import os  
import cv2  
import numpy as np  
  
""" CONSTS """  
PATH = os.path.dirname(os.path.abspath(\_\_file\_\_))  
  
  
""" 12 задание """  
def twelve():  
 """ a """  
 image\_with\_details = cv2.imread(f'{PATH}/first.jpg')  
 cv2.imshow("Default Details", image\_with\_details)  
 print('Original Dimensions : ', image\_with\_details.shape)  
 width = int(image\_with\_details.shape[1] \* 0.5)  
 height = int(image\_with\_details.shape[0] \* 0.5)  
 dim = (width, height)  
 resized = cv2.resize(image\_with\_details, dsize=dim)  
 width = int(resized.shape[1] \* 0.5)  
 height = int(resized.shape[0] \* 0.5)  
 dim = (width, height)  
 resized = cv2.resize(resized, dsize=dim)  
 width = int(resized.shape[1] \* 0.5)  
 height = int(resized.shape[0] \* 0.5)  
 dim = (width, height)  
 resized = cv2.resize(resized, dsize=dim)  
 cv2.namedWindow('Resized', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('Resized', 1280, 854)  
 print('Resized Dimensions : ', resized.shape)  
 cv2.imshow("Resized", resized)  
 cv2.waitKey(0)  
 """ b """  
 pyr\_image = cv2.pyrDown(image\_with\_details)  
 pyr\_image = cv2.pyrDown(pyr\_image)  
 pyr\_image = cv2.pyrDown(pyr\_image)  
 cv2.namedWindow('Resized', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('Resized', 1280, 854)  
 print('Resized Dimensions : ', resized.shape)  
 cv2.imshow("Resized", resized)  
 cv2.namedWindow('PyrImage', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('PyrImage', 1280, 854)  
 print('PyrImage : ', pyr\_image.shape)  
 cv2.imshow("PyrImage", pyr\_image)  
 cv2.waitKey(0)  
 """ c """  
 """   
 Изображение уменьшенное при помощи pyrDown получилось более размытым. pyrDown понижает шаг дискретизации   
 пирамиды Гаусса, в то время как resize меняет размер применяя раличные методы интерполяции.  
 """  
  
