Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Информационная безопасность»

К защите

Руководитель проекта

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по дисциплине «Защищенные геоинформационные технологии и системы»

Выполнил студент группы 342 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Маркина Полина Павловна дата сдачи на проверку, подпись

Проверил старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Назарцев Валерий Викторович оценка дата защиты, подпись

Рязань 2018

ЗАДАНИЕ

## Кодирование изображений с использованием алгоритма группового сжатия

В соответствии с алгоритмом группового сжатия для изображений с кодировкой 8 бит на пиксель группа повторяющихся байтов заменяется двумя байтами: байтом-повторителем и повторяющимся байтом, рисунок 1.

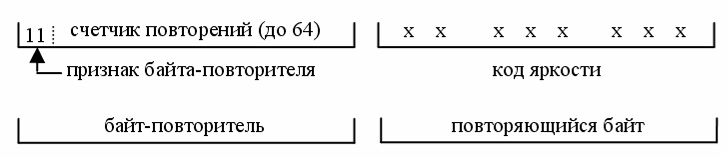


Рисунок 1 – Схема кодирования изображения по алгоритму группового сжатия

Для повторяющихся байтов поступают следующим образом. Байты со значением, меньшим C0, заносятся в файл без изменения. В противном случае значение байта пишется с байтом-повторителем, у которого счётчик равен 1.

При выполнении курсового проекта необходимо:

1) разработать алгоритм кодирования и декодирования изображений в формате TIFF с кодировкой два байта на пиксель. Код яркости каждого пикселя находится в диапазоне 0÷1023;

2) разработать программу кодирования, позволяющую открывать и визуализировать исходное изображение, сохранять закодированный файл;

3) разработать программу декодирования, позволяющую визуализировать закодированное изображение, сравнивать его с исходным и оценивать точность восстановления информации.

Исходное изображение для выполнения задания может быть получено у преподавателя.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc514563640)

[1 Анализ задачи и разработка структуры и интерфейса программы 5](#_Toc514563641)

[2 Разработка алгоритмов 6](#_Toc514563642)

[2.1 Алгоритм 1 (интерактивное задание области перекрытия изображений) 6](#_Toc514563643)

[2.2 Алгоритм 2 (фотометрическое выравнивание изображений) 8](#_Toc514563644)

[3 Программа и методики испытаний 9](#_Toc514563646)

[3.1 Тестирование методом «белого ящика» 9](#_Toc514563647)

[4 Разработка программной документации 14](#_Toc514563649)

[4.1 Руководство оператора 14](#_Toc514563650)

[4.1.1 Назначение и условия применения программы 14](#_Toc514563651)

[4.1.2 Характеристики программы 14](#_Toc514563652)

[4.1.3 Обращение к программе 14](#_Toc514563653)

[4.1.4 Входные и выходные данные 14](#_Toc514563654)

[4.2 Руководство системного программиста 14](#_Toc514563655)

[4.2.1 Назначение и условия применения программы 15](#_Toc514563656)

[4.2.2 Обращение к программе 15](#_Toc514563657)

[4.2.3 Входные и выходные данные 15](#_Toc514563658)

[ЗАКЛЮЧЕНИЕ 16](#_Toc514563659)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc514563660)

[ПРИЛОЖЕНИЕ 18](#_Toc514563661)

ВВЕДЕНИЕ

Цель курсового проектирования состоит в кодирование изображений с использованием алгоритма группового сжатия.

Результатом выполнения курсового проекта являются:

1) алгоритм кодирования и декодирования изображений в формате TIFF с кодировкой два байта на пиксель. Код яркости каждого пикселя находится в диапазоне 0÷1023;

2) программа кодирования, позволяющую открывать и визуализировать исходное изображение, сохранять закодированный файл;

3) программа декодирования, позволяющую визуализировать закодированное изображение, сравнивать его с исходным и оценивать точность восстановления информации.

Составление программы выполняется на языке программирования С#.

# Анализ задачи и разработка структуры и интерфейса программы

Поставленная в задании задача может быть разделена на 5 подзадач:

1) открытие базового (первого) изображения из файла формата TIFF с кодировкой два байта на пиксель. Код яркости каждого пикселя находится в диапазоне 0÷1023;

2) отображение изображения из файла формата TIFF;

3) кодирование изображения с использованием алгоритма группового сжатия;

4) декодирование изображения ;

5) сравнение объема исходного изображения с декодированным.

