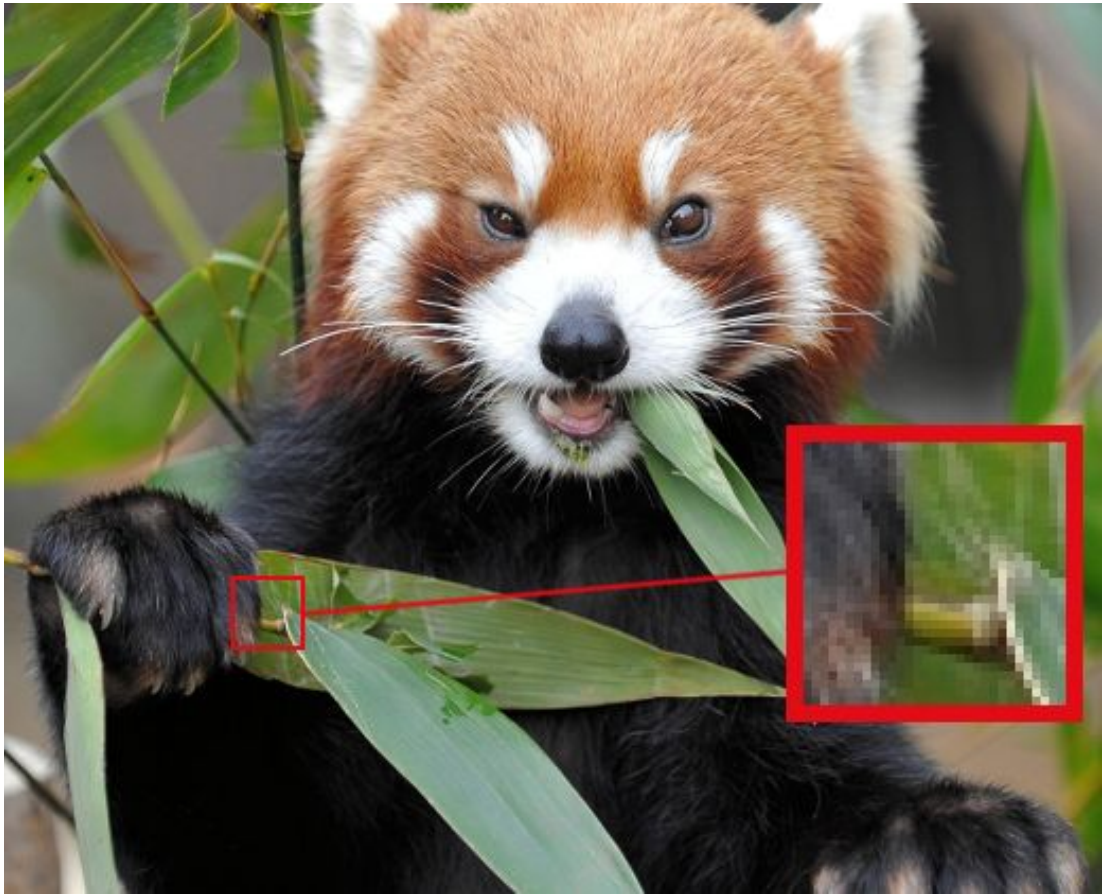
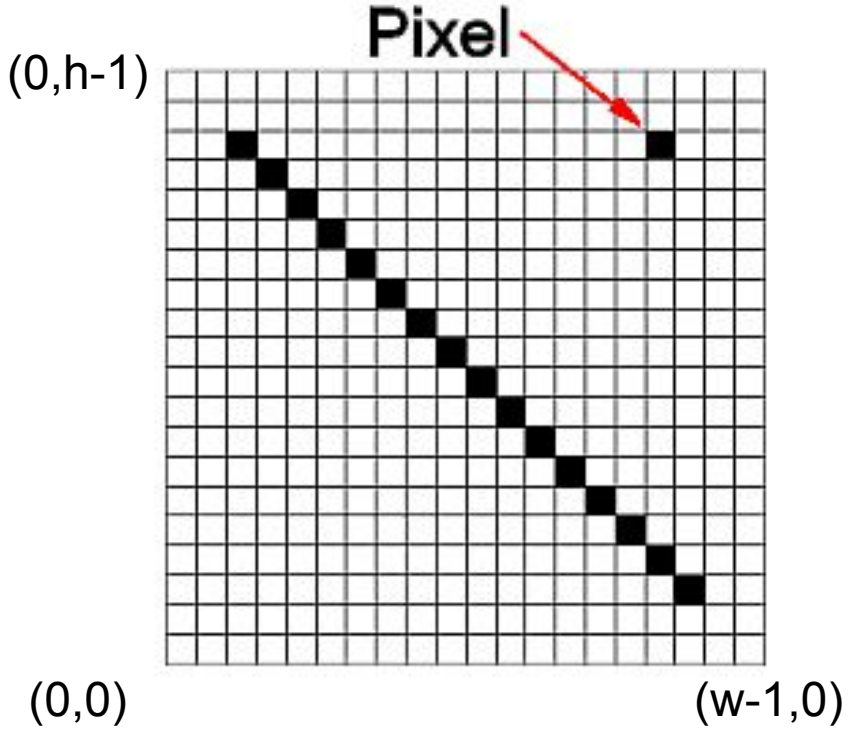
Обработка изображений

