МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа № 02

По дисциплине «Введение в цифровую культуру и программирование»

Эффективное кодирование.

Выполнил студент группы №M3102  
*Ларин Владислав Денисович*

Проверил:  
*Хлопотов Максим Валерьевич*

САНКТ-ПЕТЕРБУРГ  
2019

Исходное изображение:



Изображение в заданном формате:



цифровая последовательность

1. *Полученная цифровая последовательность средней строки изображения:*

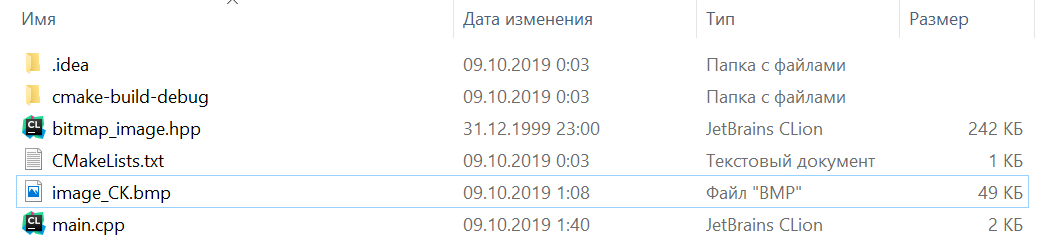
*62 62 63 67 78 97 120 141 155 161 160 158 157 157 162 167 171 172 171 170 168 165 160 150 143 138 137 145 147 154 155 148 148 154 154 151 149 152 154 156 156 157 155 152 148 142 132 111 94 88 98 114 123 122 110 101 117 134 143 142 140 131 125 127 138 153 167 159 101 99 119 137 145 149 151 160 174 192 207 214 216 216 216 221 226 227 222 222 229 222 195 147 162 179 178 152 116 106 113 132 158 179 195 207 216 220 219 214 206 191 160 126 99 83 77 80 88 97 103 105 106 107 109 111 115 119 122 124*

1. *Цифровая последовательность после квантования:*

60 60 60 60 80 100 120 140 160 160 160 160 160 160 160 160 180 180 180 180 160 160 160 160 140 140 140 140 140 160 160 140 140 160 160 160 140 160 160 160 160 160 160 160 140 140 140 120 100 80 100 120 120 120 120 100 120 140 140 140 140 140 120 120 140 160 160 160 100 100 120 140 140 140 160 160 180 200 200 220 220 220 220 220 220 220 220 220 220 220 200 140 160 180 180 160 120 100 120 140 160 180 200 200 220 220 220 220 200 200 160 120 100 80 80 80 80 100 100 100 100 100 100 120 120 120 120 120

Ход работы:

1. Выделяем цифровую последовательность путем написания программы с использованием библиотеки “bitmap\_image.hpp”.



1. Программным путем выделяем среднюю строку изображения, для каждого уникального символа вычисляем частоту появления в цифровой последовательности и производим квантование и вывод полученной последовательности.

Ниже приведен код программы данного алгоритма:

#include **<iostream>**#include **<cmath>**#include **"bitmap\_image.hpp"** *// библиотека, обрабатывающая изображения*

**using namespace** std;

**int** main()

{  
 **int** a[128][128]; *// массив для хранения числовых значений цветов изображения* bitmap\_image image(**"imagine\_CK.bmp"**);  
 **for** (**int** x = 0; x < image.width(); x++)

{  
 **for** (**int** y = 0; y < image.height(); y++)

{  
 rgb\_t color; *// структура, в которой хранятся значения хар-ки цветов* image.get\_pixel(x, y, color.blue, color.red, color.green);  
 a[x][y] = color.red;  
 }  
 }  
  
 **for** (**int** i = 0; i < 128; i++)  
 cout << a[i][64] << **' '**; *// вывод средней строки* cout << endl;  
  
 **int** b[261];**for** (**int** i = 0; i < 261; i++)  
 b[i] = 0;  
 **for** (**int** i = 0; i < 128; i++)

{  
 a[i][64] = round(**double**(a[i][64]) / 20) \* 20; *// квантование средней строки* b[a[i][64]]++; *// подсчет частоты встречаемости*  
 cout << a[i][64] << **' '**; *// вывод отквантованной строки* }  
 cout << endl;  
 **return** 0;  
}

анализ цифровой последовательности (строки)

Частота встречаемости:

\*Исходные данные упорядочены по не возрастанию частоты встречаемости

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Символ алфавита* | *A(160)* | *B(140)* | *C(120)* | *D(220)* | *E(100)* | *F(180)* | *G(200)* | *H(80)* | *I(60)* |
| *Количество*  *символов* | 33 | 23 | 18 | 15 | 14 | 8 | 7 | 6 | 4 |
| *Частота встречаемости* |  |  |  |  |  |  |  |  |  |

