Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное   
образовательное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

Лабораторная Работа №1. ЧАСТОТНЫЙ АНАЛИЗАТОР

Отчет по лабораторной работе №1 по учебной дисциплине   
«Методы защиты информации»

по специальности 09.03.01 Информатика и вычислительная техника

|  |  |
| --- | --- |
|  | Руководитель  / Т. В. Жгун  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |
|  | Студент группы 3091  / Р. А. Михайлов  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

Оглавление

[План работы 3](#_Toc181634428)

[Теоретические положения 4](#_Toc181634429)

[Алгоритм работы программы 5](#_Toc181634430)

[Приложение А 6](#_Toc181634431)

[Приложение Б 22](#_Toc181634432)

# План работы

1. Реализовать инструмент, обеспечивающий частотный анализ файлов графического (\*.bmp, 3 канала по 8 бит), и текстового формата (\*.txt, 8 бит). Кодировка символов – ASCII представлена в таблице 1. Все форматы представлены числами в диапазоне [0, 255].

Анализ файла представляет собою гистограмму текста и таблицу встречаемости символов. Все гистограммы должны быть реализованы для одного диапазона [0, 255].

Реализованный частотный анализатор должен использоваться при выполнении следующих лабораторных работ.

1. Проанализировать распределение символов для графических файлов различного свойства (картинка с преобладанием одного цвета, картинка в графическом редакторе, фото, …) и текстов разного свойства (текст учебника, статья в СМИ, литературный текст, переписка в соцсетях, …). Объем текстового файла не менее 2 тысяч символов.

В отчете представить не менее 4х гистограмм для файлов каждого типа

1. Вычислить информационную меру рассматриваемых файлов, используя информацию шага 1. Обосновать возможности сжатия текстов

# Теоретические положения

Частотный анализ — это метод, используемый в криптографии и лингвистике для изучения частоты появления различных символов, букв или слов в тексте. Он основан на предположении, что в любом языке определенные символы или комбинации символов встречаются с разной частотой.

В криптографии частотный анализ может быть использован для взлома шифров, особенно простых, таких как шифр Цезаря или шифр подстановки. Например, если в зашифрованном тексте буква "E" (которая является самой частой буквой в английском языке) встречается чаще всего, то можно предположить, что она соответствует наиболее часто встречающейся букве в зашифрованном сообщении.

Одной из функций программы будет возможность вычисления энтропии данного файла. Энтропия в теории информации измеряет среднее количество информации, которое содержится в сообщении, и определяется как мера неопределенности, связанной с вероятностным распределением возможных исходов.

Низкая энтропия: если система имеет низкую энтропию, это означает, что она предсказуема. Например, если в тексте часто повторяются одни и те же слова, то его энтропия будет низкой, так как информация в нем менее разнообразна.

Высокая энтропия: высокая энтропия указывает на большую степень неопределенности. Например, если в тексте используются множество уникальных слов с равной вероятностью, то его энтропия будет высокой, так как предсказать следующее слово будет сложно.

Энтропия вычисляется по следующей формуле:

# Алгоритм работы программы

Для реализации частотного анализатора алгоритмическим языком был выбран C++, язык для написания графического интерфейса – C++/CLI (платформа Microsoft Windows Forms), код программы представлен в Приложении А.

Для работы с программой необходимо запустить FrequencyAnalysis.exe. После запуска на экране будет представлен графический интерфейс приложения:

– Координатная плоскость, где ось x – числовое значение символа/пикселя, ось y – количество символов/пикселей с данным числовым значением

– Таблица в формате <числовое значение>. <количество символов/пикселей>, внизу таблицы находится значение энтропии выбранного файла.

– Верхняя часть окна приложения – меню с возможностью выбора файла, настройкой гистограммы и кнопкой выхода из приложения.

После выбора файла программа проходит по данному тексту/изображению и сохраняет в структуру данных std::map (ключ – значение, в нашем случае ключ это числовое значение пикселя/символа, а значение это количество символов/пикселей с данным числовым значением) количество символов с данным значением.

После того, как файл был проанализирован, программа отображает полученные данные в виде гистограммы, полученные гистограммы представлены в Приложении Б.

В таблице будут выведены числовые значения и количество символов/пикселей с такими числовыми значениями, внизу таблицы выведена энтропия данного файла.

# Приложение А

Листинг файла Analisys.h

#include <iostream>

#include <string>

#include <fstream>

#include <unordered\_map>

#include <Windows.h>

#include <cmath>

void init\_map(std::unordered\_map<char, int>& data)

{

for (int i = 0; i < 256; i++)

{

data.insert({ (char)i, 0 });

}

}

std::unordered\_map<int, int> init\_map\_RGB()

{

std::unordered\_map<int, int> data;

for (int i = 0; i < 256; i++)

{

data.insert({ i, 0 });

}

return data;

}

int find\_max(std::unordered\_map<int, int> data)

{

int max = data[0];

for (int i = 0; i < 256; i++)

{

if (data[i] > max)

max = data[i];

}

return max;

}

int find\_max(std::unordered\_map<char, int>& data)

{

int max = data['\0'];

for (int i = 0; i < 256; i++)

