

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»

КАФЕДРА Инженерной кибернетики

НАПРАВЛЕНИЕ Прикладная математика ГРУППА БПМ-16-2

КУРСОВАЯ РАБОТА

по курсу **Технологии программирования**

Тема: Цепочки фильтров для изображений

Студент Малышковский Олег Витальевич

Руководитель Полевой Дмитрий Валерьевич

Оценка выполнения курсовой работы _____

Москва 2017г.

Оглавление:

1. Введение
2. Технические задание
3. Реализация программы
 - 3.1 Интерфейс программы
 - 3.2 Реализация алгоритмов и методов
 - 3.3 Пояснение к заданию параметров фильтров
4. Инструкция по сборке

1. Введение

Необходимо реализовать пользовательское приложение, которое может по инструкциям из конфигурационного файла многократно производить фильтрацию для заданного набора изображений и сохранять их.

2. Техническое задание

Сама программа должно представлять собой консольное приложение, как некий конвейер по фильтрации входных данных. Можно будет запускать непосредственно само приложение или осуществить его запуск через командную строку. Программа должна получать на вход конфигурационный файл (JSON файл), в котором согласно определённой семантике, будет описано: откуда взять исходные данные, куда их сохранить, какие действия с ними сделать.

Синтаксис конфигурационного файла следующий:

```
{  
  "action": "...",  
  "src": "...",  
  "dst": "...",  
  "settings": {"...": ..., "...": ..., "...": ...}  
}
```

action – применяемый фильтр

src – месторасположение искомого изображения

dst – адрес сохранения отфильтрованных данных

settings – дополнительные параметры, необходимые для работы фильтра

Для получения цепочки фильтров конфигурационный файл составляется следующим образом:

```
[  
  {...}, //Описание 1-го фильтра  
  {...}, //Описание 2-го фильтра  
  {...},  
  ...  
]
```

Программа должна произвести разбор инструкций из конфигурационного файла и согласно им осуществить цепочки фильтров.

Программа должна уметь осуществлять следующие виды фильтров:

resize – изменение размера

dilation – дилатация

erosion – эрозия

smoothing - размытие

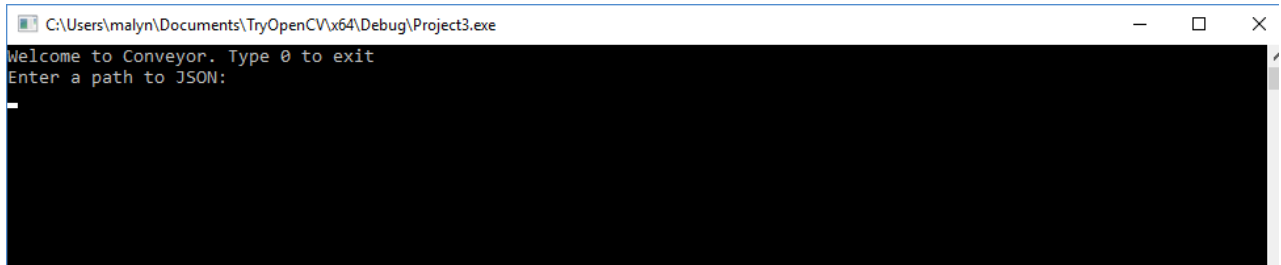
Сохранение происходит в папку, указанную в конфигурационном файле. Если по указанному адресу папки не существует, то программа должно её создать. Если не указать адрес ("src:"), то данные сохраняются в специальную папку save, которую программа при необходимости создаст в каталоге с собой.

Если происходит запуск непосредственно самого исполняемого файла, то в этом случае конвейер будет работать до тех пор, пока на вход вместо ссылки на конфигурационный файл не получит 0. В таком случае программа завершает выполнение. Если будет выполнен запуск из под командой строки, то конвейер выполнить обработку 1 раз (при этом можно выполнить цепочку фильтров, но это будет выполнено только над один набором данных один раз).

