

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №2**  
**по дисциплине «Информатика»**  
**ТЕМА: ВВЕДЕНИЕ В АРХИТЕКТУРУ КОМПЬЮТЕРА**

Студентка гр. 3341

Игнатъев К.А.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2023

## **Цель работы**

Целью работы является изучение библиотеки Pillow, решение 3 подзадач с использованием библиотеки Pillow (PIL), работа с объектом типа `<class 'PIL.Image.Image'>`

## Задание

### Вариант 4

Решить 3 подзадачи, используя библиотеку Pillow (PIL). Для реализации требуемых функций использовать numpy и PIL. Аргумент image в функциях подразумевает объект типа `<class 'PIL.Image.Image'>`

1) Рисование отрезка. Отрезок определяется:

- координатами начала
- координатами конца
- цветом
- толщиной.

Необходимо реализовать функцию `user_func()`, рисующую на картинке отрезок

Функция `user_func()` принимает на вход:

- изображение;
- координаты начала ( $x_0, y_0$ );
- координаты конца ( $x_1, y_1$ );
- цвет;
- толщину.

Функция должна вернуть обработанное изображение.

2) Преобразовать в Ч/Б изображение (любым простым способом).

Функционал определяется:

- Координатами левого верхнего угла области;
- Координатами правого нижнего угла области;
- Алгоритмом, если реализовано несколько алгоритмов преобразования изображения (по желанию студента).

Нужно реализовать 2 функции:

`check_coords(image, x0, y0, x1, y1)` - проверяет координаты области ( $x_0, y_0, x_1, y_1$ ) на корректность (они должны быть неотрицательными, не превышать размеров изображения, поскольку  $x_0, y_0$  - координаты левого

верхнего угла,  $x_1$ ,  $y_1$  - координаты правого нижнего угла, то  $x_1$  должен быть больше  $x_0$ , а  $y_1$  должен быть больше  $y_0$ );

`set_black_white(image, x0, y0, x1, y1)` - преобразовывает заданную область изображения в черно-белый (используйте для конвертации параметр '1'). В этой функции должна вызываться функция проверки, и, если область некорректна, то должно быть возвращено исходное изображение без изменений. Примечание: поскольку черно-белый формат изображения (greyscale) является самостоятельным форматом, а не вариацией RGB-формата, для его получения необходимо использовать метод `Image.convert`.

3) Найти самый большой прямоугольник заданного цвета и перекрасить его в другой цвет. Функционал определяется:

- Цветом, прямоугольник которого надо найти
- Цветом, в который надо его перекрасить.

Написать функцию `find_rect_and_recolor(image, old_color, new_color)`, принимающую на вход изображение и кортежи rgb-компонент старого и нового цветов. Она выполняет задачу и возвращает изображение. При необходимости можно писать дополнительные функции.

## Выполнение работы

### 1 задача:

Объявлена функция `def user_func(image, x0, y0, x1, y1, fill, width)`. Она принимает на вход изображение, координаты начала и конца отрезка, цвет отрезка и его толщину. С помощью метода `Draw` отрисовывается линия с использованием выше указанных этих параметров и возвращается полученное измененное изображение.

### 2 задача:

Объявляется функция `check_coords(image, x0, y0, x1, y1)`, которая проверяет корректность координат, с которыми будет производится работа. Если координаты меньше нуля, координаты больше или равны границ изображения и если координаты нижнего угла меньше координат верхнего угла, то возвращается значение `False`. В иных случаях возвращается значение `True`.

Объявляется функция `set_black_white(image, x0, y0, x1, y1)`. Если координаты корректны (функция `check_coords` вернула `True`), то исходное изображение обрезаются, переводится в чёрно-белый формат и вставляется в начальное.

### 3 задача:

Объявляется функцию `find_max_rect(pix, width, height, x, y, old_color)`. Первоначальные значения координат левого верхнего угла записываются равными `(0,0)`, нижнего правого – (максимальные координаты, являющиеся длиной изображения и высотой изображения соответственно), в массив с текущими проверяемыми пикселями сохраняются переданные значения координат `(x,y)`. Далее, пока массив не пустой, из него берутся первые координаты: Если они находятся в пределах изображения и цвет данного пикселя равен старому цвету, тогда в массив с координатами левого верхнего угла записываются минимальные найденные координаты, с координатами правого нижнего – максимальные. Далее в массив с текущими проверяемыми координатами добавляются все смежные координаты для следующей

проверки и цвет проверенного пикселя обращается в черный, чтобы избежать множественной проверки. Функция выводит кортеж с координатами левого верхнего угла и правого нижнего.