{

if (data[(char)i] > max)

{

max = data[(char)i];

}

}

return max;

}

long countTotal(std::unordered\_map<int, int> data)

{

long total = 0;

for (int i = 0; i < 256; i++)

{

total += data[i];

}

return total;

}

long countTotal(std::unordered\_map<char, int> data)

{

long total = 0;

for (int i = 0; i < 256; i++)

{

total += data[(char)i];

}

return total;

}

std::string get\_text(std::string path)

{

std::string data = "";

std::ifstream in(path);

if (in.is\_open())

{

std::string temp;

while (std::getline(in, temp))

{

data += temp;

}

}

return data;

}

void text\_to\_diagram(std::string path, std::unordered\_map<char, int>& data)

{

std::string text = get\_text(path);

for (int i = 0; i < text.size(); i++)

{

data[text[i]]++;

}

}

void ReadBMP(std::string filename, std::unordered\_map<int, int>& R, std::unordered\_map<int, int>& G, std::unordered\_map<int, int>& B)

{

R = init\_map\_RGB();

G = init\_map\_RGB();

B = init\_map\_RGB();

std::ifstream in(filename, std::ios\_base::binary);

if (!in.is\_open())

throw "Argument Exception";

unsigned char info[54]; //Массив заголовка файла bmp

in.read((char\*)info, sizeof(char) \* 54); // Чтение в массив заголовка файла

int width = \*(int\*)&info[18]; // Получение ширины изображения из массив

int height = \*(int\*)&info[22]; // Получение высоты изображения из массив

int row\_padded = (width \* 3 + 3) & (~3);

unsigned char\* data = new unsigned char[row\_padded];

char tmp;

std::string result = "";

for (int i = 0; i < height; i++)

{

in.read((char\*)data, sizeof(char) \* row\_padded);

for (int j = 0; j < width \* 3; j += 3)

{

tmp = data[j];

data[j] = data[j + 2];

data[j + 2] = tmp;

R[(int)data[j]]++;

G[(int)data[j + 1]]++;

B[(int)data[j + 2]]++;

}

}

in.close();

}

float entropy(std::unordered\_map<int, int> data)

{

long total = countTotal(data);

float result = 0.0;

for (int i = 0; i < 256; i++)

{

if (data[i] != 0)

{

double posibility = (double)data[i] / (double)total;

result += posibility \* log2(1 / posibility);

}

}

return round(result \* 1000) / 1000;

}

float entropy(std::unordered\_map<char, int> data)

{

long total = countTotal(data);

float result = 0.0;

for (int i = 0; i < 256; i++)

{

if (data[(char)i] != 0)

{

double posibility = (double)data[(char)i] / (double)total;

result += posibility \* log2(1 / posibility);

}

}

return round(result \* 1000) / 1000;

}

Листинг файла MyForm.cpp

#pragma optimize("agi", off)

#include "MyForm.h"

#include <Windows.h>

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

int main()

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

FreaquencyAnalisys::MyForm form;

Application::Run(% form);

}

Листинг файла MyForm.h

#pragma once

#include "Analisys.h"

#include <msclr\marshal\_cppstd.h>

std::unordered\_map<int, int> R, G, B;

namespace FreaquencyAnalisys {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::IO;

using namespace System::Windows::Forms::DataVisualization::Charting;

/// <summary>

/// Сводка для MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

openFileDialog1->Filter = "Текстовый файл (\*.txt)|\*.txt|Изображение (\*.bmp)|\*.bmp";

this->chart1->Series["Series1"]->Points->AddXY(0, 0);

this->chart1->ChartAreas[0]->AxisX->ScaleView->Zoomable = true;

this->chart1->ChartAreas[0]->AxisY->ScaleView->Zoomable = true;

this->trackBar1->Value = 5;

this->trackBar2->Value = 10;

this->chart1->Series->Add("R");

this->chart1->Series["R"]->Color = System::Drawing::Color::Red;

this->chart1->Series->Add("G");

this->chart1->Series["G"]->Color = System::Drawing::Color::Green;

this->chart1->Series->Add("B");

this->chart1->Series["B"]->Color = System::Drawing::Color::Blue;

this->скрытьКрасныйRКаналToolStripMenuItem->Visible = false;

this->скрытьЗелёныйGКаналToolStripMenuItem->Visible = false;

this->скрытьСинийBКаналToolStripMenuItem->Visible = false;

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::DataVisualization::Charting::Chart^ chart1;

private: System::Windows::Forms::MenuStrip^ menuStrip1;

private: System::Windows::Forms::ToolStripMenuItem^ файлToolStripMenuItem;

private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::TrackBar^ trackBar1;

private: System::Windows::Forms::TrackBar^ trackBar2;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::ToolStripMenuItem^ закрытьПриложениеToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ настройкаГистрограммыToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ выключитьСеткуToolStripMenuItem;

private: int max = 0;

private: System::Windows::Forms::ToolStripMenuItem^ скрытьКрасныйRКаналToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ скрытьЗелёныйGКаналToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ скрытьСинийBКаналToolStripMenuItem;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::Windows::Forms::DataVisualization::Charting::ChartArea^ chartArea1 = (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());

