ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Старший преподаватель		Т. А. Суетина
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

«Алгоритм сжатия изображения JPEG 2000»

по курсу: Техника аудиовизуальных средств информации

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №	4128		В.А. Воробьев
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

1. Цель работы

Получить теоретические знания по работе с цветным изображением, изучить этапы классического алгоритма JPEG 2000 для сжатия цветного изображения и практически реализовать поученные знания.

2. Задание

Для цветного изображения размерностью 16x16 пикселей выполнить сжатие изображения на основе алгоритма JPEG2000. Вычислить поученный коэффициент сжатия.

3. Ход работы

Цветное изображение для сжатия размерностью 16x16 пикселей представлено на рисунке 1.

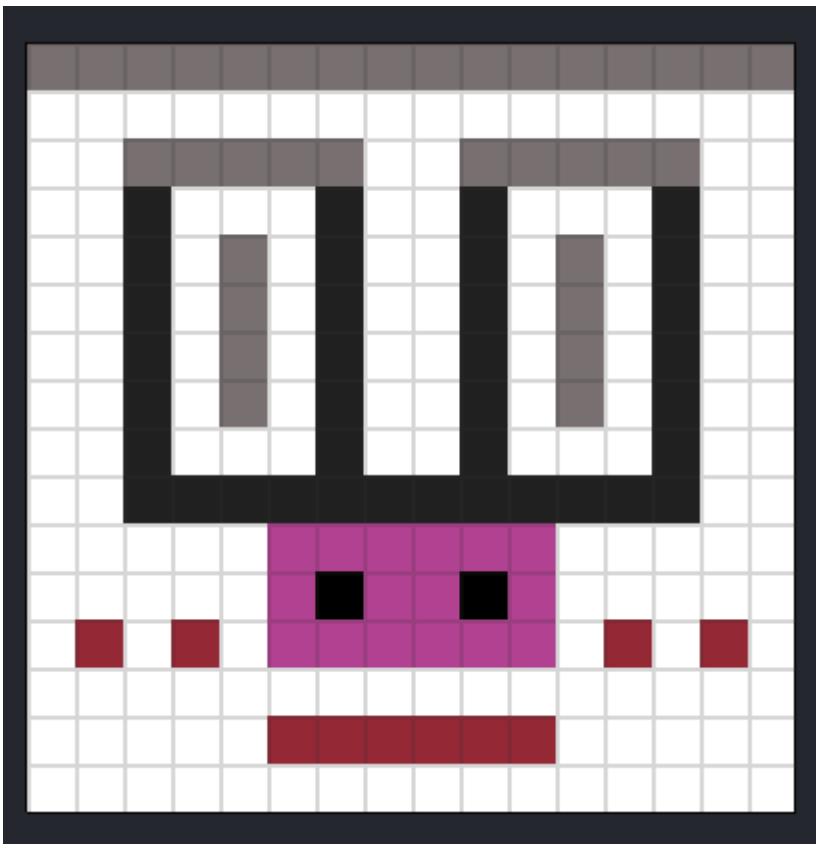

Рисунок 1 - Исходное изображение

Реализован сдвиг по яркости всех значений пикселей изображения. Изображение переведено в цветовое пространство YUV. Матрица Y разделена на четыре блока 8x8, а матрицы U (Cr) и V (Cb) прорежены. В итоге получено шесть матриц: четыре матрицы света и две цветоразностных. Полученные матрицы представлены на рисунках 2-4.

В алгоритме используется сдвиг по яркости на 31 единиц для каждого пикселя изображения перед его дальнейшей обработкой.

