Использование Doxygen для работы с исходным программным кодом ПО ГИС

Обсудить в форуме Комментариев — 0

Эта страница опубликована в основном списке статей сайта
по адресу http://gis-lab.info/qa/doxygen.html

Одна из методик работы с большими объёмами чужого кода, ориентации в нём и использовании его составных частей в своих программах.

Содержание

- 1 Введение
- 2 Устанавливаем Doxygen
- 3 Разбор кода
- 4 Результаты

Введение

Решение задачи написания ПО начинается с определения проблемы и постановки задач. Затем, в общем случае, следуют выработка требований, детальное проектирование, кодирование и отладка, тестирование и, наконец, сопровождение. Перед тем, как приступать к кодированию разработчик должен знать, что он будет писать, с помощью чего, как это должно себя вести и в каком виде будет использоваться. Поэтому стадия проектирования весьма важна. В том числе, потому что на ней необходимо определить возможность использования чужого кода, либо своего написанного до этого, а так же обдумать проблемы его интеграции в собственный проект.

На сегодняшний день существует огромное количество программного кода разной тематики, который доступен под разными лицензиями в сети и может заметно облегчить жизнь программиста, начинающего работу с задачами, связанными с ГИС.

Рассмотрим пример: допустим возник вопрос написания ПО для векторизации специфичного типа карт, среди прочего понадобится иметь на руках код, который должен осуществлять привязку карты. В сфере ГИС существует несколько открытых проектов, где эта задача уже решена, например GRASS. GRASS распространяется под лицензией GPL и каждый желающий может изучить его исходный код и, при желании, использовать их в своем программном обеспечении в соответствии с условиями лицензии (внимательно ознакомтесь с ней, прежде чем использовать данный код).

Устанавливаем Doxygen

GRASS изначально создавался для Unix платформ и его компиляция для Windows непростая задача (про установку GRASS). Изучив описание, убедимся, что в программе присутствует необходимый код. Скачав исходники легко обнаружить, что их объём составляет более 560 тыс. строк что заставляет задуматься о более эффективном их изучении.

Как ориентироваться в таком объёме чужого кода? А тем более получить оттуда отдельные участки. Надо заметить, что код написан на чистом C и это усугубляет дело, т.к. процедурный и модульный подходы в программировании делают разбор чужого зачастую достаточно сложным процессом. Пойдем наработанным путём и воспользуемся программным пакетом Doxygen(www.stack.nl/~dimitri/doxygen/).