System::Windows::Forms::DataVisualization::Charting::Legend^ legend1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Legend());

System::Windows::Forms::DataVisualization::Charting::Series^ series1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());

this->chart1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Chart());

this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());

this->файлToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->настройкаГистрограммыToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->выключитьСеткуToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->скрытьКрасныйRКаналToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->скрытьЗелёныйGКаналToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->скрытьСинийBКаналToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->закрытьПриложениеToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());

this->label1 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->trackBar1 = (gcnew System::Windows::Forms::TrackBar());

this->trackBar2 = (gcnew System::Windows::Forms::TrackBar());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label4 = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

this->label6 = (gcnew System::Windows::Forms::Label());

this->label7 = (gcnew System::Windows::Forms::Label());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->chart1))->BeginInit();

this->menuStrip1->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar2))->BeginInit();

this->SuspendLayout();

//

// chart1

//

this->chart1->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Bottom)

| System::Windows::Forms::AnchorStyles::Left)

| System::Windows::Forms::AnchorStyles::Right));

chartArea1->AlignmentOrientation = System::Windows::Forms::DataVisualization::Charting::AreaAlignmentOrientations::None;

chartArea1->AxisX->Maximum = 255;

chartArea1->AxisY->IsLabelAutoFit = false;

chartArea1->Name = L"ChartArea1";

this->chart1->ChartAreas->Add(chartArea1);

legend1->AutoFitMinFontSize = 14;

legend1->Enabled = false;

legend1->Name = L"Legend1";

this->chart1->Legends->Add(legend1);

this->chart1->Location = System::Drawing::Point(618, 42);

this->chart1->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->chart1->Name = L"chart1";

series1->ChartArea = L"ChartArea1";

series1->Legend = L"Legend1";

series1->LegendText = L"Количесвто символов в тексте";

series1->Name = L"Series1";

this->chart1->Series->Add(series1);

this->chart1->Size = System::Drawing::Size(1278, 1003);

this->chart1->TabIndex = 0;

this->chart1->Text = L"chart1";

//

// menuStrip1

//

this->menuStrip1->GripMargin = System::Windows::Forms::Padding(2, 2, 0, 2);

this->menuStrip1->ImageScalingSize = System::Drawing::Size(24, 24);

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {

this->файлToolStripMenuItem,

this->настройкаГистрограммыToolStripMenuItem, this->закрытьПриложениеToolStripMenuItem

});

this->menuStrip1->Location = System::Drawing::Point(0, 0);

this->menuStrip1->Name = L"menuStrip1";

this->menuStrip1->Size = System::Drawing::Size(1896, 33);

this->menuStrip1->TabIndex = 1;

this->menuStrip1->Text = L"menuStrip1";

//

// файлToolStripMenuItem

//

this->файлToolStripMenuItem->Name = L"файлToolStripMenuItem";

this->файлToolStripMenuItem->Size = System::Drawing::Size(142, 29);

this->файлToolStripMenuItem->Text = L"Выбрать файл";

this->файлToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::файлToolStripMenuItem\_Click);

//

// настройкаГистрограммыToolStripMenuItem

//

this->настройкаГистрограммыToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(4) {

this->выключитьСеткуToolStripMenuItem,

this->скрытьКрасныйRКаналToolStripMenuItem, this->скрытьЗелёныйGКаналToolStripMenuItem, this->скрытьСинийBКаналToolStripMenuItem

});

this->настройкаГистрограммыToolStripMenuItem->Name = L"настройкаГистрограммыToolStripMenuItem";

this->настройкаГистрограммыToolStripMenuItem->Size = System::Drawing::Size(229, 29);

this->настройкаГистрограммыToolStripMenuItem->Text = L"Настройка гистограммы";

//

// выключитьСеткуToolStripMenuItem

//

this->выключитьСеткуToolStripMenuItem->Name = L"выключитьСеткуToolStripMenuItem";

this->выключитьСеткуToolStripMenuItem->Size = System::Drawing::Size(326, 34);

this->выключитьСеткуToolStripMenuItem->Text = L"Отключить сетку";

this->выключитьСеткуToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::выключитьСеткуToolStripMenuItem\_Click);

//

// скрытьКрасныйRКаналToolStripMenuItem

//

this->скрытьКрасныйRКаналToolStripMenuItem->Name = L"скрытьКрасныйRКаналToolStripMenuItem";

this->скрытьКрасныйRКаналToolStripMenuItem->Size = System::Drawing::Size(326, 34);

this->скрытьКрасныйRКаналToolStripMenuItem->Text = L"Скрыть красный (R) канал";

this->скрытьКрасныйRКаналToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::скрытьКрасныйRКаналToolStripMenuItem\_Click);

//

// скрытьЗелёныйGКаналToolStripMenuItem

//

this->скрытьЗелёныйGКаналToolStripMenuItem->Name = L"скрытьЗелёныйGКаналToolStripMenuItem";

this->скрытьЗелёныйGКаналToolStripMenuItem->Size = System::Drawing::Size(326, 34);

this->скрытьЗелёныйGКаналToolStripMenuItem->Text = L"Скрыть зелёный (G) канал";

this->скрытьЗелёныйGКаналToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::скрытьЗелёныйGКаналToolStripMenuItem\_Click);