6) сравнение произвольных изображений

Каждая из них может быть реализована отдельными структурными частями программы (методами). Пусть каждый из методов 1, 2,3, 4, 5 ,6 будет вызываться нажатием на соответствующие кнопки.

Результат выполнения подзадач 1 и 2 – отображение изображения в области формы программы: слева – базовое, справа – выравниваемое.

Результат выполнения подзадачи 3 – кодирование изображения в формате .TIFF с использованием алгоритма группового сжатия.

Результат выполнения подзадачи 4 – декодирование изображения произвольного формата .pol.

Результат выполнения подзадачи 5 – сравнение объема исходного изображения с декодированным.

Результат выполнения подзадачи 6 – сравнение произвольных изображений загружаемых пользователем.

# Разработка алгоритмов

В пояснительной записке к данному курсовому проекту описываются только те алгоритмы, которые выполняют основную задачу рассматриваемого задания, а именно:

1) кодирование изображения формата .TIFF;

2) декодирование изображения произвольного формата .pol;

## Алгоритм 1 (кодирование изображения формата .TIFF)

Алгоритм 1 .

Рисунок 2 – Алгоритм кодирования изображения по алгоритму группового сжатия.



## Алгоритм 2 (фотометрическое выравнивание изображений)

Алгоритм 2 .

Рисунок 3 – Алгоритм декодирования файла.



# Программа и методики испытаний

## Тестирование методом «белого ящика»

Алгоритм кодирования изображения по алгоритму группового сжатия(Алгоритм 2). Исходный код метода:

private void button8\_Click(object sender, EventArgs e)

1) {

2) if (pictureBox1.Image != null & pictureBox2.Image != null)

3) {

4) if (sizeofimage == sizeofimage2)

5) MessageBox.Show("Размер файлов одинаков!");

6) else

7) MessageBox.Show("Размер файлов не совпадает!");

8) }

9) else

10) {

11) if (pictureBox1.Image == null & pictureBox1.Image == null)

12) MessageBox.Show("Загрузите изображения !");

13) else

14) if (pictureBox1.Image == null)

15) MessageBox.Show("Загрузите исходное изображение!");

16) else

17) if (pictureBox2.Image == null)

18) MessageBox.Show("Загрузите изображение для декодирования!");

19) }

20) }

Сформируем потоковый граф (рисунок 3).

Определим цикломатическое число графа:

1)V(G) = 24дуг – 20 узлов + 2 = 6

2)V(G) = 5 предикатных узла + 1 = 6

Рисунок 3 - Потоковый граф



Рисунок 3 - Потоковый граф

Цикломатическое число графа равно 6. Таким образом, множество независимых путей состоит из 6-и путей:

Путь 1: 1-2-3-4-5-19-20;

Путь 2: 1-2-3-4-6-7-8-19-20;

Путь 3: 1-2-9-10-11-12-19-20;

Путь 4: 1-2-9-10-11-13-14-15-19-20;

Путь 5: 1-2-9-10-11-14-16-17-19-20;

Путь 6: 1-2-9-10-11-14-16-17-18-19-20;

Таблица - Метод "белого ящика"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ пути** | **Исходные данные** | **Ожидаемые результаты** | **Реальные результаты** | **Полученные данные совпадают с ожидаемыми?** |
| 1 | Сравнение ранее загруженных двух изображений | Изображения равны по объему | Изображения равны по объему | Да |
| 2 | Сравнение ранее загруженных двух изображений | Изображения не равны по объему | Изображения не равны по объему | Да |
| 3 | Если пытаемся сравнить два незагруженных изображения | Изображения будут сравнены | Два изображения не могут быть сравнены | Да |
| 4 | Если хотя бы одно из изображений не загружено и мы пытаемся сравнить исходное с пустым незагруженным | Изображения будут сравнены | Два изображения не могут быть сравнены | Да |
| 5 | Если хотя бы одно из изображений не загружено и мы пытаемся сравнить исходное пустое с загруженным кодированным | Изображения будут сравнены | Два изображения не могут быть сравнены | Да |

