**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. 3344 |  | Волохов М. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Освоение работы с библиотеками в языке программирования Python.

## Задание.

Вариант 4. 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 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.

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

Были импортированы библиотеки *numpy, pillow*.

Функция user\_func, рисует линию на переданном в функцию изображении и возвращает измененное изображение.

Функции set\_black\_white() и check\_coords. Функция set\_black\_white() использует функцию check\_coords, чтобы убедиться, что указанные координаты (x0, y0, x1, y1) являются допустимыми. Если координаты корректны, функция извлекает указанную область изображения, преобразует ее в черно-белый формат, а затем вставляет черно-белую область обратно в исходное изображение.

Функция find\_rect\_and\_recolor(), динамически ищет наибольшую прямоугольную матрицу в массиве цветов изображения, а затем в цикле поэлементно заменяет цвета найденной матрицы. Функция возвращает измененное изображение.

алгоритм для поиска самого большого прямоугольника в матрице. Он использует переменные, такие как x1, x2, y1, y2 для хранения координат этого прямоугольника, area для отслеживания его площади, и массив height для хранения высоты столбцов.

В цикле по строкам изображения, алгоритм обновляет массив height, увеличивая его значения на 1, если текущий пиксель соответствует цвету old\_color, или обнуляя его, если нет. Затем используется стек для эффективного определения самого большого прямоугольника. Если высота текущего столбца меньше предыдущего, из стека извлекается предыдущий столбец, и проверяется, образует ли он с текущим столбцом прямоугольник большей площади. Если да, обновляются координаты и площадь.

Эти манипуляции повторяются, пока не пройдена вся матрица. В конце концов, функция изменяет цвет пикселей в найденной области на new\_color и возвращает измененное изображение.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | user\_func(Image.new('RGB', (300, 300), (255, 0,0 )), 100, 20, 40, 100, 'green', 10) | correct image | - |
|  | set\_black\_white(Image.new('RGB', (300, 300), (255, 0, 0)), 100, 50, 200, 100) | correct image | - |
|  | find\_rect\_and\_recolor(Image.new('RGB', (300, 300), (255, 0, 0)), (255, 0, 0), (0, 0, 255)) | correct image | - |

## Выводы

Была освоена работа с библиотеками в языке Python. Были получены навыки работы с библиотекой Pillow. Были освоены функции преобразования изображений.

# Приложение А Исходный код программы

Название файла: Volokhov\_Mikhail\_lb2.py

# 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):

if x0 >= 0 and y0 >= 0 and x1 <= image.width and y1 <= image.height and x1 >= x0 and y1 >= y0:

return True

return False

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

if not check\_coords(image, x0, y0, x1, y1):

return image

# Преобразвание изображения в ч/б

region = image.crop((x0, y0, x1, y1))

region = region.convert('1')

image.paste(region, (x0, y0, x1, y1))

return image

# Задача 3

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

img\_width, img\_height = image.size

img\_data = list(image.getdata())

img\_matrix = [img\_data[i:i+img\_width] for i in range(0, len(img\_data), img\_width)]

x1 = y1 = x2 = y2 = 0

n = img\_width

height = [0] \* (n + 1)

area = 0

row\_count = -1

for row in img\_matrix:

row\_count += 1

for i in range(n):

height[i] = height[i] + 1 if row[i] == old\_color else 0

stack = [-1]

for i in range(n + 1):

while height[i] < height[stack[-1]]:

h = height[stack.pop()]

w = i - 1 - stack[-1]

if area < h \* w:

x2 = i - 1

y2 = row\_count

x1 = stack[-1] + 1

y1 = row\_count - h + 1

area = max(area, h \* w)

stack.append(i)

for i in range(y1, y2 + 1):

for j in range(x1, x2 + 1):

img\_matrix[i][j] = new\_color

img\_data = [pixel for row in img\_matrix for pixel in row]

image.putdata(img\_data)

return image