//

// скрытьСинийBКаналToolStripMenuItem

//

this->скрытьСинийBКаналToolStripMenuItem->Name = L"скрытьСинийBКаналToolStripMenuItem";

this->скрытьСинийBКаналToolStripMenuItem->Size = System::Drawing::Size(326, 34);

this->скрытьСинийBКаналToolStripMenuItem->Text = L"Скрыть синий (B) канал";

this->скрытьСинийBКаналToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::скрытьСинийBКаналToolStripMenuItem\_Click);

//

// закрытьПриложениеToolStripMenuItem

//

this->закрытьПриложениеToolStripMenuItem->Name = L"закрытьПриложениеToolStripMenuItem";

this->закрытьПриложениеToolStripMenuItem->Size = System::Drawing::Size(203, 29);

this->закрытьПриложениеToolStripMenuItem->Text = L"Закрыть приложение";

this->закрытьПриложениеToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::закрытьПриложениеToolStripMenuItem\_Click);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(20, 43);

this->label1->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(235, 20);

this->label1->TabIndex = 2;

this->label1->Text = L"Шаг по оси X (номер символа)";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(20, 140);

this->label2->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(286, 20);

this->label2->TabIndex = 3;

this->label2->Text = L"Шаг по оси Y (количество символов)";

//

// trackBar1

//

this->trackBar1->Location = System::Drawing::Point(24, 68);

this->trackBar1->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->trackBar1->Name = L"trackBar1";

this->trackBar1->Size = System::Drawing::Size(585, 69);

this->trackBar1->TabIndex = 4;

this->trackBar1->Scroll += gcnew System::EventHandler(this, &MyForm::trackBar1\_Scroll);

//

// trackBar2

//

this->trackBar2->Location = System::Drawing::Point(24, 165);

this->trackBar2->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->trackBar2->Name = L"trackBar2";

this->trackBar2->Size = System::Drawing::Size(585, 69);

this->trackBar2->TabIndex = 5;

this->trackBar2->Scroll += gcnew System::EventHandler(this, &MyForm::trackBar2\_Scroll);

//

// textBox1

//

this->textBox1->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>(((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Bottom)

| System::Windows::Forms::AnchorStyles::Left));

this->textBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox1->Location = System::Drawing::Point(24, 272);

this->textBox1->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->textBox1->Multiline = true;

this->textBox1->Name = L"textBox1";

this->textBox1->ReadOnly = true;

this->textBox1->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->textBox1->Size = System::Drawing::Size(583, 755);

this->textBox1->TabIndex = 6;

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(33, 117);

this->label3->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(18, 20);

this->label3->TabIndex = 7;

this->label3->Text = L"1";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(33, 214);

this->label4->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(18, 20);

this->label4->TabIndex = 8;

this->label4->Text = L"1";

//

// label5

//

this->label5->AutoSize = true;

this->label5->Location = System::Drawing::Point(570, 214);

this->label5->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(36, 20);

this->label5->TabIndex = 9;

this->label5->Text = L"100";

//

// label6

//

this->label6->AutoSize = true;

this->label6->Location = System::Drawing::Point(578, 117);

this->label6->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(27, 20);

this->label6->TabIndex = 10;

this->label6->Text = L"20";

//

// label7

//

this->label7->AutoSize = true;

this->label7->Location = System::Drawing::Point(20, 248);

this->label7->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label7->Name = L"label7";

this->label7->Size = System::Drawing::Size(323, 20);

this->label7->TabIndex = 11;

this->label7->Text = L"Количество вхождений данного символа";

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(9, 20);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(1896, 1048);

this->Controls->Add(this->label7);

this->Controls->Add(this->label6);

this->Controls->Add(this->label5);

this->Controls->Add(this->label4);

this->Controls->Add(this->label3);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->trackBar2);

this->Controls->Add(this->trackBar1);

this->Controls->Add(this->label2);

this->Controls->Add(this->label1);

this->Controls->Add(this->chart1);

this->Controls->Add(this->menuStrip1);

this->MainMenuStrip = this->menuStrip1;

this->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->MaximumSize = System::Drawing::Size(2869, 1631);

this->MinimumSize = System::Drawing::Size(1906, 998);

this->Name = L"MyForm";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"Частотный анализатор";

this->WindowState = System::Windows::Forms::FormWindowState::Maximized;

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->chart1))->EndInit();

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar2))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void файлToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

{

this->chart1->Series["Series1"]->Points->Clear();

this->chart1->Series["R"]->Points->Clear();

this->chart1->Series["G"]->Points->Clear();

this->chart1->Series["B"]->Points->Clear();

if (openFileDialog1->FileName->Contains(".txt"))

text();

else if (openFileDialog1->FileName->Contains(".bmp"))

{

image();

this->скрытьКрасныйRКаналToolStripMenuItem->Text = "Скрыть красный (R) канал";

this->скрытьЗелёныйGКаналToolStripMenuItem->Text = "Скрыть зелёный (G) канал";

this->скрытьСинийBКаналToolStripMenuItem->Text = "Скрыть синий (B) канал";