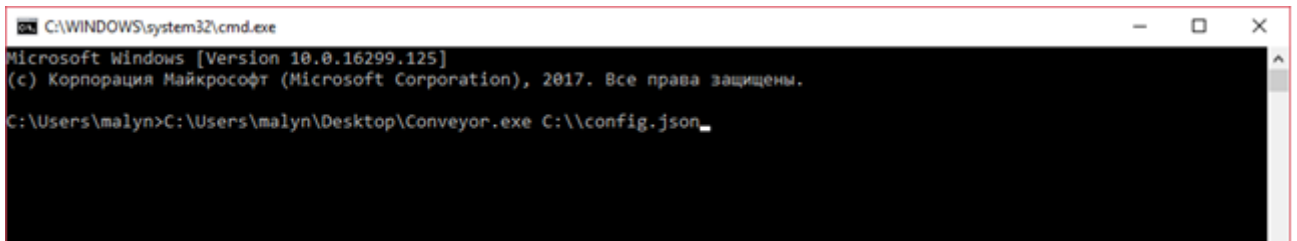
3. Структура программы

Интерфейс программы

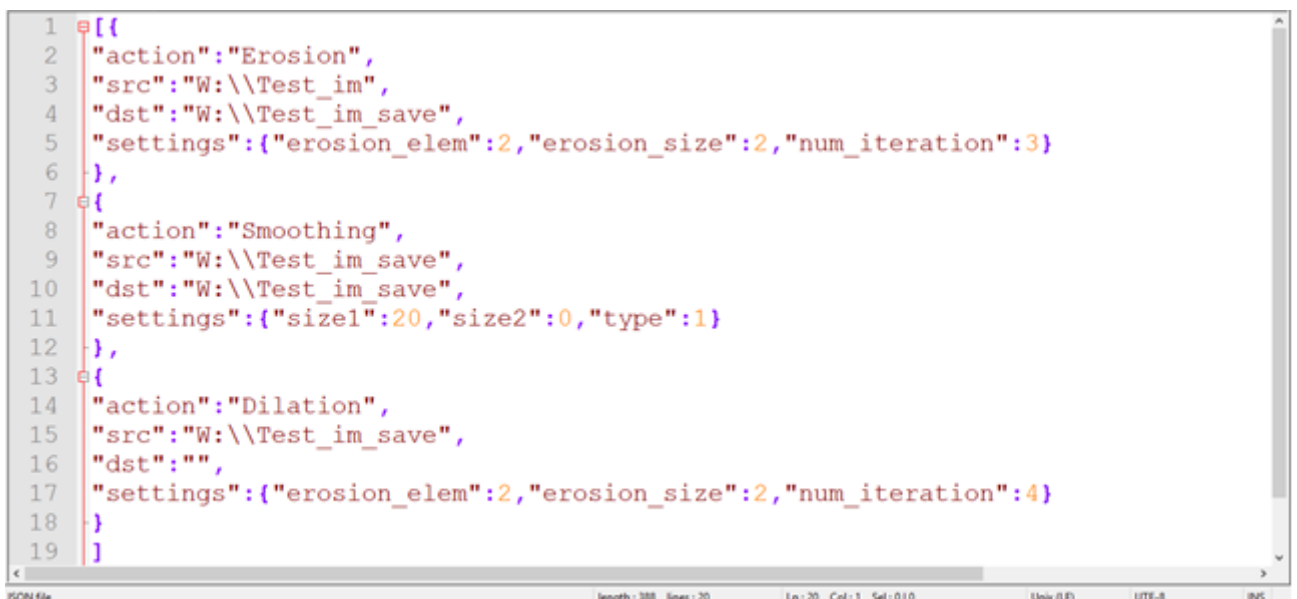
При запуске пользователю предлагается указать путь к конфигурационному файлу (также напоминает что для завершения работы нужно ввести 0).



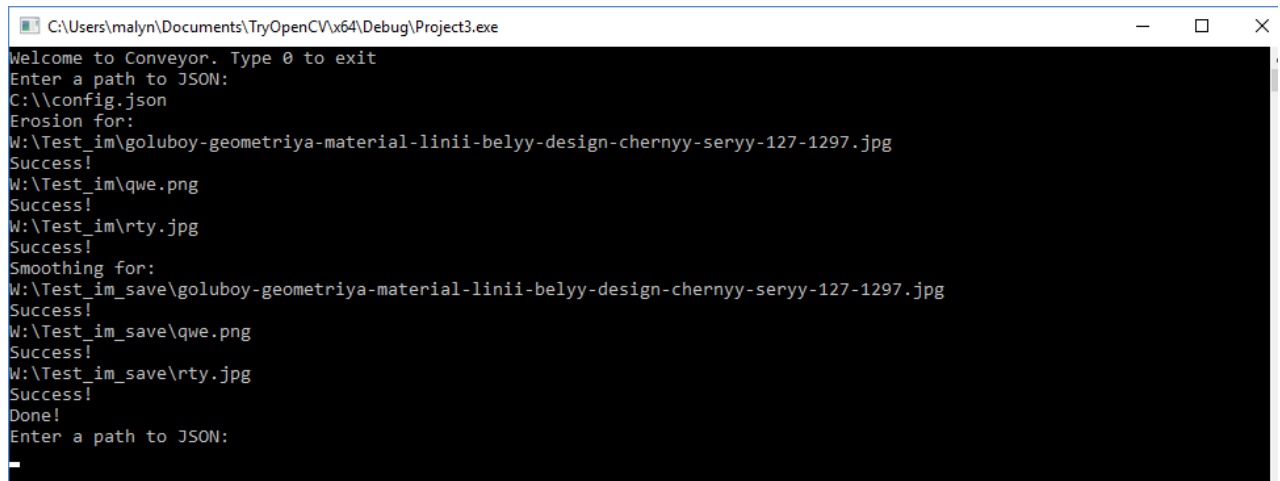
Если запускать программу из под командной строки необходимо через пробел указать входные данные- ссылку на конфигурационный файл (чтобы не указывать путь к исполняемому файлу, для удобства можно перетащить мышкой приложение в окно командной строки и ссылка на него будет автоматически указана).



