

**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

Мальцев К.Л.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2023

### **Цель работы**

Решить 3 подзадачи, используя библиотеку Pillow (PIL) и numpy. Необходимо разработать функции, которые работают с объектами типа `<class 'PIL.Image.Image'>`.

## Задание

### Вариант 2

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:

$$\phi_i = (\pi/5) * (2*i + 3/2)$$

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

$x_0, y_0$  - 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 `mix()` принимает на вход:

Изображение (`img`)

Словарь с описанием того, какие части на какие менять (`rules`)

Пример словаря `rules`:

{0:1,1:2,2:4,3:4,4:5,5:3,6:8,7:8,8:8}

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.

Пример входной картинки и словаря:

Картинка



{0:2,1:2,2:2,3:5,4:5,5:5,6:8,7:8,8:8}

Результат:



Можно реализовывать дополнительные функции.

## Основные теоретические положения

Библиотека *PIL* (*Python Imaging Library*) - это библиотека для работы с изображениями. Она предоставляет функции для открытия, изменения, сохранения и обработки изображений, а также для создания новых изображений. Для доступа к функциям библиотеки мы импортировали ее используя "*import PIL*".

Модуль *Image* из библиотеки *PIL* - это класс, предоставляющий различные методы для работы с изображениями, такие как открытие, сохранение, изменение размера, поворот, фильтрация и многое другое. Модуль импортирован из библиотеки *PIL* с помощью "*from PIL import Image*".

Модуль *ImageDraw* из библиотеки *PIL* - это класс, который предоставляет методы для рисования на изображениях. Он использован для рисования фигур и линий на изображении. Модуль импортирован из библиотеки *PIL* с помощью "*from PIL import ImageDraw*".

Модуль *ImageOps* из библиотеки *PIL* - это класс, предоставляющий различные методы для обработки изображений, такие как изменение контраста, наложение эффектов и другие операции. Использован в функции *invert* для инвертирования цветов изображения. Модуль импортирован из библиотеки *PIL* с помощью "*from PIL import ImageOps*".

Библиотека *numpy* - это библиотека для выполнения математических операций, включая многомерные массивы и функции для работы с ними. Библиотека импортирована с помощью "*import numpy as np*".

## Выполнение работы

- Импортируем библиотеки PIL (Pillow), Image, ImageDraw, ImageOps и numpy.
- Объявляем функцию pentagram с входными параметрами img, x0, y0, x1, y1, thickness, color.
- Создаем объект ImageDraw для рисования на изображении img.
- Вычисляем радиус эллипса и координаты его центра.
- Создаем пустой список nodes для хранения координат вершин пентаграммы.
- В цикле проходим по числам от 0 до 4 и вычисляем координаты вершин пентаграммы с помощью формулы.
- Сохраняем вершины пентаграммы в списке nodes.
- Устанавливаем последней вершине координаты первой вершины, чтобы пентаграмма была замкнутой.
- Рисуем эллипс на изображении с заданными координатами, цветом и толщиной контура.
- Рисуем линию, соединяющую вершины пентаграммы, с заданным цветом и толщиной.
- Возвращаем исходное изображение с нарисованной пентаграммой.
- Объявляем функцию invert с входными параметрами img, N, vertical.
- Получаем ширину и высоту изображения.
- Если vertical равно True, выполняем следующие операции для каждой нечетной части ширины.
  - Выбираем часть изображения с помощью crop.
  - Инвертируем цвета выбранной части с помощью функции invert из библиотеки ImageOps.
  - Вставляем инвертированную часть обратно в изображение с помощью paste.