## Тестирование методом «черного ящика»

Входными значениями метода являются:

* Изображение в формате tiff

Разделим область данных на классы эквивалентности (КЭ):

* предложение открыть изображение отменено;
* предложение закодировать изображение без его вывода на экран;
* предложение сравнить исходное изображение с отсутствующим;
* предложение сохранить 2 отсутствующих изображения;

Таблица - Метод "черного ящика"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ КЭ** | **Исходные данные** | **Ожидаемые результаты** | **Реальные результаты** | **Полученные данные совпадают с ожидаемыми?** |
| 1 | Предложение открыть изображение отменено (рис.5) | Отмена открытия изображения (ничего не открылось) | Открытие изображения отменено (ничего не открылось) | Да |
| 2 | Предложение закодировать изображение без его вывода на экран (рис.6) | Изображение закодируется | Без вывода на экран исходного изображения, кодирование не происходит | Да |
| 3 | Предложение сравнить объем исходного изображения с объемом отсутствующего (рис.7) | Объем изображения сравнится с чем-то | Объем изображения не был сравнен ни с чем | Да |
| 4 | Предложение сравнить 2 отсутствующих изображения (рис.8) | Изображения будут сравнены | Без вывода сравниваемых изображений ничего не сравнится | Да |

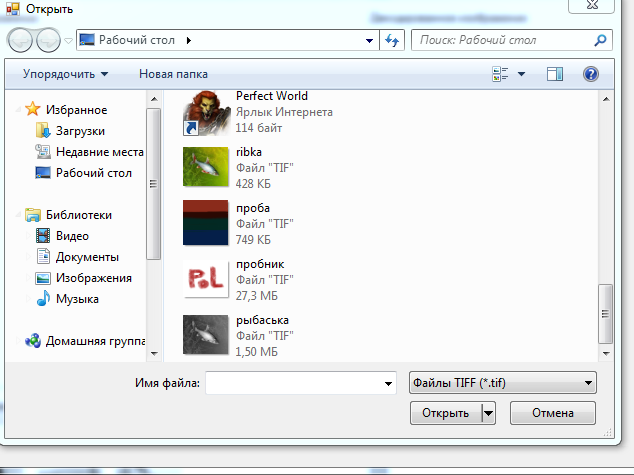


Рисунок 5 - Предложение открыть изображение отменено

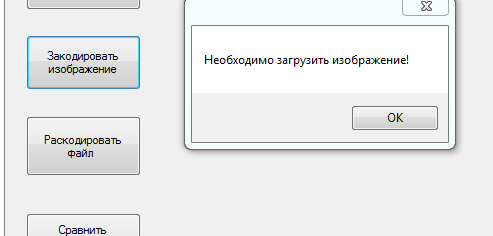


Рисунок 6 - Предложение закодировать изображение без его вывода на экран

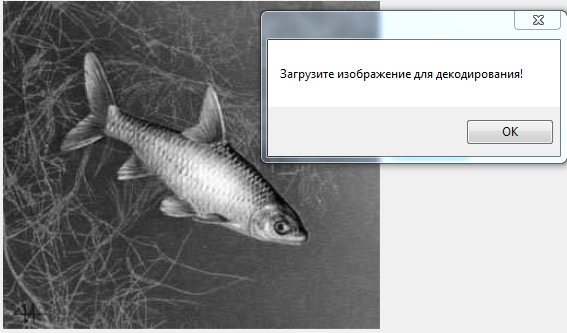


Рисунок 7 - Предложение сравнить объем исходного изображения с объемом отсутствующего

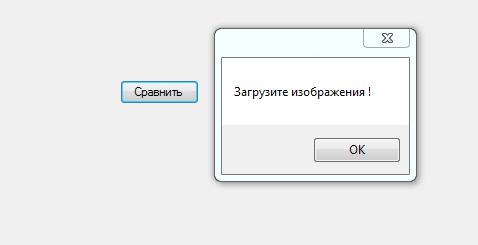


Рисунок 8 - Предложение сравнить 2 отсутствующих изображения

# Разработка программной документации

## Руководство оператора

### Назначение и условия применения программы

Кодировать изображение в формате .TIFF кодировкой два байта на пиксель. Код яркости каждого пикселя находится в диапазоне 0÷1023 :

1) открытие изображения в формате TIFF;

## 2) Кодирование изображений с использованием алгоритма группового сжатия;

Программа разработана для использования на компьютерах, оснащенных операционной системой Windows 8 и выше.

