**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 |  | Трофимов В.О. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Целью работы является изучение библиотеки Pillow языка python и его практическое применение для решения трёх задач лабораторной работы.

## Задание

Вариант 4

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 отрезка. Отрезок определяется:

координатами начала

координатами конца

цветом

толщиной.

Необходимо реализовать функцию user\_func(), рисующую на картинке отрезок

Функция user\_func() принимает на вход:

изображение;

координаты начала (x0, y0);

координаты конца (x1, y1);

цвет;

толщину.

Функция должна вернуть обработанное изображение.

2) Преобразовать в Ч/Б изображение (любым простым способом).

Функционал определяется:

Координатами левого верхнего угла области;

Координатами правого нижнего угла области;

Алгоритмом, если реализовано несколько алгоритмов преобразования изображения (по желанию студента).

Нужно реализовать 2 функции:

check\_coords(image, x0, y0, x1, y1) - проверяет координаты области (x0, y0, x1, y1) на корректность (они должны быть неотрицательными, не превышать размеров изображения, поскольку x0, y0 - координаты левого верхнего угла, x1, y1 - координаты правого нижнего угла, то x1 должен быть больше x0, а y1 должен быть больше y0);

set\_black\_white(image, x0, y0, x1, y1) - преобразовывает заданную область изображения в черно-белый (используйте для конвертации параметр '1'). В этой функции должна вызываться функция проверки, и, если область некорректна, то должно быть возвращено исходное изображение без изменений. Примечание: поскольку черно-белый формат изображения (greyscale) является самостоятельным форматом, а не вариацией RGB-формата, для его получения необходимо использовать метод Image.convert.

3) Найти самый большой прямоугольник заданного цвета и перекрасить его в другой цвет. Функционал определяется:

Цветом, прямоугольник которого надо найти

Цветом, в который надо его перекрасить.

Написать функцию find\_rect\_and\_recolor(image, old\_color, new\_color), принимающую на вход изображение и кортежи rgb-компонент старого и нового цветов. Она выполняет задачу и возвращает изображение. При необходимости можно писать дополнительные функции.

## Выполнение работы

Определяются следующие функции для решения каждой из подзадач:

1. def user\_func(image, x0, y0, x1, y1, fill, width):

2. def check\_coords(image, x0, y0, x1, y1):

3. def find\_rect\_and\_recolor(image, old\_color, new\_color):

Для решения задачи № 1 была определена функция def user\_func(image, x0, y0, x1, y1, fill, width):

Данная функция user\_func() принимает на вход изображение (image), координаты начала отрезка (x0, y0), координаты конца отрезка (x1, y1), цвет отрезка (fill) и ширину линии (width).

Сначала создается объект ImageDraw.Draw, который представляет собой класс для рисования на изображении. Затем с помощью метода line() задается рисование отрезка с координатами начала и конца, цветом fill и шириной width. После наложения отрезка на изображение, функция возвращает image.

Для решения задачи № 2 были определены функции def check\_coords(image, x0, y0, x1, y1): и set\_black\_white(image, x0, y0, x1, y1):

Функция check\_coords(image, x0, y0, x1, y1) предназначена для проверки корректности координат области на изображении. Она принимает на вход изображение и координаты (x0, y0), (x1, y1) левого верхнего и правого нижнего углов области.

Объявляются переменные width, height значение, которых получается с помощью image.size. Дальше последовательно проверяются различные условия: координаты должны быть неотрицательными, а также правый нижний угол области должен иметь большую координату x и y, чем левый верхний угол, и не должен выходить за пределы ширины и высоты изображения. В случае выполнения всех условий возвращается значение True, указывающее на корректность координат. В случае несоответствия любому из условий возвращается None или False.

Функция set\_black\_white(image, x0, y0, x1, y1) предназначена для преобразования указанной области на изображении в черно-белый цвет.

Вызывается функция check\_coords() для проверки корректности координат области. Если координаты области проверку не прошли, то функция возвращает исходное изображение без изменений. Если координаты области проверку прошли, то дальше объявляется переменная area, которая хранит в себе кортеж с координатами области (x0, y0, x1, y1). Объявляется переменная cropped\_area, в которую записывают указанную область на изображении, используя метод image.crop(area). Объявляется переменная black\_white\_cropped\_area, которая хранит в себе конвертированную в черно-белый область (cropped\_area.convert "1"). Исходная область изображения заменяется на полученную черно-белую область с помощью метода image.paste(black\_white\_cropped\_area, area). Функция возвращает изменённое изображение.

Для решения задачи № 3 были определены функции def find\_rect\_and\_recolor(image, old\_color, new\_color): и def flood\_fill(x, y, width, height, pixels, old\_color):

Функция flood\_fill(x, y, width, height, pixels, old\_color) используется для поиска граничных координат прямоугольника. Она работает по принципу "заливки".

Поиск начинается от координат x, y и помещает их в стек. Далее извлекает координаты из стека и проверяет их цвет. Если цвет равен old\_color, то он изменяется на новый цвет и координаты обновляются. После происходит добавление соседних пикселей в стек. Процесс продолжится до тех пор, пока весь прямоугольный регион не будет "залит", тем самым в результате будут известные крайние координаты самого большого прямоугольника заданного цвета. Функция возвращает координаты, представляющие граничный прямоугольник, вычисленные на основе минимальных и максимальных координат найденных пикселей.

