**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. 3343 |  | Добрякова А.А. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2023

## Цель работы

Научится работать с модулем *Pillow (PIL)*, а также с функциями *numpy*, выполнять различные графические преобразования над изображениями.

## 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 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 () принимает на вход:

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

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.

При необходимости можно писать дополнительные функции.

## Выполнение работы

Функция *pentagram()* рисует на полученном изображении img пентаграмму, используя функции и методы библиотек *pillow* и *numpy*. Координаты круга определяются по координатам описанного вокруг неё квадрата (*x0, y0* — левый верхний угол, *x1, y1* — правый нижний угол). Пятиконечная звезда рисуется пятью отдельными линиями, координаты для обоих концов которых высчитываются относительно координат центра круга и длины его радиуса.

Функция *invert()* инвертирует цвет нечётных полос ширины *N* изначального изображения и возвращает изменённое изображение img. В зависимости от значения передаваемой фунции перменной *vertical* полосы могут быть как вертикальными, так и горизонтальными. Высчитывается максимальный размер изображения по обеим осям. Цикл *for* проходит по всему изображению и поочерёдно копирует полосы исходного изображения нужно размера, инвертирует их (используется *ImageOps.invert()*) и вставляет в исходное изображение на прежнее место.

Функция *mix()* разделяет изображение на 9 прямоугольников и перемешивает их согласно инструкции, указанной в переданном ей словаре *rules*. Возвращает изменённое изображение. Определяются длина и ширина одного прямоугольника. Создаётся список, элементами которого являются вырезанные из основного прямоугольные изображения одинакового размера (нумерация идёт слева-направо сверху-вниз). Благодаря циклу *for* программа просматривает каждый из изначальных кусков изображения, и на их место вставляет нужные изображения из списка согласно словарю.

## Выводы

Были изучены различные способы преобразования изображения, написаны функции с использованием библиотек *numpy* и *pillow*.

Программа может получать на вход исходное изображать и возвращать изменённое, в зависимости от выбранной функции.

# Приложение А Исходный код программы

Название файла: main.py

import PIL

import numpy as np

from PIL import Image, ImageDraw, ImageOps

def pentagram(img, x0, y0, x1, y1, thickness, color):

drawing = ImageDraw.Draw(img)

color = tuple(color)

drawing.ellipse((x0, y0, x1, y1), None, color, thickness)

points = []

r = int((x1 - x0) / 2)

x = x0 + r

y = y0 + r

drawing\_order = [0, 2, 4, 1, 3, 0]

for i in drawing\_order:

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

points.append((int(x + r \* np.cos(phi)), int(y + r \* np.sin(phi))))

points = tuple(points)

drawing.line(points, color, thickness)

return img

def invert(img, N, vertical):

end\_x, end\_y = img.size

if vertical:

for x in range(N, end\_x, 2 \* N):

strip = img.crop((x, 0, x + N, end\_y))

strip = ImageOps.invert(strip)

img.paste(strip, (x, 0))

else:

for y in range(N, end\_y, 2 \* N):

strip = img.crop((0, y, end\_x, y + N))

strip = ImageOps.invert(strip)

img.paste(strip, (0, y))

return img

def mix(img, rules):

size = img.size

block\_x = size[0] // 3

block\_y = size[1] // 3

crooped\_images = []

for i in range(3):

for j in range(3):

x = block\_x \* j

y = block\_y \* i

img2 = img.crop((x, y, x + block\_x, y + block\_y))

crooped\_images.append(img2)

for i in range(9):

crooped\_img = crooped\_images[rules[i]]

x = i % 3 \* block\_x

y = i // 3 \* block\_y

img.paste(crooped\_img, (x, y))

return img

# Приложение Б тЕСТИРОВАНИЕ

Исходные изображения:



Рисунок 1 – Изображение для функции *pentagram*



Рисунок 2 – Изображение для функции *invert*

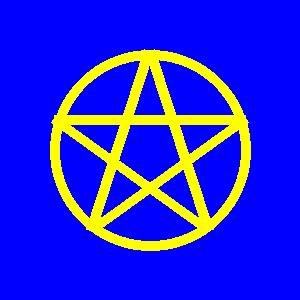


Рисунок 3 – Изображение для функции *mix*

Параметры функций:

1) Для рисунка 1 функция

*pentagram(img, 50, 50, 250, 250, 10, [255, 255, 0])*

2) Для рисунка 2 функция

*invert(img, 45, False)*

3)Для рисунка 3 функция

*mix(img, {0:2,1:2,2:2,3:5,4:5,5:5,6:8,7:8,8:8})*

Результат:

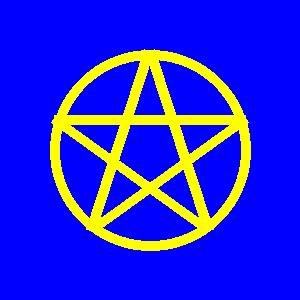


Рисунок 1 – Результат функции *pentagram*

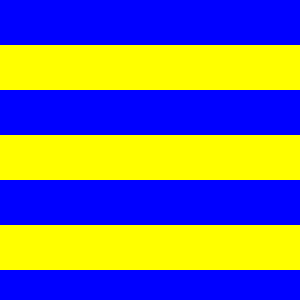


Рисунок 2 – Результат функции *invert*

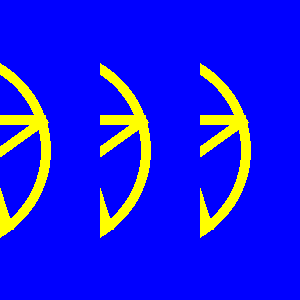


Рисунок 3 – Результат функции *mix*