ЛКШ.2017.Зима.C’

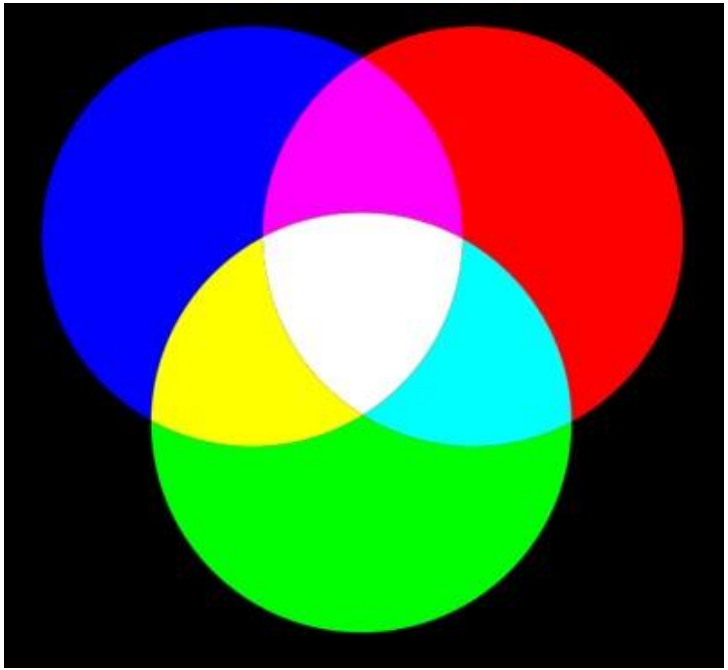
Представление изображений



Представление изображений



RGB (0..255)



Первые шаги

Установка модуля для работы с изображениями:

$ pip3 install pillow

Импорт модуля в питоне:

```
from PIL import Image
```

Базовые действия

```
img = Image.open("image.png")
pixels = img.load()
for x in range(img.width):
    for y in range(img.height):
        r, g, b = pixels[x, y]
        pixels[x, y] = g, b, r
img.save('result.jpg')
```

Для цветных изображений пиксель - это кортеж (R, G, B)

Для изображений с прозрачностью - (R, G, B, alpha)

Как сдавать задачи

Со стандартного ввода вводится:

В первой строке - имя входного файла, например test.jpg
Во второй строке - имя выходного файла, например result.jpg

В каждой задаче могут быть дополнительные параметры в следующих строках.

Программа должна преобразовать изображение из первого файла и записать результат во второй.

Код сдавайте в ejudge. Все решения проверяются вручную.

Задание 1: Негатив

(R, G, B)