Функция def find\_rect\_and\_recolor(image, old\_color, new\_color) принимает на вход изображение(image), цвет прямоугольника, который нужно изменить на новый цвет(old\_color), новый цвет(new\_color).

Сначала в функции делается копия входного изображения image. С помощью copy\_image.load() и image.load() получаем доступ к пикселям исходного и скопированного изображения соответственно. Затем происходит инициализация начальных значений для координат наибольшего прямоугольника (max\_rectangle\_coords) и его площади (max\_rectangle\_area). Далее осуществляется проход по каждому пикселю изображения, проверяя, если цвет пикселя совпадает с old\_color, то вызывается функция flood\_fill для нахождения граничных координат прямоугольника и расчета его площади. Если найденная площадь прямоугольника больше, чем текущая максимальная, то обновляются координаты и площадь максимального прямоугольника. После этого происходит проход и перекраска всех пикселей максимального прямоугольника в новый цвет new\_color. Функция возвращает изображение с перекрашенным самым большим прямоугольником заданного цвета.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | img = Image.new('RGB', (300, 200), 'white')  user\_func(img, 225, 80, 100, 20, 'blue', 15)  func(img, 0, 0, 140, 110, 0, None, 1, 'yellow')  func(img, 85, 115, 145, 165, 0, None, 1, 'yellow')  set\_black\_white(img, 105, 35, 255, 65)  find\_rect\_and\_recolor(img, 'yellow', 'red')  img.show() |  |

## Выводы

Была изучена библиотека Pillow языка python и полученные знания были подкреплены практикой решением трёх задач.

В итоге была разработаны три функции: 1) Функция может нарисовать отрезок на изображении. 2) Функция может преобразовать в чёрно-белый цвет изображение определённую область изображения. 3) Функция находит в изображении самый большой прямоугольник заданного цвета и перекрашивает его в другой цвет.

# Приложение А Исходный код программы

Название файла: lab2.py

import PIL

import math

from PIL import Image, ImageDraw

# Задача 1

def user\_func(image, x0, y0, x1, y1, fill, width):

drawing = ImageDraw.Draw(image)

drawing.line((x0,y0,x1,y1),fill,width)

return image

# Задача 2

def check\_coords(image, x0, y0, x1, y1):

width,height = image.size

if (x0 >= 0) and (x1 >= 0) and (y0 >= 0) and (y1 >= 0):

if (x1 > x0) and (y1 > y0) and (x1 < width) and (y1 < height):

return True

def set\_black\_white(image, x0, y0, x1, y1):

if not check\_coords(image,x0,y0,x1,y1):

return image

area = (x0,y0,x1,y1)

cropped\_area = image.crop(area)

black\_white\_cropped\_area = cropped\_area.convert("1")

image.paste(black\_white\_cropped\_area,area)

return image

# Задача 3

def find\_rect\_and\_recolor(image, old\_color, new\_color):

copy\_image = image.copy()

pixels = copy\_image.load()

res\_pixels = image.load()

width,height = image.size

max\_rectangle\_coords = (0,0,0,0)

max\_rectangle\_area = 0

for x in range(width):

for y in range(height):

if pixels[x,y] == old\_color:

rectangle\_coords = flood\_fill(x,y,width,height,pixels,old\_color)

coord\_area = (rectangle\_coords[2] - rectangle\_coords[0]) \* (rectangle\_coords[3] - rectangle\_coords[1]) # площадь прямоугольника

if coord\_area > max\_rectangle\_area:

max\_rectangle\_coords = rectangle\_coords

max\_rectangle\_area = coord\_area

for x in range(max\_rectangle\_coords[0],max\_rectangle\_coords[2]):

for y in range(max\_rectangle\_coords[1],max\_rectangle\_coords[3]):

res\_pixels[x,y] = new\_color

return image

def flood\_fill(x, y, width, height, pixels, old\_color):

stack = [(x,y)]

min\_coord = [width,height] # минимальные координаты по x и y

max\_coord = [0,0] # максимальные координаты по x и y

while stack:

current\_x,current\_y = stack.pop()

if 0 <= current\_x < width and 0 <= current\_y < height:

if pixels[current\_x,current\_y] == old\_color:

pixels[current\_x,current\_y] = (0,0,0,0) # изменяет цвет, чтобы не было повторной обработки пикселя

min\_coord[0] = min(min\_coord[0],current\_x)

min\_coord[1] = min(min\_coord[1],current\_y)

max\_coord[0] = max(max\_coord[0],current\_x)

max\_coord[1] = max(max\_coord[1],current\_y)

stack.append((current\_x,current\_y + 1)) #добавляются координаты "соседних" пикселей и после уже они проходят по циклу

stack.append((current\_x,current\_y - 1))

stack.append((current\_x + 1,current\_y))

stack.append((current\_x - 1,current\_y))

return (min\_coord[0],min\_coord[1],max\_coord[0] + 1, max\_coord[1] + 1)