  
""" 13 задание """  
def thirteen():  
 image = cv2.imread(f'{PATH}/second.jpg')  
 ret, thresh = cv2.threshold(image, 128, 255, cv2.THRESH\_MASK)  
 cv2.namedWindow('THRESH\_MASK', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('THRESH\_MASK', 1280, 854)  
 cv2.imshow("THRESH\_MASK", thresh)  
 ret, thresh = cv2.threshold(image, 128, 255, cv2.THRESH\_BINARY)  
 cv2.namedWindow('THRESH\_BINARY', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('THRESH\_BINARY', 1280, 854)  
 cv2.imshow("THRESH\_BINARY", thresh)  
 ret, thresh = cv2.threshold(image, 128, 255, cv2.THRESH\_BINARY\_INV)  
 cv2.namedWindow('THRESH\_BINARY\_INV', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('THRESH\_BINARY\_INV', 1280, 854)  
 cv2.imshow("THRESH\_BINARY\_INV", thresh)  
 ret, thresh = cv2.threshold(image, 128, 255, cv2.THRESH\_TRUNC)  
 cv2.namedWindow('THRESH\_TRUNC', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('THRESH\_TRUNC', 1280, 854)  
 cv2.imshow("THRESH\_TRUNC", thresh)  
 ret, thresh = cv2.threshold(image, 128, 255, cv2.THRESH\_TOZERO)  
 cv2.namedWindow('THRESH\_TOZERO', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('THRESH\_TOZERO', 1280, 854)  
 cv2.imshow("THRESH\_TOZERO", thresh)  
 ret, thresh = cv2.threshold(image, 128, 255, cv2.THRESH\_TOZERO\_INV)  
 cv2.namedWindow('THRESH\_TOZERO\_INV', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('THRESH\_TOZERO\_INV', 1280, 854)  
 cv2.imshow("THRESH\_TOZERO\_INV", thresh)  
 cv2.waitKey(0)  
 """ a """  
 image = cv2.imread(f'{PATH}/second.jpg', cv2.CV\_8UC1)  
 thresh = cv2.adaptiveThreshold(image, 128, adaptiveMethod=cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,  
 thresholdType=cv2.THRESH\_BINARY, blockSize=5, C=5)  
 cv2.namedWindow('Adaptive TRESH\_BINARY C=5', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('Adaptive TRESH\_BINARY C=5', 1280, 854)  
 cv2.imshow("Adaptive TRESH\_BINARY C=5", thresh)  
 thresh = cv2.adaptiveThreshold(image, 128, adaptiveMethod=cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,  
 thresholdType=cv2.THRESH\_BINARY\_INV, blockSize=5, C=5)  
 cv2.namedWindow('Adaptive TRESH\_BINARY\_INV C=5', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('Adaptive TRESH\_BINARY\_INV C=5', 1280, 854)  
 cv2.imshow("Adaptive TRESH\_BINARY\_INV C=5", thresh)  
 cv2.waitKey(0)  
 """ b """  
 thresh = cv2.adaptiveThreshold(image, 128, adaptiveMethod=cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,  
 thresholdType=cv2.THRESH\_BINARY, blockSize=5, C=0)  
 cv2.namedWindow('Adaptive TRESH\_BINARY C=0', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('Adaptive TRESH\_BINARY C=0', 1280, 854)  
 cv2.imshow("Adaptive TRESH\_BINARY C=0", thresh)  
 thresh = cv2.adaptiveThreshold(image, 128, adaptiveMethod=cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,  
 thresholdType=cv2.THRESH\_BINARY\_INV, blockSize=5, C=0)  
 cv2.namedWindow('Adaptive TRESH\_BINARY\_INV C=0', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('Adaptive TRESH\_BINARY\_INV C=0', 1280, 854)  
 cv2.imshow("Adaptive TRESH\_BINARY\_INV C=0", thresh)  
 cv2.waitKey(0)  
 thresh = cv2.adaptiveThreshold(image, 128, adaptiveMethod=cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,  
 thresholdType=cv2.THRESH\_BINARY, blockSize=5, C=-5)  
 cv2.namedWindow('Adaptive TRESH\_BINARY C=-5', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('Adaptive TRESH\_BINARY C=-5', 1280, 854)  
 cv2.imshow("Adaptive TRESH\_BINARY C=-5", thresh)  
 thresh = cv2.adaptiveThreshold(image, 128, adaptiveMethod=cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,  
 thresholdType=cv2.THRESH\_BINARY\_INV, blockSize=5, C=-5)  
 cv2.namedWindow('Adaptive TRESH\_BINARY\_INV C=-5', cv2.WINDOW\_KEEPRATIO)  
 cv2.resizeWindow('Adaptive TRESH\_BINARY\_INV C=-5', 1280, 854)  
 cv2.imshow("Adaptive TRESH\_BINARY\_INV C=-5", thresh)  
 cv2.waitKey(0)  
  
  
""" 14 задание """  
def fourteen():  
 image = cv2.imread(f'{PATH}/first.jpg')  
 borders = cv2.morphologyEx(image, cv2.MORPH\_GRADIENT, np.ones((3, 3)))  
 cv2.imshow("Borders", borders)  
 segment = cv2.pyrMeanShiftFiltering(image, 30, 25)  
 cv2.imshow("Default", image)  
 cv2.imshow("pyrMeanShiftFiltering", segment)  
 dst = cv2.addWeighted(borders, 0.5, segment, 0, 0)  
 cv2.imshow("Single Image with smoothing and with edges", dst)  
 cv2.waitKey(0)  
  
