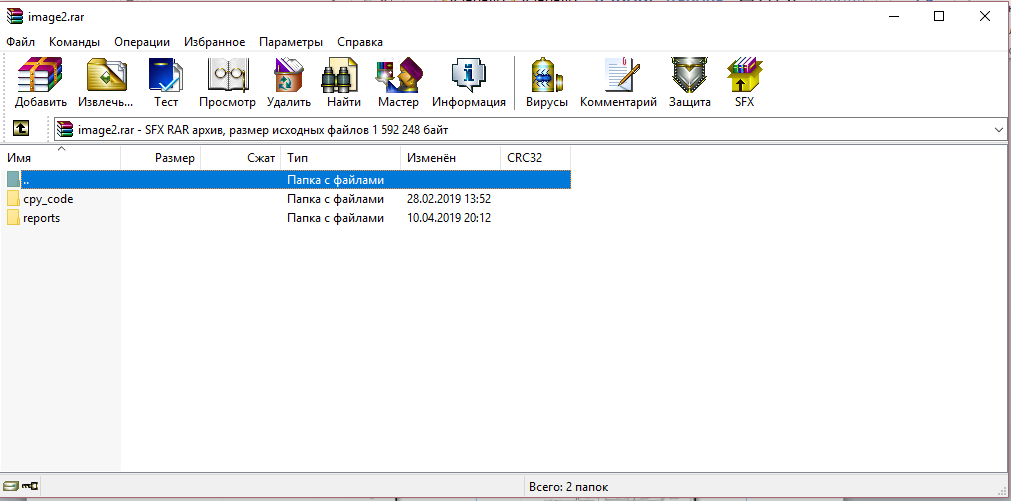


Рисунок 1 – Результат роботи

1. **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, возвраещает true, если символ в первом пикслеле 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):

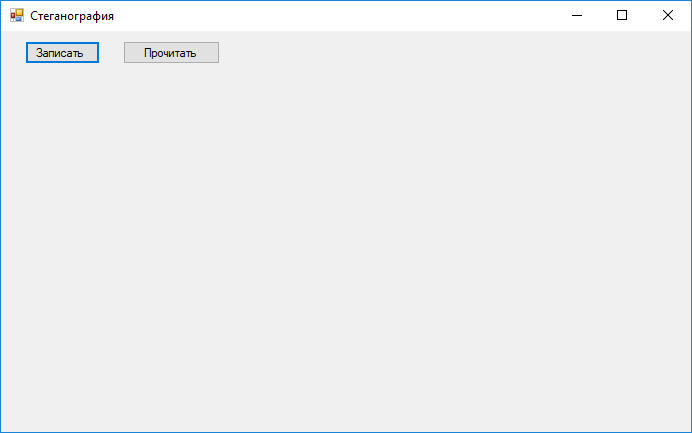


Рисунок 2

Шифруємо(рис.3-рис.4):

