

**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. 3343

Никишин С.А.

Преподаватель

Иванов Д. В.

Санкт-Петербург

2023

## **Цель работы**

Научится работать с модулем *Pillow (PIL)*, а также с функциями *numpy*, выполнять различные графические преобразования над изображениями.

## Задание

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:

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

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

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

*r* - радиус окружности

*i* - номер вершины от 0 до 4

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 () принимает на вход:

- Изображение (*img*)
- Словарь с описанием того, какие части на какие менять (*rules*)

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.

При необходимости можно писать дополнительные функции.

## Выполнение работы

Функция *pentagram()* рисует на полученном изображении *img* пентаграмму, используя функции и методы библиотек *pillow* и *numpy*. Координаты круга определяются по координатам описанного вокруг неё квадрата ( $x0$ ,  $y0$  — левый верхний угол,  $x1$ ,  $y1$  — правый нижний угол). Пятиконечная звезда рисуется пятью отдельными линиями, координаты для обоих концов которых высчитываются относительно координат центра круга и длины его радиуса.

Функция *invert()* инвертирует цвет нечётных полос ширины  $N$  изначального изображения и возвращает изменённое изображение *img*. Высчитывается максимальный размер изображения по обеим осям. Цикл *for* проходит по всему изображению и поочерёдно копирует полосы исходного изображения нужного размера, инвертирует их (используется *ImageOps.invert()*) и вставляет в исходное изображение на прежнее место.

Функция *mix()* разделяет изображение на 9 прямоугольников и перемешивает их согласно инструкции. Возвращает изменённое изображение. Определяются длина и ширина одного прямоугольника. Создаётся список, элементами которого являются вырезанные из основного прямоугольные изображения одинакового размера (нумерация идёт слева-направо сверху-вниз). Благодаря циклу *for* программа просматривает каждый из изначальных кусков изображения, и на их место вставляет нужные изображения из списка согласно словарю.

## **Выводы**

Были изучены различные способы преобразования изображения, написаны функции с использованием библиотек *numpy* и *pillow*.

Программа может получать на вход исходное изображение и возвращать изменённое, в зависимости от выбранной функции.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from PIL import Image, ImageDraw, ImageOps
from numpy import pi, cos, sin
import math

def pentagram(img, x0, y0, x1, y1, tolcschina, cvet):
    cvet = tuple(cvet)
    kartinka = ImageDraw.Draw(img)
    kartinka.ellipse(((x0, y0), (x1, y1)), None, cvet, tolcschina)
    okruglen = round((x1 - x0) / 2)
    uzli = {}
    x = round(x0 + okruglen)
    y = round(y0 + okruglen)
    for i in range(5):
        phi = (pi / 5) * (2 * i + 3 / 2)
        uzel = (int(x + okruglen * cos(phi)), int(y + okruglen * sin(phi)))
        uzli[i] = [uzel, i + 1]
        if i == 4:
            uzli[i] = [uzel, 0]
    for i in range(5):
        sleduzl = uzli[uzli[i][1]]
        posleduzl = uzli[sleduzl[1]]
        kartinka.line((uzli[i][0], posleduzl[0]), cvet, tolcschina)
    return img

def invert(img, N, vertical):
    razmer = img.size
    nomeraziastrok = int(math.ceil(razmer[0] / N))
    for i in range(nomeraziastrok):
        if vertical:
            obrez = img.crop((N * i, 0, N * (i + 1), razmer[1]))
        else:
            obrez = img.crop((0, N * i, razmer[1], N * (i + 1)))
        if i % 2 != 0:
            obrez = ImageOps.invert(obrez)
        if vertical:
            img.paste(obrez, (N * i, 0))
        else:
            img.paste(obrez, (0, N * i))
    return img

def mix(img, rules):
    razmer = img.size
    obrezrazmer = (razmer[0] // 3, razmer[1] // 3)
    obrezpikchas = []
    for i in range(3):
        for j in range(3):
            levo = (j * obrezrazmer[0], i * obrezrazmer[1])
            pravo = ((j + 1) * obrezrazmer[0], (i + 1) * obrezrazmer[1])
            obrezpikcha = img.crop((levo[0], levo[1], pravo[0], pravo[1]))
            obrezpikchas.append(obrezpikcha)
    schet = 0
```

