Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**Национальный исследовательский университет ИТМО**

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

**ЛАБОРАТОРНАЯ РАБОТА № 2**

**По дисциплине**

**Введение в цифровую культуру**

|  |
| --- |
| **Тихомиров Дмитрий Алексеевич** |

**Выполнил:**

**Проверил:**

|  |
| --- |
| **Страдина Марина Владимировна** |

Санкт-Петербург, 2020г

# Исходное изображение



## Изображение в заданном формате



# Полученная последовательность

74 12 100 8 60 43 22 111 79 28 68 49 28 39 111 73 29 12 122 186 155 54 81 130 51 37 170 168 140 160 146 120 156 216 176 130 136 142 149 139 148 140 146 156 183 152 78 78 67 99 152 55 250 246 243 237 229 144 175 176 212 136 171 136 122 143 162 140 190 200 176 171 189 182 161 201 196 164 209 101 80 109 116 151 194 89 151 164 113 96 82 137 173 175 122 146 206 157 144 180 142 113 191 202 126 136 163 150 156 114 207 195 192 142 159 190 175 106 164 183 173 155 190 90 180 161 112 137

## Результат квантования последовательности

80 20 100 0 60 40 20 120 80 20 60 40 20 40 120 80 20 20 120 180 160 60 80 140 60 40 180 160 140 160 140 120 160 220 180 140 140 140 140 140 140 140 140 160 180 160 80 80 60 100 160 60 260 240 240 240 220 140 180 180 220 140 180 140 120 140 160 140 200 200 180 180 180 180 160 200 200 160 200 100 80 100 120 160 200 80 160 160 120 100 80 140 180 180 120 140 200 160 140 180 140 120 200 200 120 140 160 160 160 120 200 200 200 140 160 200 180 100 160 180 180 160 200 100 180 160 120 140

## Реализация на языке С++

#include<iostream>

#include<cmath>

#include"bitmap\_image.hpp" // стороняя библиотека, использующаяся для работы с изображениями

using namespace std;

int main()

{

int pixels[128][128]; // структура для хранения значений пикселей

bitmap\_image img("10p1.bmp");

for (int i = 0; i < 128; i++)

{

for (int j = 0; j < 128; j++)

{

rgb\_t color;

img.get\_pixel(i, j, color.blue, color.red, color.green); // считывание значения пикселей

pixels[i][j] = color.red;

}

}

for (int i = 0; i < 128; i++) // вывод последовательности

cout << pixels[i][63] << ' ';

cout << endl;

int v[14] = { 0 };

for (int i = 0; i < 128; i++)

{

pixels[i][63] = round(pixels[i][63] / 20.0) \* 20; //квантование

v[pixels[i][63] / 20]++; //подсчет частоты встречаемости уникальных символов

cout << pixels[i][63] << ' '; //вывод последовательности после квантования

}

cout << endl;

for (int i = 0; i < 14; i++) // вывод количества для каждого уникального символа

cout << i \* 20 << ":" << v[i] << "\t";

double ent = 0; //нахождение энтропии

for (int i = 0; i < 14; i++)

ent = ent + (v[i] / 128.0) \* log2((v[i] / 128.0));

cout << -ent;

return 0;

}

# Анализ первичного алфавита:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Элемент | 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 | 220 | 240 | 260 |
| Количество | 1 | 6 | 4 | 6 | 9 | 7 | 12 | 23 | 21 | 18 | 14 | 3 | 3 | 1 |
| Частота |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Количество символов алфавита: 14

Значение энтропии: 2.48618

# Построение равномерного двоичного кода

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 | 220 | 240 | 260 |
| 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 |

## Закодированное равномерным кодом сообщение:

0100 0001 0101 0000 0011 0010 0001 0110 0100 0001 0011 0010 0001 0010 0110 0100 0001 0001 0110 1001 1000 0011 0100 0111 0011 0010 1001 1000 0111 1000 0111 0110 1000 1011 1001 0111 0111 0111 0111 0111 0111 0111 0111 1000 1001 1000 0100 0100 0011 0101 1000 0011 1101 1100 1100 1100 1011 0111 1001 1001 1011 0111 1001 0111 0110 0111 1000 0111 1010 1010 1001 1001 1001 1001 1000 1010 1010 1000 1010 0101 0100 0101 0110 1000 1010 0100 1000 1000 0110 0101 0100 0111 1001 1001 0110 0111 1010 1000 0111 1001 0111 0110 1010 1010 0110 0111 1000 1000 1000 0110 1010 1010 1010 0111 1000 1010 1001 0101 1000 1001 1001 1000 1010 0101 1001 1000 0110 0111

Минимальная длина двоичного кода: 4 символа

Количество переданной информации: 4\* 128 = 512 бит

# Коды Шеннона-Фано

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 140 | 160 | 180 | 260 | 200 | 120 | 80 | 100 | 60 | 20 | 40 | 220 | 240 | 0 |
| 010 | 011 | 000 | 001 | 100 | 1010 | 1011 | 1100 | 1101 | 11100 | 11101 | 11110 | 111110 | 111111 |

## Последовательность, закодированная методом Шеннона-Фано:

1011 11100 1100 111111 1101 11101 11100 1010 1011 11100 1101 11101 11100 11101 1010 1011 11100 11100 1010 000 011 1101 1011 010 1101 11101 000 011 010 011 010 1010 011 11110 000 010 010 010 010 010 010 010 010 011 000 011 1011 1011 1101 1100 011 1101 001 111110 111110 111110 11110 010 000 000 11110 010 000 010 1010 010 011 010 100 100 000 000 000 000 011 100 100 011 100 1100 1011 1100 1010 011 100 1011 011 011 1010 1100 1011 010 000 000 1010 010 100 011 010 000 010 1010 100 100 1010 010 011 011 011 1010 100 100 100 010 011 100 000 1100 011 000 000 011 100 1100 000 011 1010 010

## Эффективность (средняя длина символа)

Lср = = 3.5625

## Относительная избыточность кода:

Q = 1 - = 0.3021249123

Количество переданной информации: 456 бит

# Коды Хаффмана

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 140 | 160 | 180 | 200 | 120 | 80 | 100 | 20 | 40 | 240 | 60 | 220 | 0 | 260 |
| 00 | 110 | 101 | 011 | 1111 | 1001 | 1000 | 0100 | 01011 | 01010 | 11101 | 111001 | 1110001 | 1110000 |

## Полученная последовательность:

1001 0100 1000 1110001 11101 01011 0100 1111 1001 0100 11101 01011 0100 01011 1111 1001 0100 0100 1111 101 110 11101 1001 00 11101 01011 101 110 00 110 00 1111 110 111001 101 00 00 00 00 00 00 00 00 110 101 110 1001 1001 11101 1000 110 11101 1110000 01010 01010 01010 111001 00 101 101 111001 00 101 00 1111 00 110 00 1111 1111 101 101 101 101 110 1111 1111 110 1111 1000 1001 1000 1111 110 1111 1001 110 110 1111 1000 1001 00 101 101 1111 00 1111 110 00 101 00 1111 1111 1111 1111 00 110 110 110 1111 1111 1111 1111 00 110 1111 101 1000 110 101 101 110 1111 1000 101 110 1111 00

## Средняя длина:

Lср = 3.53125

## Относительная избыточность:

Q = 1 - = 0,295949

Количество переданной информации: 452 бита

# ВЫВОД

В ходе лабораторной работы выяснилось, что наибольшей эффективностью обладает код Хаффмана. На следующем месте с минимальным отрывом идет код Шеннона-Фано и затем равномерный двоичный код, который показал наихудшие результаты.