Для работы программы необходимо иметь:

* персональный компьютер;
* объем оперативной памяти не менее 20 Мб;
* монитор;
* 20 Мб свободной памяти на жестком магнитном диске;
* мышь;
* клавиатура.

### Характеристики программы

Данная программа позволяет кодировать изображение в формате .TIFF кодировкой два байта на пиксель. Код яркости каждого пикселя находится в диапазоне 0÷1023 с использованием алгоритма группового сжатия.

### Обращение к программе

Программа поставляется в виде файла WindowsFormsApplication1.exe , который необходимо запустить на выполнение.

### Входные и выходные данные

При работе с программой в качестве входных данных используются:

* изображение из файлов формата .TIFF;

Выходными данными в программе является закодированное изображение методом группового сжатия.

## Руководство системного программиста

### Назначение и условия применения программы

Программа должна выполнять фотометрическое выравнивание изображений, а именно:

1) открытие изображение из файлов формата .TIFF;

2) кодирование изображений с использованием алгоритма группового сжатия;

Программа разработана для использования на компьютерах, оснащенных операционной системой Windows 8 и выше.

Для работы программы необходимо иметь:

* персональный компьютер;
* объем оперативной памяти не менее 20 Мб;
* монитор;
* 20 Мб свободной памяти на жестком магнитном диске;
* мышь;
* клавиатура.

### Обращение к программе

Программа поставляется в виде файла WindowsFormsApplication1.exe, который необходимо запустить на выполнение.

Для открытия изображений необходимо произвести следующие действия:

1) нажать на кнопку «Вывести изображение»;

2) выбрать необходимый файл TIFF;

3) нажать кнопку «Открыть» либо «Отмена» для отклонения действий.

Для фотометрического выравнивания необходимо произвести следующие действия:

1) нажать на кнопку «Запустить фотометрическое выравнивание изображений»;

2) результат выравнивания отобразится на форме.

Для кодирования изображения с использованием алгоритма группового сжатия:

1) нажать на кнопку «Сохранить результат»;

2) дать имя новому файлу, выбрать каталог;

3) нажать на кнопку «Сохранить».

### Входные и выходные данные

При работе с программой в качестве входных данных используются:

-открытие изображения в формате TIFF;

## - кодирование изображений с использованием алгоритма группового сжатия;

Выходными данными в программе являются закодированное изображение формата .TIFF методом группового сжатия.

ЗАКЛЮЧЕНИЕ

В результате выполнения данного курсового проектирования достигнута поставленная цель. Результатом выполнения курсового проекта стали:

1) алгоритм кодирования и декодирования изображений в формате TIFF с кодировкой два байта на пиксель. Код яркости каждого пикселя находится в диапазоне 0÷1023;

2) программа кодирования, позволяющую открывать и визуализировать исходное изображение, сохранять закодированный файл;

3) программа декодирования, позволяющую визуализировать закодированное изображение, сравнивать его с исходным и оценивать точность восстановления информации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Защищенные геоинформационные технологии и системы: Методические указания к курсовому проекту / Рязан. гос. радиотехн. ун-т; Сост.: А.Е.Кузнецов, В.И. Побаруев. - Рязань, 2011. 16 с.

2. Методические указания к курсовому проектированию по дисциплине «Защищенные геоинформационные технологии и системы» / Рязан. гос. радиотехн. ун-т; - Рязань, 2012. 16 с.

