**Санкт-Петербургский государственный УНИВЕРСИТЕТ**

**Факультет прикладной математики и процессов управления**

**отчет**

**по лабораторной работе №1**

**по дисциплине «Функциональное программирование»**

**на тему «Параллельная обработка изобрабений »**

|  |  |  |
| --- | --- | --- |
| Студент гр. 22Б15 |  | Суворов Н.В. |
| Преподаватель |  | Киямов Ж.У. |

**Санкт-Петербург**

**2023 г.**

**Цель**

Разработка программы для параллельной обработки изображений с применением нескольких фильтров: увеличение резкости, сепия и изменение размера. Необходимо обеспечить безопасное сохранение обработанных изображений в выходную папку.

**Задача**

1. **Разработка архитектуры программы:**
   * Была разработана архитектура программы, позволяющая эффективно использовать многозадачность для обработки изображений.
2. **Обработка изображений с разными фильтрами:**
   * Каждое изображение было обработано тремя фильтрами: увеличение резкости, сепия и изменение размера.
3. **Поддержка параллельной обработки:**
   * Программа была разработана с поддержкой параллельной обработки с использованием потоков.
4. **Безопасное сохранение обработанных изображений:**
   * Обработанные изображения были сохранены в отдельной папке, обеспечивая безопасность данных.

**Теория**

Для реализации параллельной обработки были использованы потоки. Каждое изображение обрабатывалось отдельными потоками, что позволило эффективно использовать процессорное время.

# Рекомендации программиста

Для запуска программы необходима 64-битная операционная система Windows и Python версии не ниже 2.7. Для корректной работы программы рекомендуется использовать IDE PyCharm версии 2023.21 и pip install версии 23.1.0.

**Решение**

import os  
import threading  
import cv2  
import numpy as np  
  
#резкость  
def apply\_sharpening\_filter(input\_path, output\_path):  
 image = cv2.imread(input\_path)  
 kernel = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])  
 sharpened = cv2.filter2D(image, -1, kernel)  
 cv2.imwrite(output\_path, sharpened)  
  
#степень  
def apply\_sepia\_filter(input\_path, output\_path):  
 image = cv2.imread(input\_path)  
 kernel = np.array([[0.272, 0.534, 0.131],  
 [0.349, 0.686, 0.168],  
 [0.393, 0.769, 0.189]])  
 sepia\_toned = cv2.transform(image, kernel)  
 cv2.imwrite(output\_path, sepia\_toned)  
  
#уменьшение размера  
def resize\_image(input\_path, output\_path, width, height):  
 image = cv2.imread(input\_path)  
 resized = cv2.resize(image, (width, height))  
 cv2.imwrite(output\_path, resized)  
  
  
def process\_images(input\_folder, output\_folder):  
 for filename in os.listdir(input\_folder):  
 if filename.endswith('.jpg') or filename.endswith('.JPG'):  
 input\_path = os.path.join(input\_folder, filename)  
  
  
 sharpened\_output\_path = os.path.join(output\_folder, f'sharpened\_{filename}')  
 sepia\_output\_path = os.path.join(output\_folder, f'sepia\_{filename}')  
 resized\_output\_path = os.path.join(output\_folder, f'resized\_{filename}')  
  
 sharpening\_thread = threading.Thread(target=apply\_sharpening\_filter, args=(input\_path, sharpened\_output\_path))  
 sepia\_thread = threading.Thread(target=apply\_sepia\_filter, args=(input\_path, sepia\_output\_path))  
 resize\_thread = threading.Thread(target=resize\_image, args=(input\_path, resized\_output\_path, 800, 600))  
  
 sharpening\_thread.start()  
 sepia\_thread.start()  
 resize\_thread.start()  
  
 sharpening\_thread.join()  
 sepia\_thread.join()  
 resize\_thread.join()  
  
  
input\_folder = 'C:/Users/snw12/PycharmProjects/FP-1/foto'  
output\_folder = ('C:/Users/snw12/PycharmProjects/FP-1/1')  
process\_images(input\_folder, output\_folder)

**Пример**

Запустив программу и считав изображения из папки ./foto(рис.1), мы получим соответствующий результат в папке ./1 (рис. 2). Это и является доказательством корректной работы программы.

