

**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

Ступак.А.А.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2023

## **Цель работы**

Научиться работать с изображениями с помощью библиотеки Pillow (PIL).

## **Задание.**

### **1) Рисование пентаграммы в круге**

Необходимо написать функцию `solve()`, которая рисует на изображении пентаграмму в окружности.

Функция `solve()` принимает на вход:

- Изображение (`img`)
- координаты центра окружности (`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.

Примечание:

Вершины пентаграммы высчитывать по формуле:

$$\text{phi} = (\pi/5) * (2*i + 3/2)$$

$$\text{node\_i} = (\text{int}(x_0 + r * \cos(\text{phi})), \text{int}(y_0 + r * \sin(\text{phi})))$$

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

`r` - радиус окружности

`i` - номер вершины от 0 до 4

### **2) Поменять местами участки изображения и поворот**

Необходимо реализовать функцию `solve`, 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 `solve()` 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.

Пример входной картинке (300x300) и переданных параметров:



Рисунок 1 – Задача 2

$x_0 = 0$  ;  $y_0 = 0$  ;  $x_1 = 150$  ;  $y_1 = 150$  ;  $width = 150$  ;

Изображение после первого этапа (замена участков и поворот этих участков на 90 градусов):



Рисунок 2 – Задача 2

Итоговое изображение:



Рисунок 3 – Задача 3

### 3) Средний цвет

Необходимо реализовать функцию `solve`, которая заменяет цвет каждого пикселя в области на средний цвет пикселей вокруг (не считая сам этот пиксель).

Функция `solve()` 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 углу

Функция должна вернуть обработанное изображение, не изменяя исходное.

Средний цвет - берется целая часть от среднего каждой компоненты из rgb.  $(\text{int}(\text{sum}(r)/\text{count}), \text{int}(\text{sum}(g)/\text{count}), \text{int}(\text{sum}(b)/\text{count}))$

Пример получения среднего цвета:

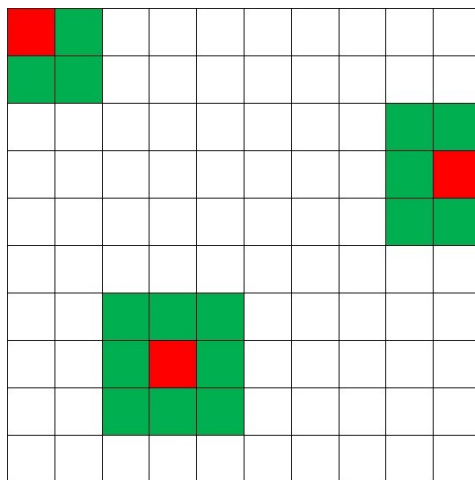


Рисунок 4 – Задача 3

Для красных пикселей, берется средний цвет зеленых пикселей вокруг него.

Можно реализовывать дополнительные функции.

## **Выполнение работы**

Для каждой задачи были разработаны функции (см. приложение А).

Функция `swar()` копирует исходное изображение, копирует нужные участки изображения с помощью метода `image.crop()`, поворачивает эти участки на -90 градусов с помощью `image.rotate()`, меняя их местами вставляет в скопированное изображение, и переворачивает изменённое изображение на -90 градусов.

Функция `avg_color()` копирует исходное изображение потом проходимся по всем пикселям нужного участка и меня каждый пиксель на средний цвет пикселей вокруг с помощью метода `Image.putpixel()`.