Зададим следующий конфигурационный файл с цепочкой фильтров из Erosion, Smoothing и укажем программе путь к нему. Путь указывается с использованием \\

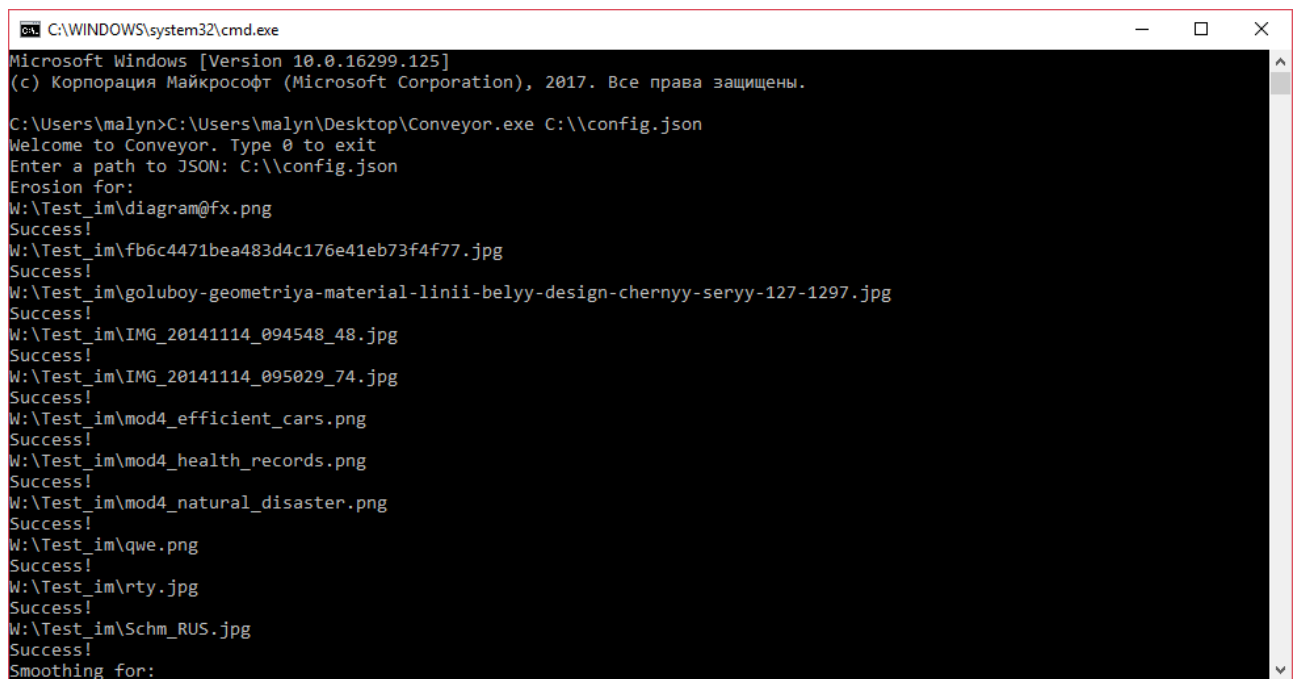


Программа успешно считала конфигурационный файл и выполнила цепочку фильтров.



