МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
"Национальный исследовательский ядерный университет "МИФИ"

ФАКУЛЬТЕТ КИБЕРНЕТИКИ

КАФЕДРА ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему:

Разработка модуля преобразования геоданных из формата OpenStreetMap в формат OpenGIS

Группа К7-361

Студент _____ (Лаврентьева М.О.)
(подпись) (Ф.И.О.)

Руководитель _____ (Муравьёв С.К.)
(подпись) (Ф.И.О.)

Оценка _____

Члены комиссии _____ ( )
(подпись) (Ф.И.О.)

_____ ( )
(подпись) (Ф.И.О.)

_____ ( )
(подпись) (Ф.И.О.)

_____ ( )
(подпись) (Ф.И.О.)

Москва 2011

Литература

- Львовский С.М. Набор и верстка в пакете LaTeX - 3-е изд. Электронное издание, 2003.
- Сандра Э.Эдди XML Справочник, изд. 'ПИТЕР', 1999, 480с.
- Конспект лекций по теории программирования C++.
- Бланшет Ж., Саммерфилд М. Qt4 программирование GUI на C++ - 2-е изд. Изд. 'КУДИЦ-ПРЕСС', Москва, 2008, 736 с.
- Документация PostgreSQL. http://postgresql.ru.net/docs.html
- Документация XML

Содержание

1 Введение

В настоящее время широкое распространение в информационных технологиях получили геоинформационные системы (ГИС). ГИС представляет собой аппаратно-программный человеко-машинный комплекс, обеспечивающий сбор, обработку, отображение и распространение геоданных.

Понятие геоданные (пространственные данные, географические данные) включает в себя цифровые данные о пространственных объектах, сведения об их местоположении и свойствах, пространственных и непространственных атрибутах.

ГИС позволяет эффективно и комплексно использовать совершенно различные геоданные при решении научных и прикладных географических задач, связанных с инвентаризацией, анализом, моделированием, прогнозированием и управлением окружающей средой и территориальной организацией общества.

Термин ГИС используется и в более узком смысле — как инструмент (программный продукт), позволяющий пользователям искать, анализировать и редактировать цифровые карты, а также получать различную дополнительную информацию. Наиболее известные и часто используемые современные ГИС - это ArcGIS, MapInfo, а также Google Maps, Yandex-карты.

Данная учебно-исследовательская работа посвящена рассмотрению задачи сбора, обмена и хранения геоданных.

Исходные картографические данные для учебно-исследовательской работы будут получены из открытого источника в интернете OpenStreetMap.org в формате OSM XML.

Однако обмен и хранение пространственных данных требует развитой и всеобъемлющей системы стандартов представления геоданных. Такой системой стандартов для многих приложений на данный момент является формат Open Geospatial Consortium (OGC) (или сокращенно OpenGIS). Таким образом, для дальнейшего анализа и представления полученных геоданных их необходимо представить в данном формате (в формате OpenGIS).

Целью работы является описание алгоритма и реализация функционала модуля преобразования геоданных из формата OpenStreetMap в формат OpenGIS.

Задачи работы:

1. Изучить структуру XML-документа с геоданными в формате OSM.

   В данном пункте предполагается изучить структуру и синтаксис XML-документа. Выявить и описать особенности структуры и синтаксиса документа OSM XML. А также описать какие сведения о местоположении и свойствах пространственных данных можно получить из документа OSM XML.

2. Построить XPath-запросы для извлечения геоданных об объектах заданного типа.

   В данном пункте предполагается изучить синтаксис языка запросов XPath, овладеть навыками построения типовых XPath-запросов и построить XPath-запросы к документу OSM XML для извлечения необходимых сведений о пространственных данных.

3. Изучить стандарт OpenGIS и разработать структуру базы данных для хранения геоданных в формате OpenGIS.

   Здесь планируется изучить особенности СУБД PostgreSQL и её расширения PostGIS для работы с геоданными в формате OpenGIS, а также разработать базу данных для хранения извлеченных геоданных.

4. Построить SQL-запросы для сохранения и извлечения геоданных в формате OpenGIS.

   В данном пункте предполагается изучить структуру и синтаксис языка SQL, овладеть навыками построения типовых, а также пространственных SQL-запросов. Построить SQL-запросы для сохранения и извлечения геоданных в формате OpenGIS.

5. Реализовать модуль преобразования геоданных из формата OpenStreetMap в формат OpenGIS.

   Данный пункт является основным. В нём предполагается составление алгоритма решения задачи и его реализация на языке C++.

2 Теоретическая часть

2.1 Язык XML

Картографические данные для решения поставленной задачи получены в формате OSM XML из открытого источника в интернете OpenStreetMap.org. В данном разделе рассмотрена структура языка XML.

2.1.1 Общие сведения

Язык XML (eXtensible Markup Language) - это расширяемый язык разметки.

Язык разметки документов - набор символов или последовательностей, вставляемых в текст для передачи информации о его выводе или строении. Такие языки принадлежат классу компьютерных языков, но не являются языками программирования. Текстовый документ, написанный с использованием языка разметки, содержит не только сам текст (как последовательность слов и знаков препинания), но и дополнительную информацию о различных его участках (например, указание на заголовки, выделения, списки и т.д.). Иногда язык разметки позволяет вставлять в документ интерактивные элементы и содержание других документов. Языки разметки используются везде, где требуется вывод форматированного текста - в типографии (SGML, TeX, PostScript, PDF), пользовательских интерфейсах компьютеров (MicrosoftWord, OpenOffice), Всемирной Сети (HTML, XHTML, XML, WML, VML, PGML, SVG, XBRL).

Общая особенность всех языков разметки состоит в том, что они перемешивают текст документа с инструкциями разметки в потоке данных или файле. Это не необходимость, для удобства работы с текстом, его можно изолировать от разметки, используя указатели, метки, идентификаторы или другие методы координации.

Родоначальником наиболее известных в наше время языков разметки является обобщенный структурированный язык разметки (Structured Generalized Language) - SGML, определенный в международном стандарте ISO 8879:1986. Языки HTML(HyperText Markup Language - язык разметки гипертекста) и XML (eXtensible Markup Language - расширяемый язык разметки) различными способами образованы из SGML.

SGML определяет базовый синтаксис, но дает возможность создавать собственные элементы (отсюда термин "обобщенный"в названии языка). Чтобы использовать SGML для описания определенного документа, необходимо продумать соответствующий набор элементов и структуру документа.

В 1996 г. группа XML Working Group консорциума World Wide Web Consortium (W3C) разработала ветвь языка SGML, специально

приспособленную для размещения информации в World Wide Web (аналогично стандартному для Web языку гипертекстовой разметки HTML (Hypertext Markup Language)). "XML предназначен для облегчения использования языка SGML в Web и выполнения задач, которые в настоящее время реализуются с помощью языка HTML. XML разработан с целью усовершенствовать применение и взаимодействие языков SGML и HTML."(спецификация версии 1.0 XML)

Как и SGML, XML дает возможность разрабатывать собственные наборы элементов при описании определенного документа и присваивать им любые имена по выбору (поэтому XML - расширяемый язык). Как и в SGML, в теле программы может быть определено XML-приложение (или словарь), которое содержит набор наиболее употребительных элементов общего назначения и структуру документа, которая может быть использована для описания документа определенного типа.

Таким образом, XML является подмножеством SGML, разработанным для упрощения процесса машинного разбора документа.

Кроме того, XML - это текстовый формат, предназначенный для хранения структурированных данных (взамен существующих файлов баз данных), для обмена информацией между программами, а также для создания на его основе более специализированных языков разметки (например, XHTML), иногда называемых словарями. Это значит, что XML можно использовать для описания практически любого документа, от музыкальной партитуры до баз данных.

2.1.2 Структура XML-документа

Несмотря на то что XML дает возможность разрабатывать собственные наборы элементов при описании документа и присваивать им любые имена по выбору, структура каждого XML-документа имеет следующие особенности:

- Первая строка XML-документа называется объявлением XML. Это необязательная строка, указывающая версию стандарта XML (обычно это 1.0). Кроме того в объявлении может быть указана кодировка символов и внешние зависимости.

- Комментарий может быть размещен в любом месте документа. XML-комментарии размещаются внутри пары тегов: начинаются <!– и заканчиваются –>. Два знака дефис (–) не могут быть применены ни в какой части внутри комментария.

- Остальная часть XML-документа состоит из вложенных элементов, некоторые из которых имеют атрибуты и содержимое. В документе может быть один и только один корневой элемент, содержащий все остальные элементы

- Элемент обычно состоит из открывающего и закрывающего тегов, обрамляющих текст и другие элементы, при этом имена элементов подчиняются правилам:
  - имя начинается с буквы, знака подчеркивания или двоеточия;
  - после первого символа в имени могут быть буквы, цифры, знаки переноса, подчеркивания, точка или двоеточие;
  - имена не могут начинаться с сочетания букв XML.
- Открывающий тег состоит из имени элемента в угловых скобках; закрывающий тег состоит из того же имени в угловых скобках, но перед именем ещё добавляется косая черта.
- Содержимым элемента называется всё, что расположено между открывающим и закрывающим тегами, включая текст и другие (вложенные) элементы.
- Кроме содержания у элемента могут быть атрибуты - пары имя=значение, добавляемые внутрь открывающего тега после названия элемента.
- Значения атрибутов всегда заключаются в кавычки (одинарные или двойные), одно и то же имя атрибута не может встречаться дважды в одном элементе.
- Для обозначения элемента без содержания, называемого пустым элементом, необходимо применять особую форму записи, состоящую из одного тега, в котором после имени элемента ставится косая черта "/".

Описанные выше правила позволяют контролировать только формальную правильность XML-документа, но не содержательную. Для решения второй задачи используются так называемые схемы.

Схема четко определяет имя и структуру корневого элемента, включая спецификацию всех его дочерних элементов. Программист должен задать, какие элементы и в каком количестве обязательны, а какие - необязательны. Схема также определяет, какие элементы содержат атрибуты, допустимые значения этих атрибутов, а также значения по умолчанию.