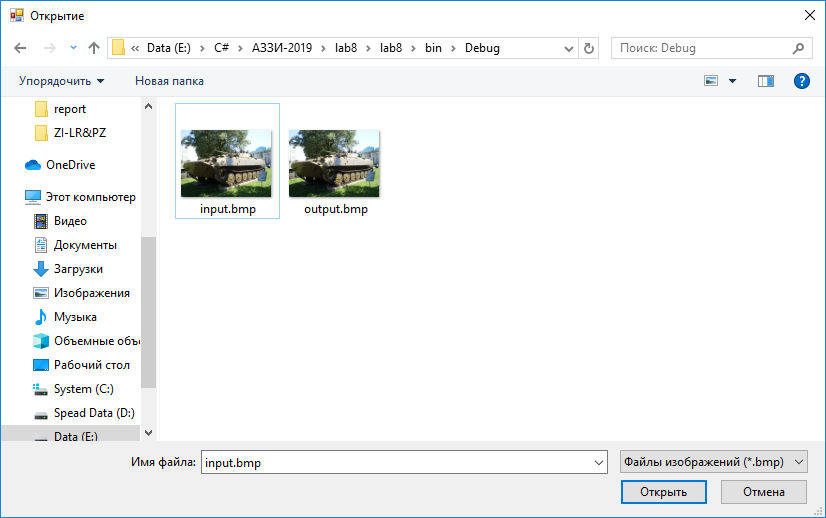


Рисунок 3

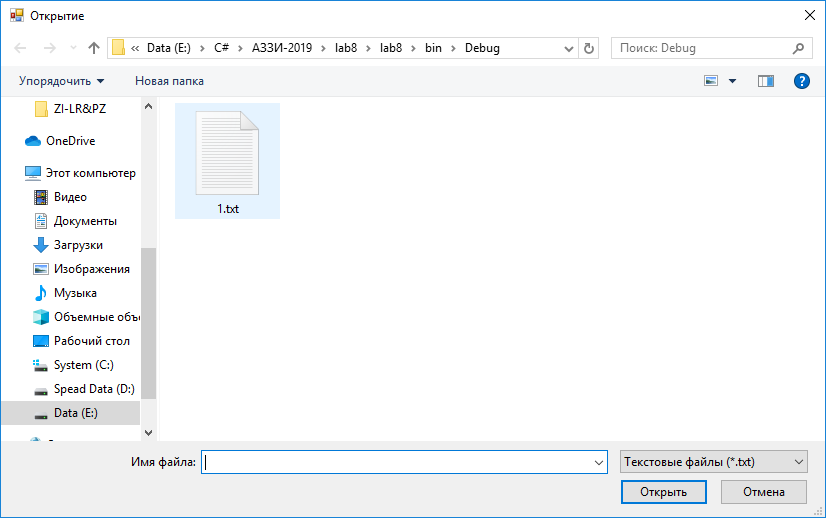


Рисунок 4

Після успішного шифрування отримуємо(рис.5):

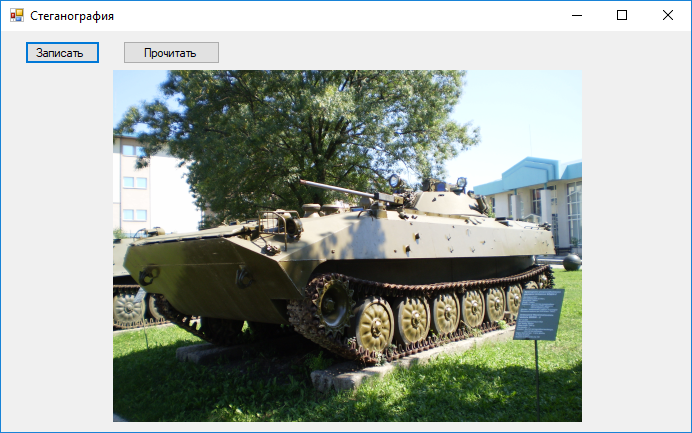


Рисунок 5

Вікно дешифрування(рис.6):