}

}

}

private: void text()

{

this->скрытьКрасныйRКаналToolStripMenuItem->Visible = false;

this->скрытьЗелёныйGКаналToolStripMenuItem->Visible = false;

this->скрытьСинийBКаналToolStripMenuItem->Visible = false;

std::unordered\_map<char, int> data;

String^ path = System::Convert::ToString(openFileDialog1->FileName);

text\_to\_diagram(msclr::interop::marshal\_as<std::string>(path), data);

int max\_value = find\_max(data);

max = max\_value;

this->chart1->ChartAreas["ChartArea1"]->AxisY->Maximum = max\_value + (max\_value \* 0.1);

this->chart1->ChartAreas["ChartArea1"]->AxisX->Maximum = 255;

this->trackBar1->Value = 5;

this->trackBar2->Value = 10;

this->chart1->ChartAreas["ChartArea1"]->AxisY->Interval = this->trackBar2->Value \* (max / 100);

this->chart1->ChartAreas["ChartArea1"]->AxisX->Interval = this->trackBar1->Value;

this->trackBar2->Maximum = 20;

this->trackBar2->Minimum = 1;

this->trackBar1->Maximum = 20;

this->trackBar1->Minimum = 1;

this->label4->Text = Convert::ToString(max / 20);

this->label5->Text = Convert::ToString(Convert::ToInt32(this->label4->Text) \* 20);

for (int i = 255; i > 0; i--)

{

if (data[(char)i] > -1)

this->chart1->Series["Series1"]->Points->AddXY(i, data[(char)i]);

}

this->textBox1->Text = gcnew String(output\_acsii(data).c\_str()) + "\r\nЭнтропия файла: " + Convert::ToString(entropy(data));

}

private: void image()

{

this->скрытьКрасныйRКаналToolStripMenuItem->Visible = true;

this->скрытьЗелёныйGКаналToolStripMenuItem->Visible = true;

this->скрытьСинийBКаналToolStripMenuItem->Visible = true;

String^ path = System::Convert::ToString(openFileDialog1->FileName);

ReadBMP(msclr::interop::marshal\_as<std::string>(path), R, G, B);

int max\_value = find\_max(R);

if (find\_max(G) > max\_value)

max\_value = find\_max(G);

if (find\_max(B) > max\_value)

max\_value = find\_max(B);

max = max\_value;

this->chart1->ChartAreas["ChartArea1"]->AxisY->Maximum = max\_value + (max\_value \* 0.1);

this->chart1->ChartAreas["ChartArea1"]->AxisX->Maximum = 255;

this->trackBar1->Value = 5;

this->trackBar2->Value = 10;

this->label4->Text = Convert::ToString(max / 20);

this->label5->Text = Convert::ToString(Convert::ToInt32(this->label4->Text) \* 20);

this->chart1->ChartAreas["ChartArea1"]->AxisY->Interval = this->trackBar2->Value \* (max / 20);

this->chart1->ChartAreas["ChartArea1"]->AxisX->Interval = this->trackBar1->Value;

this->trackBar2->Maximum = 20;

this->trackBar2->Minimum = 1;

this->trackBar1->Maximum = 20;

this->trackBar1->Minimum = 1;

std::unordered\_map<int, int> data = init\_map\_RGB();

for (int i = 255; i > 0; i--)

{

int temp = 0;

if (R[i] > -1)

{

this->chart1->Series["R"]->Points->AddXY(i, R[i]);

temp += R[i];

}

if (G[i] > -1)

{

this->chart1->Series["G"]->Points->AddXY(i, G[i]);

temp += G[i];

}

if (B[i] > -1)

{

this->chart1->Series["B"]->Points->AddXY(i, B[i]);

temp += B[i];

}

data[i] = temp;

}

String^ result = "";

for (int i = 0; i < 256; i++)

{

result += i + ". " + data[i] + "\r\n";

}

this->textBox1->Text = result + "\r\nЭнтропия файла: " + Convert::ToString(entropy(data));

}

private: System::Void trackBar2\_Scroll(System::Object^ sender, System::EventArgs^ e)

{

this->chart1->ChartAreas["ChartArea1"]->AxisY->Interval = this->trackBar2->Value \* (max / 20);

}

private: System::Void trackBar1\_Scroll(System::Object^ sender, System::EventArgs^ e)

{

this->chart1->ChartAreas["ChartArea1"]->AxisX->Interval = this->trackBar1->Value;

}

private: System::Void закрытьПриложениеToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (MessageBox::Show("Вы уверены, что хотите закрыть приложение?", "Закрыть приложение", MessageBoxButtons::OKCancel) == System::Windows::Forms::DialogResult::OK)

this->Close();

}

private: System::Void выключитьСеткуToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (this->chart1->ChartAreas["ChartArea1"]->AxisX->MajorGrid->Enabled == false)

{

this->выключитьСеткуToolStripMenuItem->Text = "Отключить сетку";

this->chart1->ChartAreas["ChartArea1"]->AxisX->MajorGrid->Enabled = true;

this->chart1->ChartAreas["ChartArea1"]->AxisY->MajorGrid->Enabled = true;

