

**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**

Студент гр. 3343

Силяев Р.А.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2023

## **Цель работы**

Научиться использовать модуль Pillow для работы с изображениями на языке Python.

## Задание

Предстоит решить 3 подзадачи, используя библиотеку **Pillow (PIL)**. Для реализации требуемых функций студент должен 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 (`x0, y0`);
- координаты конца (`x1, y1`);
- 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 (`x0, y0, x1, y1`) 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. Основные функции:

- *user\_func(image, x0, y0, x1, y1, fill, width)*: принимает на вход: изображение; координаты начала (x0, y0); координаты конца (x1, y1); цвет; толщину. Функция рисует на изображении отрезок и возвращает обработанную версию изображения.
- *check\_coords(image, x0, y0, x1, y1)*: принимает на вход координаты области и проверяет их на корректность (они должны быть неотрицательными, не превышать размеров изображения, поскольку x0, y0 - координаты левого верхнего угла, x1, y1 - координаты правого нижнего угла, то x1 должен быть больше x0, а y1 должен быть больше y0), возвращает *True* или *False*
- *set\_black\_white(image, x0, y0, x1, y1)*: принимает на вход: изображение, координаты области, с которой необходимо произвести действия. Функция преобразовывает заданную область изображения в черно-белый. В этой функции вызывается функция проверки, и, если область некорректна, то возвращается исходное изображение без изменений.
- *find\_rect\_and\_recolor(image, old\_color, new\_color)*: принимает на вход: изображение и кортежи rgb-компонент старого и нового цветов. Функция преобразует изображение в массив *nparray*, заменяет все пиксели, не соответствующие нужному цвету на 0, а соответствующие на 1, далее пробегается по массиву и выделяет непрерывные последовательности элементов нужного цвета, после снова пробегается по массиву и находит координаты прямоугольника самого большого размера. За этим заменяет цвет пикселей найденного прямоугольника на новый и возвращает изображение.

### 2. Основные переменные:

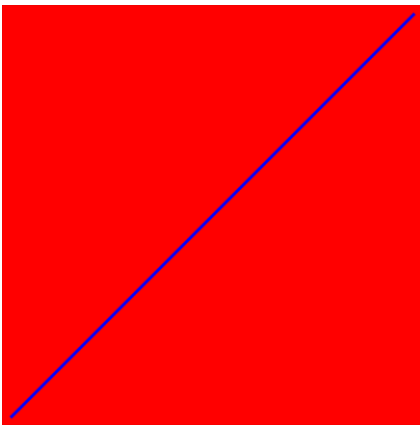
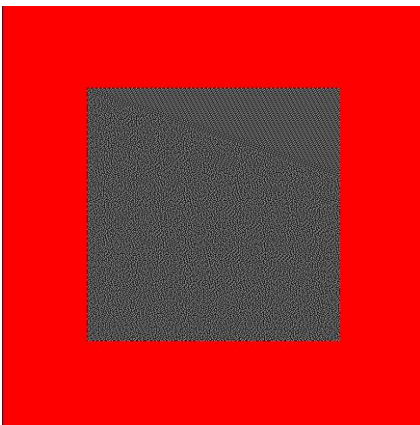
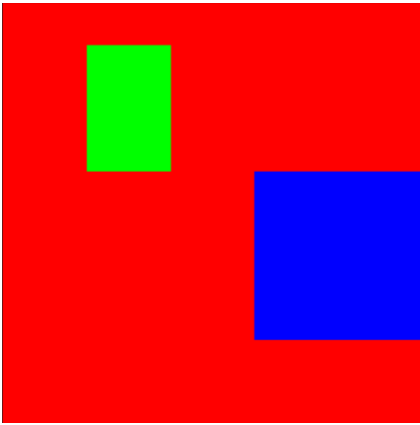
- *converted\_image* – область изображения *image*, которую необходимо преобразовать в черно-белый.
- *array* – матрица, которая содержит информацию о пикселях изображения *image*.
- *max\_size* – переменная, содержащая размер самого большого прямоугольника нужно цвета.
- *coords* - переменная, содержащая координаты самого большого прямоугольника нужно цвета.
- *curr\_size* – переменная, которая содержит информацию о размере найденного в настоящий момент времени прямоугольнике.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования содержатся в таблице 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	<code>image1 = user_func(Image.new("RGB", (500, 500), (255, 0, 0)), 10, 490, 490, 10, 'blue', 3)</code>		Тестирование функции <i>user_func()</i>
2.	<code>image1 = set_black_white(Image.new("RGB", (500, 500), (255, 0, 0)), 100, 100, 400, 400)</code>		Тестирование функции <i>check_coords()</i> и <i>set_black_white()</i>
3.	<code>image1 = Image.new("RGB", (500, 500), (255, 0, 0))</code> <code>image1.paste(Image.new("RGB", (10, 15), (0, 255, 0)), (10, 5))</code> <code>image1.paste(Image.new("RGB", (20, 20), (0, 255, 0)), (30, 20))</code>		Тестирование функции <i>find_rect_and_recolor()</i>

