**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 |  | Мильхерт А.С. |
| 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

Из библиотеки *PIL* импортируются модули *Image* и *ImageDraw*, также подключается библиотека *numpy*.

Далее описываются следующие функции:

*def swap(img, x0 ,y0 ,x1, y1, width).* Функция для решения второй подзадачи. Функция swap предназначена для обмена двух прямоугольных областей на изображении. Области задаются координатами и шириной. Её входные и выходные данные, как и входные и выходные данные всех функций для решения подзадач, совпадают с перечисленными в задании. - x0, y0: Координаты левого верхнего угла первой области. Переменные x1, y1: координаты левого верхнего угла второй области, width - ширина и высота прямоугольных областей. В функции мы вычисляем координаты границ областей, вырезаем и поворачиваем первую и вторую области на 270 градусов, создаем копии исходного изображения, вставляем повернутые области в новое изображение и поворачиваем изображение на 270.

*def get\_pixel\_color(img, x, y).* Вспомогательная функция для третьей подзадачи. Функция get\_pixel\_color вычисляет средний цвет окружности вокруг заданного пикселя, исключая сам пиксель. Получаем размеры изображения width и height. Загружаем пиксели изображения в переменную pixels. Создаем пустой список surrounding\_pixels для хранения цветов окружающих пикселей. Используем два вложенных цикла для прохода по 3x3 окрестности текущего пикселя. Добавляем цвет каждого пикселя из окрестности в список surrounding\_pixels, если пиксель находится в пределах границ изображения. Удаляем цвет текущего пикселя из списка (центральный пиксель). Создаем список result\_color для хранения суммированных значений цветов пикселей. Используем вложенные циклы для суммирования значений RGB цветов из surrounding\_pixels. Вычисляем средний цвет, разделив каждую компоненту на количество пикселей в окрестности. Возвращаем получившийся средний цвет в виде кортежа (tuple).

*def avg\_color(img, x0, y0, x1, y1).* Функция для решения третьей подзадачи. Создаем копию изображения img\_result. Загружаем пиксели копии изображения в переменную pixels. Используем два вложенных цикла для прохода по каждому пикселю в указанной области (x0, y0) до (x1, y1). Заменяем цвет каждого пикселя средним цветом окружающих его пикселей, используя функцию get\_pixel\_color. Возвращаем обработанную копию изображения. Функция возвращает *imgResult*.

*def pentagram(img, x, y, r, thickness, color).* Функция для решения первой подзадачи. - all\_lines = []: Инициализируется пустой список all\_lines для хранения координат линий пентаграммы. x1, y1 = x - r, y - r и x2, y2 = x + r, y + r: Вычисляются координаты верхнего левого и нижнего правого углов описывающего прямоугольника для эллипса (окружности) пентаграммы. drawing.ellipse(((x1, y1), (x2, y2)), fill=None, outline=tuple(color), width=thickness): Рисуется эллипс (окружность) пентаграммы с заданными параметрами (цвет, толщина линии). for i in range(0, 5):: Итерируемся по углам пентаграммы. phi1 = (np.pi / 5) \* (2 \* i + 3 / 2): Вычисляется угол для первой точки линии. node\_i1 = (int(x + r \* np.cos(phi1)), int(y + r \* np.sin(phi1))): Вычисляются координаты первой точки линии.

phi2 = (np.pi / 5) \* (2 \* (i + 2) + 3 / 2): Вычисляется угол для второй точки линии. node\_i2 = (int(x + r \* np.cos(phi2)), int(y + r \* np.sin(phi2))): Вычисляются координаты второй точки линии. all\_lines.append([node\_i1, node\_i2]): Добавляются координаты линии в список. for el in all\_lines:: Проходим по всем линиям в списке all\_lines. drawing.line((el[0], el[1]), tuple(color), thickness): Рисуется линия с заданными координатами, цветом и толщиной. return img: Возвращается изображение с нарисованной пентаграммой.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | *img=Image.new("RGB", (200, 200), "black")*  *img = pentagram(img, 100, 100, 95, 3, (255, 128, 0))* |  | Пентаграмма нарисована правильно |
|  |  |  | Области вырезаны и вставлены верно, повороты правильные |
|  |  |  | Цвета пикселей подобраны правильно, границы области соблюдены |

## 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):

x0\_1 = x0 + width

y0\_1 = y0 + width

x1\_1 = x1 + width

y1\_1 = y1 + width

image\_1 = img.crop((x0, y0, x0\_1, y0\_1))

image\_1 = image\_1.rotate(270)

image\_2 = img.crop((x1, y1, x1\_1, y1\_1))

image\_2 = image\_2.rotate(270)

answer = img.copy()

answer.paste(image\_1, (x1, y1))

answer.paste(image\_2, (x0, y0))

answer = answer.rotate(270)

return answer

def get\_pixel\_color(img, x, y):

width, height = img.size

pixels = img.load()

surrounding\_pixels = []

for i in range(-1, 2):

for j in range(-1, 2):

if ((x + i) >= 0) and ((x + i) < width):

if ((y + j) >= 0) and ((y + j) < height):

surrounding\_pixels.append(pixels[x + i, y + j])

surrounding\_pixels.remove(pixels[x, y])

result\_color = [0, 0, 0]

for color in surrounding\_pixels:

for rgb in range(3):

result\_color[rgb] += color[rgb]

for rgb in range(3):

result\_color[rgb] = int(result\_color[rgb] / len(surrounding\_pixels))

return tuple(result\_color)

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

img\_result = img.copy()

pixels = img\_result.load()

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

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

pixels[x, y] = get\_pixel\_color(img, x, y)

return img\_result

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

drawing = ImageDraw.Draw(img)

all\_lines = []

x1, y1 = x - r, y - r

x2, y2 = x + r, y + r

drawing.ellipse(((x1, y1), (x2, y2)), fill=None, outline=tuple(color), width=thickness)

for i in range(0, 5):

phi1 = (np.pi / 5) \* (2 \* i + 3 / 2)

node\_i1 = (int(x + r \* np.cos(phi1)), int(y + r \* np.sin(phi1)))

phi2 = (np.pi / 5) \* (2 \* (i + 2) + 3 / 2)

node\_i2 = (int(x + r \* np.cos(phi2)), int(y + r \* np.sin(phi2)))

all\_lines.append([node\_i1, node\_i2])

for el in all\_lines:

drawing.line((el[0], el[1]), tuple(color), thickness)

return img