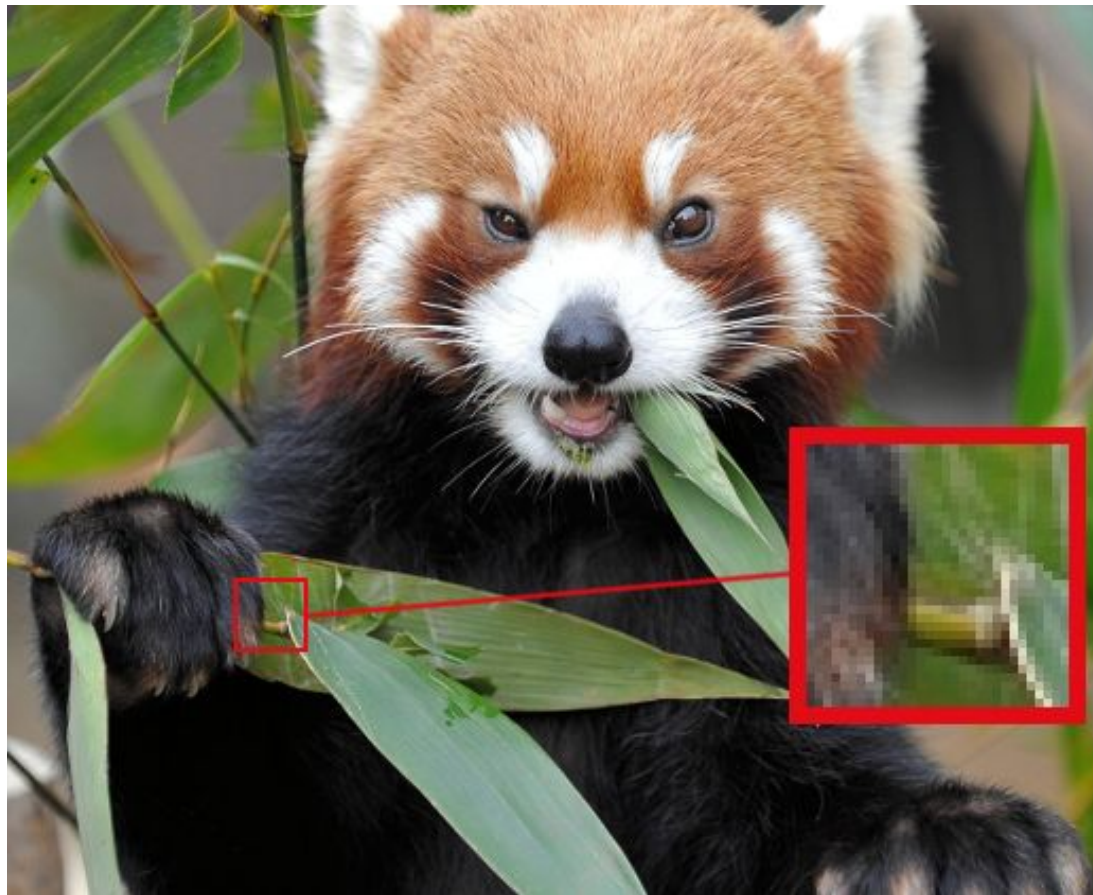


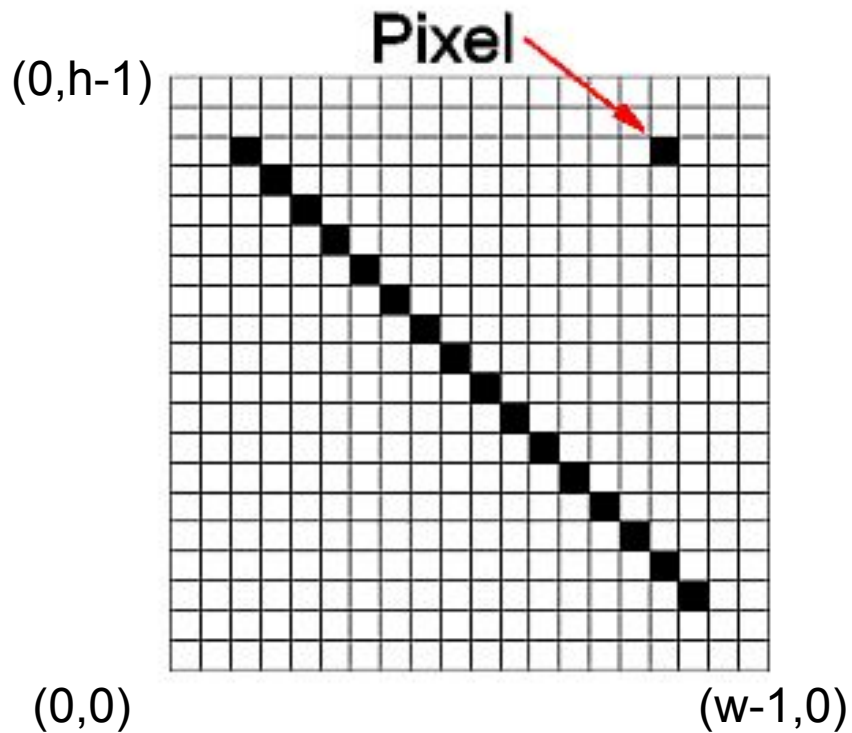
Обработка изображений

ЛКШ.2015.Зима.С'