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

Яркость

Просто: (R + G + B) / 3 | Сложно: a * R + b * G + c * B

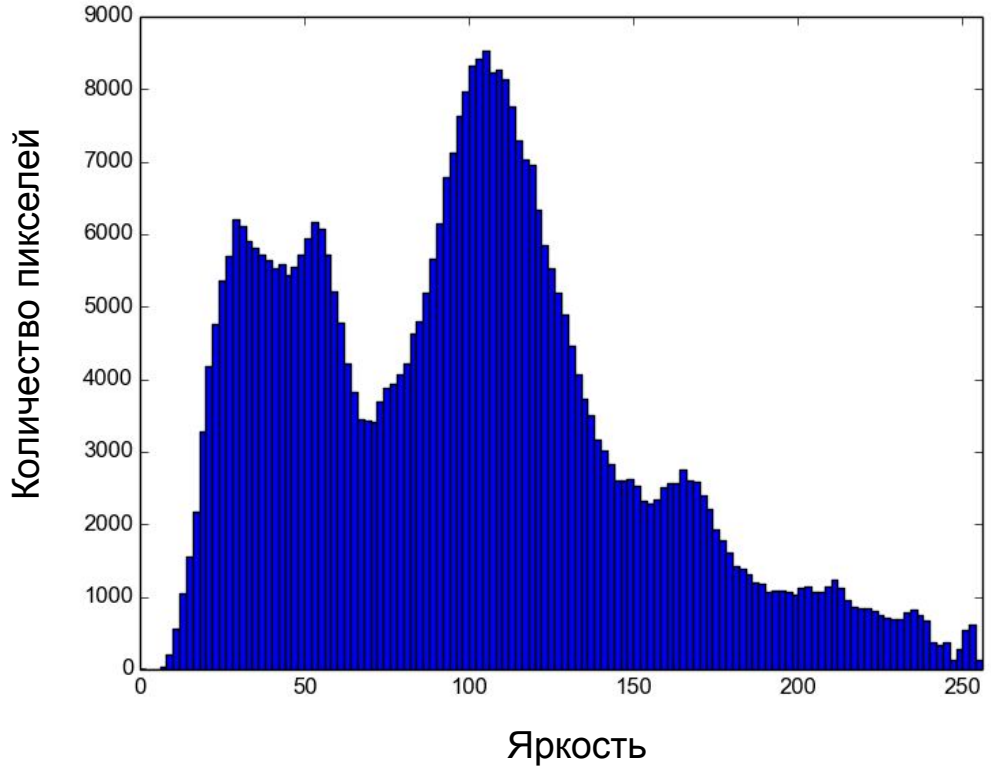
Задание 2: Яркость

Дополнительный параметр в третьей строке: C

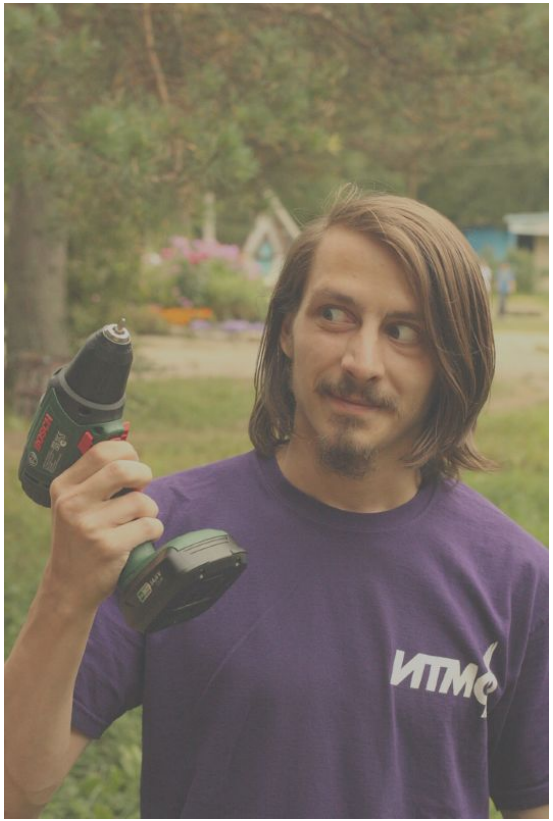
(R, G, B)