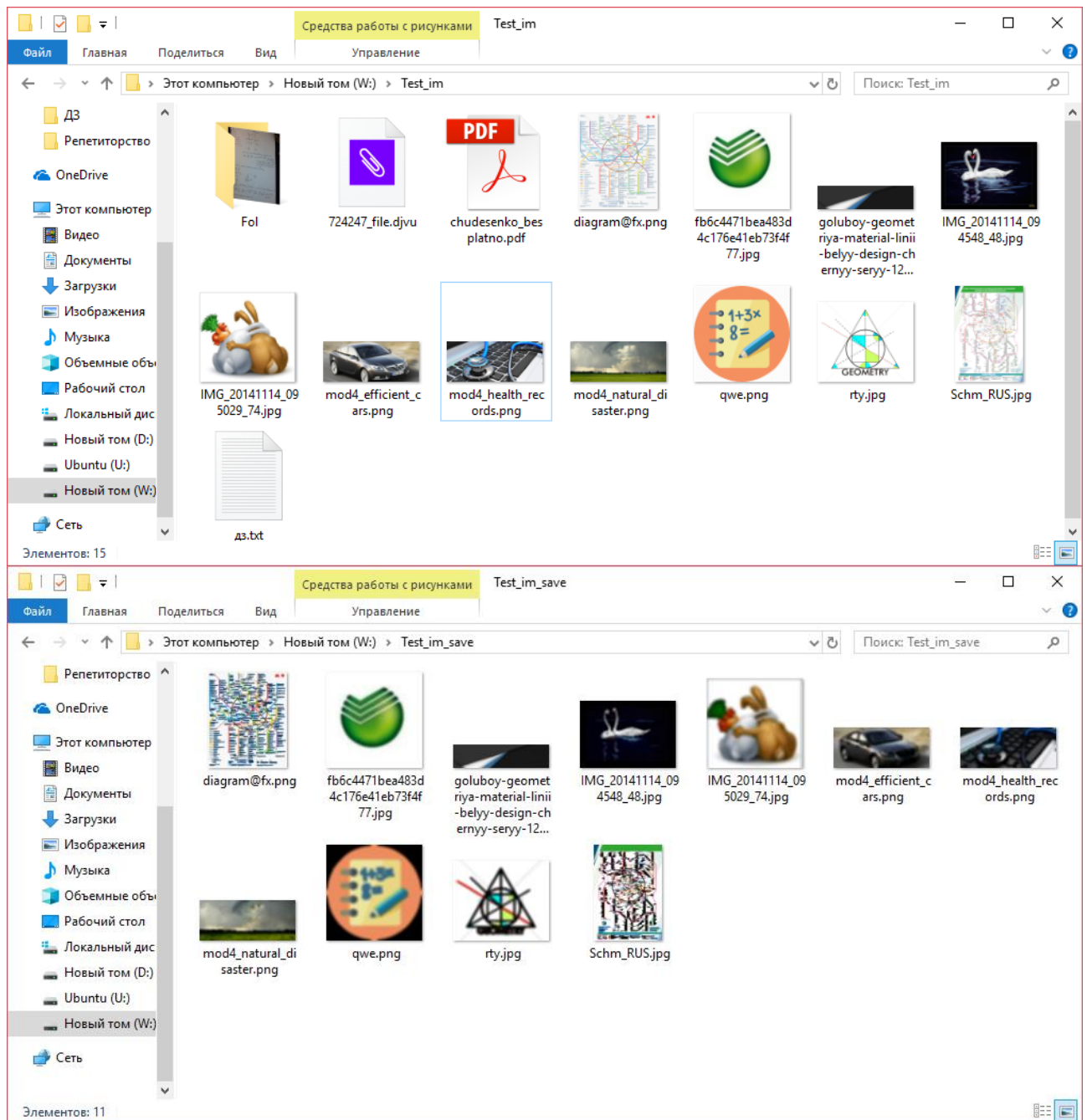
```
C:\Users\malyn\Documents\TryOpenCV\64\Debug\Project3.exe
Welcome to Conveyor. Type 0 to exit
Enter a path to JSON:
C:\\config.json
Erosion for:
W:\\Test_im\\goluboy-geometriya-material-linii-belyy-design-chernyy-seryy-127-1297.jpg
Success!
W:\\Test_im\\qwe.png
Success!
W:\\Test_im\\rty.jpg
Success!
Smoothing for:
W:\\Test_im_save\\goluboy-geometriya-material-linii-belyy-design-chernyy-seryy-127-1297.jpg
Success!
W:\\Test_im_save\\qwe.png
Success!
W:\\Test_im_save\\rty.jpg
Success!
Done!
Enter a path to JSON:
```

Из консоли:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.16299.125]
(c) Корпорация Майкрософт (Microsoft Corporation), 2017. Все права защищены.