Чаще всего для описания схемы используются следующие спецификации:

- DTD (Document Type Definition) - язык определения типа документов, который первоначально использовался в качестве язык описания структуры SGML-документа.
- XDR (XML Data Reduced) - диалект XML, разработанный Microsoft.
- XSD (язык определения схем XML) - рекомендация консорциумом W3C с 2001 года.

2.2 OpenStreetMap

2.2.1 Описание OpenStreetMap.org

OpenStreetMap ("открытая карта улиц"), сокращённо OSM, - некоммерческий сетевой картографический проект, в котором могут участвовать все желающие пользователи Интернета. Он направлен на создание подробной свободной и бесплатной географической карты всего мира. Все данные доступны для легального копирования, редактирования и коммерческого использования. База данных OSM содержит данные самого разного рода, например, дороги, тропы, здания, магазины, аптеки, памятники, деревья, заборы, детские площадки, точки Wi-Fi, почтовые ящики, телефонные будки, административное деление, адреса, часы работы, веб-сайты, телефоны и многое другое.

OpenStreetMap - единственный свободный картографический проект. Основными конкурентами являются сайты Google Map Maker, Wikimapia, Яндекс.Народная карта, но все эти проекты имеют общий недостаток - они являются 'закрытыми' и несвободными.

Все данные создаются на основе GPS-треков, которые загружают участники на основе обычных народных знаний, фотографий (также, например, Яндекс.Панорам) или же путём обрисовывания спутниковых снимков (доступно с тех пор как их впервые предоставил Yahoo!). В проекте делают свободную карту, которой могли бы пользоваться все без ограничений, поэтому тщательно следят за соблюдением авторских прав — импорты из или обрисовывание несвободных источников не допускаются.

Спутниковые снимки земной поверхности позволяют рисовать, не имея треков, карты крупных городов (для которых имеются снимки высокого разрешения). В качестве источников участникам можно пользоваться снимками, предоставленными правительственными сервисами, такими как Landsat, Prototype Global Shorelines (PGS) и TIGER, а также картографическими сервисами Yahoo! и Bing Maps(весь мир), Spot Image (Франция), Космоснимками (спутник IRS — запад России, SPOT4 — восток и Киргизия) и другими компаниями, а также данными ASTER.

Данные об основных дорогах обычно получаются из истории 'треков'. Они записаны GPS-приёмниками или GPS-трекерами. Такие треки создаются добровольцами и выполняются в результате путешествий по исследуемому району пешком, на велосипеде или на автомобиле. Затем треки экспортируются из GPS-устройства в специальную программу.

Карты проекта двухмерные, без отображения высот над уровнем моря, изолиний. С OpenStreetMap.org возможен экспорт карт в форматы PNG, JPEG, SVG, PDF, PostScript. Также существуют проекты по экспорту данных OpenStreetMap в различные форматы, например, формат карт

Garmin, а также ГисРусса.

2.2.2 Формат данных OSM

Для учебно-исследовательской работы данные получены с OpenStreetMap.org в формате OSM XML. Формат OSM представляет из себя тот же XML с определенным набором наиболее употребительных элементов общего назначения (определены в онотлогии к проекту) .

OpenStreetMap использует топологическую структуру данных, состоящую из объектов:

- node (точка) - точка с указанными координатами;

- way (линия) - упорядоченный список точек, составляющих линию или полигон (замкнутая линия);

- relation (отношение) - группы точек, линий и других отношений, которым назначаются некоторые свойства;

- tag (тег) - пары 'ключ - значение', могут назначаться точкам, линиям и отношениям.

Рассмотрим структуру OSM-документа.

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
    <bounds minlat="55.1196000" minlon="37.3708000" maxlat="55.1814000" maxlon="37.5074000"/>
    <node id="76475011" lat="55.0269826" lon="37.4271725" user="kol" uid="73184" visible="true" version="13" changeset="813236" timestamp="2009-01-19T09:05:54Z"/>
    <node id="139744952" lat="55.0294985" lon="37.4277889" user="u07" uid="69700" visible="true" version="7" changeset="3437942" timestamp="2009-12-23T21:29:34Z">
        <tag k="railway" v="level_crossing"/>
    </node>
    ...
    <way id="115595846" user="Dimon62" uid="457129" visible="true" version="2" changeset="8303598" timestamp="2011-05-31T17:43:16Z">
        <nd ref="1305572872"/>
        <nd ref="1305572955"/>
        <nd ref="1305573037"/>
        ...
        <tag k="shop" v="supermarket"/>
        <tag k="name" v="Карусель"/>
        ...
    </way>
    ...
    <relation id="421030" user="Hind" uid="79836" visible="true" version="1" changeset="4008870" timestamp="2010-03-01T14:05:55Z">
        <member type="relation" ref="421012" role="is_in"/>
        ...
        <member type="way" ref="124027968" role="outer"/>
        ...
        <member type="node" ref="124027968" role="outer"/>
        ...
        <tag k="name" v="Чеховский район"/>
        <tag k="type" v="address"/>
        ...
    </relation>
    ...
</osm>
```

Документ OSM состоит из двух основных частей: пролога и тела документа.

Как и обычный XML-документ пролог OSM-документа может содержать идентификационную информацию о нем или может быть абсолютно пустым.

В прологе документа обычно содержится объявление, указывающее на то, что это XML-документ, и содержащее номер версии (version) и используемую кодировку (encoding declaration):

<?xml version="1.0"encoding="UTF-8»

Второй основной частью OSM-документа является тело документа (Документ или корневой элемент), который в свою очередь

содержит дополнительные элементы. К телу документа относится все, что заключено между тегами <osm> ... </osm>. Элементы, заключенные между этими тегами, несут в себе информацию, содержащуюся в документе (в нашем случае это информация о геообъектах, такая как тип объекта, его название, координаты и т.д.).

В OpenStreetMap нет ограничений на вводимые теги (так же как и для обычного XML-документа), но все же, как упоминалось ранее, для структурированной работы существует рекомендованный набор объектов и соответствующих им тегов. Основными такими тегами являются вложенные элементы <bounds/>, <node/>, <way/>, <relaition/>, <tag/>, а также некоторые другие, определеные в онотлогии к проекту.

Рассмотрим свойства основных тегов.

1. Элемент bounds. Включает в себя границы рассматриваемой области, которые определены атрибутами minlat, maxlat, minlon, maxlon.

2. Базовые элементы: точка (node), линия(way) и полигон(area).

   - точка (node) - базовый элемент в структуре данных OSM. Точка имеет атрибуты 'широта' и 'долгота'. Точки используются для того, чтобы определить 'линию' (см. ниже), однако точка может являться и самостоятельным элементом карты, и использоваться для обозначения отдельного ни с чем не связанного объекта (например, телефонной будки, кафе, для указания координат, к которым привязано название населённого пункта или любого другого места. Отдельные точки (т.е. не входящие в состав линий или областей) всегда должны иметь хотя бы одно свойство.

     Точки, входящие в состав линии, часто не имеют свойств и нужны только для описания линии; однако это не является незыблемым правилом.

   - линия (way). Линия представляет собой ломаную линию, проходящую через точки. Линия состоит, как минимум, из двух точек. Обычно линиями обозначаются улицы, дороги и т.д. Одна точка может принадлежать нескольким линиям одновременно.

     Линия характеризуется свойствами, которые распространяются на неё на всём протяжении. Например, для линии, обозначающей дорогу, такими свойствами могут являться тип и качество покрытия, допустимая скорость движения и т.п. Если при уточнении выясняется, что не все свойства линии сохраняются на всём её протяжении (например, на дороге, которой соответствует линия, имеется участок с другим типом покрытия), то линия может быть разделена на части.

     Для того, чтобы считаться корректно определённой, линия должна иметь хотя бы одно свойство.

- полигон (area) или замкнутая линия (closed way) - элемент карты, предназначенный для описания участков поверхности. Полигон формируется замкнутой линией (т.е. первая точка линии совпадает с последней) и является совокупностью этой самой линии и области, находящейся внутри контура этой линии. В этом смысле полигон не является самостоятельным типом элементов, а лишь псевдо-элементом, особой разновидностью линии с соответствующими свойствами.

  Полигоны используются для обозначения участка поверхности, обладающего общими свойствами. Например, полигоны используются для описания водоёмов и лесов.

  Корректно определённый полигон должен иметь хотя бы одно свойство.

  Для описания вырезов, 'дыр' в полигонах, например, для описания участка занятого лесом, внутри которого имеется вырубленный участок рисуются мультиполигоны.

3. Элемент relation (отношение). Отношение группирует объекты по определенному признаку и для определенной цели. 'Участниками' (member) отношения могут любые объекты (точки, линии, области) и даже другие отношения. Эти элементы — участники отношения и каждому участнику присваивается 'роль' (role) в отношении. Как и другие типы элементов могут иметь теги. Один объект может входить в несколько отношений, а также несколько раз в одно и сто же отношение.

   Атрибут 'тип' (type) устанавливает разновидность отношения. Отношения могут описывать замкнутые дороги, запреты поворотов и т.д.

   Расположение участников в отношении постоянно и определяется очередностью добавления. Повторяющиеся объекты сохраняют их определенный порядок.

4. Элемент tag (тег). Тег строго говоря не элемент, а свойство объекта. Каждый тег имеет атрибуты - ключ (k) и значение (v). Ключи и их значения описывают объект и наделяют его свойствами.

Для всех элементов существуют общие атрибуты:

- user - имя пользователя, который совершал последнее изменение объекта;
- uid - числовой идентификатор пользователя, который совершил последнее изменение объекта;
- timestap - время последнего изменения;

- visible - является ли объект удаленным в базе данных. Если visible='false', то объект должен быть возвращен вызовом истории изменений;
- version - версия редакции объекта. Версия вновь созданных объектов равна 1, это значение увеличивается на сервере, когда клиент добавляет новую версию объекта. Сервер будет отклонять новую версию объекта, если
  версия присланная клиентом не соответствует текущей версии объекта в базе данных;
- chengeset - пакет правок (история) в котором указаны создание и изменения объектов.

Особенности элемента 'точка'.

- атрибут id - числовой идентификационный номер, который уникален только среди точек. (Линия может иметь такое же id как и точка.);
- атрибут lat - координаты широты;
- атрибут lon - координаты долготы;
- теги - множество пар тегов (ключ - значение) с уникальным ключом;

Особенности элемента 'линия'.

- атрибут id - числовой идентификационный номер, не являются уникальными, линия может иметь такой же идентификатор, как точка;
- nodes - теги (nd) с атибутом ref (идентификатор точки), являющиеся списком всех точек идентификаторы которых составляют линию;
- теги - множество пар тегов (ключ - значение) с уникальным ключом;

Особенности элемента 'отношение'.

- атрибут id - числовой идентификационный номер, не являются уникальными, линия может иметь такой же идентификатор, как точка;
- теги - множество пар тегов (ключ - значение) с уникальным ключом;
- members - упорядоченный список примитивов с атрибутами роль (где ролью может быть любой текст), тип (тип примитива), ref (идентификатор примитива).

2.3 XPath

2.3.1 Общие сведения

Для получения необходимой информации из документа OSM в учебно-исследовательской работе используется язык XPath.

Язык XML Path (XPath) является набором синтаксических и семантических правил для ссылок на части XML-документов. XPath используется в URL для записи путей, обеспечивающих навигацию по иерархической структуре XML документа.

Главная задача языка XPath - адресация частей в XML документе. Для достижения этой цели язык дополнительно наделен основными функциями для манипулирования строками, числами и булевыми значениями. В XPath используется компактный синтаксис, отличающийся от принятого в XML и облегчающий использование языка XPath. XPath работает не с внешним синтаксисом XML документа, а с его абстрактной логической структурой. Язык XPath спроектирован так, что помимо поддержки адресации он обладает естественным набором элементов, которые могут использоваться для сравнения (проверки, соответствует ли узел некому шаблону).

XPath представляет XML-документ как дерево узлов. Узлы могут быть разных типов, таких как узлы элементов, узлы атрибутов или узлы текста. Для каждого типа узлов в XPath определяется способ вычисления строкового значения. Некоторые типы узлов имеют также имя. XPath полностью поддерживает пространства имен XML. В результате, имя любого узла в этом языке образуется из двух частей: локальной части и URI (унифицированный (единообразный) идентификатор ресурса - последовательность символов, идентифицирующая абстрактный или физический ресурс) некого пространства имен (возможно, нулевого). Такая комбинация называется расширенным именем.

Специальным типом узла является корневой узел. XML-документ содержит только один такой узел и он содержит весь XML-документ, т.е. корневой узел содержит корневой элемент, а также любые узлы инструкций обработки, объявлений или комментариев, которые появляются до или после корневого элемента.

Не существует типа узла для XML-декларации (такой, как <?xml version='1.0' encoding='UTF-8'?>). Следовательно, на такие сущности нельзя ссылаться в XPath.

Элементные узлы представляют каждый элемент в XML-документе. Узлы атрибутов принадлежат элементным узлам и представляют атрибуты. Однако, атрибуты, которые начинаются с xmlns: представляются в XPath с узлами пространств имен. Другие типы узлов включают в себя текстовые узлы, узлы инструкций обработки и узлы комментариев.

Главной синтаксической конструкцией языка XPath является выражение. В результате обработки выражения получается объект, относящийся к одному из четырех основных типов:

- набор узлов (node-set) - неупорядоченный набор узлов без дубликатов;
- булево значение (boolean) - true или false;
- число (number) - число с плавающей точкой;
- строка (string) - последовательность UCS символов.

2.3.2 Типовые XPath-запросы

Рассмотрим XPath-запросы на примере документа OSM:

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
    <bounds minlat="55.1196000" minlon="37.3708000" maxlat="55.1814000" maxlon="37.5074000"/>
    <node id="1" lat="55.0269826" lon="37.4271725" user="kol" uid="73184" visible="true" version="13" changeset="813236" timestamp="2009-01-19T09:05:54Z"/>
    <node id="2" lat="55.0294985" lon="37.4277889" user="u07" uid="69700" visible="true" version="7" changeset="3437942" timestamp="2009-12-23T21:29:34Z">
        <tag k="railway" v="level_crossing"/>
    </node>
    <way id="10" user="Dimon62" uid="457129" visible="true" version="2" changeset="8303598" timestamp="2011-05-31T17:43:16Z">
        <nd ref="1"/>
        <nd ref="2"/>
        <tag k="shop" v="supermarket"/>
        <tag k="name" v="Карусель"/>
    </way>
    <relation id="20" user="Hind" uid="79836" visible="true" version="1" changeset="4008870" timestamp="2010-03-01T14:05:55Z">
        <member type="way" ref="10" role="outer"/>
        <member type="node" ref="1" role="outer"/>
        <member type="node" ref="2" role="outer"/>
        <tag k="name" v="Чеховский район"/>
        <tag k="type" v="address"/>
    </relation>
</osm>
```