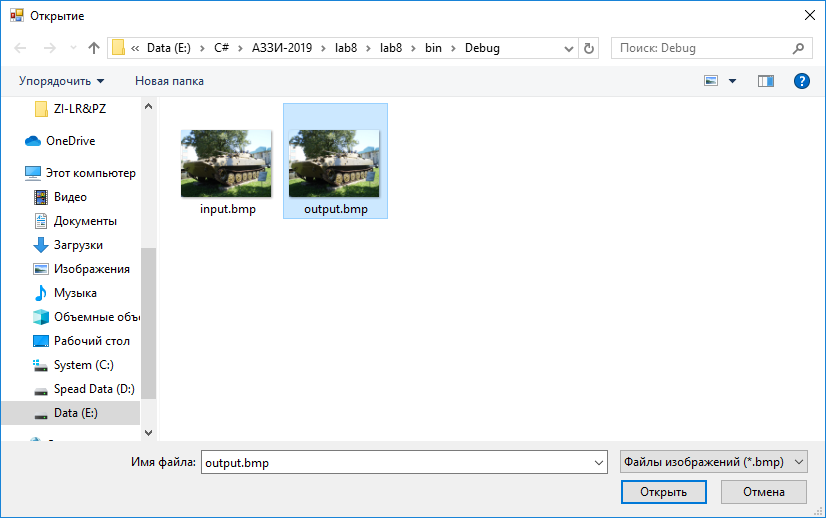


Рисунок 6

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

1. **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

6,602352

6,882231

10,4114

11,72546

10,08114

7,488267

9,154365

9,017269

9,071028

8,712497

15,49204

13,59591

4,368248

9,805822

10,2925

13,31227

15,37749

14,04251

14,48361

15,53108

18,71885

21,12788

17,07851

17,51955

18,39029

16,26345

18,51371

14,57256

16,22443

18,76369

19,08683

17,81503

18,43302

19,52293

24,00557

17,21871

15,77192

30,20258

19,97088

18,53673

27,31954

20,78723

17,30277

22,58024

16,61148

20,69745

19,95212

21,99485

21,10829

22,10979

22,91933

23,64973

28,77957

25,51016

21,07578

25,72509

25,63928

24,617

26,89802

28,15198

22,40869

24,35066

26,45418

21,77562

20,91942

24,84826

15,8903

24,00463

25,71889

26,1856

19,89659

22,7258

20,92413

23,55863

17,76337

22,54622

23,76629

21,39973

30,29913

22,88251

20,89263

23,06738

28,21039

26,11608

24,96853

32,25324

26,54238

28,89034

30,60033

29,36459

24,73317

27,01699

25,06766

29,417

29,90853

36,64503

32,45295

33,11042

32,52485

35,32042

29,54469

29,23939

32,25732

36,53647

37,91182

34,74085

38,0374

33,18106

39,00436

40,74645

32,1642

35,88916

34,77683

30,35462

28,86841

42,37957

33,35374

27,9877

36,30834

30,37259

35,60266

31,29341

36,29037

39,86333

39,49258

31,30599

31,51244

25,75222

32,28563

21,55489

33,00792

33,01551

39,56182

39,41772

28,46305

38,5104

31,92656

35,41357

30,17971

35,73144

45,44111

50,14728

47,79624

40,65534

35,58175

40,56166

37,25591

42,29782

39,11872

43,30504

31,76235

38,13852

43,4772

42,72965

46,1685

39,67209

42,10828

37,84917

34,03489

42,60714

43,84552

40,53698

36,6781

54,29446

47,8241

42,26297

49,2329

29,29431

34,69809

35,50072

38,30441

39,9416

42,60372

44,97622

34,10729

40,58337

34,54359

38,2799

33,84617

40,04724

33,51714

50,21315

46,96571

46,38053

48,75936

49,75933

40,7524

43,55427

31,97704

37,4986

31,8292

41,44631

42,21203

33,86383

30,93434

40,10492

29,16931

40,06855

37,91342

32,68763

43,81496

37,18126

30,46997

43,61493

41,94183

34,7701

35,42057

41,80043

34,17306

35,25652

39,25831