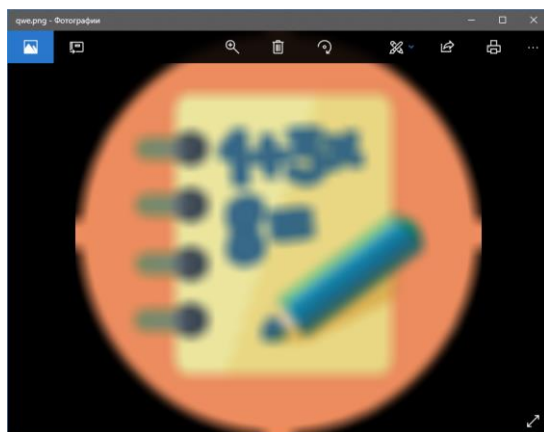
C:\Users\malyn>C:\Users\malyn\Desktop\Conveyor.exe C:\\config.json
Welcome to Conveyor. Type 0 to exit
Enter a path to JSON: C:\\config.json
Erosion for:
W:\\Test_im\\diagram@fx.png
Success!
W:\\Test_im\\fb6c4471bea483d4c176e41eb73f4f77.jpg
Success!
W:\\Test_im\\goluboy-geometriya-material-linii-belyy-design-chernyy-seryy-127-1297.jpg
Success!
W:\\Test_im\\IMG_20141114_094548_48.jpg
Success!
W:\\Test_im\\IMG_20141114_095029_74.jpg
Success!
W:\\Test_im\\mod4_efficient_cars.png
Success!
W:\\Test_im\\mod4_health_records.png
Success!
W:\\Test_im\\mod4_natural_disaster.png
Success!
W:\\Test_im\\qwe.png
Success!
W:\\Test_im\\rty.jpg
Success!
W:\\Test_im\\Schm_RUS.jpg
Success!
Smoothing for:
```



До обработки:



После:



Реализация алгоритмов и методов

К программе подключены две сторонние библиотеки: OpenCV и Jsoncpp.

Jsoncpp – библиотека для работы с конфигурационными файлами в формате JSON.

OpenCV – библиотека компьютерного зрения. Она лежит в основе всех реализуемых фильтров и при её помощи происходят все манипуляции с самими картинками.

При получении конфигурационного файла есть ссылка на сам каталог, но нет полных имен лежащих в ней файлов, их нужно получить. Есть заголовок `filesystem` для доступа к классам и функциям, обработки и получения сведений о пути, файлов и каталогов. Он содержит класс `directory_iterator`, который описывает итератор ввода, выполняющий последовательный перебор имен файлов в каталоге файловой системы.

```
#include "jsoncpp-master\dist\json\json.h"
#include "jsoncpp-master\dist\json\json-forwards.h"
#include <filesystem>
namespace fs = std::experimental::filesystem;
std::ifstream config_doc(pathJSON, std::ifstream::binary);
Json::Value root;
config_doc >> root;
std::vector<std::string> ref_to_files;
for (auto& p : fs::directory_iterator(root[i]["src"].asString())){
    if (IsImage(p.path().string()))
        ref_to_files.insert(ref_to_files.end(), p.path().string());
}
```

Таким образом полные ссылки на все картинки в каталоге записываются в массив ссылок. `IsImage` – функция для проверки, что отдельно взятый путь указывает на изображение, а не каталог или файл с иным расширением.

Для выбора конкретного метода используется инструкция `switch`, но так как она не принимает `string`, придётся связать названия фильтров с числовым значением. Используем в этих целях класс `map` стандартной библиотеки.

```
typedef std::map <const std::string, int> ImageMap;
ImageMap IM{ { "Erosion",0 }, { "Dilation",1 }, { "Smoothing",2 }, {...}...};
switch (IM.at(root[i]["action"].asString()))
{
    case 0:
        for (int index(0); index < ref_to_files.size(); index += 1)
            Erosion(...);
        break;
    case 1:
        for (int index(0); index < ref_to_files.size(); index += 1)
            Dilation(...);
        break;
    . . .
    default:
        break;
}
```

Теперь можно послать в качестве параметра для фильтра ссылку на конкретную картинку, но пока нельзя сказать как её сохранить, имея лишь путь к каталогу для сохранения. Поэтому из ссылки для взятие изображения берут имя файла с его расширением и добавляют его к ссылке на каталог, получая полный путь для сохранения. После этого возможно сохранить результат фильтрации.

Поэтому введена функция `GetFileName` для разбора этих строк.