Базовый синтаксис языка

Базовый синтаксис языка XPath похож на адресацию в файловой системе. База языка - это оси. Оси служат для определения набора узлов относительно данного узла. В дополнение к базе существует набор функций, разделенных на 5 групп. Описание осей и функций находится в Приложении А.

Основные оси языка XPath:

- attribute:: — возвращает множество атрибутов текущего элемента (можно заменить на '@');
- child:: — возвращает множество потомков на один уровень ниже (часто просто опускают);
- descendant:: — возвращает полное множество потомков (можно заменить на './/');
- parent:: — возвращает предка на один уровень назад (можно заменить на '..');
- self:: — возвращает текущий элемент (можно заменить на '.').

Если путь начинается с символа '/', то он представляет абсолютный путь к заданному элементу. Запрос '/osm' выберет корневой элемент <osm> ... </osm>. Этот запрос эквивалентен '/child::osm', но, как отмечалось ранее, подобная запись упрощается.

'/osm/node/tag' - выбираются все элементы tag, являющиеся детьми элементов <node>, которые в свою очередь являются детьми корневого узла <osm>:

<tag k=’railway’ v=’level_crossing’></tag>.

Если путь начинается с //, то будут выбраны все элементы документа, которые соответствуют указанному шаблону, например, ’//nd’ - будут выбраны все элементы документа, соответствующие <nd>.

<nd ref=’1’></nd>
<nd ref=’2’></nd>

’//way/tag’ - все элементы tag, являющиеся детьми <way>:

<tag k=’shop’ v=’supermarket’></tag>
<tag k=’name’ v=’Карусель’></tag>

Символ ’*’ указывает, что надо выбрать все элементы, соответствующие пути перед ней. Например, ’/osm/node/*’ - будут выбраны все элементы, являющиеся прямыми потомками /osm/node :

<tag k=’railway’ v=’level_crossing’></tag>

’/*/*/tag’ - все элементы tag, имеющие двух предков.

<tag k=’railway’ v=’level_crossing’></tag>
<tag k=’shop’ v=’supermarket’></tag>
<tag k=’name’ v=’Карусель’></tag>
<tag k=’name’ v=’Чеховский район’></tag>
<tag k=’type’ v=’address’></tag>

’//*’ - будут выбраны все возможные элементы.

Выражение в квадратных скобках позволяет задавать более четкие критерии для элемента. Так число в квадратных скобках обозначает позицию элемента в выбранном множестве.

’/osm/node[1]’ - будет выбран первый потомок <node> элемента <osm>:

<node id=’1’ lat=’55.0269826’ lon=’37.4271725’ user=’kolen’ ...>

Несколько путей можно скомбинировать с помощью разделителя |.

’//member | //bounds’ - Выбираются все элементы member и bounds:

<bounds minlat=’55.1196000’ minlon=’37.3708000’ ...></bounds>
<member type=’way’ ref=’10’ role=’admin_centre’></member>
<member type=’node’ ref=’1’ role=’outer’></member>
<member type=’node’ ref=’2’ role=’outer’></member>

’//tag[@v]’ - выбираются элементы tag, имеющие атрибут ’@v’ (может быть также записан как ’//tag[attribute::v]):

<tag k=’railway’ v="level_crossing»></tag>

<tag k=’shop’ v=’supermarket’></tag>
<tag k=’name’ v=’Карусель’></tag>
<tag k=’name’ v=’Чеховский район’></tag>
<tag k=’type’ v=’address’></tag>

Ось descendant содержит потомков контекстного узла; потомком является дочерний элемент, дочерний элемент дочернего элемента и так далее; таким образом ось descendant не содержит узлов атрибутов и пространств имен.

’//node/descendant::tag’ - выбираются элементы tag, имеющие в качестве предка элемент node:

<tag k=’railway’ v=’level_crossing’/>

Ось parent содержит родителя контекстного узла, если он существует:

’//bounds/parent::*’ - результат:

<osm version=’0.6’ generator=’CGImap 0.0.2’></osm>;

2.4 PostgreSQL и OpenGIS

2.4.1 PostgreSQL

Геоданные предполагается хранить в базе данных.

База данных (БД) - это поименнованная совокупность структурированных данных, относящихся к определенной предметной области.

Для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации используется комплекс программных и языковых средств - система управления базами данных (СУБД). СУБД позволяют структурировать, систематизировать и организовывать данные для их компьютерного хранения и обработки.

Основные функции СУБД:

- физическое размещение в памяти данных и их описаний;
- реализация механизмов поиска запрашиваемых данных;
- решение проблем, возникающих при одновременном запросе одних и тех же данных многими пользователями (прикладными программами);
- обеспечение защиты данных от некорректных обновлений и (или) несанкционированного доступа;
- поддержание баз данных в актуальном состоянии;
- обеспечение целостности и непротиворечивости данных.

СУБД различаются по типу используемых моделей данных, а также по способу доступа к базе данных.

По типу используемых моделей данных СУБД различают как:

- иерархические;
- сетевые;
- реляционные;
- объектно-ориентированные;
- объектно-реляционные.