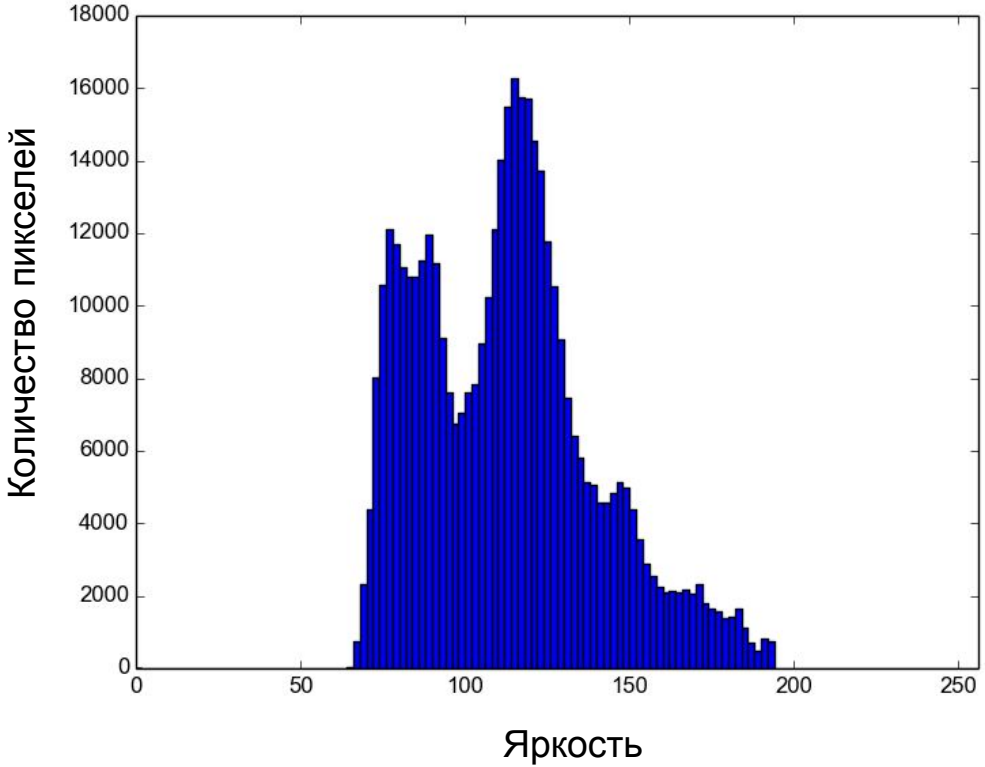
(R + C, G + C, B + C)

Контрастность - 1

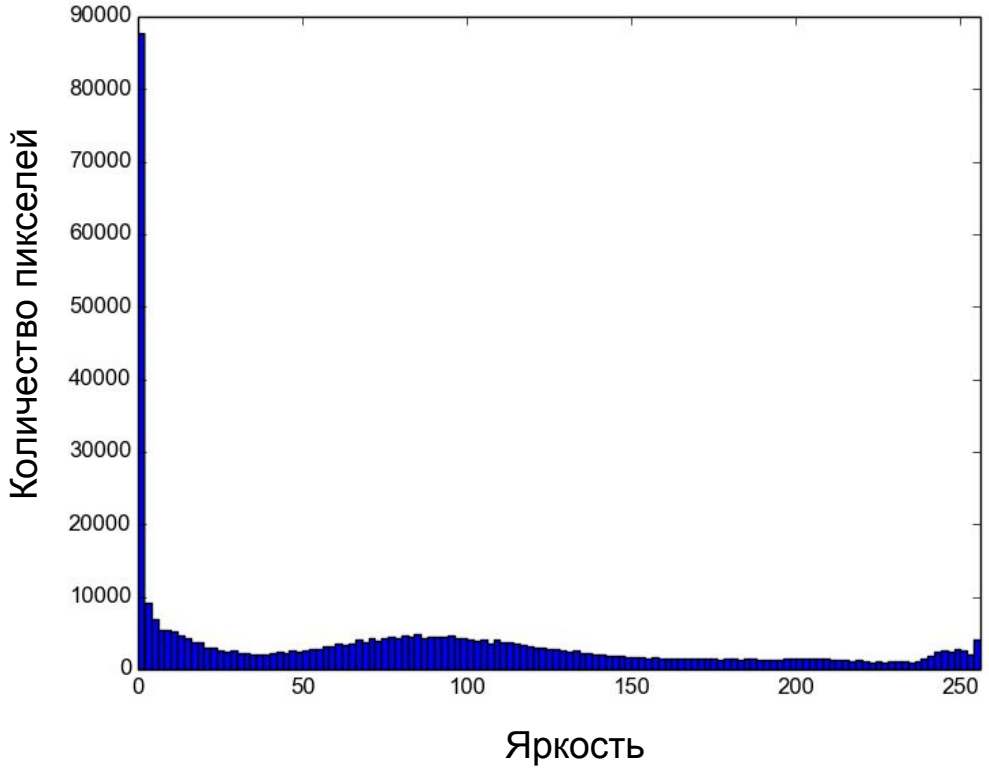


Контрастность - 2





Контрастность - 3



Задание 3: Изменение контрастности

Задача: изменить контрастность изображения с заданным коэффициентом, не меняя средней яркости.