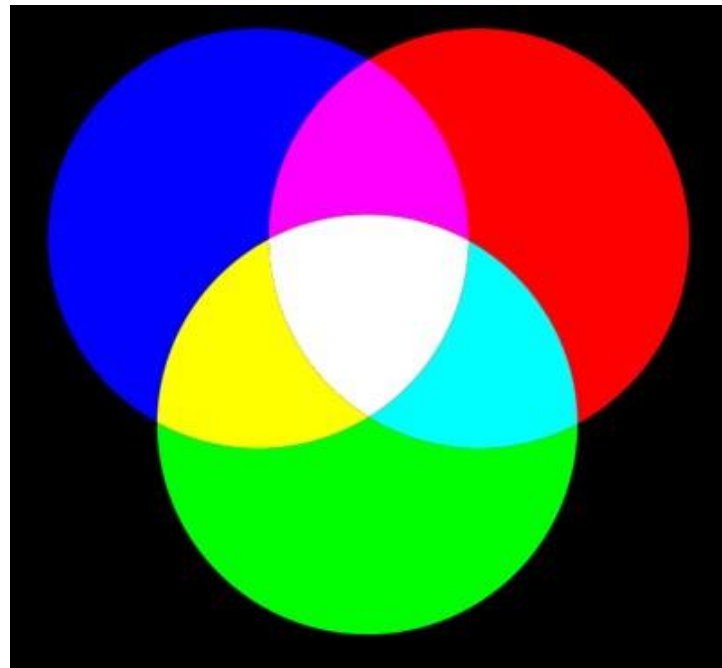
Представление изображений



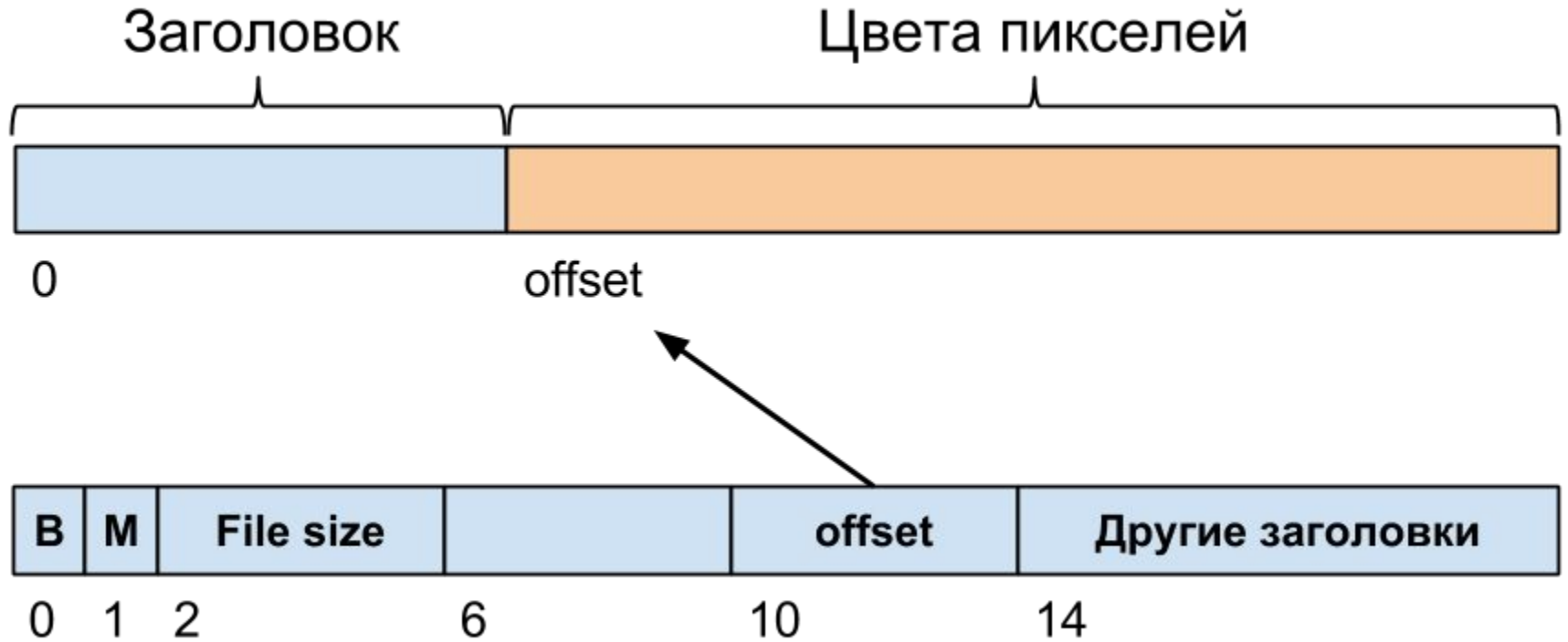
Представление изображений



RGB (0..255)



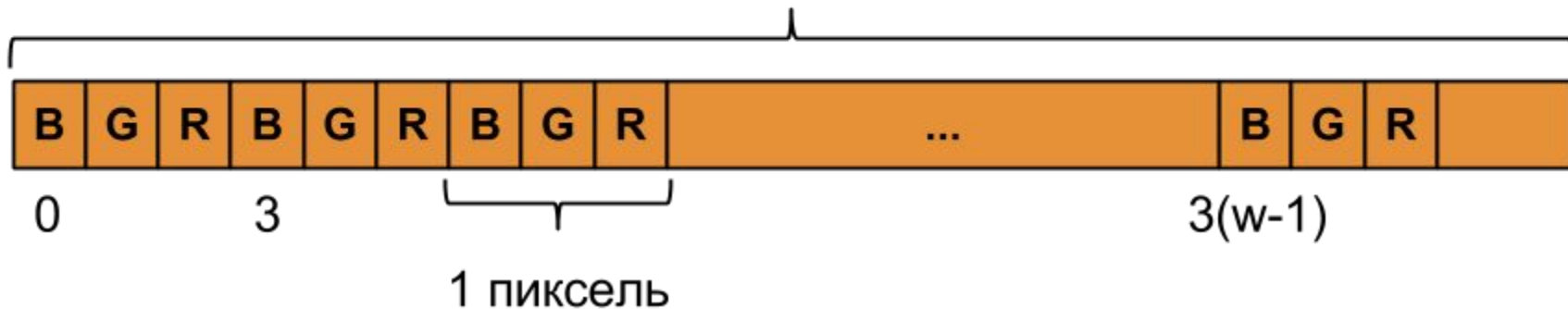
Формат BMP



Формат BMP: цвета пикселей



Количество байт должно быть кратно 4



Код: считывание изображения

```
def readImage(file_name):  
    with open(file_name, 'rb') as f:  
        raw_data = f.read()  
        image_data_offset =  
            bytesToNumber(raw_data[10:14])  
    return Image(  
        list(raw_data[:image_data_offset]),  
        list(raw_data[image_data_offset:])  
    )
```

```
Image readImage(const string &file_name) {  
    ifstream f(file_name, ios::binary);  
  
    std::vector<char> header(14);  
    std::vector<char> data;  
  
    f.read(header.data(), header.size());  
    int size = bytesToNumber(header, 2, 6);  
    int image_data_offset = bytesToNumber(header, 10, 14);  
  
    header.resize(image_data_offset);  
    f.read(header.data() + 14, image_data_offset - 14);  
  
    data.resize(size - image_data_offset);  
    f.read(data.data(), data.size());  
  
    return Image(header, data);  
}
```