ПРИЛОЖЕНИЕ

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

bool pictlod = false;

string filename;

byte[] ByteArray;

byte[] ByteArray2;

long sizeofimage,sizeofimage2;

public Form1()

{

InitializeComponent();

}

private void button3\_Click(object sender, EventArgs e)

{

var openFileDialog1 = new OpenFileDialog();

openFileDialog1.DefaultExt = "\*.TIFF";

openFileDialog1.Filter = "Файлы TIFF (\*.tif)|\*.tif";

if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

FileStream fstream = File.OpenRead(openFileDialog1.FileName);

pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);

filename = openFileDialog1.FileName;

ByteArray = new byte[fstream.Length]; //создали массив размеров с файл , загрузили туда байты

fstream.Read(ByteArray, 0, ByteArray.Length);

fstream.Dispose();

pictlod = true;

}

else {

pictlod = false;

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (pictlod == false)

{

MessageBox.Show("Необходимо загрузить изображение!");

return;

}

string path = "";

if (path.Length == 0)

{

SaveFileDialog save = new SaveFileDialog();

save.DefaultExt = "\*.POLPAL";

save.Filter = "Файлы POLPAL (\*.pol)|\*.pol";

if (save.ShowDialog() == DialogResult.OK)

{

path = save.FileName; //тут просто диалоговое окно открывалось для названия файла

}

else

{

return;

}

}

FileStream fstream = new FileStream(path, FileMode.OpenOrCreate); //создали поток файла

BinaryWriter compr = new BinaryWriter(fstream);

long i = 0;

int povt = 1;

byte cv;

sizeofimage = ByteArray.Length;

while (i < sizeofimage-1) //открываем цикл ,будем все делать до тех пор пока массив не кончится

{

povt = 1;

if ((i + 3)> (sizeofimage - 1)) // если мы пришли к концу файла и у нас осталось

//2 байта,конечно мы их не будем сравнивать со следующими ,у нас конец массива

{

if (ByteArray[i] < 0xC0)

{

compr.Write(ByteArray[i]);

compr.Write(ByteArray[i + 1]); //кароч если байт меньше С0 то пишем его таким какой он есть (неповторяющийся байт)

break;

}

else

{

cv = Convert.ToByte(0xC1);

compr.Write(cv);

compr.Write(ByteArray[i]); // если байт больше С0,то пишем С1-значение байта повторителя+значение байта

compr.Write(ByteArray[i + 1]);

break;

}

}

if (ByteArray[i] == ByteArray[i + 2] & ByteArray[i + 1] == ByteArray[i + 3]) //если байты повторяются

{

while (ByteArray[i] == ByteArray[i + 2] & ByteArray[i + 1] == ByteArray[i + 3])

{ //тут уже все интереснее,мы сравниваем байт со следующим байтом ,если одинаковые стакуем повторения

povt++;

i = i + 2;

if (povt == 63) //помним ,что повторения нельзя превышать 64,настаковали 63,обнулили группу

{

cv = Convert.ToByte(povt + 0xC0);

compr.Write(cv);

compr.Write(ByteArray[i]);

compr.Write(ByteArray[i + 1]);

povt = 1;

i = i + 2;

}

}

cv = Convert.ToByte(povt + 0xC0); //тут все пишем норм ,если повторений не много (cv-счетчик)

compr.Write(cv);

compr.Write(ByteArray[i]);

compr.Write(ByteArray[i + 1]);

povt = 1;

i = i + 2;

}

else //если байты одиночны

{

if (ByteArray[i] < 0xC0)

{

compr.Write(ByteArray[i]);

compr.Write(ByteArray[i + 1]); //кароч если байт меньше С0 то пишем его таким какой он есть (неповторяющийся байт)

i = i + 2;

povt = 1;

}

else

{

cv = Convert.ToByte(0xC1);

compr.Write(cv);

compr.Write(ByteArray[i]); // если байт больше С0,то пишем С1-значение байта повторителя+значение байта

compr.Write(ByteArray[i + 1]);

i = i + 2;

povt = 1;

}

}

}

fstream.Close();

compr.Close();

}

private void button4\_Click(object sender, EventArgs e)

{

Close();

}

private void button1\_Click(object sender, EventArgs e)

{

}

private void button1\_Click\_1(object sender, EventArgs e)

{

var openFileDialog1 = new OpenFileDialog();

openFileDialog1.DefaultExt = "\*.POLPAL";

openFileDialog1.Filter = "Файлы POLPAL (\*.pol)|\*.pol";

if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

FileStream fstream = File.OpenRead(openFileDialog1.FileName);

filename = openFileDialog1.FileName;