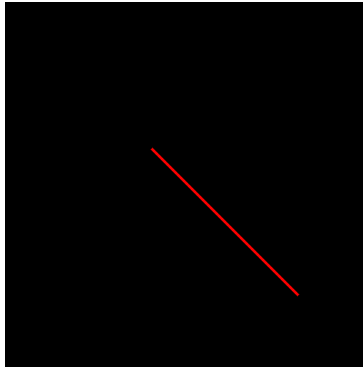
Объявляется функция `find_rect_and_recolor(image, old_color, new_color)`. Создается копия изображения для работы с ним и используется функция `load()` для перевода в пиксели. Определяется длина и ширина изображения с помощью функции `size`. Программа проходит по пикселям копии и если находится пиксель старого цвета, в переменную `coords` сохраняется кортеж из координат левого верхнего и правого нижнего угла прямоугольника, найденных с помощью вспомогательной функции `find_max_react`. Далее определяется площадь найденного прямоугольника и если она оказывается больше площади максимальной – сохраняются координаты углов прямоугольника и максимальная площадь и приравнивается к найденной. Далее программа проходит по пикселям наибольшего прямоугольника в заданном изображении и меняем их цвет на новый. Функция возвращает измененное изображение.


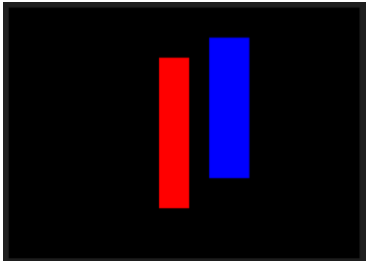
См. приложение А.

## **Тестирование**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования



п/п	Входные данные	Выходные данные	Комментарии
	<pre>image=Image.new("RGB", (500, 500), (0,0,0))  img=user_func(image, 400, 400, 200, 200, (255,0,0,0), 3)  Image._show(img)</pre>		№1
	<pre>set_black_white(Image.open("input.png"), 250, 200, 700, 500)</pre>		№2
	<pre>image = Image.new("RGB", (350, 250), 'black')  image.paste(Image</pre>		№3



<pre>e.new("RGB", (30,150), 'red'), (150, 50))      image.paste(Image e.new("RGB", (40,140), 'red'), (200, 30))      find_rect_and_re color(image, (255, 0 , 0), (0, 0, 255))</pre>		
---	--	--

## **Выводы**

Изучена библиотека Pillow, решены 3 подзадачи с использованием библиотеки Pillow (PIL).

Реализована программа, состоящая из трех задач, под каждую из которых выделена отдельная функция.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import PIL
from PIL import Image, ImageDraw

def user_func(image, x0, y0, x1, y1, fill, width):
    drawing = ImageDraw.Draw(image)
    drawing.line((x0,y0,x1,y1),fill,width)
    return image

def check_coords(image, x0, y0, x1, y1):
    width, height = image.size
    if x0 < 0 or y0 < 0 or x1 < 0 or y1 < 0:
        return False
    if x0 >= width or x1 >= width or y0 >= height or y1 >= height:
        return False
    if x1 <= x0 or y1 <= y0:
        return False
    return True

def set_black_white(image, x0, y0, x1, y1):
    if check_coords(image, x0, y0, x1, y1):
        region = image.crop((x0, y0, x1, y1))
        region = region.convert('1')
        image.paste(region, (x0, y0, x1, y1))
    return image

def find_max_rect(pix, width, height, x,y,old_color):
    min_cord = [0, 0]
    max_cord = [width, height]
    curr = [(x, y)]
    while len(curr) > 0:
        x1, y1 = curr.pop()
        if (0 <= x1 < width and 0 <= y1 < height and pix[x1, y1] ==
old_color):
            max_cord = [min(max_cord[0], x1), min(max_cord[1], y1)]
            min_cord = [max(min_cord[0], x1), max(min_cord[1], y1)]
            curr += [(x1 - 1, y1), (x1 + 1, y1), (x1, y1 - 1), (x1, y1
+ 1)]
            pix[x1, y1] = (0, 0, 0)
    return (max_cord[0], max_cord[1], min_cord[0], min_cord[1])

def find_rect_and_recolor(image, old_color, new_color):
    img=image.copy()
    pix=img.load()
    width, height=img.size[0], image.size[1]
    max_rect=0
    for x in range(width):
        for y in range(height):
            if pix[x,y]==old_color:
                cords=find_max_rect(pix, width, height, x,y,old_color)
                rect= (cords[3]+1 - cords[1])*(cords[2]+1 - cords[0])
                if rect>max_rect:
                    max_cords=cords
```

```
        max_rect=rect
res_img=image.load()
for x in range (max_cords[0], max_cords[2]+1):
    for y in range (max_cords[1], max_cords[3]+1):
        res_img[x,y]=new_color
return image
```