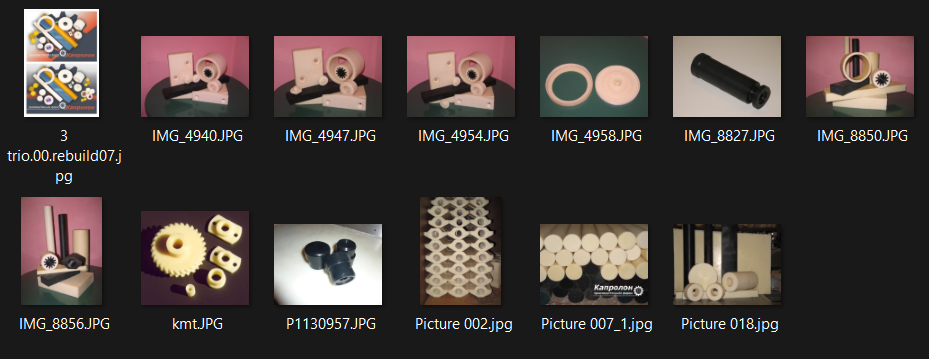


Рис.1

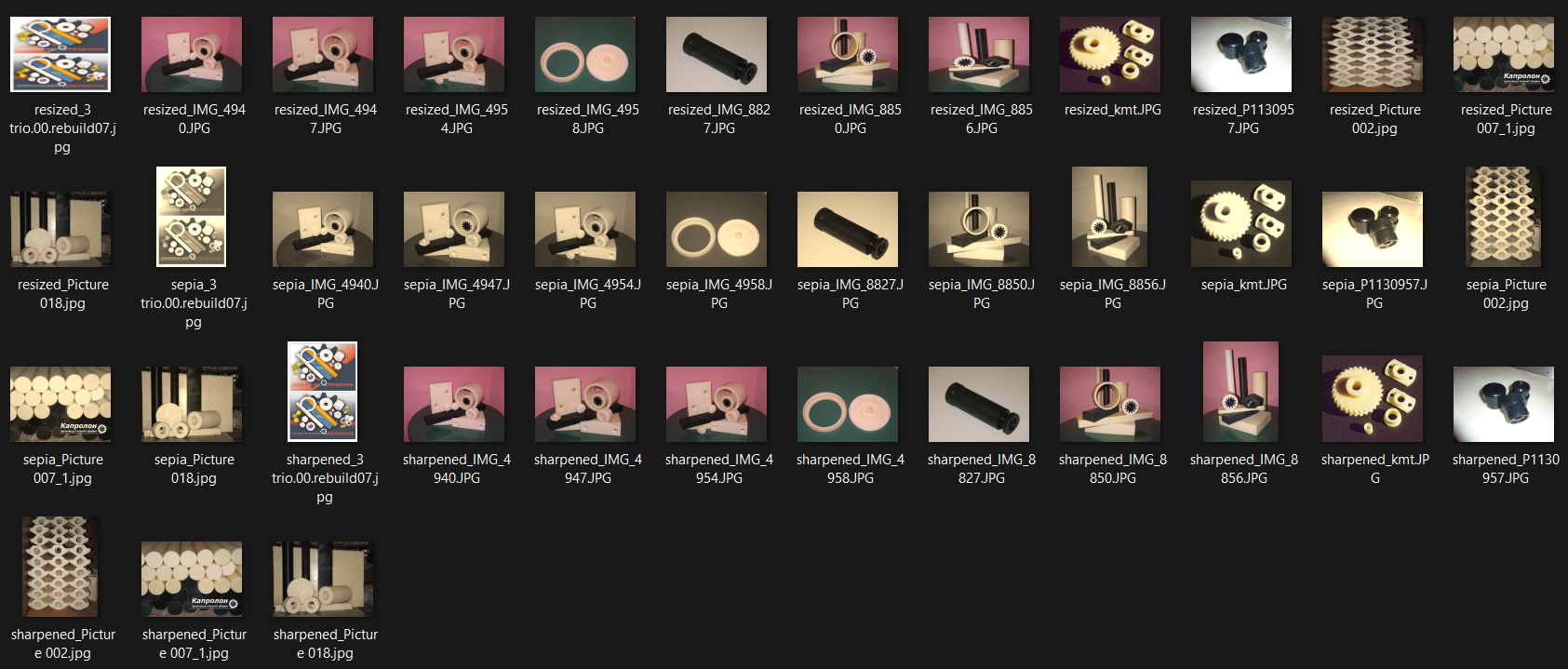


Рис.2

**Вывод**

Реализованная программа успешно обрабатывает изображения с применением нескольких фильтров параллельно. Программа позволяет эффективно использовать ресурсы и избегать конфликтов при параллельной обработке данных.