1. Иерархические СУБД (например, Information Management System (IMS)) - поддерживают древовидную организацию информации. Связи между записями
   выражаются в виде отношений предок/потомок, а у каждой записи есть ровно одна родительская запись. Это помогает поддерживать ссылочную целостность. Когда запись удаляется из дерева, все ее потомки также должны быть удалены.

Иерархические базы данных имеют централизованную структуру, т.е. безопасность данных легко контролировать. Поиск записи осуществляется методом прямого обхода дерева. Отсюда следует необходимость правильно упорядочивать записи, чтобы время их поиска было минимальным. Это трудно, так как не все отношения, существующие в реальном мире, можно выразить в иерархической базе данных. Отношения 'один ко многим' являются естественными, но практически невозможно описать отношения 'многие ко многим' или ситуации, когда запись имеет несколько предков.

2. Сетевые СУБД (например, Integrated Database Management System (IDMS)). Сетевая модель расширяет иерархическую модель СУБД, позволяя группировать связи между записями в множества. С логической точки зрения связь — это не сама запись. Связи лишь выражают отношения между записями. В сетевой модели допускаются отношения 'многие ко многим', а записи не зависят друг от друга. При удалении записи удаляются и все ее связи, но не сами связанные записи.

   В сетевой модели требуется, чтобы связи устанавливались между существующими записями во избежание дублирования и искажения целостности. Данные можно изолировать в соответствующих таблицах и связать с записями в других таблицах.

   Программисту не нужно, при проектировании СУБД, заботиться о том, как организуется физическое хранение данных на диске. Это ослабляет зависимость приложений и данных. Но в сетевой модели требуется, чтобы программист помнил структуру данных при формировании запросов.

3. Реляционные СУБД (например, System R, Ingres). В сравнении с рассмотренными выше моделями реляционная модель требует от сервера СУБД гораздо более высокого уровня сложности. В ней делается попытка избавить программиста от выполнения рутинных операций по управлению данными, характерных для иерархической и сетевой моделей.

   В реляционной модели база данных представляет собой централизованное хранилище таблиц, обеспечивающее безопасный одновременный доступ к информации со стороны многих пользователей. В строках таблиц часть полей содержит данные, относящиеся непосредственно к записи, а часть — ссылки на записи других таблиц. Таким образом, связи между записями являются неотъемлемым свойством реляционной модели. Каждая запись таблицы имеет одинаковую структуру.

   В реляционной модели СУБД достигается информационная и структурная независимость. Записи не связаны между собой

настолько, чтобы изменение одной из них затронуло остальные.

В реляционных СУБД применяется язык SQL, позволяющий формулировать произвольные, нерегламентированные запросы. Реляционные базы данных страдают от различий в реализации языка SQL, хотя это и не проблема реляционной модели. Каждая реляционная СУБД реализует какое-то подмножество стандарта SQL плюс набор уникальных команд, что усложняет задачу программистам, пытающимся перейти от одной СУБД к другой.

4. Объектно-ориентированные СУБД (O2, ORION, Iris). Позволяют программистам интерпретировать все свои информационные сущности как объекты, хранящиеся в оперативной памяти. Дополнительный интерфейсный уровень абстракции обеспечивает перехват запросов, обращающихся к тем частям базы данных, которые находятся в постоянном хранилище на диске. Изменения, вносимые в объекты, оптимальным образом переносятся из памяти на диск.

   Преимуществом ООСУБД является упрощенный код. Приложения получают возможность интерпретировать данные в контексте того языка программирования, на котором они написаны. Реляционная база данных возвращает значения всех полей в текстовом виде, а затем они приводятся к локальным типам данных. В ООБД этот этап ликвидирован. Методы манипулирования данными всегда остаются одинаковыми независимо от того, находятся данные на диске или в памяти.

   С помощью ООСУБД решаются две проблемы. Во-первых, сложные информационные структуры выражаются в них лучше, чем в реляционных базах данных, а во вторых, устраняется необходимость транслировать данные из того формата, который поддерживается в СУБД.

   В объектной модели данных поддерживаются нерегламентированные запросы, но языком их составления не обязательно является SQL. Логическое представление данных может не соответствовать реляционной модели, поэтому применение языка SQL станет бессмысленным. Зачастую удобнее обрабатывать объекты в памяти, выполняя соответствующие виды поиска.

   Большим недостатком объектно-ориентированных баз данных является их тесная связь с применяемым языком программирования. К данным, хранящимся в реляционной СУБД, могут обращаться любые приложения, тогда как, к примеру, Java-объект, помещенный в ООСУБД, будет представлять интерес лишь для приложений, написанных на Java.

5. Объектно-реляционные СУБД (PostgreSQL). Объединяют в себе черты реляционной и объектной моделей. Их возникновение объясняется тем,

что реляционные базы данных хорошо работают со встроенными типами данных и гораздо хуже — с пользовательскими, нестандартными. Когда появляется новый важный тип данных, приходится либо включать его поддержку в СУБД, либо заставлять программиста самостоятельно управлять данными в приложении.

Объектно-реляционная СУБД позволяет загружать код, предназначенный для обработки 'нетипичных' данных. Таким образом, база данных сохраняет свою табличную структуру, но способ обработки некоторых полей таблиц определяется извне, т.е. программистом.

По способу доступа к базе данных СУБД различаются как:

1. Файл-серверные СУБД. В файл-серверных СУБД файлы данных располагаются централизованно на файл-сервере СУБД. Ядро СУБД располагается на каждом клиентском компьютере. Доступ к данным осуществляется через локальную сеть. Синхронизация чтений и обновлений осуществляется посредством файловых блокировок. Преимуществом этой архитектуры является низкая нагрузка на ценральный процессор сервера, а недостатком — высокая загрузка локальной сети.

2. Клиент-серверные СУБД. Такие СУБД состоят из клиентской части (которая входит в состав прикладной программы) и сервера СУБД. Клиент-серверные СУБД, в отличие от файл-серверных, обеспечивают разграничение доступа между пользователями и мало загружают сеть и клиентские машины. Сервер является внешней по отношению к клиенту программой, и по надобности его можно заменить другим. Недостаток клиент-серверных СУБД в самом факте существования сервера СУБД (что плохо для локальных программ — в них удобнее встраиваемые СУБД) и больших вычислительных ресурсах, потребляемых сервером.

3. Встраиваемые СУБД. Встраиваемая СУБД — библиотека, которая позволяет унифицированным образом хранить большие объёмы данных на локальной машине. Доступ к данным может происходить через SQL либо через особые функции СУБД. Встраиваемые СУБД быстрее обычных клиент-серверных и не требуют установки сервера, поэтому востребованы в локальном ПО, которое имеет дело с большими объёмами данных.

В учебно-исследовательской работе использована свободная объектно-реляционная клиент-серверная СУБД PostgreSQL.

История PostgreSQL начинается с 1977 года с разработки в Калифорнийском университете Беркли базы данных Ingres. В 1996 был открыт исходный код PostgreSQL для программистов, и

началось её развитие под девизом Open Source. Данная СУБД была выбрана для проекта не только из-за того, что она является открытой. PostgreSQL выбрана из-за того, что в распоряжении PostgreSQL есть расширение PostGIS, осуществляющее поддержку хранения геоданных в соответствии со стандартом хранения геоданных OpenGis.

2.4.2 Стандарт OpenGIS

Хранение геоданных — одна из самых актуальных задач в области хранения данных. Будь то каталог автозаправок, база данных городов, система отслеживания пробок — везде требуется эффективная система хранения больших объемов географических данных.

За стандартизацию этой области на данный момент отвечает Open Geospatial Consortium (OGC) или сокращенно Open GIS, в который входит более 370 организаций. В течение нескольких лет были составлены 28 стандартов.

OpenGIS даёт следующие преимущества:

- позволяет хранить геометрический/географический объект в виде одного кортежа и строить реляционные связи между этими объектами и другими данными БД;
- позволяет строить индексы по геометрическим полям и выполнять выборки с их помощью;
- предоставляет набор функций для манипулирования и выявления отношений между объектами (включение, пересечение, расстояние, соприкасание и т.п.);

Все разработчики СУБД стремятся создавать расширения для работы с геоданными в соответствии со стандартами OpenGIS. Как отмечалось ранее у PosgreSQL существует расширение - PostGIS, позволяющее этой СУБД хранить геоданные в соответствии со стандартами OpenGIS.

Спецификация OpenGIS определяет стандартные типы объектов ГИС, функции для манипуляции ими, и набор таблиц метаданных. В целях сохранения корректности метаданных, такие операции как создание и удаление столбцов с пространственными данными осуществляются с помощью специальных процедур, определенных OpenGIS.

Спецификация OpenGIS определяет два стандартных способа определения пространственных объектов: в форме Well-Known Text (WKT) и в форме Well-Known Binary (WKB). WKT и WKB включают информацию о типе объекта и координаты, составляющие объект.

Примеры текстового представления (WKT) пространственных объектов приведены ниже:

– POINT(0 0)

– LINESTRING(0 0,1 1,1 2)

– POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))

– MULTIPOINT(0 0,1 2)

– MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))

– MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))

– GEOMETRYCOLLECTION(POINT(2 3),LINESTRING((2 3,3 4)))

Кроме того, стандартом OpenGIS определены две таблицы метаданных: SPATIAL_REF_SYS и GEOMETRY_COLUMNS. Эти две таблицы и обеспечивают поддержку стандарта OpenGIS для PostGIS.

Таблица SPATIAL_REF_SYS содержит числовые идентификаторы и текстовые описания систем координат, используемых в пространственной базе данных, и определяется следующим образом:

```
CREATE TABLE spatial_ref_sys (
    srid INTEGER NOT NULL PRIMARY KEY,
    auth_name VARCHAR(256),
    auth_srid INTEGER,
    srtext VARCHAR(2048),
    proj4text VARCHAR(2048)
    )
```

В этой таблице столбец srid - уникальный идентификатор пространственной системы координат (Spatial Referencing System, SRS) в пределах этой таблицы. SRID представляет из себя числовой код, которому соответствует некоторая система координат. Например, распространенный код EPSG 4326 соответствует географической системе координат WGS84.