Код: bytesToNumber

```
def bytesToNumber(bs):  
    x = 0  
    for b in reversed(bs):  
        x = x * 256 + b  
    return x
```

```
int bytesToNumber(  
    const vector<char> &bytes,  
    int start, int end) {  
    int x = 0;  
    for (int i = end - 1; i >= start; --i) {  
        x = x * 256 + (unsigned char)bytes[i];  
    }  
    return x;  
}
```

Код: класс Image

```
class Image:
    def __init__(self, header, data):
        self.header = header
        self.data = data
        self.width = bytesToNumber(header[18:22])
        self.height = bytesToNumber(header[22:26])
        self.row_size = (3 * self.width + 3) // 4 * 4

    def write(self, file_name):
        with open(file_name, "wb") as f:
            f.write(bytes(self.header))
            f.write(bytes(self.data))
```

```
struct Image {
    vector<char> header;
    vector<char> data;
    int width, height, row_size;

    Image(const vector<char> &h, const vector<char> &d) {
        header = h;
        data = d;
        width = bytesToNumber(header, 18, 22);
        height = bytesToNumber(header, 22, 26);
        row_size = (3 * width + 3) / 4 * 4;
    }

    void write(const string &file_name) {
        ofstream f(file_name, ios::binary);
        f.write(header.data(), header.size());
        f.write(data.data(), data.size());
    }
};
```

Код: класс Color

```
class Color:
    def __init__(self, r, g, b):
        self.r, self.g, self.b = r, g, b

    def __getitem__(self, k):
        return [self.r, self.g, self.b][k]

    def __setitem__(self, k, v):
        if k == 0:
            self.r = v
        elif k == 1:
            self.g = v
        elif k == 2:
            self.b = v
```

```
struct Color {
    int r, g, b;

    Color(int red, int green, int blue) {
        r = red;
        g = green;
        b = blue;
    }

    const int& operator [](int i) const {
        if (i == 0) return r;
        if (i == 1) return g;
        if (i == 2) return b;
    }

    int& operator [](int i) {
        if (i == 0) return r;
        if (i == 1) return g;
        if (i == 2) return b;
    }
};
```

Код: Image.getPixel и Image.setPixel

```
def fit(x, mn, mx):  
    return min(max(x, mn), mx)
```

```
def getPixel(self, x, y):  
    offset = y * self.row_size + 3 * x  
    return Color(  
        self.data[offset + 2],  
        self.data[offset + 1],  
        self.data[offset])
```

```
def setPixel(self, x, y, color):  
    offset = y * self.row_size + 3 * x  
    self.data[offset + 2] = fit(int(color.r), 0, 255)  
    self.data[offset + 1] = fit(int(color.g), 0, 255)  
    self.data[offset] = fit(int(color.b), 0, 255)
```

```
int fit(int x, int mn, int mx) {  
    return min(max(x, mn), mx);  
}
```

```
Color getPixel(int x, int y) {  
    int offset = y * row_size + 3 * x;  
    return Color(  
        (unsigned char)data[offset + 2],  
        (unsigned char)data[offset + 1],  
        (unsigned char)data[offset]);  
}
```

```
Color setPixel(int x, int y, const Color &color) {  
    int offset = y * row_size + 3 * x;  
    data[offset + 2] =  
        (unsigned char)fit(color.r, 0, 255);  
    data[offset + 1] =  
        (unsigned char)fit(color.g, 0, 255);  
    data[offset] =  
        (unsigned char)fit(color.b, 0, 255);  
}
```

Код: пример использования

```
image = readImage('sample.bmp')
result = image.copy()

for y in range(image.height):
    for x in range(image.width):
        c = image.getPixel(x, y)
        # Здесь мог бы быть ваш код
        result.setPixel(x, y, new_color)

result.write('result.bmp')
```

```
int main() {
    Image image = readImage("roma.bmp");
    Image result(image);

    for (int y = 0; y < image.height; ++y) {
        for (int x = 0; x < image.width; ++x) {
            Color c = image.getPixel(x, y);
            // Здесь мог бы быть ваш код
            result.setPixel(x, y, new_color);
        }
    }

    result.write("result.bmp");
    return 0;
}
```

