**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. 3341 |  | Пчелкин Н.И. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Ознакомиться с функционалом библиотеки Pillow и решить 3 подзадачи с использованием её возможностей и библиотеки Numpy.

## Задание

Вариант 3

1) Рисование пентаграммы в круге

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

*Функция pentagram() принимает на вход:*

* *Изображение (img)*
* *координаты центра окружность (x,y)*
* *радиус окружности*
* *Толщину линий и окружности (thickness)*
* *Цвет линий и окружности (color) - представляет собой список (list) из 3-х целых чисел*

*Функция должна изменить исходное изображение и вернуть его изображение.*

2) Поменять местами участки изображения и поворот

*Необходимо реализовать функцию swap(), которая меняет местами два квадратных, одинаковых по размеру,  участка изображений и поворачивает эти участки на 90 градусов по часовой стрелке, а затем поворачивает изображение на 90 градусов по часовой стрелке.*

*Функция swap() принимает на вход:*

* *Квадратное изображение (img)*
* *Координаты левого верхнего угла первого квадратного участка(x0,y0)*
* *Координаты левого верхнего угла второго квадратного участка(x1,y1)*
* *Длину стороны квадратных участков (width)*

*Функция должна сначала поменять местами переданные участки изображений. Затем повернуть каждый участок на 90 градусов по часовой стрелке. Затем повернуть всё изображение на 90 градусов по часовой стрелке.*

*Функция должна вернуть обработанное изображение, не изменяя исходное.*

3) Средний цвет

*Необходимо реализовать функцию avg\_color(), которая заменяет цвет каждого пикселя в области на средний цвет пикселей вокруг (не считая сам этот пиксель).*

*Функция avg\_color() принимает на вход:*

* *Изображение (img)*
* *Координаты левого верхнего угла области (x0,y0)*
* *Координаты правого нижнего угла области (x1,y1)*

*Функция должна заменить цвета каждого пикселя в этой области на средний цвет пикселей вокруг.*

*Пиксели вокруг:*

* *8 самых близких пикселей, если пиксель находится в центре изображения*
* *5 самых близких пикселей, если пиксель находится у стенки*
* *3 самых близких пикселя, если пиксель находится в угле*

*Функция должна вернуть обработанное изображение, не изменяя исходное.*

## Основные теоретические положения

*Pillow* - это форк библиотеки *PIL* *(Python Imaging Library*), предназначенной для работы с изображениями в Python. *Pillow* предоставляет средства для открытия, редактирования и сохранения различных форматов изображений.

*Image* - это модуль, предоставляющий класс с тем же именем (*Image*), который используется для представления изображений в *Pillow*. Этот класс обеспечивает широкий спектр методов для работы с изображениями, включая открытие изображений из файлов, создание новых изображений и выполнение различных операций редактирования.

*ImageDraw* - это модуль, предоставляющий класс с тем же именем (*ImageDraw*), который позволяет рисовать на изображении. Этот класс содержит методы для рисования геометрических фигур, текста и других элементов на изображении.

## Выполнение работы

Функции:

* def pentagram(img, x, y, r, thickness, color);

Функция рисует пентаграмму. На вход подаются изображение, координаты окружности, в которую вписана пентаграмма, а также толщина и цвет линий. Функция рисует на изображении окружность, а затем по вычисленным вершинам звезду внутри этой окружности.

* def swap(img, x0, y0, x1, y1, width);

Функция создает копию изображения, в котором она меняет местами кусочки изображения и поворачивает их на 90 градусов по часовой стрелке, а затем и само полученное изображение поворачивается на тот же угол. На вход подаются изображение, координаты левого верхнего угла первого и второго квадратный участков, а также сторону квадрата.

* def avg\_color(img, x0, y0, x1, y1);

Функция меняет в заданной области цвет каждого пикселя на средний из цветов пикселей вокруг (делает размытие изображения в области). На вход подаются изображение, координаты левого верхнего и правого нижнего угла области. С помощью модуля numpy функция аккумулирует в матрице цвета, а затем делит их на количество близлежащих пикселей, чтобы получить среднее арфиметическое.