Колонка auth_name содержит название стандарта или стандартизирующего органа, на который ссылается данная справочная система. Например, EPSG будет правильным значением для auth_name.

Столбец auth_srid - идентификатор пространственной системы координат, указанной в auth_name. В случае EPSG, это должен быть код проекции EPSG.

Srtext содержит в себе WNT (well-known text) представление системы координат.

PostGIS использует библиотеку Proj4 для преобразований систем координат. Столбец proj4text содержит строковое определение координат Proj4 для данного SRID.

Таблица GEOMETRY_COLUMNS хранит информацию о таблицах базы данных, содержащих пространственную информацию и определяется следующим образом:

```
CREATE TABLE geometry_columns (
    f_table_catalog VARCHAR(256) NOT NULL,
    f_table_schema VARCHAR(256) NOT NULL,
    f_table_name VARCHAR(256) NOT NULL,
    f_geometry_column VARCHAR(256) NOT NULL,
    coord_dimension INTEGER NOT NULL,
    srid INTEGER NOT NULL,
    type VARCHAR(30) NOT NULL
    )
```

Здесь столбцы f_table_catalog, f_table_schema, f_table_name содержат в себе все составляющие имени таблицы с геометрическим столбцом, аналогично Oracle. В PosgreSQL нет аналога понятия 'каталог', поэтому эта колонка всегда пуста. В колонке 'схема' хранится имя схемы в PostgreSQL (по умолчанию всегда 'public').

Столбец f_geometry_column содержит имя геометрического столбца в основной таблице.

В колонке coord_dimension указывается геометрическая размерность столбца (2, 3, 4-мерное измерение).

Srid - внешний ключ на таблицу SPATIAL_REF_SYS, содержит идентификаторы пространственной справочной системы, используемой для координатной геометрии в указанной таблице.

Колонка type содержит типы пространственных объектов. Ограничивает пространственный столбец единственным типом, одним из следующих: POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION (или POINTM, соответствующим XYM-версиям), LINESTRINGM, POLYGONM, MULTIPOINTM, MULTILINESTRINGM, MULTIPOLYGONM, GEOMETRYCOLLECTIONM. Для разнородных (смешанный тип) коллекций возможно использование 'GEOMETRY' как типа. Этот атрибут не является частью спецификации OpenGIS, но он необходим для обеспечения типового единообразия.

Заполнение этой таблицы осуществляется вручную, либо с помощью специальной процедуры OGC AddGeometryColumn().

2.5 SQL

Для помещения информации о геообъектах в базу данных использован язык SQL. В данной главе описывается структура и синтаксис языка SQL, а также приведены типовые SQL-запросы и пространственные запросы.

2.5.1 Общие сведения

Язык SQL (Structured Query Language - структурированный язык запросов) представляет собой стандартный высокоуровневый язык описания данных и манипулирования ими в системах управления базами данных (СУБД), построенных на основе реляционной модели данных.

Язык SQL разработан фирмой IBM в конце 70-х годов. Первый международный стандарт языка принят международной стандартизирующей организацией ISO в 1989 году, а новый (более полный) - в 1992 году. Официальное название стандарта - Международный стандарт языка баз данных SQL (1992) (International Standart Database Language SQL), неофициальное название - SQL/92, или SQL-92, или SQL92. Язык SQL стал фактически стандартным языком доступа к базам данных. Все производители реляционных СУБД поддерживают с различной степенью соответствия стандарт SQL92.

Язык SQL представляет собой совокупность операторов, инструкций и вычисляемых функций.

Основу языка SQL составляют операторы, условно разбитые не несколько групп по выполняемым функциям.

Можно выделить следующие группы операторов:

1. Операторы DDL (Data Definition Language) - операторы определения и манипулирования схемой базы данных (CREATE, DROP, ALTER).

   CREATE создает объект БД (саму базу, таблицу, представление, пользователя и т. д.);

   ALTER изменяет объект;

   DROP удаляет объект.

2. Операторы DML (Data Manipulation Language) - операторы манипулирования данными (SELECT, INSERT, DELETE, UPDATE).

   SELECT считывает данные, удовлетворяющие заданным условиям;

   INSERT добавляет новые данные;

   UPDATE изменяет существующие данные;

   DELETE удаляет данные.

3. DCD (Data Control Language)- операторы защиты и управления данными (GRANT, REVOKE).

GRANT предоставляет пользователю (группе) разрешения на определенные операции с объектом;

REVOKE отзывает ранее выданные разрешения;

DENY задает запрет, имеющий приоритет над разрешением.

2.5.2 Типовые SQL-запросы

SQL-запрос состоит из ключевых слов и слов, определяемых пользователем. Ключевые слова являются постоянной частью языка SQL и имеют фиксированное значение. Их следует записывать в точности так, как это установлено, нельзя разбивать на части для переноса с одной строки на другую. Слова, определяемые пользователем, задаются им самим (в соответствии с синтаксическими правилами) и представляют собой идентификаторы или имена различных объектов БД. Слова в запросе размещаются также в соответствии с установленными синтаксическими правилами.

Каждый запрос начинается с глагола, т.е. ключевого слова, описывающего выполняемое действие, и заканчивается точкой с запятой (SELECT, CREATE, INSERT, DELETE и т.д.)

После глагола следует одно или несколько предложений. Они описывают данные, с которыми работает запрос, или содержат уточняющую информацию о действии, выполняемом запросом. Каждое предложение начинается с ключевого слова, например WHERE (где), FROM (откуда), INTO (куда) и HAVING(имеющий). Одни предложения в запросе могут изменяться, а другие - нет. При этом конкретная структура и содержимое предложения также могут изменяться. Многие предложения содержат имена таблиц или столбцов; некоторые из них могут содержать дополнительные ключевые слова, константы и выражения.

Глаголы, с которых начинаются запросы, и ключевые слова (слова, которые в SQL зарезервированы для специального использования и являются частью его синтаксиса) записываются заглавными буквами, чтобы отличать их от имен столбцов и таблиц. Но в общем случае синтаксис SQL-запросов не чувствителен к расположению текста по строкам и к регистру символов.

Чтобы получить информацию, хранящуюся в базе данных используется запрос **SELECT**. Базовое действие этого запроса ограничено одной таблицей, хотя существуют конструкции, обеспечивающие выборку с нескольких таблиц одновременно. Для того, чтобы получить все строки данных для специфических столбцов, используется запрос такого вида:

SELECT column1, column2 FROM table_name;

Также, можно получить все столбцы из таблицы, используя

подстановочный знак «*»:

SELECT * FROM table_name;

Это может быть полезно в том случае, когда необходимо выбрать данные с определенным условием WHERE. Следующий запрос возвратит все столбцы со всех строк, где «column1» содержит значение «3»:

SELECT * FROM table_name WHERE column1=3;

Кроме «=» (равно), существуют следующие условные операторы:

- = равно;
- <> не равно;
- > больше;
- < меньше;
- >= больше или равно;
- <= меньше или равно;

Запрос **INSERT** используется для создания новой строки данных. Для обновления уже существующих данных или пустых полей строки нужно использовать запрос UPDATE.

Примерный синтаксис запроса INSERT:

INSERT INTO table_name (column1, column2, column3) VALUES ('data1', 'data2', 'data3');

Если предполагается вставлять все значения в порядке, в котором находятся столбцы таблицы, то можно и не указывать имена столбцов, хотя для удобочитаемости это предпочтительнее. Кроме того, если столбцы перечисляются, то необязательно указывать их по порядку нахождения в базе данных, пока вводимые значения, соответствуют этому порядку. Не нужно перечислять столбцы, в которые не вводится информация.

Изменяется уже существующая информация в базе данных очень похожим образом.

UPDATE используется для того, чтобы изменить существующие значения или освободить поле в строке, поэтому новые значения должны соответствовать существующему типу данных и обеспечивать приемлемые значения. Если нужно изменить значения не во всех строках, то нужно использовать условие WHERE.

UPDATE table_name SET column1 = 'data1', column2 = 'data2' WHERE column3 = 'data3';

Можно использовать WHERE для любого столбца, включая тот, который необходимо изменить. Это используется когда необходимо заменить одно определенное значение на другое.

UPDATE table_name SET FirstName = ‘Василий’ WHERE FirstName = ‘Василий’ AND LastName = ‘Сидоров’;

Запрос **DELETE** полностью удаляет строку из базы данных. Если необходимо удалить одно единственное поле, то нужно использовать запрос UPDATE и установить для этого поля значение, которое будет являться аналогом NULL в программе. Необходимо ограничивать запрос DELETE условием WHERE, иначе можно потерять все содержимое таблицы.

DELETE FROM table_name WHERE column1 = ‘data1’;

Как только строка была удалена из базы данных, она не подлежит восстановлению.

2.6 Язык C++ и библиотека QT

Функциональность модуля реализована с помощью языка C++ и библиотеки Qt.

Язык C++ — компилируемый статически типизированный язык программирования общего назначения. Сочетает в себе свойства как высокоуровневых, так и низкоуровневых языков за счет поддержки разных парадигм программирования.

Являясь одним из самых популярных языков программирования, C++ широко используется для разработки программного обеспечения. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (например, видеоигры).

Стандарт C++ на 2003 год состоит из двух основных частей: описание ядра языка и описание стандартной библиотеки. Кроме того, существует огромное количество библиотек C++, не входящих в стандарт. Одной из таких библиотек является библиотека Qt.

Библиотека Qt - инструментарий для быстрой разработки графических интерфейсов (GUI) приложений на языке C++, с целью упростить перенос GUI-приложений с одной платформы на другую, т.е. такие приложения должны работать и в среде Windows, и в среде Unix/Linux под X11, и на компьютерах Macintosh одинаково.

В настоящее время Qt предоставляет использующему её программисту целостный фреймворк (framework), позволяющий при написании большей части приложения использовать только 'родные' классы Qt и практически полностью отказаться от написания системно-зависимого кода, использования системных вызовов или от изобретения собственных кросс-платформенных обёрток. Классы Qt удовлетворяют почти всем потребностям программиста. В Qt предусмотрены классы и для работы со строками, и для работы с файлами, сетью, базами данных, XML, и для обеспечения многопоточности в приложении, и многое-многое другое.