```
std::string Conveyor::GetFileName(const char* source,const char* save)
{
    std::string fileName("");
    int i = std::string(source).length() - 1;
    while (std::string(source)[i--] != '\\');
    while (i++ != std::string(source).length())
        fileName += std::string(source)[i];
    if (std::string(save).empty())
        save = "save\\";
    if (!fs::exists(std::string(save)))
        fs::create_directory(std::string(save));
    return std::string(save)+fileName;
}
```

Пример фильтра выполняющего эрозию изображения:

```
void Conveyor::Erosion(const char* source,const char* save,const int&
erosion_elem,const int& erosion_size, const int& iterate)
{
    using namespace cv;
    std::cout << source << std::endl;
    IplImage* input = cvLoadImage(source, 1);
    int erosion_type;
    if (erosion_elem == 0) { erosion_type = CV_SHAPE_RECT; }
    else if (erosion_elem == 1) { erosion_type = CV_SHAPE_CROSS; }
    else if (erosion_elem == 2) { erosion_type = CV_SHAPE_ELLIPSE; }
    IplConvKernel *structure_element;
    structure_element = cvCreateStructuringElementEx((2 * erosion_size + 1),
(2 * erosion_size + 1), erosion_size, erosion_size, erosion_type);
    IplImage* input1(input);
    cvErode(input, input1, structure_element, iterate);
    cvSaveImage(GetFileName(source,save).c_str(), input1);
    std::cout << "Success!" << std::endl;
}
```

Пояснения к заданию параметров фильтров

Erosion:

int erosion_elem - от 0 до 2, определяет тип эрозии

(Если 0, то CV_SHAPE_RECT, 1 - CV_SHAPE_CROSS, 2 - CV_SHAPE_ELLIPSE)

int erosion_size - определяет ядро преобразования

int iterate - количество проходов

Dilation:

int dilation_elem - от 0 до 2, определяет тип дилатации

(Если 0, то CV_SHAPE_RECT, 1 - CV_SHAPE_CROSS, 2 - CV_SHAPE_ELLIPSE)

int dilation_size - определяет ядро преобразования

int iterate - количество проходов

Smoothing:

int size1 - ширина апертуры

int size2 - высота апертуры

int smtype - от 0 до 4 определяет тип сглаживания (0 - простое с масштабированием и 1 - без, 2 - по Гауссу, 3 - медианное, 4 - двустороннее)

Resize:

int interpolation - от 0 до 3, определяет тип (0 - метод ближайшего соседа; 1 - билинейная (по умолчанию); 2 - бикубическая; 3 - интерполяция на основе соотношений площади пикселей.)

int size1 - ширина получаемого изображения

int size2 - его высота

4. Инструкция по сборке

1. Необходимо установить библиотеку OpenCV 3.2.0

Для Windows: В «Переменные среды» добавляем во-первых новую переменную OpenCV на путь ...\\bin (куда был распакован архив) и добавляем в переменную Path тот же путь (тут в конце знак ; и перед вставкой тоже должен он быть).

Заходим в свойства проекта по нажатию правой клавиши. Выставляем Конфигурация: все конфигурации и Платформа: все платформы.

Открываем каталоги VC++, вкладку «каталоги исполняемых файлов». Нажимаем изменить. Внутри пишем ...\\vc14(куда был распакован архив).

Переходим в C/C++, Общие, Дополнительные каталоги включаемых файлов, изменить и аналогично прошлому пункту добавить туда ...\\opencv\\build\\include

Переходим в Компоновщик, Общие, Дополнительные каталоги библиотек и добавляем ...\\lib

Переходим в Компоновщик, Ввод, Дополнительные зависимости. Там указываем opencv_world320.lib

Переходим в каталоги VC++, каталоги библиотек. Указываем ...\\lib. Применяем все написанные выше настройки нажатием кнопки «Ок».

В коде:

```
#include <opencv\\cv.h>
#include <opencv2\\opencv.hpp>
#include <opencv\\highgui.h>
#include <opencv\\cxcore.h>
```

2. Необходимо установить библиотеку Jsoncpp 1.8.0

Инструкция по ссылке:

<https://github.com/open-source-parsers/jsoncpp/wiki/Amalgamated>

3. Исходные тексты лежат в папке project в следующем репозитории:

https://mysvn.ru/malynkovsky/malynkovsky_o_v/