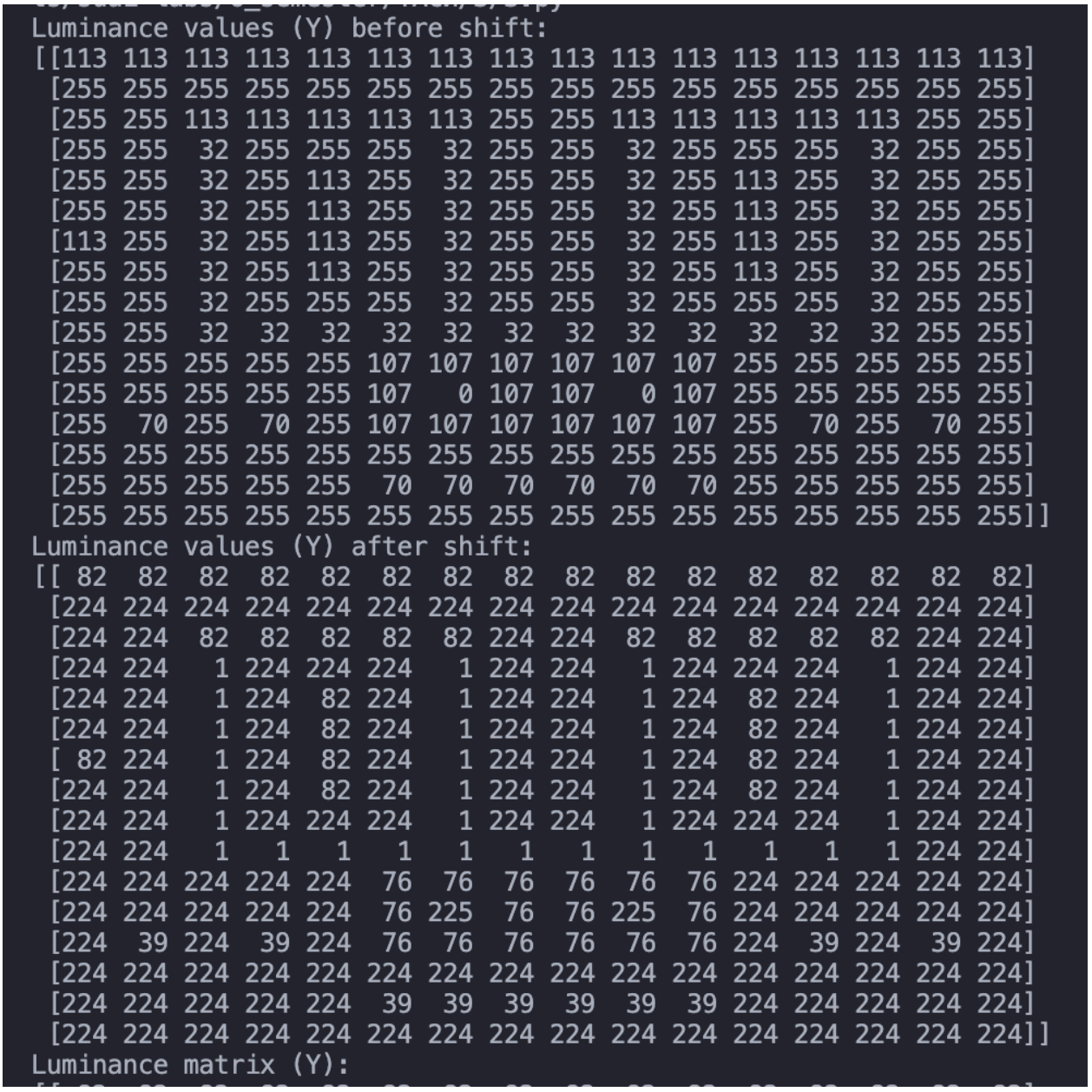

Рисунок 1 – матрицы до и после сдвига

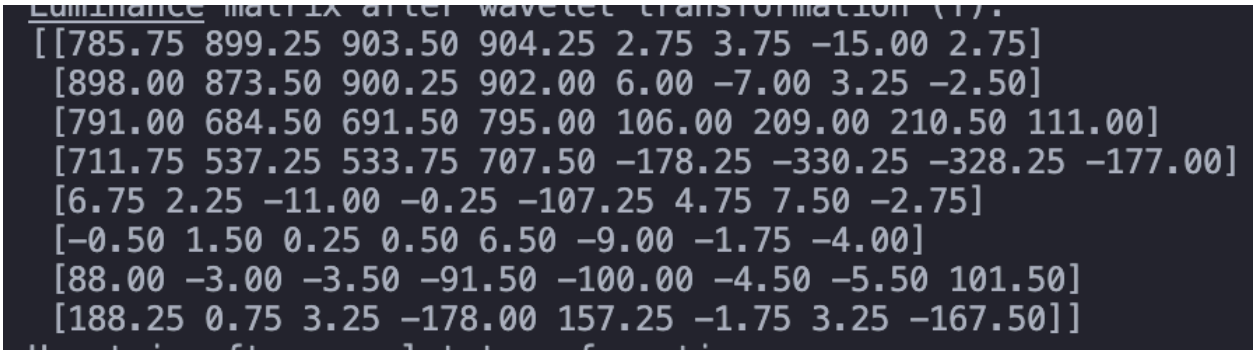

Рисунок 2.1 - Матрица Y после преобразования

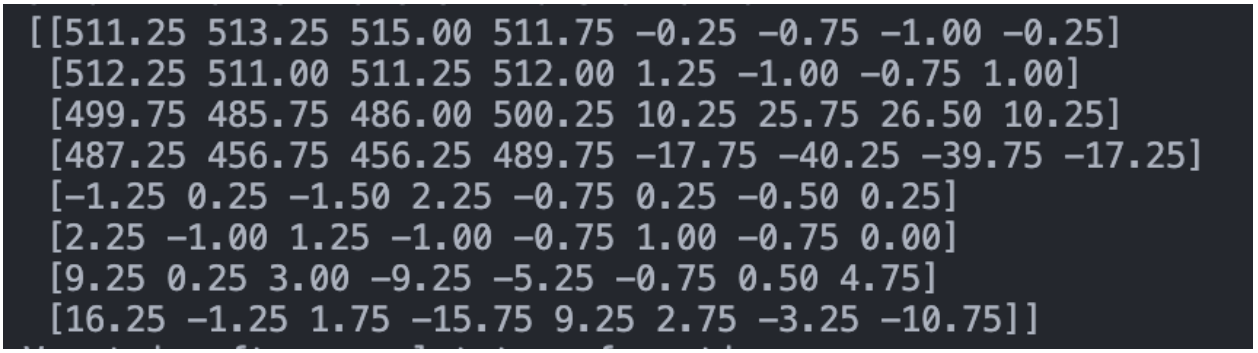