Задание 1: Негатив



(R, G, B)



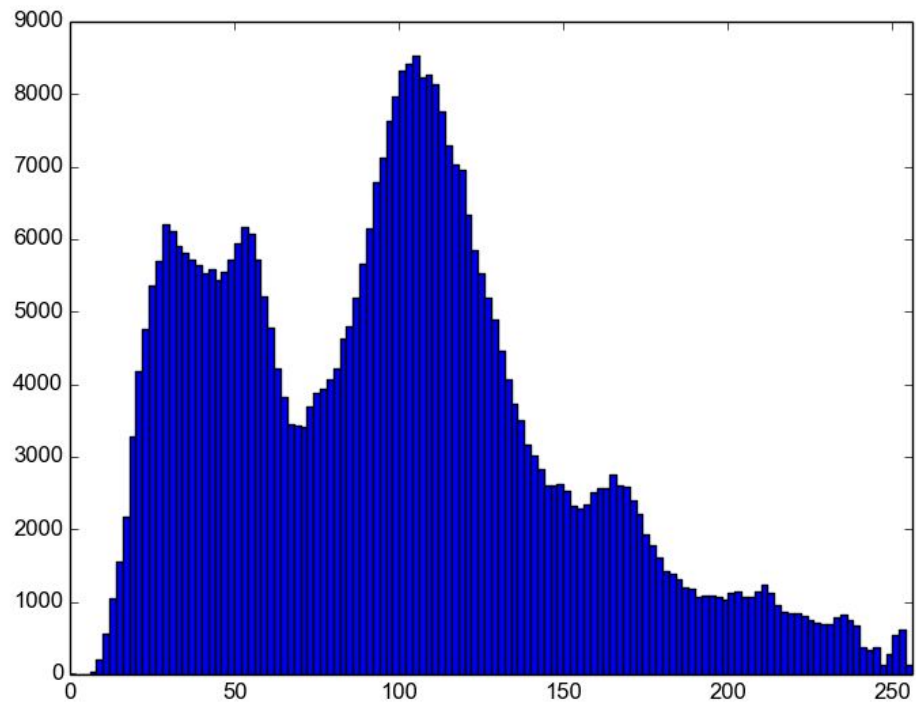
(255 - R, 255 - G, 255 - B)

