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. 3344 _____ Якимова Ю.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Освоение обработки изображений на языке Python.

Задание.

Вариант 2

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 пентаграммы в круге

Необходимо написать функцию *pentagram*(), которая рисует на изображении пентаграмму в круге.

Функция *pentagram*() принимает на вход:

Изображение (*img*)

координаты левого верхнего и правого нижнего угла квадрата, в который вписана окружность (*x0,y0,x1,y1*)

Толщину линий и окружности (*thickness*)

Цвет линий и окружности (*color*) - представляет собой список (*list*) из 3-х целых чисел

Функция должна вернуть обработанное изображение.

Примечание:

Вершины пентаграммы высчитывать по формуле:

phi = (pi/5)(2*i+3/2)*

*node_i = (int(x0+r*cos(phi)),int(y0+r*sin(phi)))*

x0,y0 - координаты центра окружности, в который вписана пентаграмма

r - радиус окружности

i - номер вершины от 0 до 4

Подсказка: Округляйте все вычисляемые вами значения (кроме значений углов) до целых чисел.

2) Инвертирование полос

Необходимо реализовать функцию *invert*, которая делит изображение на "полосы" и инвертирует цвет нечетных полос (счёт с нуля).

Функция *invert*() принимает на вход:

Изображение (*img*)

Ширину полос в пикселах (*N*)

Признак того, вертикальные или горизонтальные полосы(*vertical* - если *True*, то вертикальные)

Функция должна разделить изображение на вертикальные или горизонтальные полосы шириной *N* пикселей. И инвертировать цвет в нечетных полосах (счет с нуля). Последняя полоса может быть меньшей ширины, чем *N*.

3) Поменять местами 9 частей изображения

Необходимо реализовать функцию *mix*, которая делит квадратное изображение на 9 равных частей (сторона изображения делится на 3), и по правилам, записанным в словаре, меняет их местами.

Функция *mix*() принимает на вход:

Изображение (*img*)

Словарь с описанием того, какие части на какие менять (*rules*)

Пример словаря *rules*:

{0:1,1:2,2:4,3:4,4:5,5:3,6:8,7:8,8:8}

Элементы нумеруются слева-направо, сверху-вниз.

В данном случае нулевой элемент заменяется на первый, первый на второй, второй на четвертый, третий на четвертый и так далее.

Функция должна вернуть обработанное изображение.

Выполнение работы

Перед началом работы были импортированы библиотеки *PIL*, *numpy.*

Далее было реализовано 3 функции:

Функция *def pentagram(img, x0, y0, x1, y1, thickness, color)*, принимающая на вход изображение - объект типа *<class 'PIL.Image.Image'>,* координаты левого верхнего и правого нижнего угла квадрата, в который вписана окружность, толщину линий и окружности, цвет линий и окружности, рисующая пентаграмму. Для отрисовки окружности на изображении был вызван *ImageDraw.Draw(img)*, у которого, в свою очередь, был вызван метод *ellipse.* Далее были рассчитаны координаты центра окружности и ее радиус. Потом с помощью цикла *for* были получены значения координат вершин пентаграммы и занесены в массив *coords.* Для отрисовки линий пентаграммы использовался цикл *for* и метод *line* у *ImageDraw.Draw(img)*. Функция возвращает изображение с пентаграммой.

Функция *def invert(img, N, vertical)*, принимающая на вход изображение, ширину полос для инвертирования в пикселях, признак расположения полос. В ней было реализовано инвертирование цвета всех нечетных полос. Были получены ширина и высота изображения (*width* и *height)*. Далее, если признак расположения полос являлся *True*, то выполнялся цикл *for*, который проходил по всем четным индексам частей ширины. В каждой итерации цикла создавался кортеж координат для части, которую надо инвертировать. С помощью метода *invert* у *ImageChops*, который принимал часть исходного изображения, создавалась инвертированная часть *part=ImageChops.invert(img.crop(box)).* Далее методом *paste* в исходное изображение вставлялась его инвертированная часть. img.paste(part, box). Аналогично, если признак расположения полос не являлся *True*. Функция возвращает отредактированное изображение.

Функция *def mix(img, rules)*, принимающая на вход изображение, словарь с описанием того, какие части на какие менять. Была инициализирована переменная *sp*, которая была равна ширине пикселей одной части *sp = img.width*

// 3. Был создан массив кортежей вида *((x0, y0, x1, y1), <PIL.Image.Image>)*, где *(x0, y0, x1, y1)* – кортеж координат одной части, *<PIL.Image.Image>* - сама часть исходного изображения. Это массив был сформирован с помощью функции *map*, первым аргументом являлась лямбда функция возвращающая кортеж координат и части, вторым аргументом являлся список кортежей координат всех частей в нужном порядке. *parts = [*map(lambda box: (box, img.crop(box)), [((w - 1) * sp, (h - 1) * sp, w * sp, h * sp) for h in range(1, 4) for w in range(1, 4)])]*. Далее с помощью цикла *for*, проходящим по парам ключ-значение словаря с правилами, были заменены старые части на новые. Функция возвращает отредактированное изображение.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	pentagram(Image.new("RGB", (300, 300)), 67, 82, 139, 154, 4, [197, 114, 130])	img	-
2.	invert(Image.new("RGB", (400, 400), "black"), 75, False)	img	-
3.	mix(Image.open("krab1.jpeg"), {0:2,1:2,2:2,3:5,4:5,5:5,6:8,7:8,8:8})	img	-

Выводы

Была освоена обработка изображений на языке Python. Были получены базовые навыки работы с пакетом *Pillow*. Были освоены функции рисования фигур и линий.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Yakimova_Yuliya_lb2.py

```
import numpy as np
from PIL import ImageDraw, ImageChops


def pentagram(img, x0, y0, x1, y1, thickness, color):
    color = tuple(color)
    draw = ImageDraw.Draw(img)
    draw.ellipse(((x0, y0), (x1, y1)), outline=color, width=thickness)

    r = (x1 - x0) // 2
    x = (x1 + x0) // 2
    y = (y1 + y0) // 2
    coords = []

    for i in range(5):
        phi = (np.pi / 5) * (2 * i + 3 / 2)
        node_i = (int(x + r * np.cos(phi)), int(y + r * np.sin(phi)))
        coords.append(node_i)

    for i in range(2):
        xy = (coords[i - 2], coords[i], coords[i + 2])
        draw.line(xy, fill=color, width=thickness)
    draw.line((coords[2], coords[-1]), fill=color, width=thickness)

    return img


def invert(img, N, vertical):
    width = img.width
    height = img.height

    if vertical:
        for i in range(2, width // N + (width % N != 0) + 1, 2):
            box = ((i - 1) * N, 0, i * N, height)
            part = ImageChops.invert(img.crop(box))
            img.paste(part, box)
    else:
        for i in range(2, height // N + (height % N != 0) + 1, 2):
            box = (0, (i - 1) * N, width, i * N)
            part = ImageChops.invert(img.crop(box))
            img.paste(part, box)

    return img


def mix(img, rules):
    sp = img.width // 3

    parts = [*map(lambda box: (box, img.crop(box)),
```

```
                    [((w - 1) * sp, (h - 1) * sp, w * sp, h * sp) for h in
range(1, 4) for w in range(1, 4)])]

    for old, new in rules.items():
        img.paste(parts[new][1], parts[old][0])

    return img
```