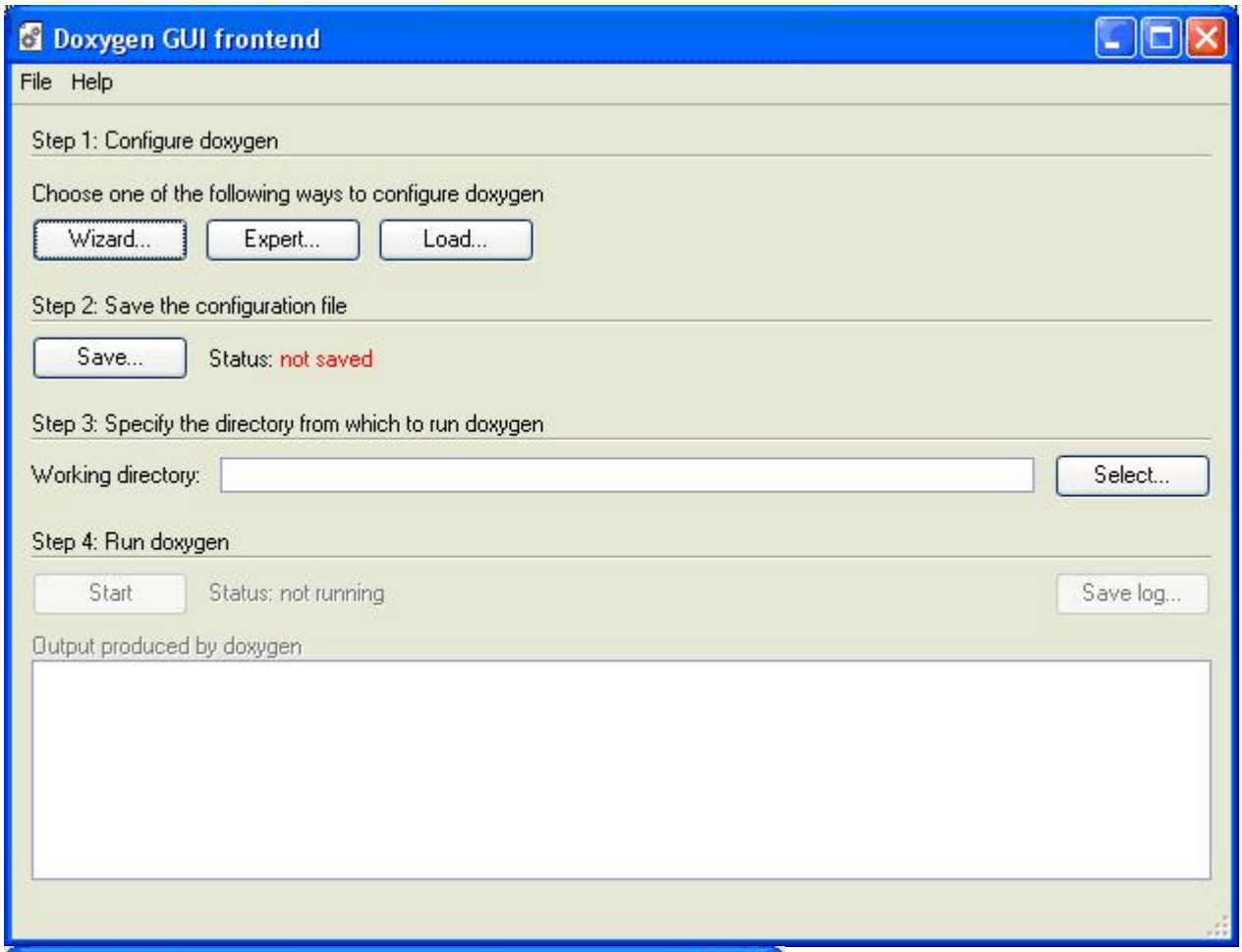
Doxygen — это кросс-платформенная система документирования исходных текстов, которая поддерживает C++, Java, IDL и, частично, PHP, C#. Doxygen генерирует документацию на основе набора исходных текстов и также может быть настроен для извлечения структуры программы из недокументированных исходных кодов.

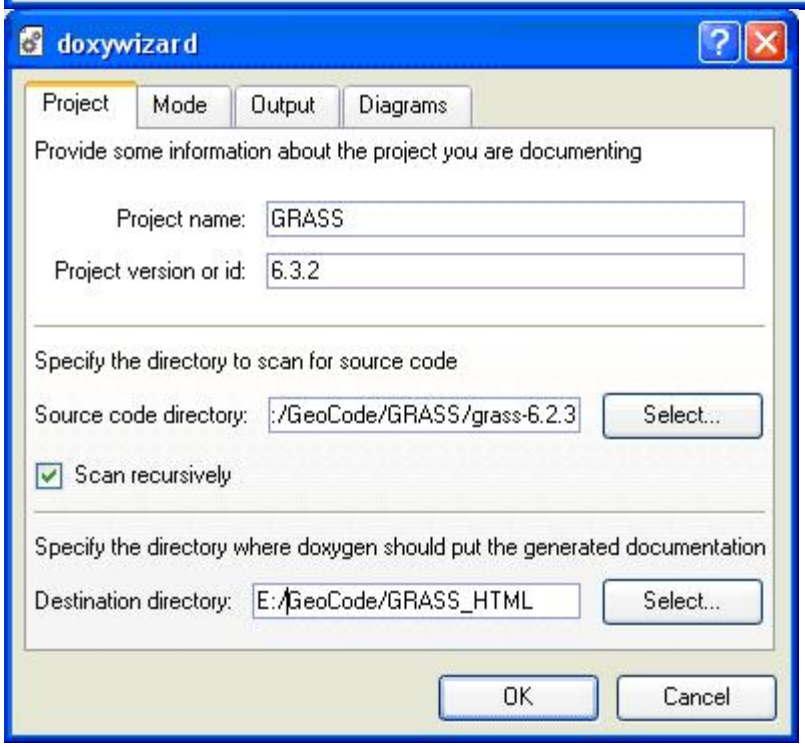
Возможно составление графов зависимостей программных объектов, диаграмм классов и исходных кодов с гиперссылками. Doxygen имеет встроенную поддержку генерации документации в формате HTML, , man, RTF и XML. Также вывод может быть легко сконвертирован в CHM, PostScript, PDF (подробнее о программе).

Скачаем и установим Doxygen для Windows, имеющий графический интерфейс пользователя.

