**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. 3342 |  | Корниенко А.Е. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2023

## Цель работы

Научиться применять библиотеку Pillow языка программирования Python. Основной задачей лабораторной работы является реализация 3-ёх функций согласно заданию лабораторной работы, используя функции библиотеки.

## Задание

1.

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 кортеж.

Функция должна вернуть обработанное изображение.

2.

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.

3.

Функция mix делит изображение на 9 равных частей и меняет их местами.

Функция mix() принимает на вход:

* Изображение (img).
* Словарь с описанием какие части менять местами (rules).

## Выполнение работы

Для работы с изображениями и математических расчетов размеров фигур были использованы библиотека *Pillow.Ф*ункции:

1. *def pentagram(img, x0, y0, x1, y1, thickness, color)*: Рисует пентаграмм в круге. Сначала рисуем круг при помощи функции ellipse, а затем вычисляем координаты точек и соединяет их, которые находятся на круге.
2. *def invert(img, N, vertical): с помощью циклов пробегаем по изображению и инвертируем нечётные полосы при помощи функции invert.*
3. *def mix(img, rules):* с помощью функции crop вырезаем части изображения и создаём новое изображение.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
| 1. | def pentagram(img, 60, 110, 110, 160, 4, [35, 60, 80]): | Верно обработанное изображение |
| 2. | def invert(img, 9, True) | Верно обработанное изображение |
| 3. | def mix(img, {0:1,1:2,2:4,3:4,4:5,5:3,6:8,7:8,8:8}): | Верно обработанное изображение |

## Выводы

Была изучена технология Pillow, написаны функции для обработки изображений.

# Приложение А Исходный код программы

Название файла: main.py

import PIL

from PIL import Image, ImageDraw, ImageOps

import math

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

center\_x = x1 - (abs(x1 - x0) // 2)

center\_y = y1 - (abs(y1 - y0) // 2)

radius = abs(x1 - x0) // 2

draw = ImageDraw.Draw(img)

draw.ellipse([(center\_x - radius, center\_y - radius), (center\_x + radius, center\_y + radius)], outline=tuple(color),

width=thickness)

node\_coordinates = []

for i in range(5):

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

x = int(center\_x + radius \* math.cos(phi))

y = int(center\_y + radius \* math.sin(phi))

node\_coordinates.append((x, y))

for i in range(5):

draw.line([node\_coordinates[i], node\_coordinates[(i + 2) % 5]], fill=tuple(color), width=thickness)

return img

def invert(img, N, vertical):

length, height = img.size

parity = 0

if vertical:

for x in range(0, length, N):

if parity % 2 == 1:

img\_2 = img.crop((x, 0, x + N, height))

invert\_img = ImageOps.invert(img\_2)

img.paste(invert\_img, (x, 0))

parity += 1

else:

for y in range(0, height, N):

if parity % 2 == 1:

img\_2 = img.crop((0, y, length, y + N))

invert\_img = ImageOps.invert(img\_2)

img.paste(invert\_img, (0, y))

parity += 1

return img

def invert(img, N, vertical):

w, h = img.size

k = 0

if vertical:

for x in range(0, w, N):

if k % 2 == 1:

part = img.crop((x, 0, x + N, h))

inv\_part = ImageOps.invert(part)

img.paste(inv\_part, (x, 0))

k += 1

else:

for y in range(0, h, N):

if k % 2 == 1:

part = img.crop((0, y, w, y + N))

inv\_part = ImageOps.invert(part)

img.paste(inv\_part, (0, y))

k += 1

return img

def mix(img, rules):

width, height = img.size

part\_width = width // 3

part\_height = height // 3

result = Image.new('RGB', (width, height))

for i in range(3):

for j in range(3):

source\_index = i \* 3 + j

target\_index = rules[source\_index]

source\_x = (source\_index % 3) \* part\_width

source\_y = (source\_index // 3) \* part\_height

target\_x = (target\_index % 3) \* part\_width

target\_y = (target\_index // 3) \* part\_height

source\_part = img.crop((target\_x, target\_y, target\_x + part\_width, target\_y + part\_height))

result.paste(source\_part, (source\_x, source\_y))

return result