**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**](https://e.moevm.info/mod/quiz/view.php?id=2018) **в архитектуру компьютера**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Пушко К.Д. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Целью работы является освоение работы с функциями и библиотекой Pillow.

## Задание

Вариант 4.

Задача 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.

Задача 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 x0, y0 - координаты левого верхнего угла, x1, y1 - координаты правого нижнего угла, то x1 должен быть больше x0, а y1 должен быть больше y0);
* 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.

Задача 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

Написанная программа написана на языке Python с использованием библиотеки Pillow. Она состоит из 3-функций, которые вызываются сразу на сайте https://e.moevm.info.

Функция user\_func получает на вход изображение, координаты начала (x0, y0), координаты конца (x1, y1), цвет отрезка и её толщину. Рисуется изображение при помощи метода “Image.Draw”. На созданной картинке, рисуется отрезок по таким параметрам как: координаты, цвет отрезка, его толщина.

Функция check\_coords получает на вход изображение и координаты. Данная функция, при помощи условных операторов, проверяет корректность заданных координат.

Функция set\_black\_white\_получает на вход: изображение и координаты. Функция начинается с того, что проверяет корректность координат при помощи функции check\_coord далее, из картинки вырезается обрабатываемая область c помощью метода crop, затем используя метод convert получаем Ч/Б изображение. Это изображение вставляется в изначальное место исходной картинки, а затем возвращается из функции.

Функция find\_rect\_and\_recolor получает на вход изображение, старый цвет и новый цвет. Создается массив, в котором картинка преобразовывается в двоичную матрицу, где элементы массива равные единице – это искомый цвет, а остальные равны нулю. После чего мы ищем в ней максимальный квадрат из единиц.

Функция max\_area\_histogram принимает на вход гистограммы из функции find\_rect\_and\_recolor и находит максимальную площадь из них.

Данная программа демонстрирует использование функций библиотеки Pillow и работу функций на языке Python для выполнения различных графических операций

Разработанный программный код см. в приложении А.

## Выводы

Была освоена библиотека Pillow. Полученные знания были применены на практике. Были разработаны такие функции как: рисования отрезка, преобразование изображения в черно-белый цвет, нахождения прямоугольника заданного цвета и его перекрашивания в другой цвет.

# Приложение А Исходный код программы

Название файла: main.py

import numpy as np

from PIL import Image, ImageDraw

# Задача 1

def user\_func(image, x0, y0, x1, y1, fill, width):

coordinates = (x0, y0, x1, y1)

drawing = ImageDraw.Draw(image)

drawing.line(coordinates, fill, width)

return image

# Задача 2

def check\_coords(image, x0, y0, x1, y1):

x, y = image.size

return (x >= x1) and (x1 > x0) and (x0 >= 0) and (y >= y1) and (y1 > y0) and (y0 >= 0)

def set\_black\_white(image, x0, y0, x1, y1):

if check\_coords(image, x0, y0, x1, y1):

crop\_img = image.crop((x0, y0, x1, y1))

crop\_img = crop\_img.convert("1")

image.paste(crop\_img, (x0, y0))

return image

# Задача 3

def largestRectangleArea(heights):

n, heights, st, ans = len(heights), [0] + heights + [0], [], 0

for i in range(n + 2):

while st and heights[st[-1]] > heights[i]:

ans = max(ans, heights[st.pop(-1)] \* (i - st[-1] - 1))

st.append(i)

return ans

def check\_sqr(x, pixel, sqr, max\_sqr, n, ans, coordinates):

for y in range(len(pixel[x])):

if n <= pixel[x][y]:

sqr += n

if y == len(pixel[x]) - 1 or pixel[x][y + 1] < n:

if max\_sqr < sqr:

max\_sqr = sqr

coordinates = (y - max\_sqr // n + 1, x - n + 1, y, x)

if max\_sqr == ans:

return True, max\_sqr, sqr, coordinates

sqr = 0

return False, max\_sqr, sqr, coordinates

def find\_big\_rect(image, old\_color):

pixel = np.array(image).tolist()

for x in range(len(pixel)):

for y in range(len(pixel[x])):

pixel[x][y] = 1 if pixel[x][y] == list(old\_color) else 0

pixel = np.array(pixel)

heights = [0] \* len(pixel[0])

ans = 0

for x in range(len(pixel)):

for y in range(len(pixel[x])):

if pixel[x][y] == 0:

heights[y] = 0

else:

heights[y] += 1

ans = max(ans, largestRectangleArea(heights))

for x in range(1, len(pixel)):

for y in range(len(pixel[x])):

if pixel[x][y] == 0:

pixel[x][y] = 0

else:

pixel[x][y] += pixel[x - 1][y]

max\_sqr = 0

coordinates = (0, 0, 0, 0)

for x in range(len(pixel)):

sqr = 0

for n in set(pixel[x]):

fl, max\_sqr, sqr, coordinates = check\_sqr(x, pixel, sqr, max\_sqr, n, ans, coordinates)

if fl:

return coordinates

else:

continue

return coordinates

def find\_rect\_and\_recolor(image, old\_color, new\_color):

coordinates = find\_big\_rect(image, old\_color)

res = np.array(image)

res[coordinates[1]:coordinates[3] + 1, coordinates[0]:coordinates[2] + 1, :3] = new\_color

image = Image.fromarray(res)

return image