Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет телекоммуникаций и
информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Представление Графической Информации
Расчетно-графическое Задание

Вариант №9

Выполнил:
Студент IV курса ИВТ,
группы ИП-713
Михеев Никита Алексеевич

Работу проверил:
доцент кафедры ПМиК
Перцев И.В.

Новосибирск 2020 г.

1. Задание

Преобразовать True Color PCX файл в 256-цветный BMP файл.

2. Описание алгоритма

Алгоритм был разработан на языке программирования Python, версии 3.9 для True Color PCX файлов.

Сначала считываются все данные из файла в объект image, а все данные о пикселях сохраняются в объекте colors. Далее идет просчет упрощенной палитры, состоящей из 256 цветов. Количество цветов уменьшается зачёт вычисления дельт между соседними цветами по формуле:

delta = $(red1 - red2)^2 + (green1 - green2)^2 + (blue1 - blue2)^2$, где red, green, blue – цвета пикселей в цветовой палитре RGB. После вычисления сокращенной палитры, происходит замена цветов в исходном изображении на ближайшие цвета из полученной палитры, для этого опять же используется обозначенная выше формула для каждого пикселя исходного изображения.

После всех преобразований инициализируется новый BMP файл с размерами исходного изображения по ширине и высоте, в него записывается полученная информация о цветах пикселей в файле rgr-result.bmp, изображение выводится на экран.

3. Результат работы программы

Рис.1 – оригинальное изображение.

Рис.2 – преобразованное изображение.

4. Листинг

Rgr.py:

```
from PIL import Image


def count_delta(left, right):
    return sum([(x - y) ** 2 for x, y in zip(left, right)], 0)


def find_palette(colors_count):
    left = 0
    right = 10000
    result = set()
    while left <= right:
        middle = (right + left) // 2
        print(middle)
        palette = set()
        is_fit = True
        for i in colors_count:
            if all([count_delta(i[0], j) >= middle for j in palette]):
                palette.add(i[0])
            if len(palette) >= 256:
                is_fit = False
                break
        if is_fit:
            right = middle - 1
        else:
            result = palette
            left = middle + 1
    return result


def find_closest(color_palette, color):
    res = (0, 0, 0)
```

```
    for i in color_palette:
        if count_delta(res, color) > count_delta(i, color):
            res = i
    return res


def main() -> int:
    image = Image.open('TrueColor.pcx')
    image.show()
    colors = image.getdata()

    colors_count = {}
    for color in colors:
        colors_count[color] = colors_count[color] + 1 if color in
colors_count else 1
    colors_count = list(colors_count.items())
    colors_count.sort(key=lambda x: x[1], reverse=True)

    color_palette = list(find_palette(colors_count))
    print(color_palette)
    colors = [find_closest(color_palette, color) for color in colors]

    converted_image = Image.new('RGB', (image.width, image.height))
    converted_image.putdata(colors)
    converted_image.save('rgr-result.bmp')
    converted_image.show()
    return 0


if __name__ == '__main__':
    exit(main())
```