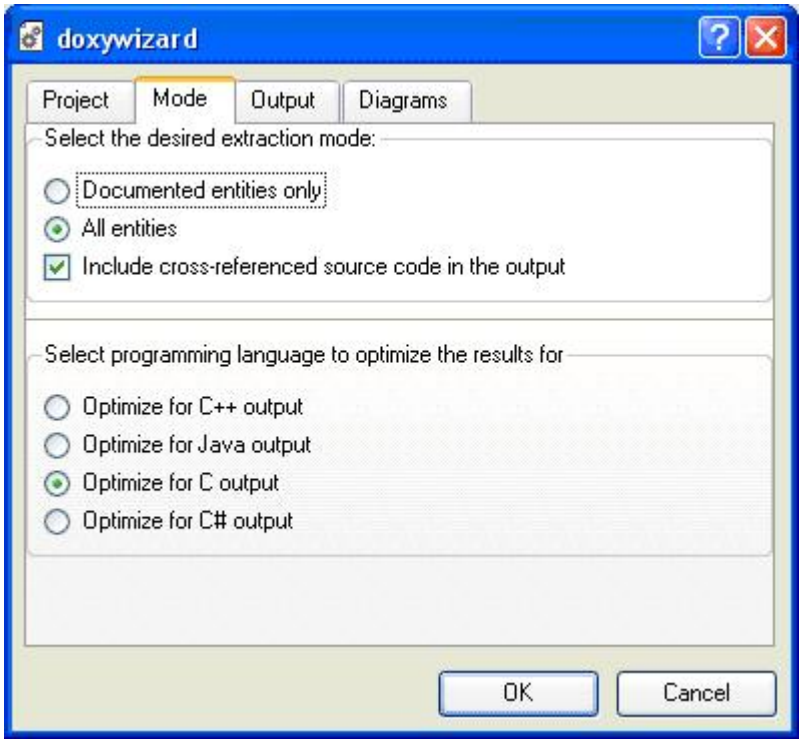
Разбор кода

Воспользуемся для начала работы Помощником (Wizard).