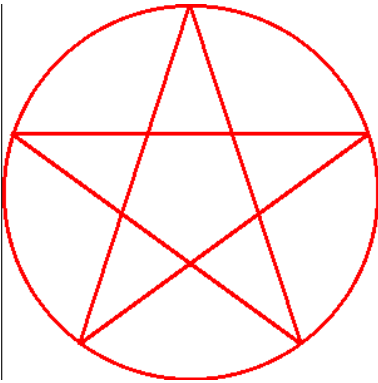
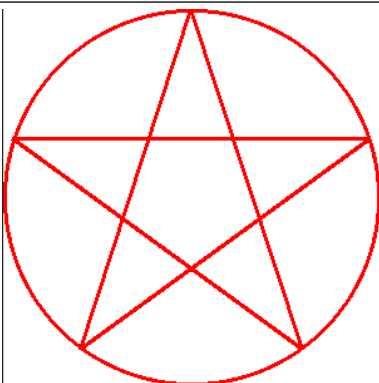
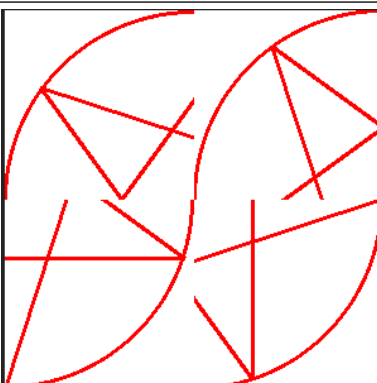
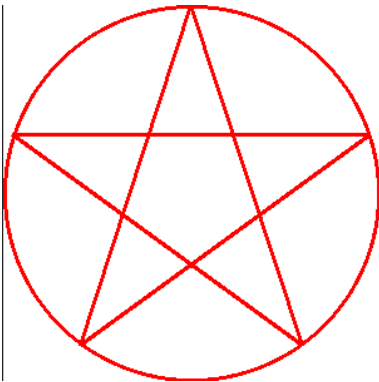
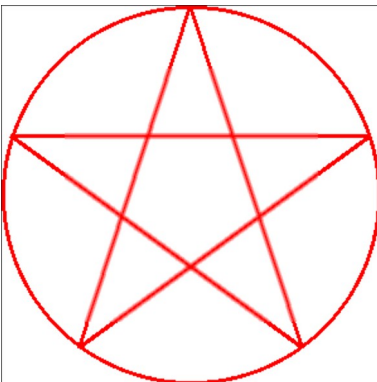
Функция `avg()` для нужного пикселя с помощью метода `Image.getpixel()` передаёт значение цвета пикселей вокруг в список и потом ищет средний цвет.

Функция `pentagram()` рисует пентаграмму используя метод `ImageDraw.line()` и окружность вокруг неё с помощью метода `ImageDraw.ellipse()`

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Изображение залитое белым цветом 300x300, x = 150, y = 150, r = 150, thickness = 3 , color = (255,0,0)		OK
2.	 x0 = 0, y0 = 0, x1 = 150, y1 = 150, width = 150		OK
3.	 x0 = 50, y0 = 50, x1 = 250, y1 =		OK



	250		
--	-----	--	--

### **Выводы**

В данной лабораторной работе была изучена библиотека Pillow и применены на практике такие методы как `Image.crop()`, `Image.rotate()`, `Image.past()`, `Image.putpixel()`, `Image.getpixel()`, `ImageDraw.Draw()`, `ImageDraw.line()` и `ImageDraw.ellipse()`.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: app-001.py

```
from PIL import ImageDraw, Image
import math
```

```
def swap(img, x0, y0, x1, y1, width):
    img2 = img.copy()
    im1 = img.crop((x0, y0, x0 + width, y0 + width))
    im2 = img.crop((x1, y1, x1 + width, y1 + width))
    im1 = im1.rotate(-90)
    im2 = im2.rotate(-90)
    img2.paste(im1, (x1, y1))
    img2.paste(im2, (x0, y0))
    img2 = img2.rotate(-90)
    return img2
```

```
def avg_color(img, x0, y0, x1, y1):
    img1 = img.copy()
    w, h = img1.size
    for x in range(x0, x1 + 1):
        for y in range(y0, y1 + 1):
            img1.putpixel((x, y), avg(img, x, y, w, h))
    return img1

pass
```

```

def avg(img1, x, y, w, h):
    if ((x - 1 < 0) and (y - 1 < 0)):
        colors = [img1.getpixel((x, y + 1)), img1.getpixel((x + 1, y)),
img1.getpixel((x + 1, y + 1))]
        answ = (int(sum(col[0] for col in colors) / 3),
                int(sum(col[1] for col in colors) / 3),
                int(sum(col[2] for col in colors) / 3))
    elif ((x + 1 > w) and (y - 1 < 0)):
        colors = [img1.getpixel((x - 1, y)), img1.getpixel((x - 1, y + 1)),
img1.getpixel((x, y + 1))]
        answ = (int(sum(col[0] for col in colors) / 3),
                int(sum(col[1] for col in colors) / 3),
                int(sum(col[2] for col in colors) / 3))
    elif ((x - 1 < 0) and (y + 1 > h)):
        colors = [img1.getpixel((x, y - 1)), img1.getpixel((x + 1, y)),
img1.getpixel((x + 1, y - 1))]
        answ = (int(sum(col[0] for col in colors) / 3),
                int(sum(col[1] for col in colors) / 3),
                int(sum(col[2] for col in colors) / 3))
    elif ((x + 1 > w) and (y + 1 > h)):
        colors = [img1.getpixel((x - 1, y)), img1.getpixel((x - 1, y - 1)),
img1.getpixel((x, y - 1))]
        answ = (int(sum(col[0] for col in colors) / 3),
                int(sum(col[1] for col in colors) / 3),
                int(sum(col[2] for col in colors) / 3))
    elif (x - 1 < 0):
        colors = [img1.getpixel((x, y + 1)), img1.getpixel((x, y - 1)),
img1.getpixel((x + 1, y)),
img1.getpixel((x + 1, y + 1)), img1.getpixel((x + 1, y - 1))]