}

else

{

this->выключитьСеткуToolStripMenuItem->Text = "Включить сетку";

this->chart1->ChartAreas["ChartArea1"]->AxisX->MajorGrid->Enabled = false;

this->chart1->ChartAreas["ChartArea1"]->AxisY->MajorGrid->Enabled = false;

}

}

private: System::Void очиститьКоординатнуюПлоскостьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

this->chart1->Series["Series1"]->Points->Clear();

this->textBox1->Text = "";

this->chart1->Series["R"]->Points->Clear();

this->chart1->Series["G"]->Points->Clear();

this->chart1->Series["B"]->Points->Clear();

this->chart1->Series["Series1"]->Points->AddXY(0, 0);

}

private: System::Void скрытьКрасныйRКаналToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (this->скрытьКрасныйRКаналToolStripMenuItem->Text == "Скрыть красный (R) канал")

{

this->скрытьКрасныйRКаналToolStripMenuItem->Text = "Показать красный (R) канал";

this->chart1->Series["R"]->Points->Clear();

this->chart1->Series["Series1"]->Points->AddXY(0, 0);

}

else

{

this->скрытьКрасныйRКаналToolStripMenuItem->Text = "Скрыть красный (R) канал";

for (int i = 255; i > 0; i--)

{

this->chart1->Series["R"]->Points->AddXY(i, R[i]);

}

}

}

private: System::Void скрытьЗелёныйGКаналToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (this->скрытьЗелёныйGКаналToolStripMenuItem->Text == "Скрыть зелёный (G) канал")

{

this->скрытьЗелёныйGКаналToolStripMenuItem->Text = "Показать зелёный (G) канал";

this->chart1->Series["G"]->Points->Clear();

this->chart1->Series["Series1"]->Points->AddXY(0, 0);

}

else

{

this->скрытьЗелёныйGКаналToolStripMenuItem->Text = "Скрыть зелёный (G) канал";

for (int i = 255; i > 0; i--)

{

this->chart1->Series["G"]->Points->AddXY(i, G[i]);

}

}

}

private: System::Void скрытьСинийBКаналToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (this->скрытьСинийBКаналToolStripMenuItem->Text == "Скрыть синий (B) канал")

{

this->скрытьСинийBКаналToolStripMenuItem->Text = "Показать синий (B) канал";

this->chart1->Series["B"]->Points->Clear();

this->chart1->Series["Series1"]->Points->AddXY(0, 0);

}

else

{

this->скрытьСинийBКаналToolStripMenuItem->Text = "Скрыть синий (B) канал";

for (int i = 255; i > 0; i--)

{

this->chart1->Series["B"]->Points->AddXY(i, B[i]);

}

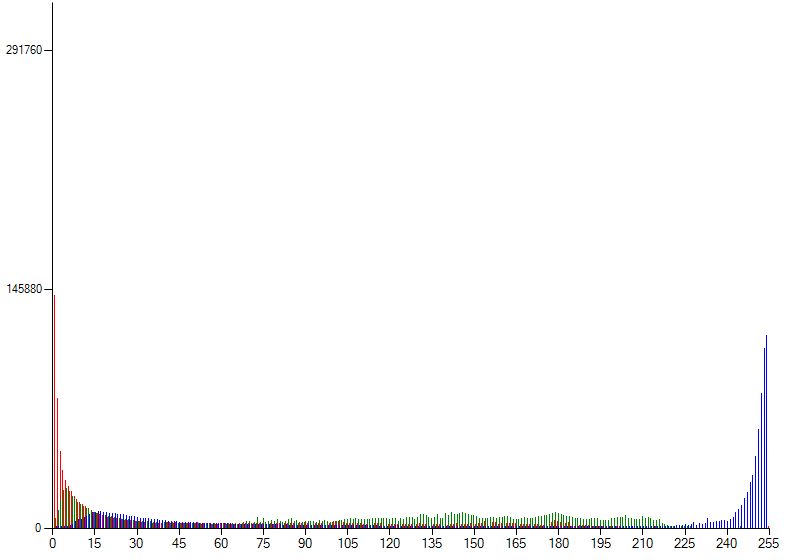
}

}

};