40,39367

35,876

34,1584

39,22456

33,49566

30,39234

39,38005

33,45901

37,36061

30,44549

46,98542

42,52323

43,03728

40,55154

42,98789

41,52327

49,71091

34,71738

50,14543

38,8946

40,0382

40,97203

54,97368

37,65459

36,67244

31,65229

49,43009

44,93486

38,26728

44,6308

45,11111

35,04748

35,84376

43,15501

43,98745

49,17834

38,50854

51,80996

44,39981

43,5591

37,65248

31,50911

39,74249

33,35456

39,88101

42,43529

39,89148

39,16735

38,49815

38,01804

44,00561

45,63758

41,12879

42,59253

44,80273

44,29272

48,75478

44,0441

44,4375

40,20526

46,71464

35,16461

35,31532

45,45911

49,50489

40,59302

39,42157

43,18629

41,72626

54,97544

45,83641

37,8209

38,39762

39,11563

39,51297

33,15423

34,73637

32,27465

42,82469

37,02657

33,55157

31,07365

36,77075

40,29977

31,96975

29,42743

40,52412

29,42583

31,58653

24,40223

29,93193

27,65165

37,03685

44,57619

31,82221

30,14185

36,91786

30,23227

29,77549

45,34048

41,00381

34,94019

31,41444

39,17629

40,35654

33,79017

29,86265

45,31366

39,48125

31,87279

37,39197

35,68297

35,03404

37,69159

32,86285

43,92839

37,13186

39,15253

34,37307

29,1833

37,01498

37,48056

34,29092

34,94159

22,92594

30,72241

36,91498

30,68996

29,29756

27,01113

34,83069

35,16977

26,06658

40,49747

40,71859

28,91859

25,64052

36,66591

31,03383

32,35425

25,62327

32,11322

34,21517

30,86046

31,42785

35,85347

27,66432

35,40471

40,32709

21,88405

33,45871

38,63847

29,50723

34,01437

37,79895

34,22886

34,93753

35,13897

34,39285

31,63271

27,59285

31,5464

37,40307

36,13053

34,56627

39,44002

38,92864

41,75237

27,90025

37,61999

43,30811

34,60318

37,09918

29,98084

34,37227

36,21127

34,19172

38,72752

39,57607

38,19257

40,71127

35,10137

43,9899

41,04857

36,80381

34,79934

39,58604

38,65533

47,12251

46,93114

43,71571

45,9427

47,12938

36,88138

47,52624

44,7867

41,92608

38,89398

29,98594

45,022

40,46099

39,50034

44,89994

32,92783

42,14116

36,39652

34,19759

30,93514

41,78849

33,13213

37,35632

39,44782

35,94606

31,72992

30,71639

37,14606

36,53402

32,3359

34,77979

31,08962

34,42636

Результат хи квадратів рядків зображення з вкладеною інформацією:

42,47453

54,4408

65,59309

83,59988

54,27842

39,26973

20,99938

26,85778

36,38498

23,66479

25,47545

66,75601

64,36694

47,26804

60,92719

45,21439

37,83136

58,70046

66,99435

79,36099

48,95057

36,50655

38,85399

42,12949

53,5116

31,38219

35,31183

39,10818

39,11304

41,38117

42,24396

63,681

51,08801

51,73277

51,1414

57,33867

50,60395

49,15891

41,64053

41,14334

44,04578

52,32316

54,56229

67,04174

69,25237

76,57677

73,47035

60,10399

55,56709

45,84217

47,46931

66,22114

72,61024

71,98864

57,97213

61,64906

56,12095

42,14645

34,0197

37,13741

41,22444

53,22012

51,18861

67,28111

39,31367

52,52551

76,11854

59,01086

47,03923

59,89692

69,57568

52,89431

72,52607

69,69469

72,81008

47,86933

32,37724

44,32936

44,9172

43,49032

51,27681

40,41265

63,75618

56,65478

44,72974

79,04656

53,22759

41,05281

45,53221

52,359

50,19253

52,92986

52,62099

60,63887

52,78805

56,22968

35,70281

60,37237

74,96816

62,2673

67,151

64,79539

60,89849

81,91236

66,12454

45,72314

40,58092

57,7103

52,79189

51,24915

49,93408

66,08736

51,72562

56,57706

55,14745

58,02719

49,50146

48,85485

43,12928

40,23577

50,50664

56,13168

56,55489

68,54079

45,32204

54,7087

67,01881

56,81571

56,90122

47,66026

61,69001

60,51334

42,77676

60,87002

50,41034

47,26809

67,1585

79,25841

61,53725

52,86876

57,97093

60,78809

67,25588

74,50051

58,5841

66,61465

66,17934

70,12425

69,8591

56,38532

66,93418

68,41708

76,45185

63,28655

62,71111

56,49332

56,70868

65,16451

53,27174

57,58268

53,44176

65,99594

56,01403

50,76433

73,34885

72,10496

58,67938

55,19647

59,40428

52,16621

49,74672

50,55278

56,24935

49,73215

45,44344

44,64343

66,18188

56,30234

59,33621

61,08214

64,86573

68,92293

55,39114

62,49053

46,46656

56,75907

63,76674

62,67404

72,65446

73,54232

45,29131

42,60632

65,87994

51,549

51,97898

53,95005

64,96548

62,00703

60,69299

70,41791

51,1603

57,5417

71,68217

51,2576

44,42337

48,71173

56,97649

33,75999

38,53487

47,62765

57,9001

67,92023

53,4127

48,58256

56,12619

47,95124

77,38607

66,57689

67,00589

61,40928

63,59022

81,16699

63,90564

64,13677

58,18629

72,73596

68,02479

64,86551

86,286

77,00802

65,7515

81,87862

83,57922

63,84426

64,14471

74,10134

59,39745

97,5839

88,20016

60,13615

70,64928

44,25771

51,54741

80,87861

83,3726

79,6158

48,14482

78,08871

60,68181

55,39999

58,92553

58,02767

64,86452

80,99428

68,81754

63,25925

54,67317

50,13046

52,29858

73,91283

66,34473

51,48182

67,83221

68,61536

66,89694

78,4477

55,3736

66,04953

67,65064

58,70349

61,56452

56,93673

67,83335

55,08116

79,62293

64,07375

47,24936

60,53736

77,36625

59,61318

52,75139

48,74036

62,10619

49,94729

50,73326

56,17179

57,20036

43,18262

72,1612

50,41838

72,81551

60,27036

43,37765

67,22698

48,93403

51,26209

50,22253

50,82164

52,40303

58,04147

55,28946

45,81351

55,43174

54,56239

44,36292

47,46068

49,55946

47,08215

58,31717

46,78362

59,65697

42,82506

72,28587

70,3337

52,21385

53,39967

56,67577

48,86791

49,20096

41,51067

53,76774

64,15732

62,60519

44,36513

61,013

64,74513

71,99778

82,03628

60,50557

60,01382

56,75117

64,64355

58,34174

40,61874

54,84371

46,66391

35,7288

36,99012

39,83581

51,83533

54,70419

44,39528

50,10118

47,46539

34,69413

65,3563

41,67408

50,24082

46,48997

39,42767

68,04929

45,02218

38,88076

43,57238

53,25527

59,35398

45,49922

40,64647

39,25702

47,10947

41,29514

33,66933

54,4841

45,1784

57,65304

52,78015

52,79571

43,41679

44,21023

53,00783

50,81821

54,52293

54,87702

43,30005

48,81794

39,86977

44,82701

64,1391

62,75515

61,05027

40,56471

40,97718

43,01525

40,80488

51,69314

41,83514

63,91231

46,67363

39,73546

65,55203

42,94699

50,60548

38,02135

46,38654

51,4328

45,15878

40,08523

54,788

46,4454

64,78365

55,35313

70,75896

47,10146

56,87395

42,75156

48,72091

36,00398

57,9824

71,91811

61,57063

49,91731

40,30049

69,75895

48,47874

73,39046

61,84441

55,72882

58,49981

54,54104

51,67847

66,93414

67,06931

56,87725

41,12015

66,43913

88,27955

85,28542

54,97636

88,69685

48,84365

69,3123

42,06428

41,33057

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