**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Информатика»**

**Тема: Введение в архитектуру компьютера**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Пивоев Н.М. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Создание программы на языке Python с использованием основных управляющих конструкций языка, а также ознакомление с модулем *Pillow* и углублённое изучения модуля *numpy*, применение их в созданном проекте.

## 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 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

Созданный проект включает несколько функций, направленных на редактирование изображения.

Первое задание. *user\_func*. Она на основе входных данных рисует линию на image с помощью метода *line*. *x0, y0, x1, y1* задают координаты начала и конца отрезка, *fill* - его цвет, а *width* – ширину линии. Функция возвращает отредактированное изображение.

Второе задание. *set\_black\_white*. Она изменяет изображение *image*, превращая его в чёрно-белое. На вход подаётся изображение и координаты, определяющие область, которую необходимо превратить в чёрно-белое изображение. Функция вызывает *check\_coords* для проверки валидности введённых координат. Функция возвращает отредактированное изображение.

*check\_coords*. Она проверяет координаты на валидность. Если координаты правого нижнего угла выше или левее координат левого верхнего угла, то возвращает *False*. Если координаты левого верхнего угла отрицательные, то возвращает *False*. Если координаты правого нижнего выходят за границы изображения, то возвращает *False*. Если введённые данные прошли все проверки, то возвращает *True.*

Третье задание. *find\_rect\_and\_recolor*. Она вызывает функцию *get\_largest\_rectangle* для определения координат наибольшего прямоугольника заданного цвета. После этого проводится проверка на присутствие запрашиваемого цвета в изображении. Далее изображение раскладывается на пиксели. Те, что находятся в области прямоугольника перекрашиваются в цвет *new\_color*. Изменения применяются к исходному изображению. Функция отредактированное изображение.

*get\_largest\_rectangle*. Она находит координаты координаты наибольшего прямоугольника заданного цвета. Сначала идёт разложение изображения по пикселям, цвет пикселей хранится в формате RGB в массиве *pixels*. Значение пикселей, отличных от необходимого цвета обнуляются, а при совпадении заменяются на единицу. Далее идёт проход по всем пикселям и сохранение в каждой ячейке числа непрерывных элементов нужного цвета, находящихся в том же столбце выше. Затем вычисляется самый большой прямоугольник и сохраняются его координаты. В *temp* хранится текущая площадь прямоугольника, которая в каждой итерации сравнивается с максимальной площадью. Если *temp* больше, то вызывается *count\_max\_area* для нахождение координат левого верхнего и правого нижнего углов прямоугольника, а *temp* становится новым максимумом. Если следующий пиксель отличается по цвету от текущего, то *temp* обнуляется. Так как для реализации обнуления необходимо сократить на 1 число итераций, то последний пиксель каждой строки рассматривается отдельно. Сначала рассматривается случай, когда пиксель нужного цвета единственный и находится в самой правой позиции. Затем идёт проверка случая равенства последних символов строки. Функция возвращает координаты границ нужного прямоугольника.

*count\_max\_*area. Она высчитывает координаты левой верхней и правой нижней точек, основываясь на массиве пикселей, максимальной площади и текущей позиции в циклах. Координата *x0* вычисляется как разность координаты (*x1*+1*)* и числа строк в прямоугольнике. Координата *y0* вычисляется как разность (*y1*+1) и значения в текущей ячейке. Функция возвращает координаты границ прямоугольника.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования содержатся в таблице 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | user\_func(Image.new(’RGB’,(500,500), (255,0,255)),100,100,400,400,(0,0,0),5) |  | Тестирование функции *user\_func()* |
|  | set\_black\_white(Image.new(’RGB’,(500,500),(255,0,0)),0,0,500,500) |  | Тестирование функций *check\_coords()* и *set\_black\_white()* |
|  | find\_rect\_and\_recolor(Image.new(‘RGB‘,(500,500),(255,0,0)),(255,0,0),(0,255,0)) |  | Тестирование функций *find\_rect\_and\_recolor(), get\_largest\_rectangle() и count\_max\_area()* |

## Выводы

В результате работы были реализованы функции с использованием основных управляющих конструкций языка и модулей *numpy* и *Pillow*, также изучены основные методы этих модулей. Реализованный проект успешно выполняет поставленные задачи, направленные на редактирование изображений.

# Приложение А Исходный код программы

Название файла: main.py

from PIL import Image,ImageDraw

import numpy as np

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

drawing = ImageDraw.Draw(image)

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

return image

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

if (x0 > x1 or y0 > y1):

return False

if (x0 < 0 or y0 < 0):

return False

if (image.width < x1 or image.height < y1):

return False

return True

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

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

return image

colorless\_image = image.crop((x0, y0, x1, y1))

colorless\_image = colorless\_image.convert("1")

image.paste(colorless\_image,(x0, y0, x1, y1))

return image

def count\_max\_area(pixels, max\_area, i, j):

value = pixels[i][j]

x1 = j

y1 = i

if max\_area == 0:

x0 = x1

y0 = y1

return x0, y0, x1, y1

x0 = x1 - (max\_area // value) + 1

y0 = i - value + 1

return x0, y0, x1, y1

def get\_largest\_rectangle(image, color):

#Преобразование изображения в массив RGB цветов

color = list(color)

pixels=np.array(image).tolist()

#Обнуление ячеек, отличных от требуемого цвета

for i in range(len(pixels)):

for j in range(len(pixels[i])):

pixels[i][j] = 1 if pixels[i][j] == color else 0

pixels=np.array(pixels)

#Сохранение в ячейках числа непрерывных элементов

#необходимого цвета, находящихся выше в том же столбце

for i in range (1,len(pixels)):

for j in range(len(pixels[i])):

if pixels[i][j] == 1:

pixels[i][j] += pixels[i-1][j]

max\_area = 0

x0 = y0 = x1 = y1 = -1

#Вычисление координат самого большого прямоугольника заданного цвета

for i in range(len(pixels)):

temp = 0

for j in range(len(pixels[i])-1):

#Вычисление координат прямоугольника

temp += pixels[i][j]

if temp > max\_area:

max\_area = temp

x0, y0, x1, y1 = count\_max\_area(pixels,max\_area,i,j)

#Обнуление размера при изменении цвета следущего элемента

if pixels[i][j] != pixels[i][j+1]:

temp = 0

#Случай, когда пиксель нужного цвета единственный и

#находится в самой правой позиции

if max\_area == 0:

max\_area = pixels[i][len(pixels[i])-1]

x0, y0, x1, y1 = count\_max\_area(pixels,max\_area,i,len(pixels[i])-1)

#Проверка равенства двух предпоследних символов в строке

if pixels[i][len(pixels[i])-1] == pixels[i][len(pixels[i])-2]:

temp += pixels[i][len(pixels[i])-1]

if temp > max\_area:

max\_area = temp

x0, y0, x1, y1 = count\_max\_area(pixels,max\_area,i,len(pixels[i])-1)

return (x0,y0,x1,y1)

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

coords=get\_largest\_rectangle(image,color)

#Проверка на присутствие запрашиваемого цвета в изображении

if any([i == -1 for i in coords]):

return image

#Перекраска изображения

pixels=np.array(image).tolist()

for i in range(len(pixels)):

for j in range(len(pixels[i])):

if coords[1] <= i <= coords[3] and coords[0] <= j <= coords[2]:

pixels[i][j] = new\_color

image = Image.fromarray(np.uint8(pixels))

return image