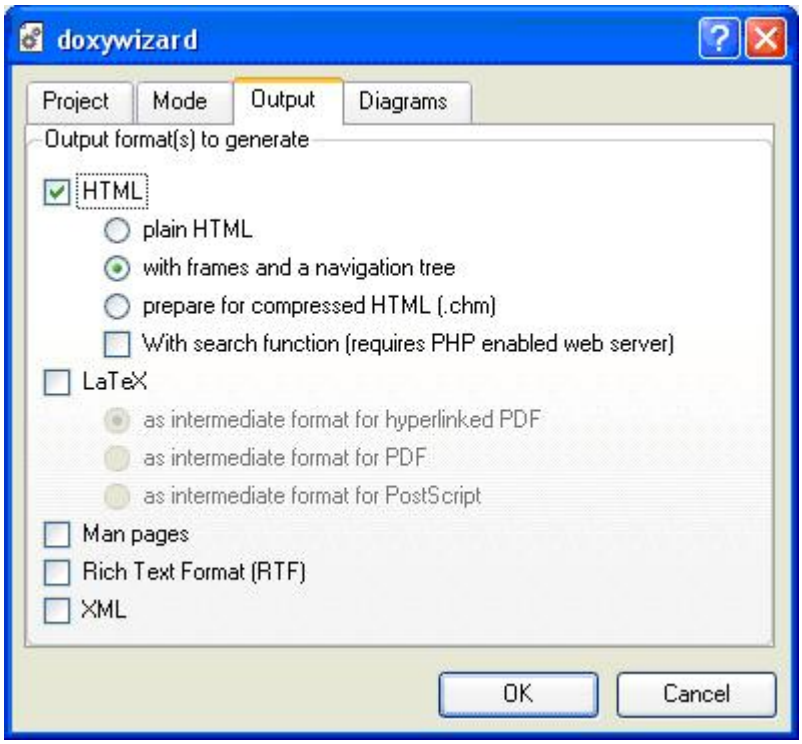




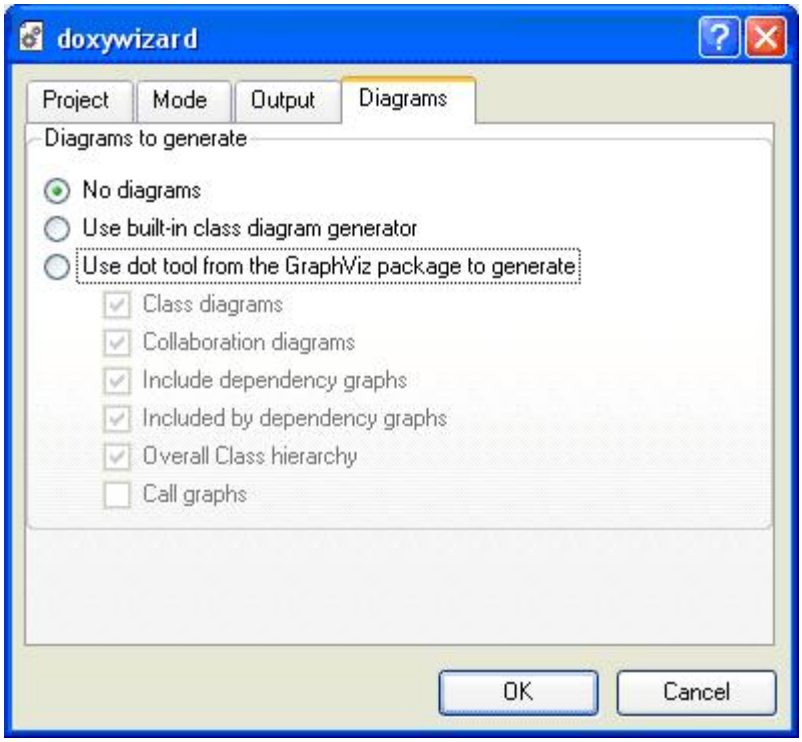
Значение параметров на вкладке «Project» вопросов вызвать не должно, заполним ее в соответствии со иллюстрацией и, переходим на вкладку «Mode». Здесь мы говорим программе обрабатывать весь код и указываем язык, на котором написан наш проект (GRASS написан на C).



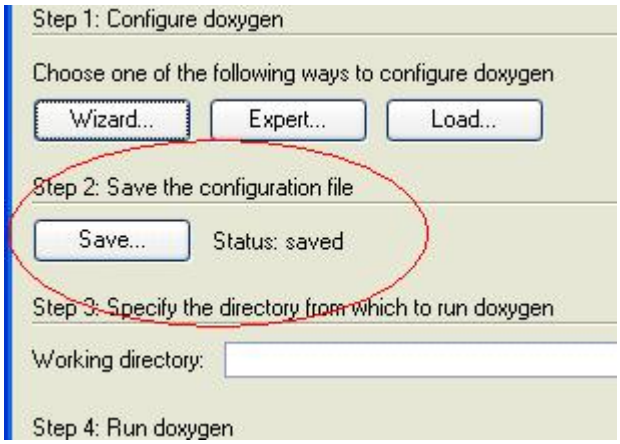
Следующий этап – настройка результирующих данных. В данном случае выбрана только генерация html-файлов с навигационным деревом и встроенными фреймами, но можно настроить вывод в том формат, который более удобен вам.



Здесь, опять же, на ваше усмотрение.