```

```

        answ = (int(sum(col[0] for col in colors) / 5),
                int(sum(col[1] for col in colors) / 5),
                int(sum(col[2] for col in colors) / 5))
    elif (x + 1 > w):
        colors = [img1.getpixel((x, y + 1)), img1.getpixel((x, y - 1)),
img1.getpixel((x - 1, y)),
                img1.getpixel((x - 1, y + 1)), img1.getpixel((x - 1, y - 1))]
        answ = (int(sum(col[0] for col in colors) / 5),
                int(sum(col[1] for col in colors) / 5),
                int(sum(col[2] for col in colors) / 5))
    elif (y - 1 < 0):
        colors = [img1.getpixel((x, y + 1)), img1.getpixel((x + 1, y)),
img1.getpixel((x + 1, y + 1)),
                img1.getpixel((x - 1, y)), img1.getpixel((x - 1, y + 1))]
        answ = (int(sum(col[0] for col in colors) / 5),
                int(sum(col[1] for col in colors) / 5),
                int(sum(col[2] for col in colors) / 5))
    elif (y + 1 > h):
        colors = [img1.getpixel((x, y - 1)), img1.getpixel((x + 1, y)),
img1.getpixel((x + 1, y - 1)),
                img1.getpixel((x - 1, y)), img1.getpixel((x - 1, y - 1))]
        answ = (int(sum(col[0] for col in colors) / 5),
                int(sum(col[1] for col in colors) / 5),
                int(sum(col[2] for col in colors) / 5))
    else:
        colors = [img1.getpixel((x - 1, y - 1)), img1.getpixel((x, y - 1)),
img1.getpixel((x + 1, y - 1)),
                img1.getpixel((x - 1, y)), img1.getpixel((x + 1, y)), img1.getpixel((x
- 1, y + 1)),
                img1.getpixel((x, y + 1)), img1.getpixel((x + 1, y + 1))]

```

```

    answ = (int(sum(col[0] for col in colors) / 8),
            int(sum(col[1] for col in colors) / 8),
            int(sum(col[2] for col in colors) / 8))
return answ

```

```

def pentagram(img, x, y, r, thickness, color):
    color = tuple(color)
    x1 = x - r
    y1 = y - r
    x2 = x + r
    y2 = y + r
    coordinates = ((x1, y1), (x2, y2))
    draw = ImageDraw.Draw(img)
    coor = []
    for i in range(5):
        phi = (math.pi / 5) * (2 * i + 3 / 2)
        node_i = (int(x + r * math.cos(phi)), int(y + r * math.sin(phi)))
        coor.append(node_i)
    draw.line((coor[0], coor[2]), color, thickness)
    draw.line((coor[2], coor[4]), color, thickness)
    draw.line((coor[4], coor[1]), color, thickness)
    draw.line((coor[1], coor[3]), color, thickness)
    draw.line((coor[3], coor[0]), color, thickness)
    draw.ellipse(coordinates, None, color, thickness)
    return img

```