```
for i in range(3):
    for j in range(3):
        obrepikcha = obrepikchas[rules[schet]]
        schet += 1
        img.paste(obrepikcha, (j * obrezrazmer[0], i *
obrezrazmer[1]))
    return img
```



## ПРИЛОЖЕНИЕ Б

### ТЕСТИРОВАНИЕ

Исходные изображения:

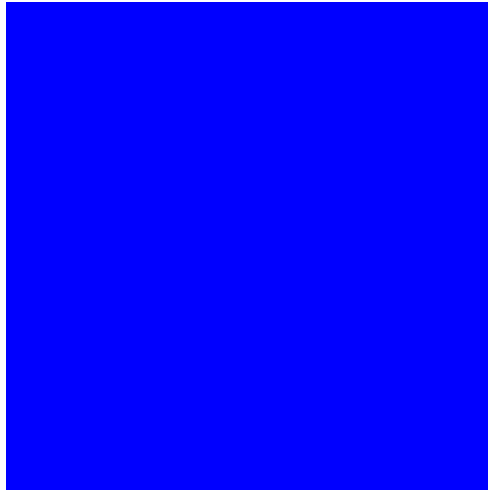


Рисунок 1 – Изображение для функции *pentagram*

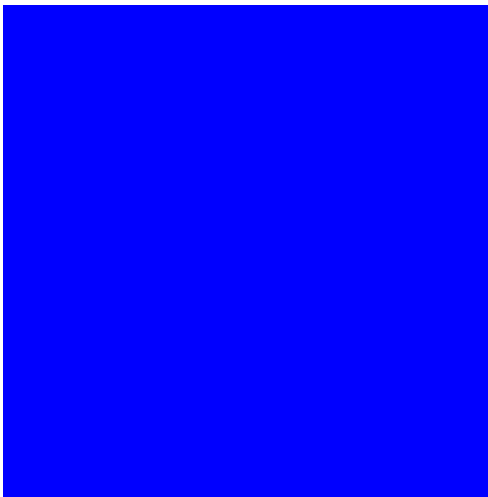


Рисунок 2 – Изображение для функции *invert*

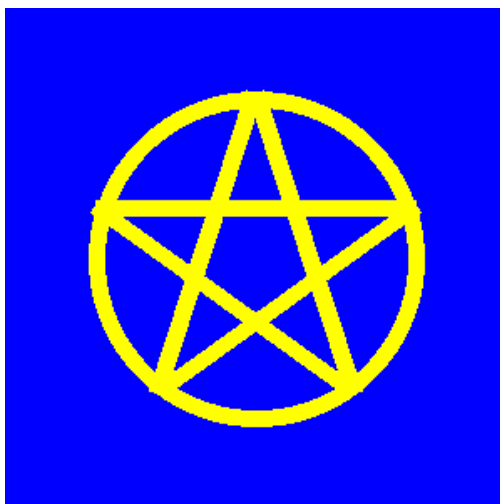


Рисунок 3 – Изображение для функции *mix*

Параметры функций:

1) Для рисунка 1 функция

*pentagram(img, 50, 50, 250, 250, 10, [255, 255, 0])*

2) Для рисунка 2 функция

*invert(img, 45, False)*

3) Для рисунка 3 функция

*mix(img, {0:2,1:2,2:2,3:5,4:5,5:5,6:8,7:8,8:8})*

Результат:

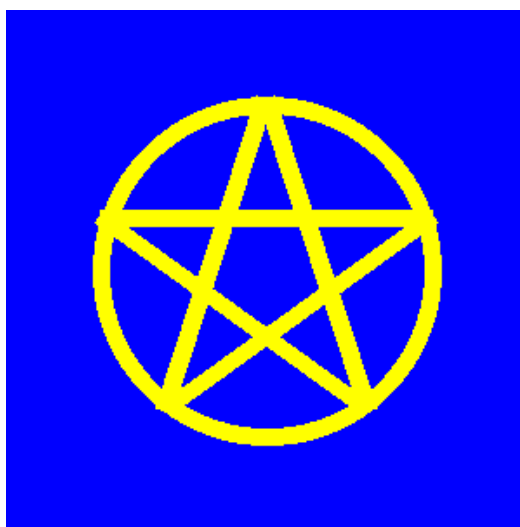


Рисунок 1 – Результат функции *pentagram*

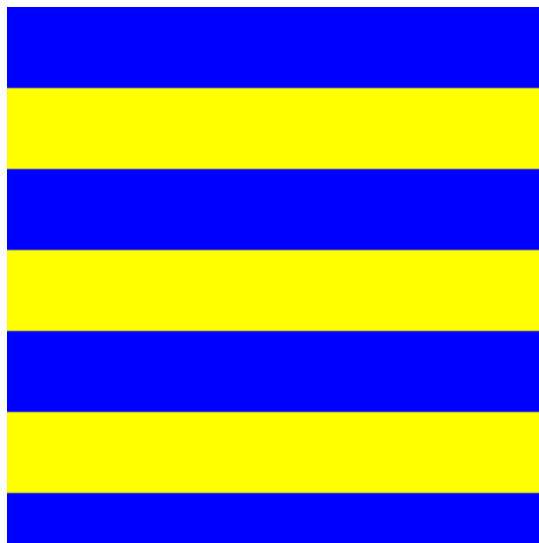


Рисунок 2 – Результат функции *invert*

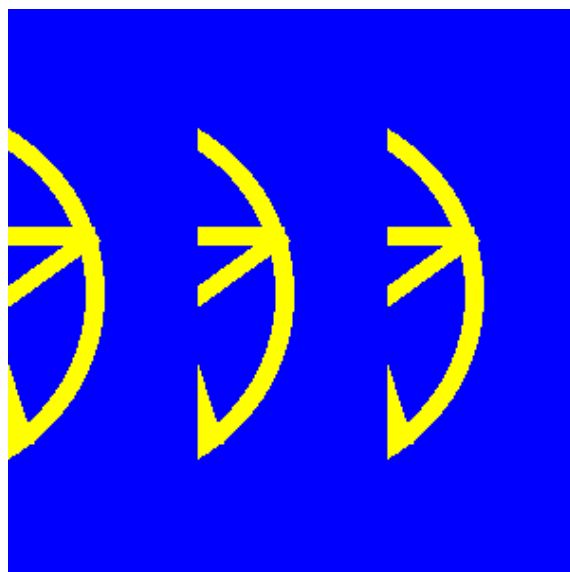


Рисунок 3 – Результат функции *mix*