Рисунок 2.2 - Матрица Y после преобразования

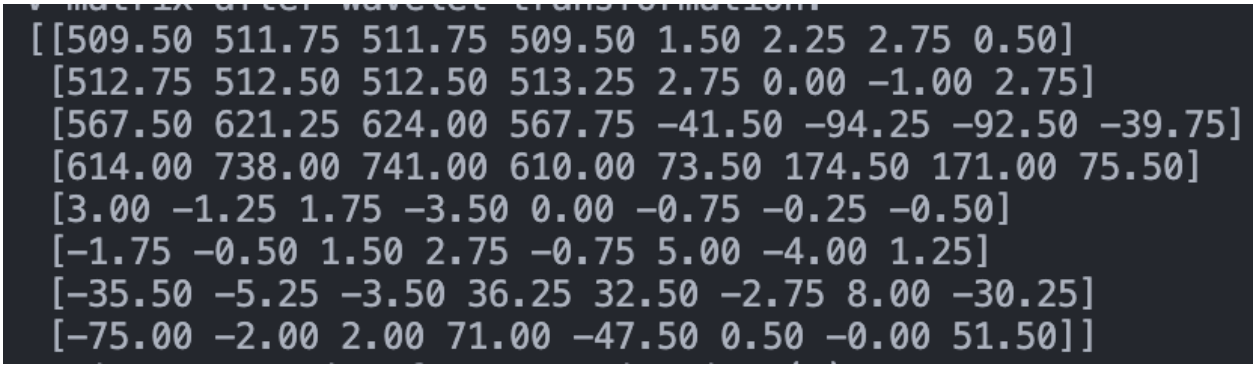

Рисунок 2.3 - Матрица Y после преобразования

Матрица, полученная в результате квантования, значения представлены на рисунке ниже.