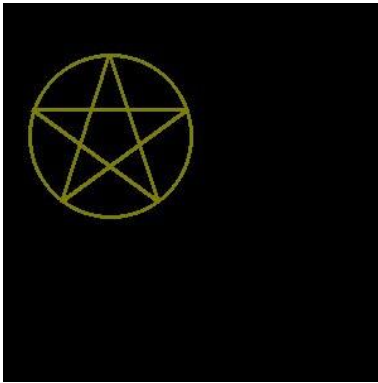
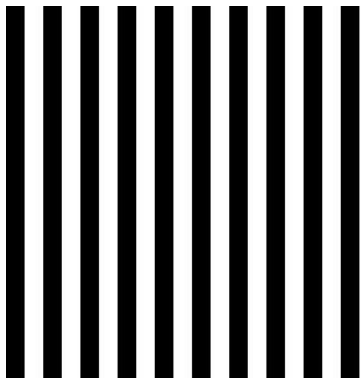



- Если `vertical` равно `False`, выполняем аналогичные операции для каждой нечетной части высоты.
- Возвращаем исходное изображение с инвертированными частями.
- Объявляем функцию `mix` с входными параметрами `img` и `rules`.
- Получаем ширину и высоту изображения.
- Создаем пустой список `parts` для хранения частей изображения.
- Вложенными циклами проходим по 9 частям изображения (3 по горизонтали и 3 по вертикали).
  - Выбираем часть изображения с помощью `crop`.
  - Добавляем выбранный фрагмент и его координаты в список `parts`.
  - Проходим по всем элементам словаря `rules`.
  - Извлекаем значение элемента словаря, используя значение как индекс в списке `parts`.
  - Извлекаем фрагмент изображения и его координаты, используя значение ключа как индекс в списке `parts`.
  - Вставляем фрагмент изображения обратно в изображение, используя его координаты.
- Возвращаем исходное изображение с перемешанными частями.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	<code>pentagram(Image.new("RGB", (300, 300), 0), 20, 40, 150, 170, 3, [128, 128, 0])</code>		-
2.	<code>invert(Image.new("RGB", (300, 300), 0), 15, True)</code>		-
3.	<code>mix(Image.open('krab1'), {0:2, 1:2, 2:2, 3:5, 4:5, 5:5, 6:8, 7:8, 8:8})</code>		-

## **Выводы**

Разработаны функции, которые могут работать с объектами типа `<class 'PIL.Image.Image'>`, а также решены 3 подзадачи, используя библиотеки Pillow (PIL) и numpy.

Таким образом, проект успешно достиг поставленных целей по разработке функций для работы с объектами типа `<class 'PIL.Image.Image'>` и решению 3 подзадач. Реализация данных функций позволяет удобно и эффективно обрабатывать изображения и выполнять необходимые операции с ними.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import PIL
from PIL import Image, ImageDraw, ImageOps
import numpy as np

def pentagram(img, x0, y0, x1, y1, thickness, color):
    drawing = ImageDraw.Draw(img)

    radius = abs(x1 - x0) // 2
    center_x = x1 - (abs(x1 - x0) // 2)
    center_y = y1 - (abs(y1 - y0) // 2)

    nodes = []
    for i in range(5):
        phi = (np.pi)*(2*i+3/2)/5
        node_i = ( int(center_x + radius*np.cos(phi)), int(center_y
+ radius*np.sin(phi)) )
        nodes.append(node_i)

    nodes = [nodes[0], nodes[2], nodes[4], nodes[1], nodes[3],
nodes[0]]

    drawing.ellipse((x0, y0, x1, y1), outline=tuple(color),
width=thickness)
    drawing.line(nodes, fill=tuple(color), width=thickness)

    return img

def invert(img, N, vertical):
    width, height = img.size
    if (vertical):
        for j in range(1, width//N+1, 2):
            inverted_part = img.crop( (j*N, 0, (j+1)*N, height) )
            inverted_part = ImageOps.invert(inverted_part)
            img.paste(inverted_part, (j*N, 0) )
    else:
        for i in range(1, height//N+1, 2):
            inverted_part = img.crop( (0, i*N, width, (i+1)*N) )
            inverted_part = ImageOps.invert(inverted_part)
            img.paste(inverted_part, (0, i*N) )
    return img

def mix(img, rules):
    width, height = img.size
    parts = []
    for j in range(3):
        for i in range(3):
            part = img.crop( ( i*(height//3), j*(width//3),
(i+1)*(height//3), (j+1)*(width//3) ) )
```

```
        parts.append( [part, (i*(height//3), j*(width//3))] )
for i in rules:
    img.paste(parts[rules[i]][0], parts[i][1])
return img
```