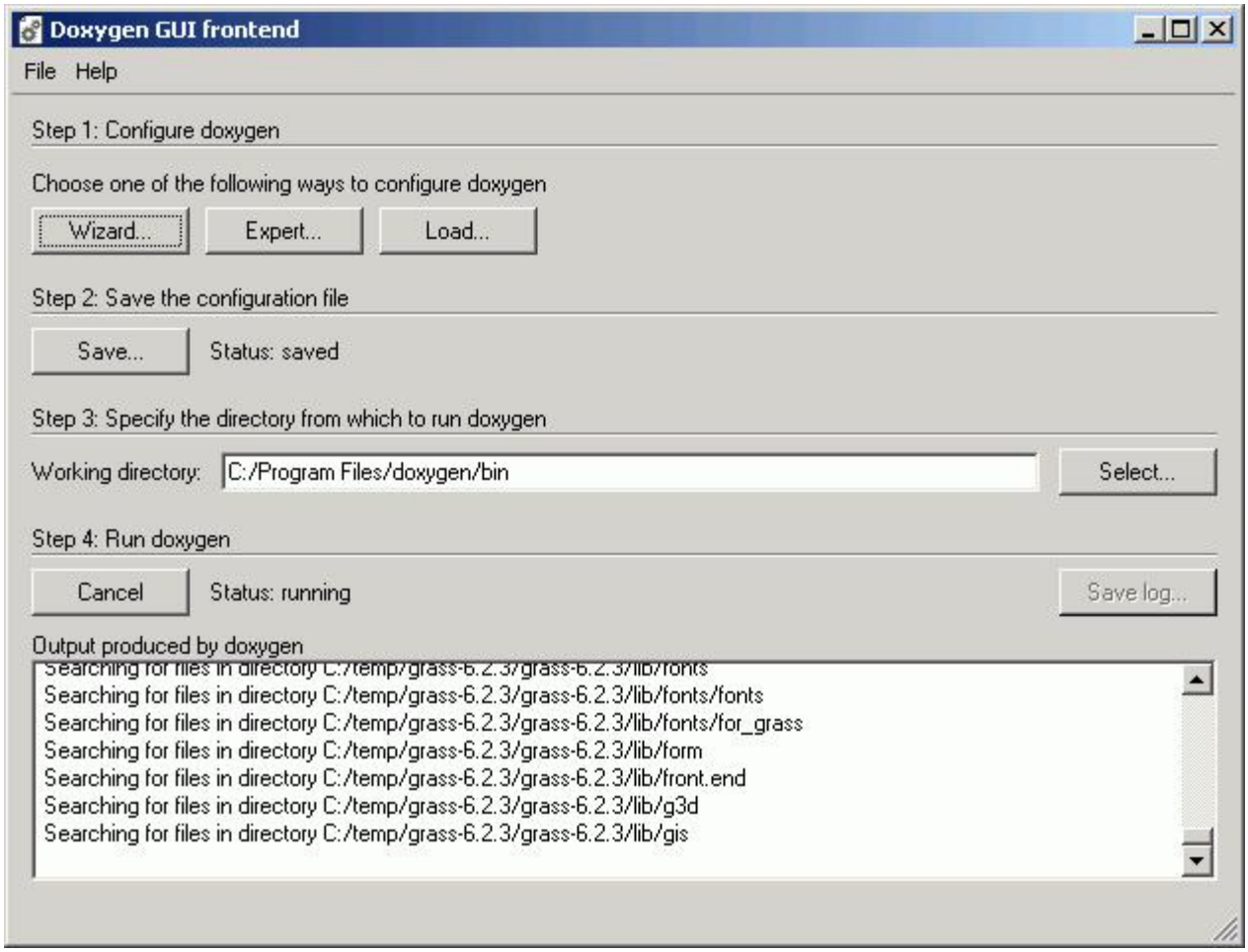


Нажимаем «OK» и сохраняем проект:



В «Step 3» нужно указать папку, откуда должен быть загружен Doxygen (например C:\Program Files\doxygen\bin\)

Затем переходим к четвертому шагу и нажимаем «Start». Процесс обработки достаточно длительный и может занять несколько минут.