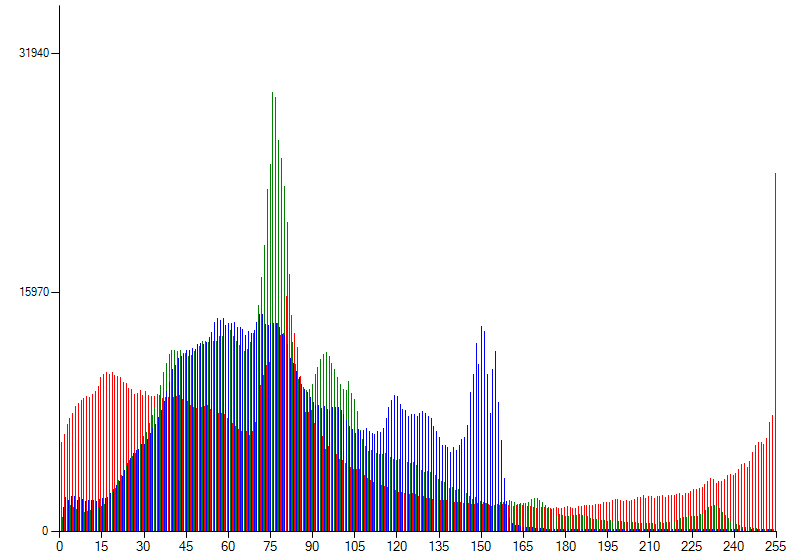
}

# Приложение Б



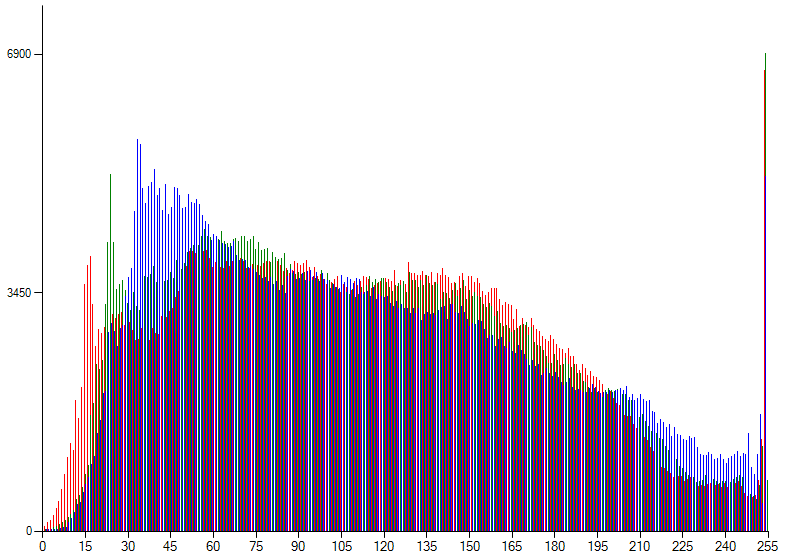
H(А) = 7,332 бит

Приложение Б.1 Рисунок 1 – Гистограмма картинки с преобладанием одного цвета



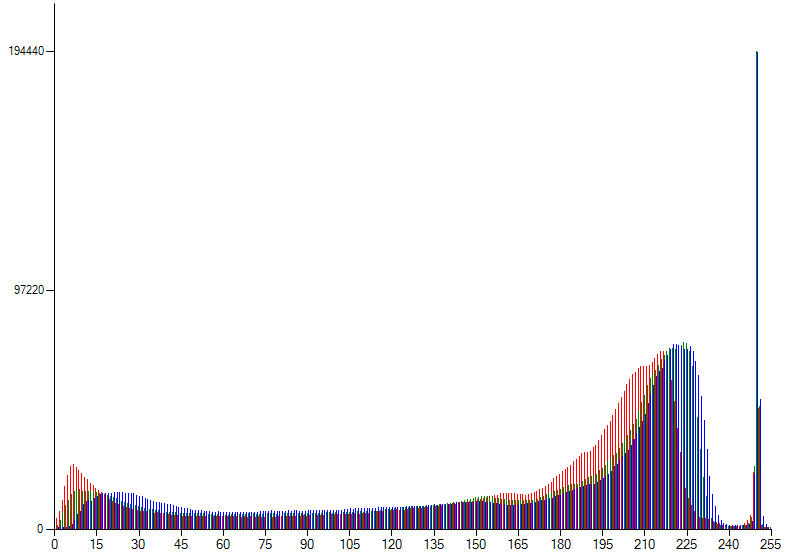
H(x) = 7,532 бит

Приложение Б.2 Рисунок 2 – Гистограмма картинки в графическом редакторе



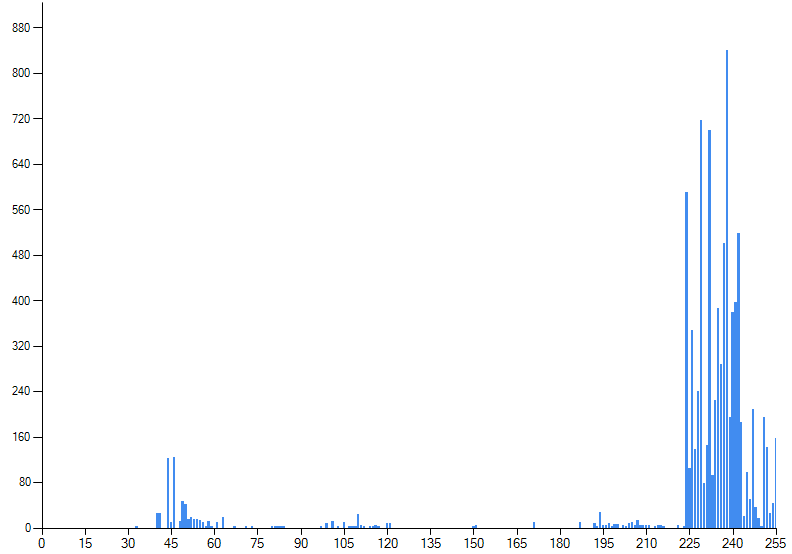
H(x) = 7,819 бит

Приложение Б.3 Рисунок 3 – Гистограмма фотографии №1



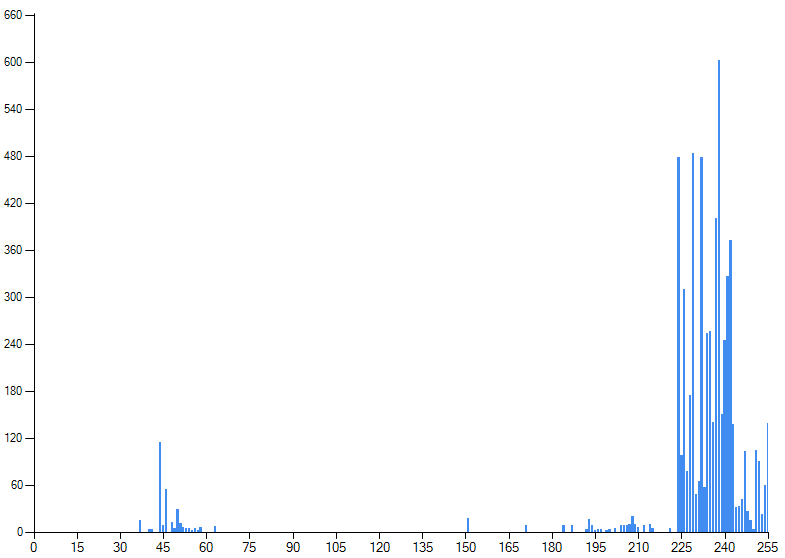