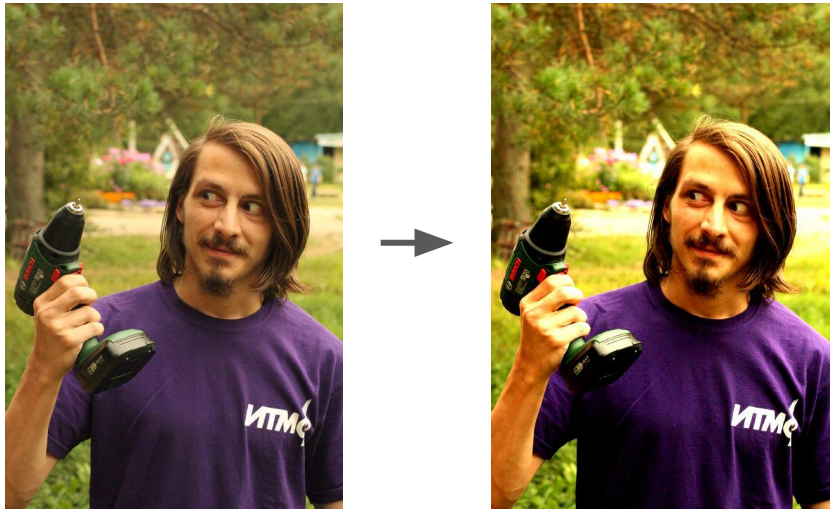
Параметры: C - коэффициент изменения контрастности

Идея: растягивает яркости относительно среднего значения

Как:

- Посчитать среднюю яркость пикселей по всему изображению (`L_avg`)
- Пересчитать новые значения по формулам:

```
R -> L_avg + (R - L_avg) * C
G -> L_avg + (G - L_avg) * C
B -> L_avg + (B - L_avg) * C
```



Баланс белого

Задание 4: Серый мир

Задача: исправить баланс белого

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

Как:

- Посчитать средние значения компонент:

```
R_avg = sum(R(x, y)) / (W * H)
G_avg = sum(G(x, y)) / (W * H)
B_avg = sum(B(x, y)) / (W * H)
L_avg = (R_avg + G_avg + B_avg) / 3
```

- Пересчитать новые значения:

```
R -> R * L_avg / R_avg
G -> G * L_avg / G_avg
B -> B * L_avg / B_avg
```

Задание 5: Автоконтраст

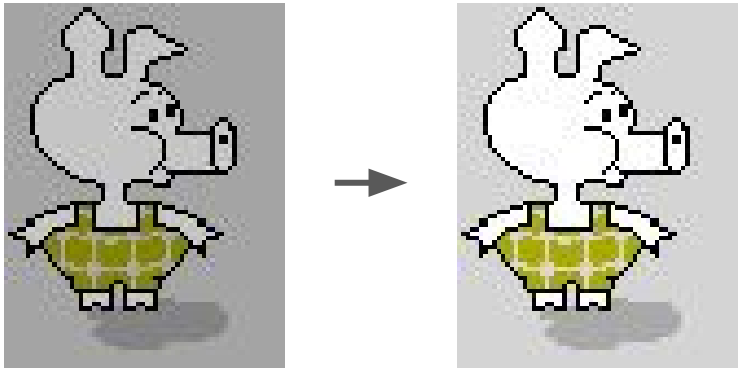
Задача: увеличить контрастность изображения

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 -> (R - R_min) / (R_max - R_min) * 255
G -> (G - G_min) / (G_max - G_min) * 255
B -> (B - B_min) / (B_max - B_min) * 255
```



Задание 6: Обрезание краёв

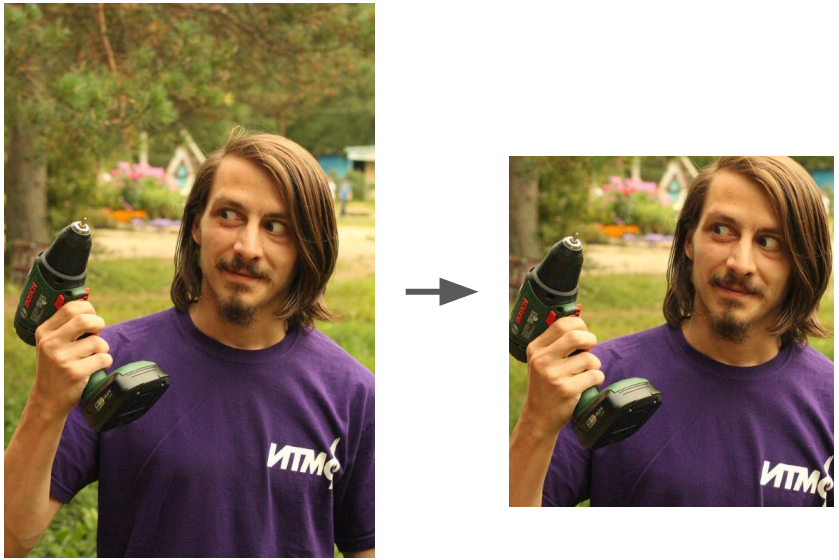
Задача: обрезать L пикселей слева, R пикселей справа, T пикселей сверху и B пикселей снизу. Сохранить новое изображение меньшего размера в файл.

Параметры: L - слева, R - справа, T - сверху, B - снизу, 4 числа через пробел

Как:

```
# Цветовая схема, размеры, изначальный цвет
img = Image.new("RGB", (W, H), "white")
```

В качестве цветовой схемы можно передавать цветовую схему другого изображения. Её можно получить написав `img.mode`.



Задания на выбор

Из следующих заданий можете выбирать и делать любые

Задание 7*: Автоконтраст (5%)

Задача: увеличить контрастность изображения

Идея: в каждом из каналов изменить значения цвета так, чтобы 5% самых тёмных соответствовало значение 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 заданию про Автоконтраст

Задание 8: Фильтр «Стекло»

Задача: в результирующем изображении каждый пиксель с координатами `(x, y)` является случайным пикселем из окрестности `(x-D..x+D, y-D..y+D)` исходного изображения

Параметры: D - размер области

Как: для выбора случайного смещения можно использовать функцию `randrange` из модуля `random`:

```
from random import randrange

# delta - случайное целое число от -5 до 5
delta = randrange(-5, 6)
```

Задание 9: Фильтр «256 оттенков серого»

Задача: сделать изображение чёрно-белым

Идея: для каждого пикселя присваиваем компонентам R, G и B значение яркости этого пикселя

Задание 10: Фильтр «Сепия»

Задача: сделать фото устаревшим, чёрно-белым с сильно коричневым оттенком

Параметры: k - степень коричневого

Идея: сделаем фото чёрно-белым, добавим коричневого (красного и зелёного)

Как:

```
middle = (R + G + B) / 3
R -> middle + 2 * k
G -> middle + k
B -> middle
```

Задание 11: Фильтр «Mayfair»

Задача: затемнить края, сделать центр более ярким

Идея: изменять яркость в зависимости от расстояния до центра изображения

Параметры: два числа - изменение яркости в центре и в самой удалённой точке

Задание 12: Бинаризация изображения

Задача: оставить в изображении только два цвета - чёрный и белый

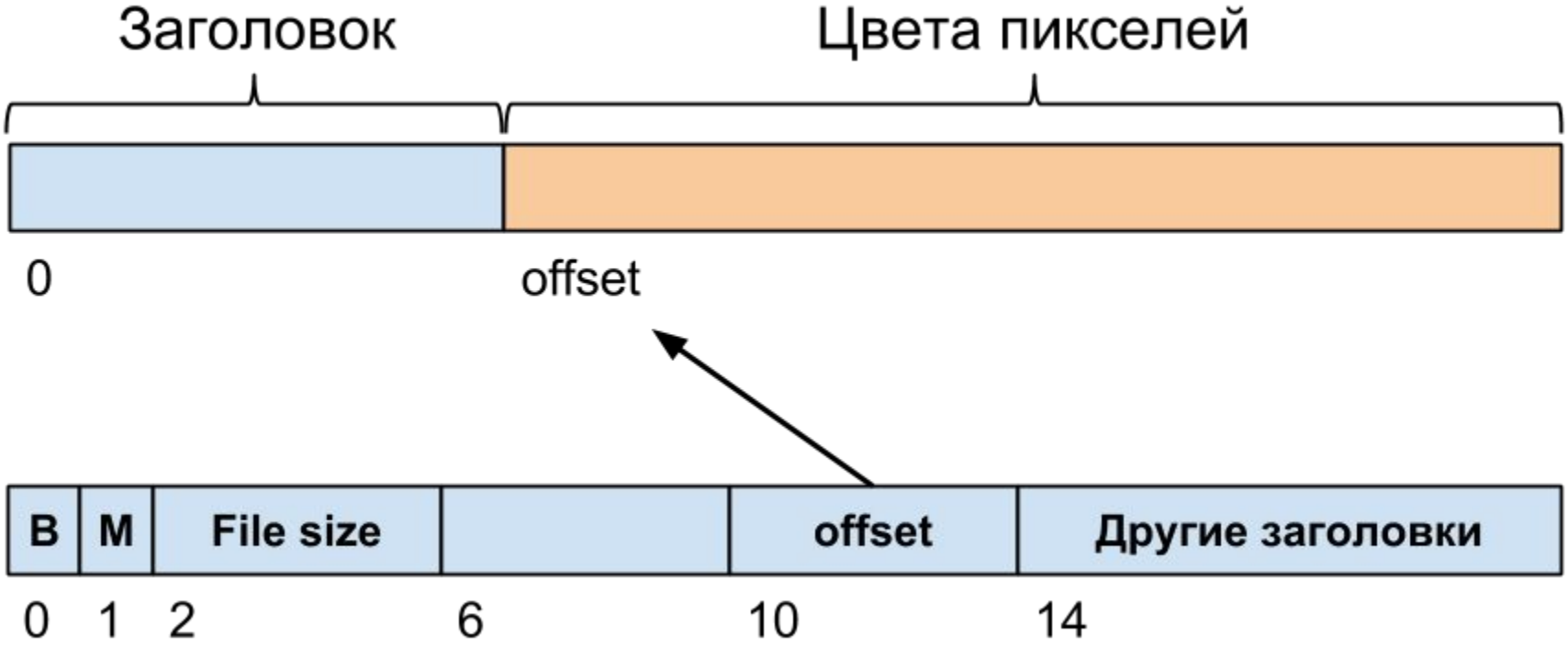
Идея: все пиксели с яркостью меньше заданного порога делаем чёрными, больше порога - белыми

Параметры: порог яркости

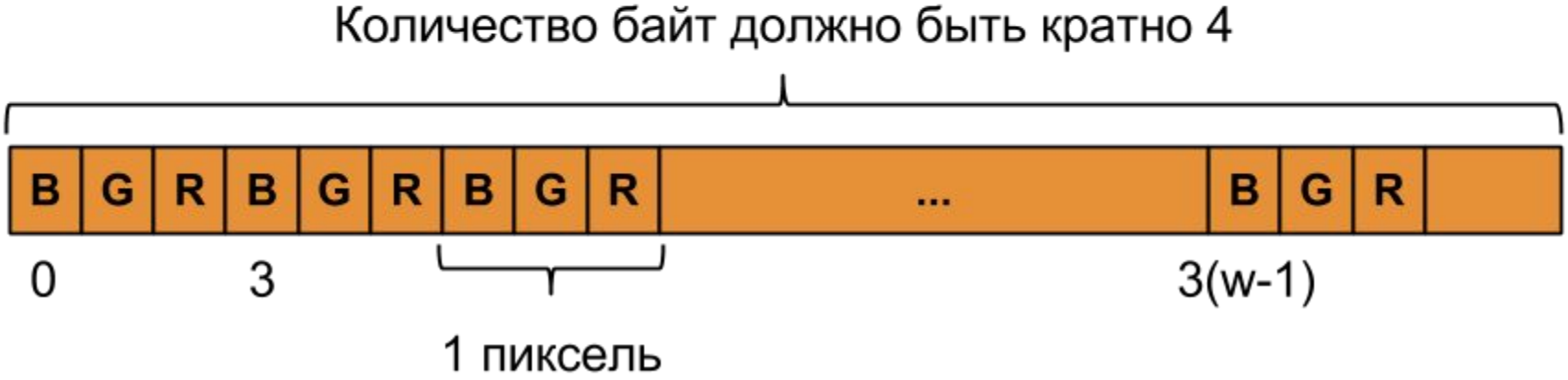
Модификация: определять порог автоматически как среднее значение яркости

Старые слайды

Формат BMP


Заголовок
Цвета пикселей
0
offset
B
M
File size
offset
Другие заголовки
0
1
2
6
10
14

Формат BMP: цвета пикселей



Код: считывание изображения

```python
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:])
    )
```

```cpp
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

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

```cpp
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

```python
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))
```

```cpp
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

```python
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
```

```cpp
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

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

```python
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)
```

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

```cpp
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

```python
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')
```

```cpp
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 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 file_size, width и height заголовка, а также размер и количество строчек в области с данными.