  
""" 15 задание """  
def fifteen():  
 image = cv2.imread(f'{PATH}/first.jpg')  
 kernel = np.array([[0,0,0,0,0,0,1,0,0,],  
 [0,0,0,0,0,0,0,0,0,],  
 [0,0,0,0,0,1,0,0,0,],  
 [0,0,0,0,0,0,0,0,0,],  
 [0,0,0,0,1,0,0,0,0,],  
 [0,0,0,0,0,0,0,0,0,],  
 [0,0,0,1,0,0,0,0,0,],  
 [0,0,0,0,0,0,0,0,0,],  
 [0,0,1,0,0,0,0,0,0]])  
  
 dst = cv2.filter2D(image, cv2.CV\_8U, kernel)  
  
 cv2.imshow("Deleting 60 degrees lines", dst)  
 cv2.waitKey(0)  
  
  
""" 16 задание """  
def sixteen():  
 image = cv2.imread(f'{PATH}/first.jpg')  
 cv2.imshow("Default", image)  
 gaus\_kernel = np.array([[1, 2, 1], [2, 4, 6], [1, 2, 1]]) / 16  
 """ a """  
 gaus\_image = cv2.filter2D(image, cv2.CV\_8U, gaus\_kernel)  
 cv2.imshow("Gaus", gaus\_image)  
 """ b """  
 kernel\_line = np.array([1, 2, 1]) / 4  
 kernel\_column = np.array([[1], [2], [1]]) / 4  
 line\_image = cv2.filter2D(image, cv2.CV\_8U, kernel\_line)  
 column\_line\_image = cv2.filter2D(line\_image, cv2.CV\_8U, kernel\_column)  
 cv2.imshow("Column-Line", column\_line\_image)  
 cv2.waitKey(0)  
 """ c """  
 """ Вместо 9 операций в a мы делаем 6 операций в b."""  
  
  
""" 18 задания """  
def eighteen():  
 mishen = cv2.imread(f'{PATH}/mishen.jpg')  
 mishen = cv2.resize(mishen, None, fx=0.8, fy=0.8)  
 """ a """  
 cv2.imshow("Mishen", mishen)  
 cv2.waitKey(0)  
 """ b """  
 d1\_3x3 = cv2.Sobel(mishen, -1, 1, 0, ksize=3)  
 d2\_3x3 = cv2.Sobel(mishen, -1, 2, 0, ksize=3)  
 d1\_5x5 = cv2.Sobel(mishen, -1, 1, 0, ksize=5)  
 d2\_5x5 = cv2.Sobel(mishen, -1, 2, 0, ksize=5)  
 d1\_9x9 = cv2.Sobel(mishen, -1, 1, 0, ksize=9)  
 d2\_9x9 = cv2.Sobel(mishen, -1, 2, 0, ksize=9)  
 d1\_13x13 = cv2.Sobel(mishen, -1, 1, 0, ksize=13)  
 d2\_13x13 = cv2.Sobel(mishen, -1, 2, 0, ksize=13)  
 cv2.imshow("Default", mishen)  
 cv2.imshow('dx1 3x3 and dx2 3x3', np.hstack((d1\_3x3, d2\_3x3)))  
 cv2.imshow('dx1 5x5 and dx2 5x5', np.hstack((d1\_5x5, d2\_5x5)))  
 cv2.imshow('dx1 9x9 and dx2 9x9', np.hstack((d1\_9x9, d2\_9x9)))  
 cv2.imshow('dx1 13x13 and dx2 13x13', np.hstack((d1\_13x13, d2\_13x13)))  
 cv2.waitKey(0)  
 """   
 Линия по оси x стирается, если дифференцировать по x, и по оси y если дифференцировать по y.   
 Но окружности в мешени стираются по оси y, если дифференцировать по x и по y, если дифференцировать по x.  
 """  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 while True:  
 print("Введите номер упражнения.")  
 flag = input('>>')  
 if flag == '12':  
 twelve()  
 if flag == '13':  
 thirteen()  
 if flag == '14':  
 fourteen()  
 if flag == '15':  
 fifteen()  
 if flag == '16':  
 sixteen()  
 if flag == '18':  
 eighteen()  
 if flag == 'exit':  
 break

**Images :**

first.jpg

second.jpg

mishen.jpg