Количество уникальных символов алфавита: 9

Значение энтропии:

Вычисляется по формуле

Воспользуемся программной реализацией:

**double** e = 0; *// энтропия***for** (**int** i = 0; i < 261; i++)  
{  
 **if** (b[i] > 0)  
 {  
 cout << i << **' '** << b[i] << endl;  
 e = е - (**double**(b[i]) / 128) \* log2(**double**(b[i]) / 128); *//подсчет энтропии* }  
}  
cout << e; *//вывод значения энтропии*

Получим значение 2.9013

двоичный равномерный код

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Символ алфавита* | A(160) | B(140) | C(120) | D(220) | E(100) | F(180) | G(200) | H(80) | I(60) |
| *Равномерный код* | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 |

Закодированное равномерным кодом сообщение:

1000 1000 1000 1000 0111 0100 0010 0001 0000 0000 0000 0000 0000 0000 0000 0000 0101 0101 0101 0101 0000 0000 0000 0000 0001 0001 0001 0001 0001 0000 0000 0001 0001 0000 0000 0000 0001 0000 0000 0000 0000 0000 0000 0000 0001 0001 0001 0010 0100 0111 0100 0010 0010 0010 0010 0100 0010 0001 0001 0001 0001 0001 0010 0010 0001 0000 0000 0000 0100 0100 0010 0001 0001 0001 0000 0000 0101 0110 0110 0011 0011 0011 0011 0011 0011 0011 0011 0011 0011 0011 0110 0001 0000 0101 0101 0000 0010 0100 0010 0001 0000 0101 0110 0110 0011 0011 0011 0011 0110 0110 0000 0010 0100 0111 0111 0111 0111 0100 0100 0100 0100 0100 0100 0010 0010 0010 0010 0010

Расчетная длина двоичного кода:

4 (бит)

Количество переданной информации:

4\*128 = 512 (бит)

Коды шеннона-фано

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Символ алфавита* | *A(160)* | *B(140)* | *C(120)* | *D(220)* | *E(100)* | *F(180)* | *G(200)* | *H(80)* | *I(60)* |
| *Количество*  *символов* | *33* | *23* | *18* | *15* | *14* | *8* | *7* | *6* | *4* |
| *Код Шеннона-Фано* | *11* | *10* | *011* | *010* | *0011* | *0010* | *0001* | *00001* | *00000* |

Закодированная полученным кодом Шеннона–Фано последовательность:

00000000000000000000000010011011101111111111111111001000100010001011111111101010101011111010111111101111111111111110101001100110000100110110110110110011011101010101001101110111111001100110111010101111001000010001010010010010010010010010010010010000110110010001011011001101110110010000100010100100100100001000111011001100001000010000100001001100110011001100110011011011011011011

Средняя длина кодовой комбинации (эффективность):

*Iср = = 2.9453125*

Относительная избыточность кода:

*Q = 1 - ~ 0.014943*

Количество переданной информации:

*= 377 (бит)*

код хаффмана

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Символ алфавита* | *A(160)* | *B(140)* | *C(120)* | *D(220)* | *E(100)* | *F(180)* | *G(200)* | *H(80)* | *I(60)* |
| *Количество символов* | *33* | *23* | *18* | *15* | *14* | *8* | *7* | *6* | *4* |
| *Код Хаффмана* | *00* | *010* | *011* | *100* | *111* | *1010* | *1011* | *1100* | *1101* |

Закодированная полученным кодом Хаффмана последовательность:

*010000100001000010000100100000110111111111111111111001001001001111111110110110110110111111011011111111011111111111111110110110100100001001000001001001001000001101101101101101001001101111111000000001101101101111110001010101011011011011011011011011011011011010110111100100110010000011011110001010101011011011011010101011100100001001010010100101001000000000000000000001001001001001*

1

1

1

1

1

1

1

1

0

0

0

0\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

0

0

0

0

HI(10)

DFG(30)

H(6)

G(7)

F(8)

EHI(24)

DFGEHI(54)

E(14)

FG(15)

D(15)

I(4)

C(18)

B(23)

BC(41)

A(33)

ABC(74)

ABCDFGEHI(128)

Средняя длина кодовой комбинации (эффективность):

*Iср = = 2.9375*

Относительная избыточность кода:

*Q = 1 - ~ 0.0123234*

Количество переданной информации:

*2\*33 + 3\*(23+18+15+14) + 4\*(8+7+6+4) = 376 (бит*)

Вывод

В ходе лабораторной работы я познакомился с такими методами сжатия информации, как метод Шеннона-Фано и метод Хаффмана. Оба метода оказались более эффективными, чем равномерное кодирование. В свою очередь, кодирование методом Хаффмана оказалось эффективнее кодирования методом Шеннона-Фано. Разница в количестве переданной информации составила один бит.