Разработанный программный код см. в приложении А.

## Выводы

В результате выполнения работы были освоены основные возможности библиотеки Pillow, а также была написана программа, использующая библиотеки Pillow и Numpy и реализующая 3 подзадачи по обработке изображений и выполнению операций с ними.

# Приложение А Исходный код программы

Название файла: main.py

import numpy as np

from PIL import Image, ImageDraw

def swap(img, x0, y0, x1, y1, width):

result\_img = img.copy()

result\_img.paste(img.crop((x0, y0, x0+width, y0+width)).rotate(270), (x1, y1))

result\_img.paste(img.crop((x1, y1, x1 + width, y1 + width)).rotate(270), (x0, y0))

return result\_img.rotate(270)

def avg\_color(img, x0, y0, x1, y1):

result\_img = img.copy()

pixels = img.load()

pixels\_new = result\_img.load()

width, height = img.size

x0\_isborder = x0 == 0

x1\_isborder = x1 == height-1

y0\_isborder = y0 == 0

y1\_isborder = y1 == width-1

if x0\_isborder and x1\_isborder:

if y0\_isborder:

pixels\_new[x0, y0] = tuple((np.array(pixels[x0 + 1, y0]) + np.array(pixels[x0, y0 + 1]) + np.array(pixels[x0 + 1, y0 + 1])) // 3)

pixels\_new[x1, y0] = tuple((np.array(pixels[x1-1, y0]) + np.array(pixels[x1, y0 + 1]) + np.array(pixels[x1-1, y0 + 1])) // 3)

if y1\_isborder:

pixels\_new[x0, y1] = tuple((np.array(pixels[x0+1, y1]) + np.array(pixels[x0, y1-1]) + np.array(pixels[x0+1, y1-1])) // 3)

pixels\_new[x1, y1] = tuple((np.array(pixels[x1-1, y1]) + np.array(pixels[x1, y1-1]) + np.array(pixels[x1-1, y1-1])) // 3)

y0 += int(y0\_isborder)

y1 -= int(y1\_isborder)

for y in range(y0, y1+1):

result1 = np.array(pixels[x0, y + 1]) + np.array(pixels[x0, y - 1])

result2 = np.array(pixels[x1, y + 1]) + np.array(pixels[x1, y - 1])

for i in range(3):

result1 += np.array(pixels[x0 + 1, y + 1 - i])

result2 += np.array(pixels[x1 - 1, y + 1 - i])

result1 //= 5

result2 //= 5

pixels\_new[x0, y] = tuple(result1)

pixels\_new[x1, y] = tuple(result2)

y0 -= int(y0\_isborder)

y1 += int(y1\_isborder)

elif x0\_isborder:

if y0\_isborder:

pixels\_new[x0, y0] = tuple((np.array(pixels[x0 + 1, y0]) + np.array(pixels[x0, y0 + 1]) + np.array(pixels[x0 + 1, y0 + 1])) // 3)

pixels\_new[x1, y0] = tuple((np.array(pixels[x1 - 1, y0]) + np.array(pixels[x1, y0 + 1]) + np.array(pixels[x1 - 1, y0 + 1])) // 3)

if y1\_isborder:

pixels\_new[x0, y1] = tuple((np.array(pixels[x0 + 1, y1]) + np.array(pixels[x0, y1 - 1]) + np.array(pixels[x0 + 1, y1 - 1])) // 3)

pixels\_new[x1, y1] = tuple((np.array(pixels[x1 - 1, y1]) + np.array(pixels[x1, y1 - 1]) + np.array(pixels[x1 - 1, y1 - 1])) // 3)

y0 += int(y0\_isborder)

y1 -= int(y1\_isborder)

for y in range(y0, y1+1):

result = np.array(pixels[x0, y + 1]) + np.array(pixels[x0, y - 1])

for i in range(3):

result += np.array(pixels[x0 + 1, y + 1 - i])

result //= 5

pixels\_new[x0, y] = tuple(result)