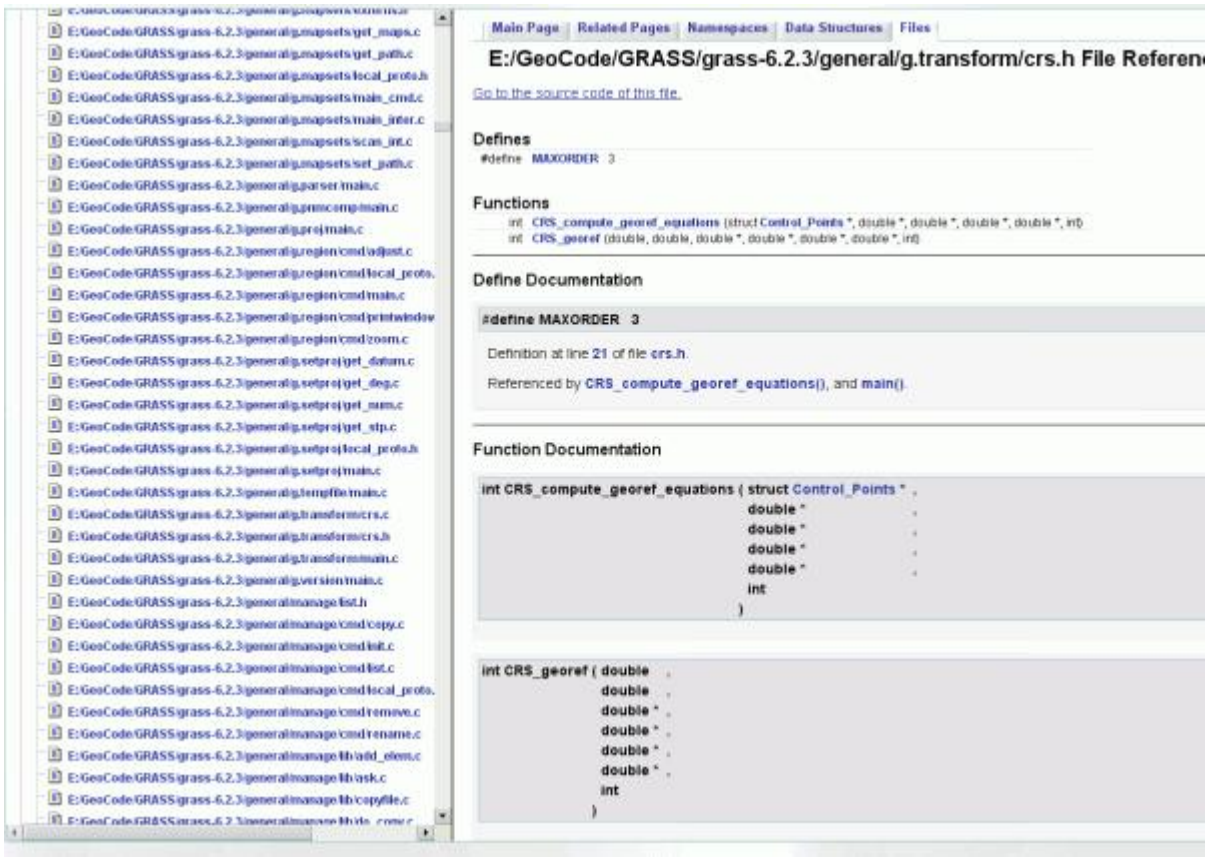
После окончания, в папке, указанной на закладке «Project» в пункте «Destination directory», появится каталог html, в котором необходимо найти файл index.html и запустить. Вы должны увидеть нечто вроде:

Doxygen, помимо того, что обработал исходники, так же нашёл документацию и включил её в общую структуру. Среди прочего, он ищет папки с именами docs/doc/manual/etc (список настраивается в экспертном режиме работы), берёт оттуда файлы и включает их в общую структуру. Кстати, сама документацию GRASS обработана так же с помощью этой программы.

Далее встаёт вопрос поиска необходимого кода, в этом может помочь чтение документации и изучение соответствующих функций программы. В нашем случае, зная, что команда трансформации называется

g.transform, ищем соответствующий ей код: grass-6.2.3/general/g.transform/crs.c|h. В данном случае сложно найти общий подход, как правило локализовать необходимый код.

Раскрываем слева в дереве пункт «File list» и ищем необходимые файлы.



Название каждой структуры, функций и констант является ссылкой, поэтому общий подход далее состоит в изучении кода этого модуля и вытаскивании необходимых функций и типов в отдельный проект. GRASS написан профессионалами, код хорошо отформатирован и имеет комментарии, поэтому его анализ не составит большого труда.

Результаты

Оформив найденный код в отдельный класс, получаем простой и удобный способ привязки карты:

```
int _tmain(int argc, _TCHAR* argv[])
{
    // Создание класса привязки
    GeoRef::GeoReferencing georef;

    // Добавление контрольных точек
    georef.AddControlPoint(448, 942, 103.0, 53.3333);
    georef.AddControlPoint(1761, 938, 103.5, 53.3333);
    georef.AddControlPoint(444, 3871, 103.0, 52.6667);
    georef.AddControlPoint(3072, 928, 104.0, 53.3333);
    georef.AddControlPoint(1774, 3870, 103.5, 52.6667);
    georef.AddControlPoint(3104, 3858, 104.0, 52.6667);

    // Вычисление коэффициентов (решение системы уравнений)
    georef.CalcCoef(2);

    // Пример использования
    double Lat,Lon;
    georef.TransformPoint(1106, 2403, &Lon, &Lat, 2);
    cout << setprecision(16) << Lat << " " << Lon << endl;

    return 0;
```

```
}
```

В приложении к статье Вы можете найти данный класс и пример его использования в MSVC++ 8.0.

С собственно математикой процесса расчетов коэффициентов для полиномиального преобразования с минимальным количество точек можно ознакомиться в специальной статье. Другие примеры реализации и проверки преобразования так же доступны в соответствующих заметках.

Обсудить в форуме Комментариев — 0

Последнее обновление: 2014-05-15 00:46

Дата создания: 04.04.2008
Автор(ы): Александр Грайвер