ByteArray2 = new byte[fstream.Length]; //создали массив размеров с файл , загрузили туда байты

fstream.Read(ByteArray2, 0, ByteArray2.Length);

fstream.Dispose();

pictlod = true;

}

else

{

return;

}

string path = "";

if (path.Length == 0)

{

SaveFileDialog save = new SaveFileDialog();

save.DefaultExt = "\*.TIFF";

save.Filter = "Файлы TIFF (\*.tif)|\*.tif";

if (save.ShowDialog() == DialogResult.OK)

{

path = save.FileName; //тут просто диалоговое окно открывалось для названия файла

}

else

{

return;

}

}

FileStream fstream2 = new FileStream(path, FileMode.OpenOrCreate); //создали поток файла

BinaryWriter compr2 = new BinaryWriter(fstream2);

long i = 0;

int povt = 0;

sizeofimage2 = ByteArray2.Length;

byte[] BufArray = new byte[2] ; //2 элемента

while (i < sizeofimage2 - 1) //открываем цикл ,будем все делать до тех пор пока массив не кончится

{

if (ByteArray2[i] < 0xC0)

{

compr2.Write(ByteArray2[i]);

compr2.Write(ByteArray2[i + 1]); //кароч если байт меньше С0 то пишем его таким какой он есть (неповторяющийся байт)

i = i + 2;

}

else

{

povt = ByteArray2[i] - 0xC0;

BufArray[0] = ByteArray2[i + 1];

BufArray[1] = ByteArray2[i + 2];

for (int j = povt; j >= 1; j--)

{

compr2.Write(BufArray[0]);

compr2.Write(BufArray[1]);

}

BufArray[0] = 0;

BufArray[1] = 0;

povt = 0;

i = i + 3;

}

}

fstream2.Close();

compr2.Close();

pictureBox2.Image = Image.FromFile(path);

}

private void button5\_Click(object sender, EventArgs e)

{

if (pictureBox1.Image != null)

{

pictureBox1.Image.Dispose();

pictureBox1.Image = null;

pictlod = false;

}

}

private void button6\_Click(object sender, EventArgs e)

{

byte[] ByteSrav;

byte[] ByteSrav2;

long sizeofimage1;

long sizeofimage3;

var openFileDialog1 = new OpenFileDialog();

openFileDialog1.DefaultExt = "\*.TIFF";

openFileDialog1.Filter = "Файлы TIFF (\*.tif)|\*.tif";

if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

FileStream fstream = File.OpenRead(openFileDialog1.FileName);

pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);

filename = openFileDialog1.FileName;

ByteSrav = new byte[fstream.Length];

sizeofimage1 = ByteSrav.Length;

}

else

{

return;

}

var openFileDialog2 = new OpenFileDialog();

openFileDialog2.DefaultExt = "\*.TIFF";

openFileDialog2.Filter = "Файлы TIFF (\*.tif)|\*.tif";

if (openFileDialog2.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

FileStream fstream = File.OpenRead(openFileDialog2.FileName);

pictureBox2.Image = Image.FromFile(openFileDialog2.FileName);

filename = openFileDialog2.FileName;

ByteSrav2 = new byte[fstream.Length];

sizeofimage3 = ByteSrav2.Length;

}

else

{

return;

}

if (sizeofimage1 == sizeofimage3)

MessageBox.Show("Размер файлов одинаков!");

else

MessageBox.Show("Размер файлов не совпадает!");

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void button8\_Click(object sender, EventArgs e)

{

if (pictureBox1.Image != null & pictureBox2.Image != null)

{

if (sizeofimage == sizeofimage2)

MessageBox.Show("Размер файлов одинаков!");

else

MessageBox.Show("Размер файлов не совпадает!");

}

else

{

if (pictureBox1.Image == null & pictureBox1.Image == null)

MessageBox.Show("Загрузите изображения !");

else

if (pictureBox1.Image == null)

MessageBox.Show("Загрузите исходное изображение!");

else

if (pictureBox2.Image == null)

MessageBox.Show("Загрузите изображение для декодирования!");

}

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button7\_Click(object sender, EventArgs e)

{

if (pictureBox2.Image != null)

{

pictureBox2.Image.Dispose();

pictureBox2.Image = null;

}

}

}

}