y0 -= int(y0\_isborder)

y1 += int(y1\_isborder)

elif x1\_isborder:

if y0\_isborder:

pixels\_new[x0, y0] = tuple((np.array(pixels[x0 + 1, y0]) + np.array(pixels[x0, y0 + 1]) + np.array(pixels[x0 + 1, y0 + 1])) // 3)

pixels\_new[x1, y0] = tuple((np.array(pixels[x1 - 1, y0]) + np.array(pixels[x1, y0 + 1]) + np.array(pixels[x1 - 1, y0 + 1])) // 3)

if y1\_isborder:

pixels\_new[x0, y1] = tuple((np.array(pixels[x0 + 1, y1]) + np.array(pixels[x0, y1 - 1]) + np.array(pixels[x0 + 1, y1 - 1])) // 3)

pixels\_new[x1, y1] = tuple((np.array(pixels[x1 - 1, y1]) + np.array(pixels[x1, y1 - 1]) + np.array(pixels[x1 - 1, y1 - 1])) // 3)

y0 += int(y0\_isborder)

y1 -= int(y1\_isborder)

for y in range(y0, y1+1):

result = np.array(pixels[x1, y + 1]) + np.array(pixels[x1, y - 1])

for i in range(3):

result += np.array(pixels[x1 - 1, y + 1 - i])

result //= 5

pixels\_new[x1, y] = tuple(result)

y0 -= int(y0\_isborder)

y1 += int(y1\_isborder)

if y0\_isborder and y1\_isborder:

x0 += int(x0\_isborder)

x1 -= int(x1\_isborder)

for x in range(x0, x1+1):

result1 = np.array(pixels[x+1, y0]) + np.array(pixels[x-1, y0])

result2 = np.array(pixels[x+1, y1]) + np.array(pixels[x-1, y1])

for i in range(3):

result1 += np.array(pixels[x+1-i, y0+1])

result2 += np.array(pixels[x+1-i, y1-1])

result1 //= 5

result2 //= 5

pixels\_new[x, y0] = tuple(result1)

pixels\_new[x, y1] = tuple(result2)

x0 -= int(x0\_isborder)

x1 += int(x1\_isborder)

elif y0\_isborder:

x0 += int(x0\_isborder)

x1 -= int(x1\_isborder)

for x in range(x0, x1+1):

result = np.array(pixels[x + 1, y0]) + np.array(pixels[x - 1, y0])

for i in range(3):

result += np.array(pixels[x + 1 - i, y0 + 1])

result //= 5

pixels\_new[x, y0] = tuple(result)

x0 -= int(x0\_isborder)

x1 += int(x1\_isborder)

elif y1\_isborder:

x0 += int(x0\_isborder)

x1 -= int(x1\_isborder)

for x in range(x0, x1+1):

result = np.array(pixels[x + 1, y1]) + np.array(pixels[x - 1, y1])

for i in range(3):

result += np.array(pixels[x + 1 - i, y1 - 1])

result //= 5

pixels\_new[x, y1] = tuple(result)

x0 -= int(x0\_isborder)

x1 += int(x1\_isborder)

x0 += int(x0\_isborder)

x1 -= int(x1\_isborder)

y0 += int(y0\_isborder)

y1 -= int(y1\_isborder)

for x in range(x0, x1+1):

for y in range(y0, y1+1):

result = np.array(pixels[x+1, y]) + np.array(pixels[x-1, y])

for i in range(3):

result += np.array(pixels[x+1-i, y+1]) + np.array(pixels[x+1-i, y-1])

result //= 8

pixels\_new[x, y] = tuple(result)

return result\_img

def pentagram(img, x, y, r, thickness, color):

color = tuple(color)

coordinates = []

for i in range(1, 7):

phi = (np.pi/5)\*(2\*((1+i\*3)%5)+3/2)

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

draw = ImageDraw.Draw(img)

draw.ellipse(((x-r, y-r, x+r, y+r)), outline=color, width=thickness)

draw.line(coordinates, fill=color, width=thickness)

return img