3 Описание алгоритма работы модуля

3.1 Описание логики алгоритма

Алгоритм включает в себя следующие этапы:

1. Выбор файла OpenStreetMap в формате OSM, в котором содержится необходимая информация.

   Выбор файла OSM осуществляется пользователем с помощью диалогового окна из конфигурационного файла.

2. Формирование запроса к OSM-файлу для извлечения геоданных об объектах необходимого типа.

   Формирование запроса происходит без участия пользователя. Для него этот процесс скрыт.

3. Формирование результата запроса, вывод извлеченных геоданных на экран.

   Результаты запроса выводятся на экран в соответствующем окне программы.

4. Формирование SQL-запроса к базе данных для сохранения извлеченных геоданных в формате OpenGIS.

   SQL-запрос формируется до запуска модуля. Формируется без участия пользователя.

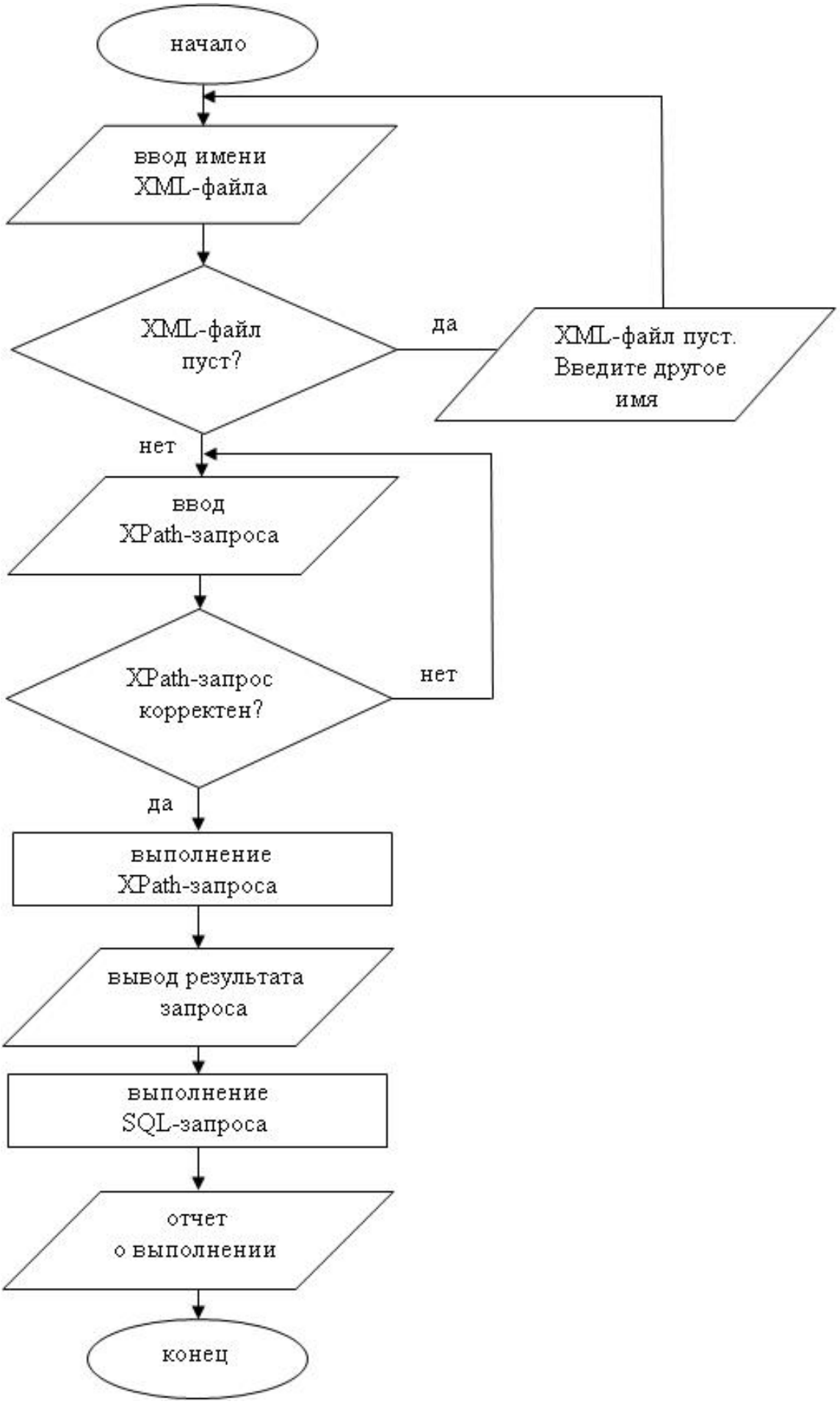
5. Формирование отчёта о занесении геоданных в базу данных.

   Отчёт о занесении геоданных в БД выводится на экран в соответствующем
   окне программы.

6. Графическое представление извлеченных геоданных.

   В соответствующем окне программы представляется графическое изображение полученных геоданных.

3.2 Блок-схема алгоритма



4 Практическая часть

4.1 Фрагменты программного кода

В этом разделе описаны основные XPath-выражения и SQL-запросы, задействованные в решении поставленной задачи, а также структура базы данных. Программный код модуля приведен в Приложении В.

4.1.1 Используемые XPath-выражения

Рассмотрим документ OSM XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
    <bounds minlat="55.1196000" minlon="37.3708000" maxlat="55.1814000" maxlon="37.5074000"/>
    <node id="1" lat="55.0269826" lon="37.4271725" user="kol" uid="73184" visible="true" version="13" changeset="813236" timestamp="2009-01-19T09:05:54Z"/>
    <node id="2" lat="55.0294985" lon="37.4277889" user="u07" uid="69700" visible="true" version="7" changeset="3437942" timestamp="2009-12-23T21:29:34Z">
        <tag k="railway" v="level_crossing"/>
    </node>
    <way id="10" user="Dimon62" uid="457129" visible="true" version="2" changeset="8303598" timestamp="2011-05-31T17:43:16Z">
        <nd ref="1"/>
        <nd ref="2"/>
        <tag k="shop" v="supermarket"/>
        <tag k="name" v="Карусель"/>
    </way>
    <relation id="20" user="Hind" uid="79836" visible="true" version="1" changeset="4008870" timestamp="2010-03-01T14:05:55Z">
        <member type="way" ref="10" role="outer"/>
        <member type="node" ref="1" role="outer"/>
        <member type="node" ref="2" role="outer"/>
        <tag k="name" v="Чеховский район"/>
        <tag k="type" v="address"/>
    </relation>
</osm>
```

Для извлечения координат границ рассматриваемой области используется XPath-запрос следующего вида:

'/osm/bounds/@minlat' - минимальная широта, результат minlat='55.1196000'
'/osm/bounds/@minlon' - минимальная долгота, результат minlon='37.3708000';
'/osm/bounds/@maxlat' - максимальная широта, результат maxlat='55.1814000';
'/osm/bounds/@maxlon' - максимальная долгота, результат maxlon='37.5074000'.

Для извлечения всех названий типа node (точка) и way (линия), используется XPath-запросы:

'/osm/way[*/@k='name']/@id' - результат: id='10';

Для получения широты и долготы точки используется запрос:

'/osm/node[@id='1']/@lat' - результат: lat='55.0269826'. Для долготы аналогично.

Остальные запросы представляют из себя комбинацию тех или иных запросов, уже описанных в этой главе и главе, посвященной XPath-запросам.

4.1.2 Структура базы данных

Для решения поставленной задачи создана пространственная база данных, включающая в себя таблицы GEOMETRY_COLUMNS и SPATIAL_REF_SYS. Кроме этого в базе данных созданы таблицы objects и tags, которые нужны непосредственно для хранения извлеченных геоданных.

Таблица objects хранит идентификаторы объектов и непосредственно сами объекты. Определяется следующим образом:

```
CREATE TABLE objests (
id_obj INTEGER NOT NULL,
geometry_obj GEOMETRY,
)
```

Колонка id_obj содержит идентификаторы объектов.

Колонка geometry_obj является геометрическим столбцом. Объект может иметь один из указанных ранее типов, например, POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON и т.д.

Таблица tags хранит в себе имя и тип тегов, относящихся к определенному объекту из таблицы objects, определяется следующим образом:

```
CREATE TABLE tags(
id_tag integer NOT NULL,
key_tag character varying(50),
value_tag character varying(50),
id_obj integer NOT NULL
)
```

Колонка id_tag содержит идентификаторы тегов.

Колонка key_tag включает тип тега (или название характеристики, или атрибута объекта) в том виде, в котором они представлены в XML-документе, например 'name', 'building', 'street' и т.д.

Value_tag содержит значения характристик (или атриутов, типов) объекта, например, у объекта с атрибутом 'name' может быть значение 'Московская область', у 'street' - 'Улица Гагарина', и т.д.

Колонка id_obj хранит в себе внешний ключ на таблицу objects, содержит идентификаторы объектов.

4.1.3 Используемые SQL-выражения

Основные SQL-запросы, использующиеся для решения поставленной задачи.

Во-первых, это запросы с использованием операторов DDL. В ходе работы созданы 4 таблицы: objects, tags, GEOMETRY_COLUMNS и SPATIAL_REF_SYS. Запрос создания таблицы objects с атрибутами id_obj и name_obj выглядит так:

```
CREATE TABLE objects (
id_obj INTEGER NOT NULL,
name_obj VARCHAR(50),
CONSTRAINT tag_pkey PRIMARY KEY (id_obj)
);
```

Запрос создания таблицы tags:

```
CREATE TABLE tags (
id_tag integer NOT NULL,
key_tag character varying(50),
value_tag character varying(50),
id_obj integer NOT NULL
);
```

Остальные таблицы построены аналогичным образом.
Если необходимо удалить таблицу, то используют следующее выражение:

```
DROP objects;
```

Если необходимо что-то изменить в таблице (например добавить новую колонку name), то выражение будет выглядеть так:

```
ALTER TABLE objects ADD name VARCHAR(50);
```

Вторая группа запросов, которая активно использовалась в работе - это операторы DML, а именно оператор INSERT. Например, чтобы внести данные в таблицу objects, например, внести объект ТОЧКА с координатами 54.4342 - широта и 34.2535 - долгота, использовано следующее выражение:

```
INSERT INTO objects (id_obj, geometry_obj)
VALUES (1, POINT(54.4342, 34.2535));
```

Для того, чтобы поместить данные в таблице tags использовано выражение:

```
INSERT INTO tags (id_tag, key_tag, value_tag, id_obj)
VALUES(1, 'name', 'Улица Гагарина', 10);
```

В таблицу tags вставлена запись, которая ссылается на запись с идентификатором 10 в таблице objects и присваивает ей тип 'name' и имя 'Улица Гагарина'.
Все остальные записи занесены в таблицы аналогичным образом.
Извлечь данные из таблиц можно с помощь оператора SELECT, например:

```
SELECT id_obj, AsText(geometry_obj) FROM objects;
```

При этом геометрические данные, которые хранятся в бинарном коде в таблице, будут извлечены из нее в текстовом узнаваемом формате, например, POINT(54.4342, 34.2535).

4.2 Описание результатов работы

Результат выполнения учебно-исследовательской работы - подключаемая статическая библиотека.

Код статической библиотеки компонуется к основной программе.

Библиотека обладает следующими возможностями:

1. Извлечение геоданных об объектах указанного типа из файла формата OSM.
2. Занесение в базу данных извлеченных геоданных в формате OpenGIS.
3. Возможность схематичного графического представления извлеченных геоданных.