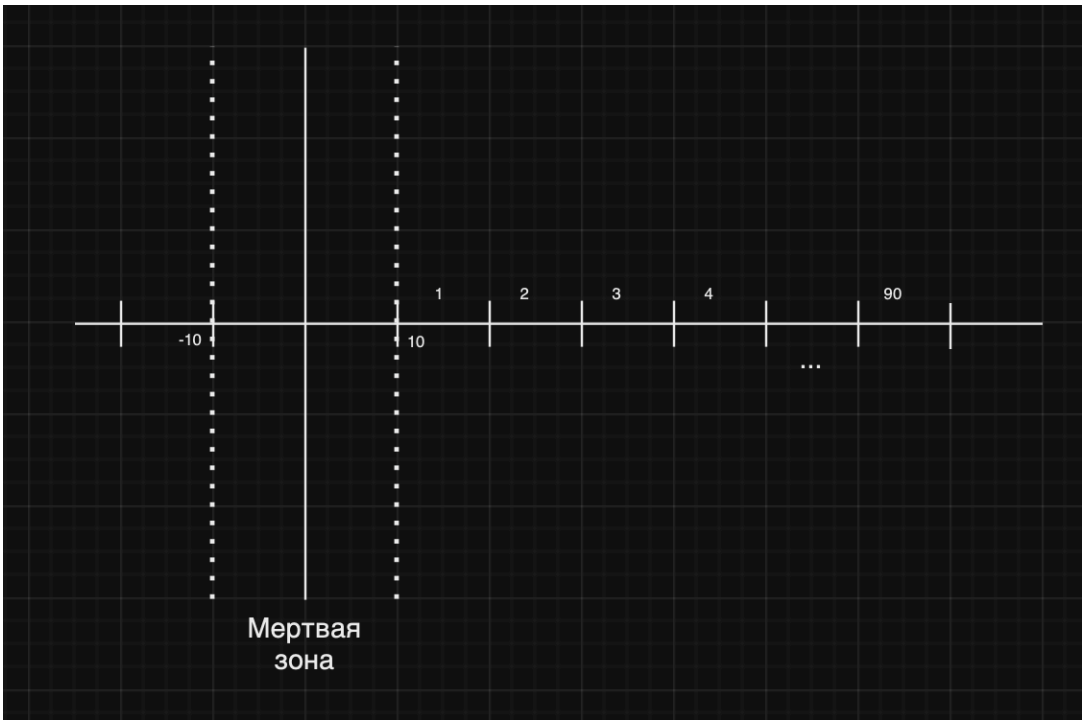

Рисунок 3 - Взятая мертвая зона

```
[[79.00 90.00 90.00 90.00 0.00 0.00 -2.00 0.00]
 [90.00 87.00 90.00 90.00 0.00 0.00 0.00 0.00]
 [79.00 68.00 69.00 80.00 11.00 21.00 21.00 11.00]
 [71.00 54.00 53.00 71.00 -18.00 -33.00 -33.00 -18.00]
 [0.00 0.00 -1.00 0.00 -11.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [9.00 0.00 0.00 -9.00 -10.00 0.00 0.00 10.00]
 [19.00 0.00 0.00 -18.00 16.00 0.00 0.00 -17.00]]
```

Рисунок 4.1 – Квантованные коэффициенты

```
[[51.00 51.00 52.00 51.00 0.00 0.00 0.00 0.00]
 [51.00 51.00 51.00 51.00 0.00 0.00 0.00 0.00]
 [50.00 49.00 49.00 50.00 1.00 3.00 3.00 1.00]
 [49.00 46.00 46.00 49.00 -2.00 -4.00 -4.00 -2.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [2.00 0.00 0.00 -2.00 0.00 0.00 0.00 -1.00]]
```

