ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра прикладной математики и кибернетики

Расчетно-графическое задание

По дисциплине «Представление графической информации»

Вариант 4

Выполнил:  
Студент 4 курса ИВТ,

группы ИП-813

Бурдуковский Илья Александрович

Работу проверил: доцент кафедры ПМиК

Перцев И.В.

Новосибирск 2021 г.

**Оглавление**

[Задание 3](#_Toc68878292)

[Описание алгоритма 4](#_Toc68878293)

[Результат работы программы 5](#_Toc68878294)

[Листинг 7](#_Toc68878295)

# Задание

Написать программу-конвертор количества цветов в изображении.

Предлагаемый алгоритм. Для уменьшения количества цветов выбираются наиболее часто встречаемые цвета в исходном изображении. Причем эти цвета не должны быть слишком похожими друг на друга. Для сравнения цветов вычисляются разности между RGB составляющими.

*Delta=(R1-R2)2 + (G1-G2)2+ (B1-B2)2*

После формирования новой палитры цвета в заменяются на наиболее похожие из записанных в палитру.

Можно использовать любой другой алгоритм преобразования цветов (например медианного сечения) главное требование – алгоритм должен быть реализован самостоятельно.

Программа должны выводить изображение на экран до и после конвертирования.

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

# Описание алгоритма

Полученная программа считывает изображение, формирует новую палитру из нужного количества цветов и, с помощью расчёта разностей между RGB составляющими, подбирает новые цвета для картинки:

Метод count\_delta - принимает 2 набора rgb и возвращает рассчитанную для них дельту.

Метод find\_palette - принимает список всех пикселей, проходя по нему сглаживает оттенок, обнуляя крайние правые 4 бита, и считает количество повторений каждого сглаженного цвета. Затем метод сортирует цвета по количеству повторений и возвращает нужное количество для палитры из начала отсортированного списка.

Метод find\_closest - принимает новую палитру и список всех пикселей картинки, подбирает для каждого пикселя новый цвет и возвращает список пикселей с новым цветом.

После всех преобразований инициализируется новый PCX файл с размерами исходного изображения по ширине и высоте, в него записывается полученная информация о цветах пикселей.