Яркость

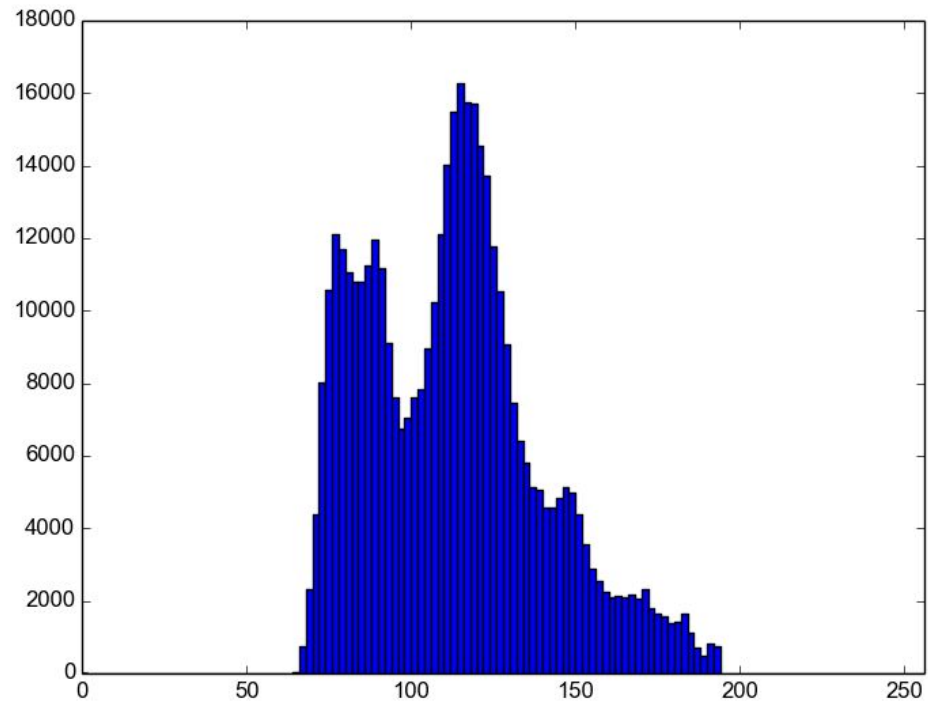


Просто: $R + G + B$ | Сложно: $a * R + b * G + c * B$

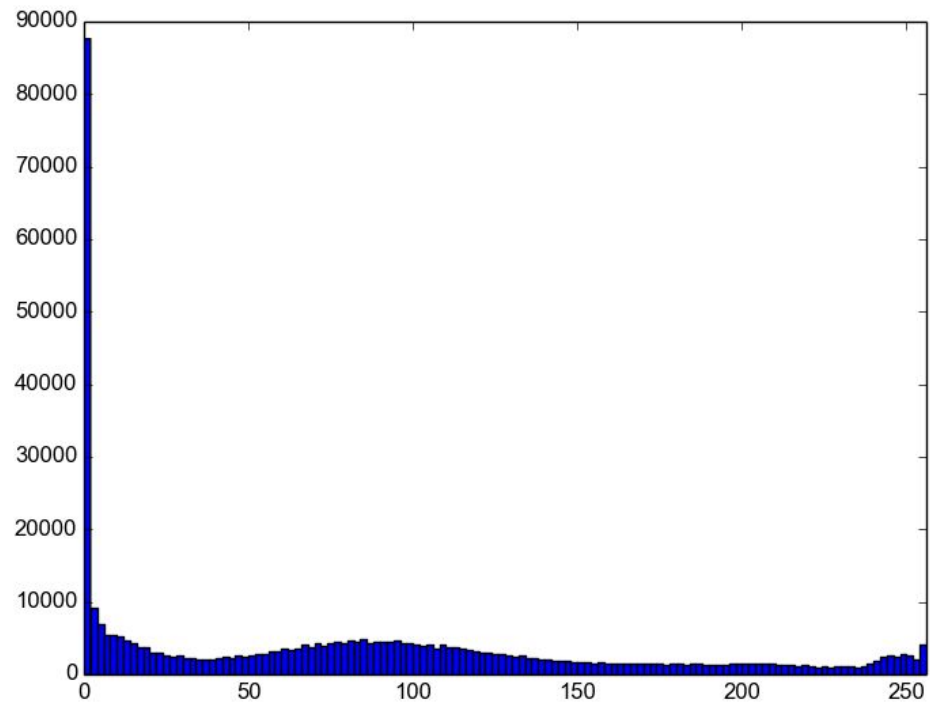
Контрастность - 1



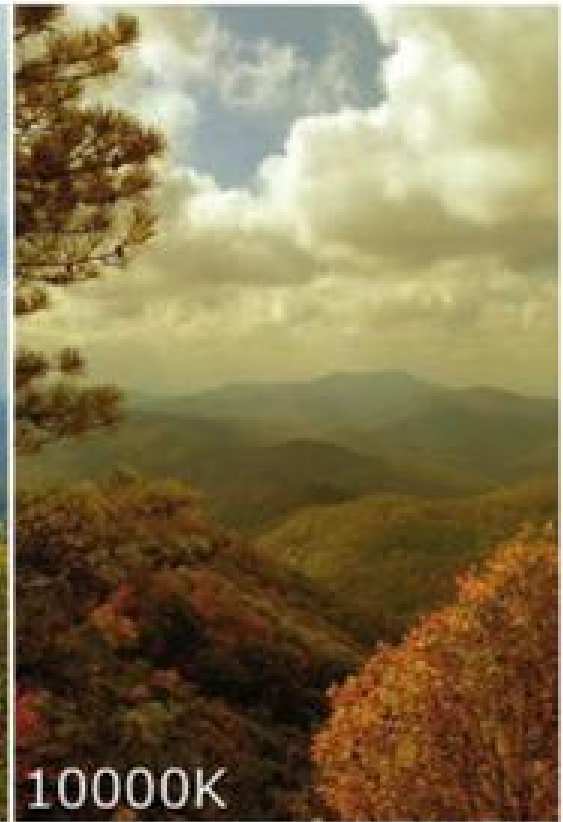
Контрастность - 2



Контрастность - 3



Баланс белого



Задание 2: Серый мир

Задача: исправить баланс белого

Идея: сделать равными средние значения красной, синей и зелёной компонент

Как:

- Посчитать средние значения компонент:

$$R_{avg} = \frac{\text{sum}(R(x, y))}{W \cdot H}, \quad G_{avg} = \frac{\text{sum}(G(x, y))}{W \cdot H}$$

$$B_{avg} = \frac{\text{sum}(B(x, y))}{W \cdot H}, \quad Avg = \frac{R_{avg} + B_{avg} + G_{avg}}{3}$$

- Пересчитать новые значения:

$$R_{new} = R \cdot \frac{Avg}{R_{avg}}, \quad G_{new} = G \cdot \frac{Avg}{G_{avg}}, \quad B_{new} = B \cdot \frac{Avg}{B_{avg}}$$



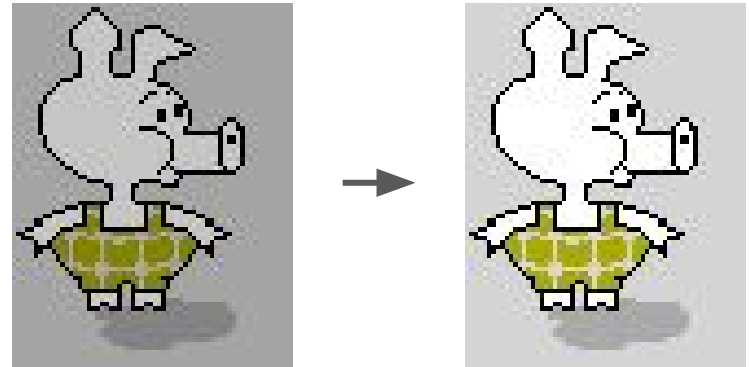
Задание 3: Автоконтраст

Задача: увеличить контрастность изображения

Идея: в каждом из каналов изменить значения цвета так, чтобы самому тёмному соответствовало значение 0, а самому яркому - 255. Все остальные значения изменить равномерно

Как:

- Найти минимальное и максимальное значения цвета для каждого из каналов
- Пересчитать новые значения по формулам
 $r \rightarrow (r - r_{\min}) / (r_{\max} - r_{\min}) * 255$
 $g \rightarrow (g - g_{\min}) / (g_{\max} - g_{\min}) * 255$
 $b \rightarrow (b - b_{\min}) / (b_{\max} - b_{\min}) * 255$



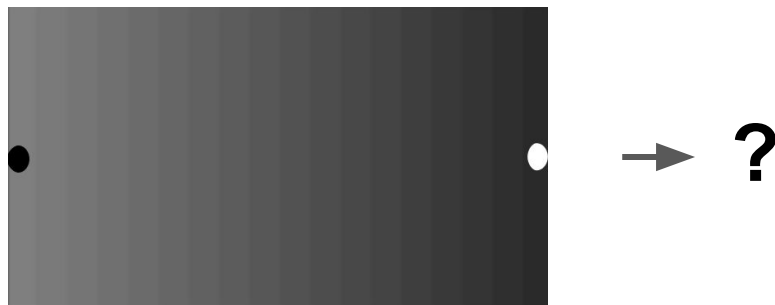
Задание 4*: Автоконтраст (5%)

Задача: увеличить контрастность изображения

Идея: в каждом из каналов изменить значения цвета так, чтобы 5% самых тёмных соответствовало значению 0, а 5% самых светлых - 255. Все остальные значения изменить равномерно.

Как:

- Для каждого канала и каждого значения цвета посчитать количество раз, которое он встречается в изображении
- Найти максимальное среди 5% самых тёмных и минимальное среди 5% самых светлых значения цвета для каждого канала
- Пересчитать новые значения по формулам, аналогичным предыдущему заданию



Задание 5*: Обрезание краёв

Задача: обрезать L пикселей слева, R пикселей справа, T пикселей сверху и B пикселей снизу. Сохранить полученное изображение меньшего размера в файл.

Идея: изменить класс Image так, что бы он поддерживал изменение размера изображения. Для этого придётся изменять поля width и height заголовка, а также размер и количество строчек в области с данными.