H(x) = 7,346 бит

Приложение Б.4 Рисунок 4 – Гистограмма фотографии №2



H(x) = 4,949 бит

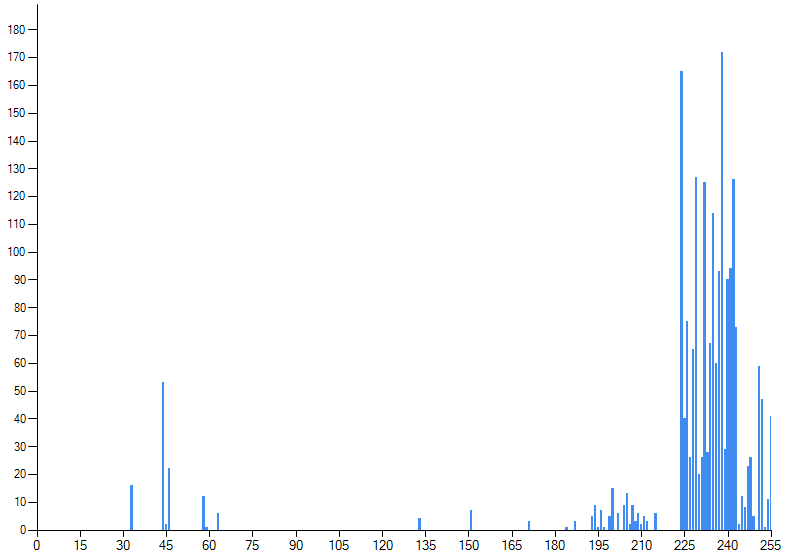
Приложение Б.5 Рисунок 5 – Гистограмма текста учебника



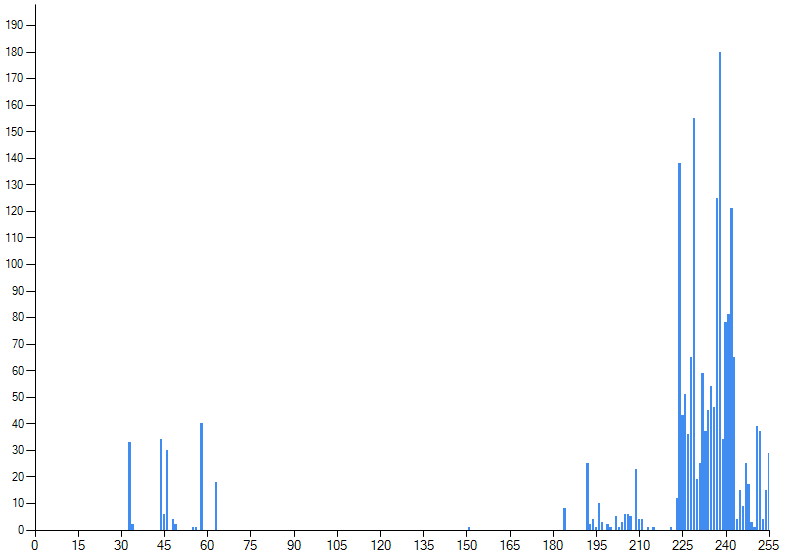
H(x) = 4,829 бит

Приложение Б.6 Рисунок 6 – Гистограмма статьи в СМИ

H(x) = 4,965 бит



Приложение Б.7 Рисунок 7 – Гистограмма литературного текста



H(x) = 5,057 бит

Приложение Б.8 Рисунок 8 – Гистограмма текста переписки в соцсетях