# Результат работы программы

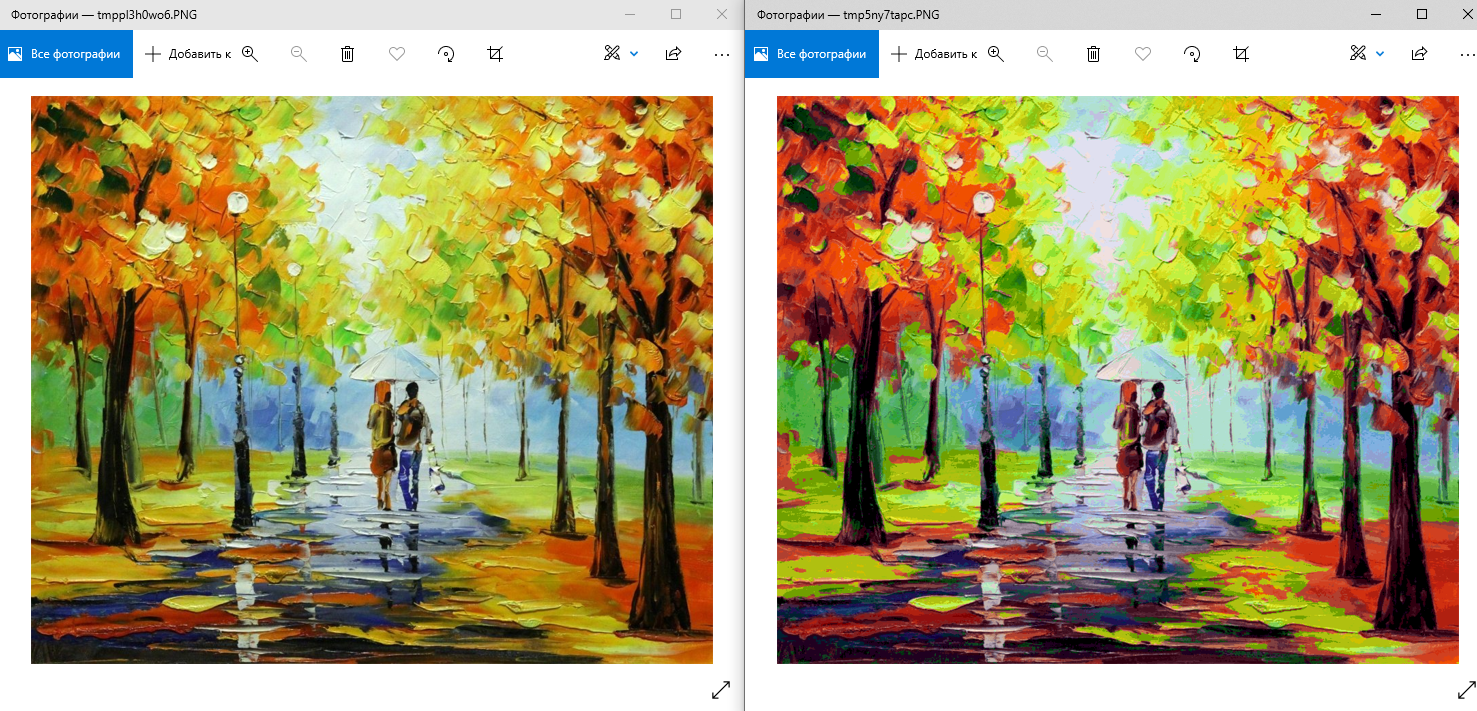
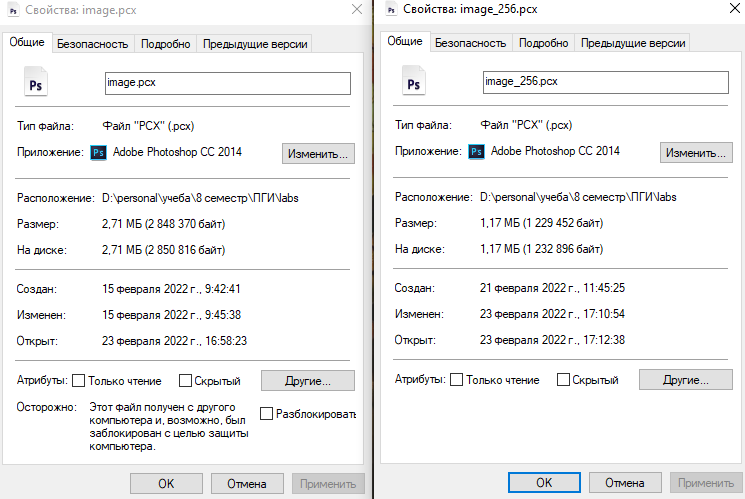


image.pcx – оригинальное изображение image\_256.pcx – преобразованное



Благодаря методу сжатия палитры и методу нахождению максимально приближенных цветов в новой палитре – изображение после преобразования осталось максимально схожим.

Преобразованный файл в 2 раза меньше исходного, как можно увидеть по свойствам.

# Листинг

from PIL import Image  
  
smooth\_coefficient = 15  
colors\_count\_mode = 256

def count\_delta(left, right):  
 return sum([(x - y) \*\* 2 for x, y in zip(left, right)])  
  
  
def find\_palette(pixels):  
 colors = {}   
 for color in pixels:  
 smoothed\_color = (color[0] >> 4 << 4, color[1] >> 4 << 4, color[2] >> 4 << 4)  
 colors[smoothed\_color] = colors[smoothed\_color] + 1 if smoothed\_color in colors else 1  
 colors = list(colors.items())  
 colors.sort(key=lambda x: x[1], reverse=True)  
 new\_colors\_palette = []  
 for i in range(colors\_count\_mode):  
 color = colors.pop()[0]  
 new\_colors\_palette.append(color)  
 return new\_colors\_palette  
  
  
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('image.pcx')  
 image.show()  
 pixels = image.getdata()  
 color\_palette = find\_palette(pixels)   
 pixels = [find\_closest(color\_palette, pixel) for pixel in pixels]  
 converted\_image = Image.new('RGB', (image.width, image.height))  
 converted\_image.putdata(pixels)  
 converted\_image.save('image\_256\_another.pcx')  
 converted\_image.show()  
 return 0  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 exit(main())