Рисунок 4.2 – Квантованные коэффициенты

```
[[51.00 51.00 51.00 51.00 0.00 0.00 0.00 0.00]
 [51.00 51.00 51.00 51.00 0.00 0.00 0.00 0.00]
 [57.00 62.00 62.00 57.00 -4.00 -9.00 -9.00 -4.00]
 [61.00 74.00 74.00 61.00 7.00 17.00 17.00 8.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
 [-4.00 0.00 0.00 4.00 3.00 0.00 0.00 -3.00]
 [-7.00 0.00 0.00 7.00 -5.00 0.00 0.00 5.00]]
```

Рисунок 4.3 – Квантованные коэффициенты

К полученным последовательностям применено сжатие алгоритмом арифметического кодирования (часть результата представлена на рисунках 5-7).

```
Luminance matrix (Y):
79.0: [0, 0.25]
79.0: low = 0, high = 0.25
90.0: [0.0625, 0.09375]
90.0: low = 0.0625, high = 0.09375
79.0: [0.06250, 0.0703125]
79.0: low = 0.06250, high = 0.0703125
71.0: [0.0654296875, 0.0664062500]
71.0: low = 0.0654296875, high = 0.0664062500
0.0: [0.0659179687500, 0.0661621093750]
0.0: low = 0.0659179687500, high = 0.0661621093750
0.0: [0.0660400390625000, 0.0661010742187500]
0.0: low = 0.0660400390625000, high = 0.0661010742187500
9.0: [0.0660858154296875000, 0.0660934448242187500]
9.0: low = 0.0660858154296875000, high = 0.0660934448242187500
19.0: [0.06609249114990234375000, 0.06609344482421875000000]
19.0: low = 0.06609249114990234375000, high = 0.06609344482421875000000
l0 = 0.06609249114990234375000
h0 = 0.06609344482421875000000
```

Рисунок 5 - Результаты арифметического кодирования

```
U matrix:
51.0: [0, 0.25]
51.0: low = 0, high = 0.25
51.0: [0.00, 0.0625]
51.0: low = 0.00, high = 0.0625
50.0: [0.015625, 0.0234375]
50.0: low = 0.015625, high = 0.0234375
49.0: [0.0185546875, 0.0195312500]
49.0: low = 0.0185546875, high = 0.0195312500
0.0: [0.0190429687500, 0.0194091796875]
0.0: low = 0.0190429687500, high = 0.0194091796875
0.0: [0.0192260742187500, 0.0193634033203125]
0.0: low = 0.0192260742187500, high = 0.0193634033203125
0.0: [0.0192947387695312500, 0.0193462371826171875]
0.0: low = 0.0192947387695312500, high = 0.0193462371826171875
2.0: [0.0193397998809814453125, 0.0193462371826171875000]
2.0: low = 0.0193397998809814453125, high = 0.0193462371826171875000
l0 = 0.0193397998809814453125
h0 = 0.0193462371826171875000
```

Рисунок 6 - Результаты арифметического кодирования

```
Total file size: 43.58781487706386
EC: 140.95682514318938
```

Рисунок 7 - Результаты сжатия

Вывод

В ходе выполнения лабораторной работы проведено сжатия изображения в цветовом пространстве RGB размером 16x16 пикселей алгоритмом JPEG2000. Получены теоретические знания по работе с цветным изображением, изучены этапы алгоритма JPEG2000 для сжатия цветного изображения и практически реализованы поученные знания.

Проведены следующие операции: сдвиг по яркости, перевод изображения в цветовое пространство YUV, прореживание цветоразностных компонентов, вейвлет-преобразование, квантование, сжатие алгоритмом арифметического кодирования

Полученный коэффициент сжатия равен примерно 140.