	<pre> image1      = find_rect_and_recol or(image1, (0,255,0),(0,0,255)) </pre>		
--	--	--	--



## **Выводы**

В результате работы были изучены методы работы с модулем *Pillow*, в частности пакеты *ImageDraw*, *Image*. Была разработана программа включающая в себя 4 функции: *user\_func(image, x0, y0, x1, y1, fill, width)*, *check\_coords(image, x0, y0, x1, y1)*, *set\_black\_white(image, x0, y0, x1, y1)*, *find\_rect\_and\_recolor(image, old\_color, new\_color)*. Принцип их работы был описан выше.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from PIL import Image, ImageDraw
import numpy as np
# Задача 1
def user_func(image, x0, y0, x1, y1, fill, width):
    draw = ImageDraw.Draw(image)
    draw.line(((x0, y0), (x1, y1)), fill, width)
    return image

# Задача 2
def check_coords(image, x0, y0, x1, y1):
    size_x = image.size[0]
    size_y = image.size[1]
    if(x0 > 0 and y0 > 0 and x1 > 0 and y1 > 0) and (x1 > x0 and
y1 > y0)
    and (x1 < size_x and y1 < size_y):
        return True
    else:
        return False
def set_black_white(image, x0, y0, x1, y1):
    if check_coords(image, x0, y0, x1, y1):
        converted_image = image.crop((x0, y0, x1, y1))
        converted_image = converted_image.convert("1")
        image.paste(converted_image, (x0, y0))
    return image

# Задача 3
def find_rect_and_recolor(image, old_color, new_color):
    color = list(old_color)
    array = np.array(image).tolist()
    for i in range(len(array)):
        for j in range(len(array[i])):
            array[i][j] = int(array[i][j] == color)
    array = np.array(array)
    for i in range(1, len(array)):
        for j in range(len(array[i])):
            if array[i][j] == 1:
                array[i][j] += array[i-1][j]
    max_size = 0
    coords = (0, 0, 0, 0)
    for i in range(len(array)):
        curr_size = 0
        last_j = 0
        for j in range(len(array[i])-1):
            curr_size += array[i][j]
            if curr_size > max_size:
                max_size = curr_size
                coords = (j - (max_size // array[i][j]) + 1, i -
array[i
[j] + 1, j, I)
            if array[i][j] != array[i][j+1]:
                curr_size = 0
                last_j = j
        if array[i][last_j] == array[i][last_j+1]:
```

```

        curr_size += array[i][last_j+1]
        if curr_size > max_size:
            max_size = curr_size
            coords = (last_j - (max_size // array[i][last_j+1])
+ 1,i
        - array[i][last_j+1] + 1, last_j+1, I)
        array = np.array(image).tolist()
        for i in range(coords[1], coords[3]+1):
            for j in range(coords[0], coords[2]+1):
                array[i][j] = new_color
        image = Image.fromarray(np.uint8(array))
        return image

```