Работа выполнена с использованием написанной на языке Python программы. Исходный код программы представлен в приложении А.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД JPEG2000.PY

```
import cv2

import pywt

import numpy as np

from decimal import Decimal, getcontext



def pixel(rgb):

    yuv = cv2.cvtColor(rgb, cv2.COLOR_BGR2YUV)



    print('Luminance matrix (Y):')

    print(yuv[:, :, 0])

    print('U matrix:')

    print(yuv[:, :, 1])

    print('V matrix:')

    print(yuv[:, :, 2])

    return yuv
```

```
def wavelet(yuv, n, need_print = False):
    wave = np.empty((n, n, 3))

    for i in range(3):
        coeffs = pywt.dwt2(yuv[:, :, i], 'haar')
        a, (h, v, d) = coeffs
        wave[:, :, i] = np.vstack((np.hstack((a, h)), np.hstack((v, d))))

    a = np.empty((int(n / 2), int(n / 2), 3))

    for i in range(3):
        for x in range(int(n / 2)):
            for y in range(int(n / 2)):
                a[x, y, i] = wave[x, y, i]

    np.set_printoptions(precision=2, suppress=True, formatter={'all': lambda
```

```
x: f'{x:0.2f}'})
        if need_print:
            print('Luminance matrix after wavelet transformation (Y):')
            print(wave[:, :, 0])
            print('U matrix after wavelet transformation:')
            print(wave[:, :, 1])
            print('V matrix after wavelet transformation:')
            print(wave[:, :, 2])

        return wave, a


    def quant(yuv, q, n):
        for i in range(3):
            for x in range(n):
                for y in range(n):
                    if np.abs(yuv[x, y, i]) < q:
                        yuv[x, y, i] = 0
```

```
            else:
                yuv[x, y, i] = (np.round(yuv[x, y, i] / q))


    print('Luminance matrix after quantization (Y):')
    print(yuv[:, :, 0])
    print('U matrix after quantization:')
    print(yuv[:, :, 1].astype(int))
    print('V matrix after quantization:')
    print(yuv[:, :, 2].astype(int))


    return yuv


def bypass(yuv, n):
    item = ["Luminance matrix (Y):", "U matrix:", "V matrix:"]
    V = 0
    for i in range(3):
        print(item[i])
```

```
        for y in range(n):
            l, h, v = arithmetic(yuv[:, y, i])
            V += v
            print(f'l{y} = {l}\nh{y} = {h}\n')
    print(f'Total file size: {V}')
    print(f'EC: {6144 / V}')


def arithmetic(input_string):
    symbol_freq = {}
    for char in input_string:
        if char in symbol_freq:
            symbol_freq[char] += 1
        else:
            symbol_freq[char] = 1

    v = 0
```

```
        interval_start = Decimal(0.0)

        interval_end = Decimal(1.0)

        for char, freq in symbol_freq.items():

            v += -np.log2(freq / len(input_string)) * freq / len(input_string)

            symbol_freq[char] = [interval_start, interval_start + Decimal(freq /
len(input_string))]

            interval_start = interval_start + Decimal(freq / len(input_string))

        symbol_freq['EOF'] = [interval_start, interval_end - interval_start]



        low = Decimal(0.0)

        high = Decimal(1.0)

        for char in input_string:

            range_size = Decimal(high - low)

            high = Decimal(low + range_size * symbol_freq[char][1])

            low = Decimal(low + range_size * symbol_freq[char][0])



            print(f'{char}: [{low}, {high}]')
```

```
            print(f'{char}: low = {low}, high = {high}')


        return low, high, v
import cv2
import pywt
import numpy as np
from decimal import Decimal, getcontext
from math_helper import *



def shift(ST):
    rgb = cv2.imread('/Users/razrab-ytka/Documents/Projects/suai-labs/6_semester/ТАСИ/2_fix/image.jpg')
    yuv = cv2.cvtColor(rgb, cv2.COLOR_BGR2YUV)


    print("Luminance values (Y) before shift:")
    print(yuv[:, :, 0])
```

```
    for x in range(16):
        for y in range(16):
            yuv[x, y, 0] -= 2 ** (ST[0]) - 1

    print("Luminance values (Y) after shift:")
    print(yuv[:, :, 0])

    rgb_shifted = cv2.cvtColor(yuv, cv2.COLOR_YUV2BGR)
    return rgb_shifted


ST = [5, 5, 5]
q = 10
rgb = shift(ST)

yuv = pixel(rgb)
```

```
print(yuv)



yuv, a = wavelet(yuv, 16)

yuv, a = wavelet(a, 8, need_print=True)



yuv = quant(yuv, q, 8)



getcontext().prec = 38

bypass(yuv, 8)
```