Для демонстрации возможностей библиотеки в рамках учебно-исследовательской работы дополнительно создано GUI-приложение. Графический интерфейс приложения см. Рис.1.

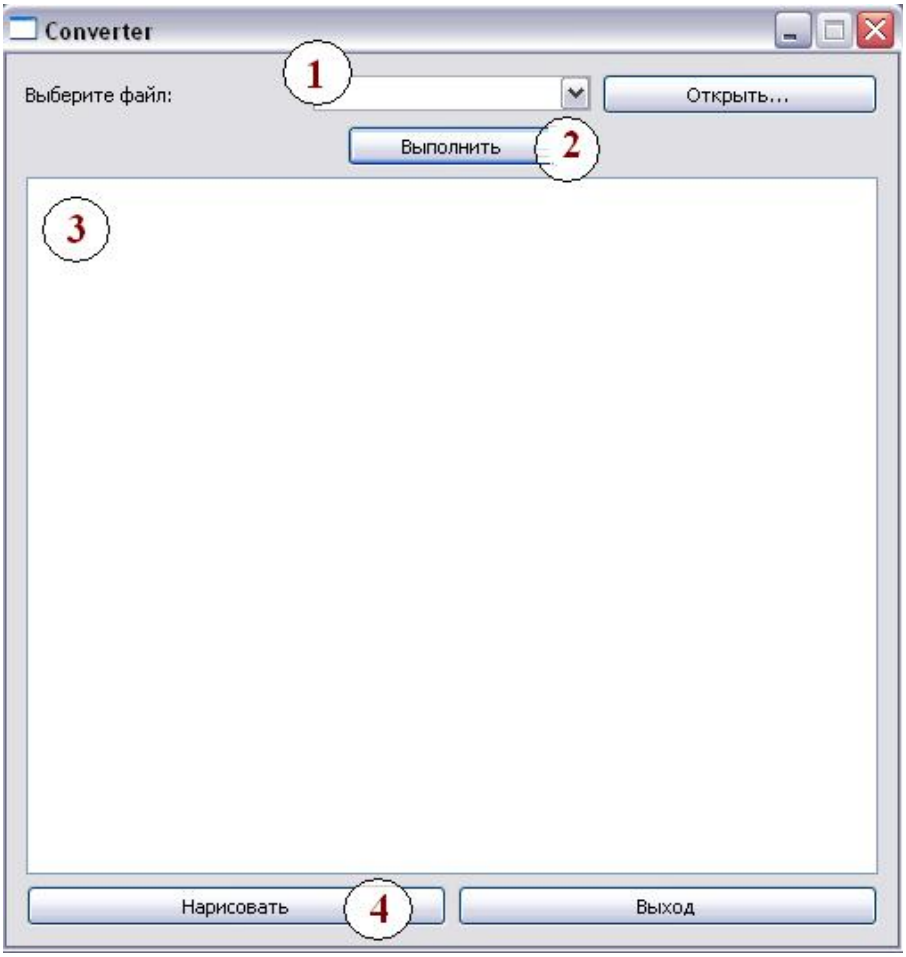


Рис.1

Цифрами 1-4 обозначены области приложения.

Приложение в совокупности с подключенной библиотекой обладает следующими возможностями:

- Область 1. Выбор интересующего файла OSM XML для извлечения из него нужной геоинформации (см. Рис.2).

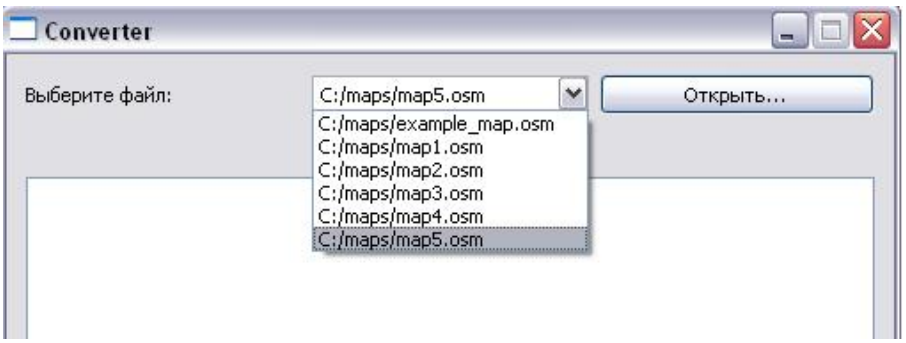


Рис.2

- Область 2. Кнопка начала извлечения геоданных.

- Область 3. Просмотр извлеченной геоинформации и занесение геоданных в базу данных в формате OpenGIS (см. Рис.3).

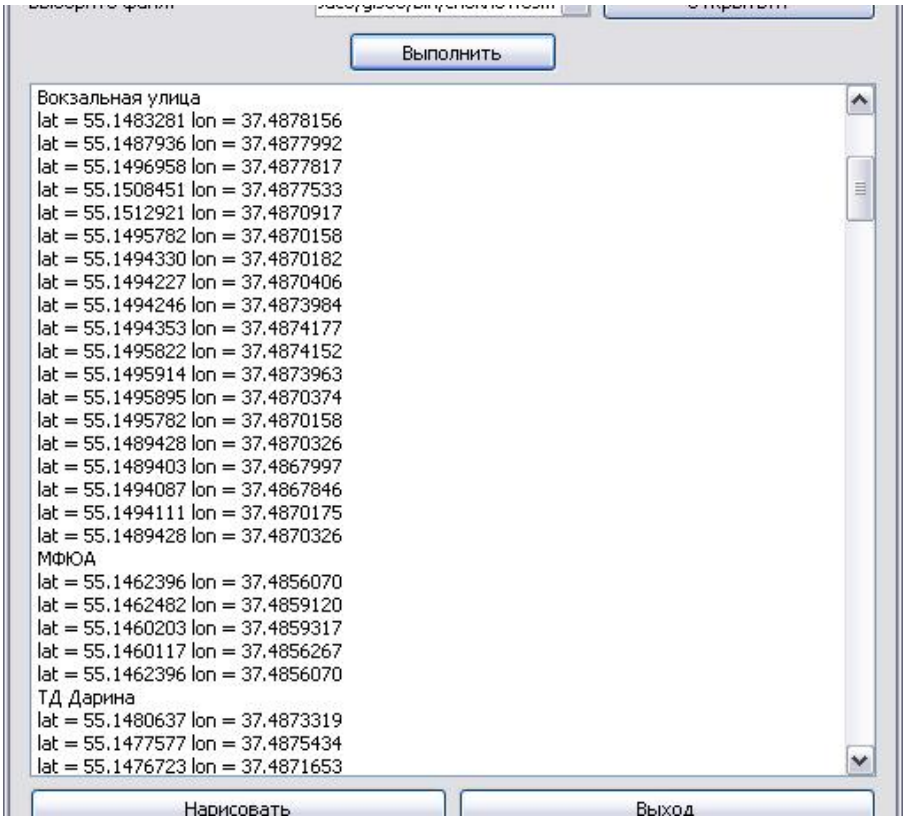


Рис.3

- Область 4. Кнопка вызова нового окна приложения, в котором схематично представляется извлеченная геоинформация (см. Рис.4).

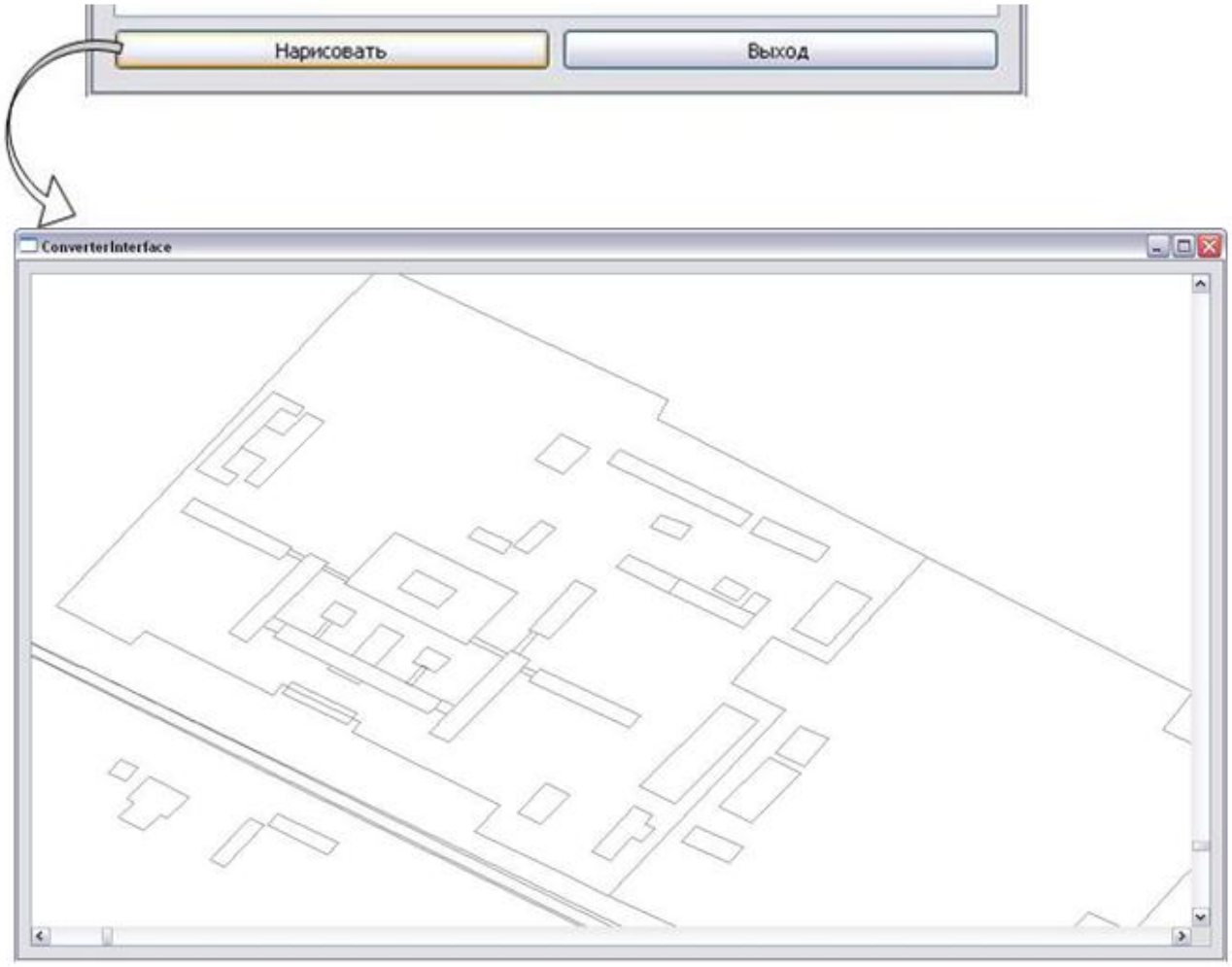


Рис.4

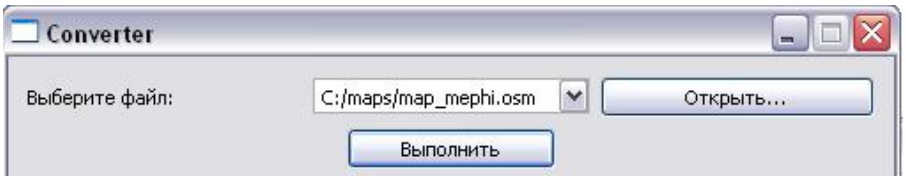
Пример работы модуля приведен ниже.

Разработанная библиотека может быть подключена к любому другому приложению, как отдельный независимый переносимый модуль.

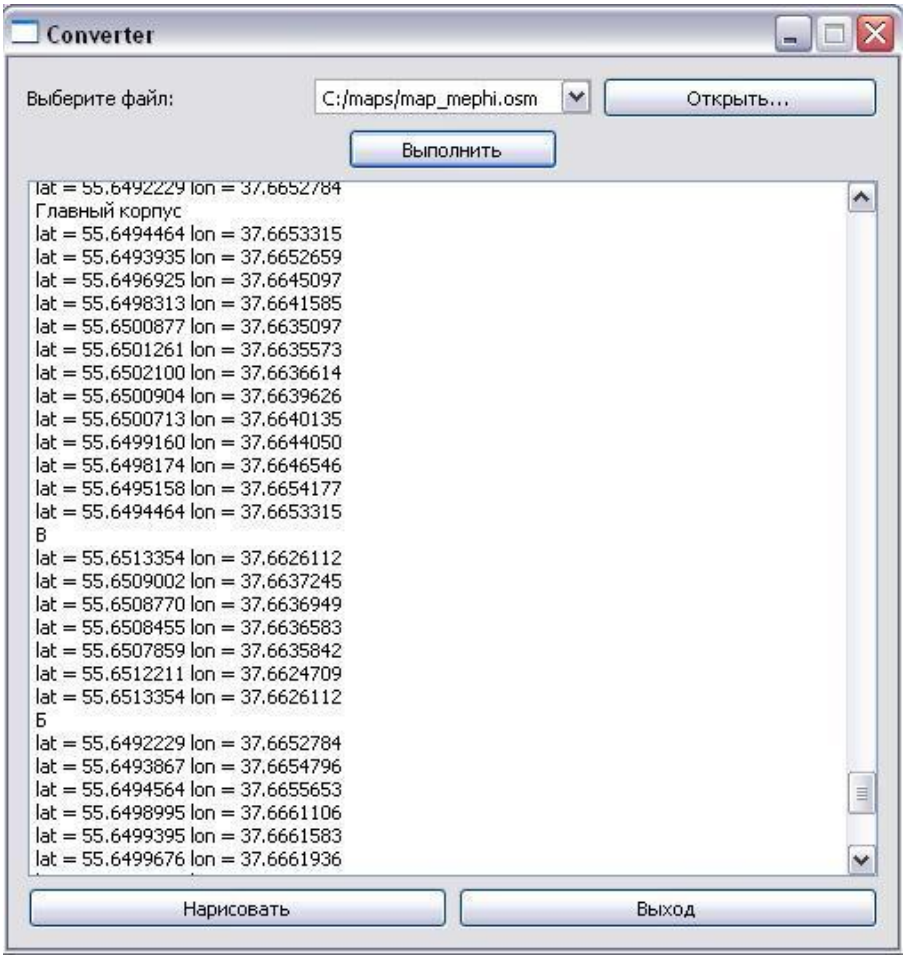
Пример работы модуля.

Для демонстрации работы модуля рассмотрим OSM-файл, в котором содержится информация о территории МИФИ.

Выбираем файл 'map_mephi.osm'.

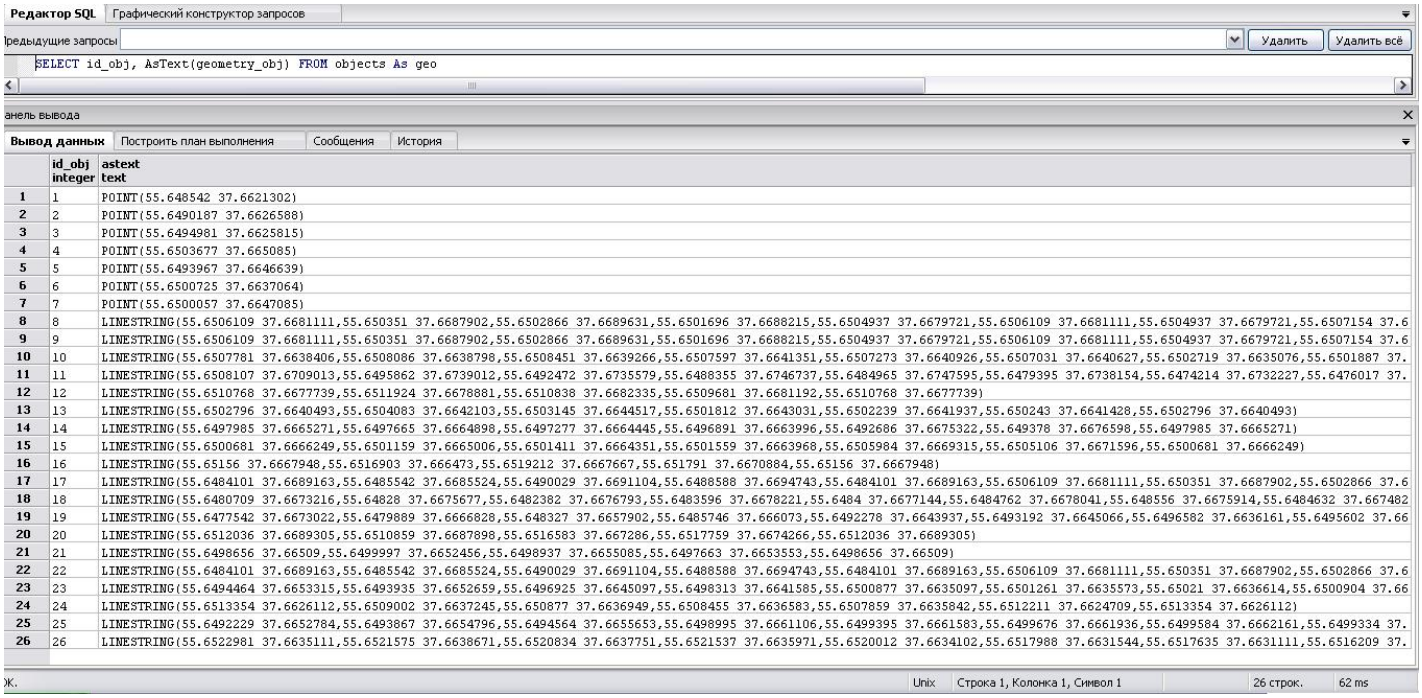


Нажатием кнопки 'Выполнить' происходит разбор OSM-файла.



Совместно с этим действием происходит занесение геоданных в базу данных. С помощью открытого графического инструмента администрирования PostgreSQL для Windows pgAdmin можно посмотреть, какие данные занесены в таблицы tags и objects.

Содержимое таблицы objects:



Редактор SQL | Графический конструктор запросов

Предыдущие запросы | Удалить | Удалить всё

```
SELECT id_obj, AsText(geometry_obj) FROM objects As geo
```

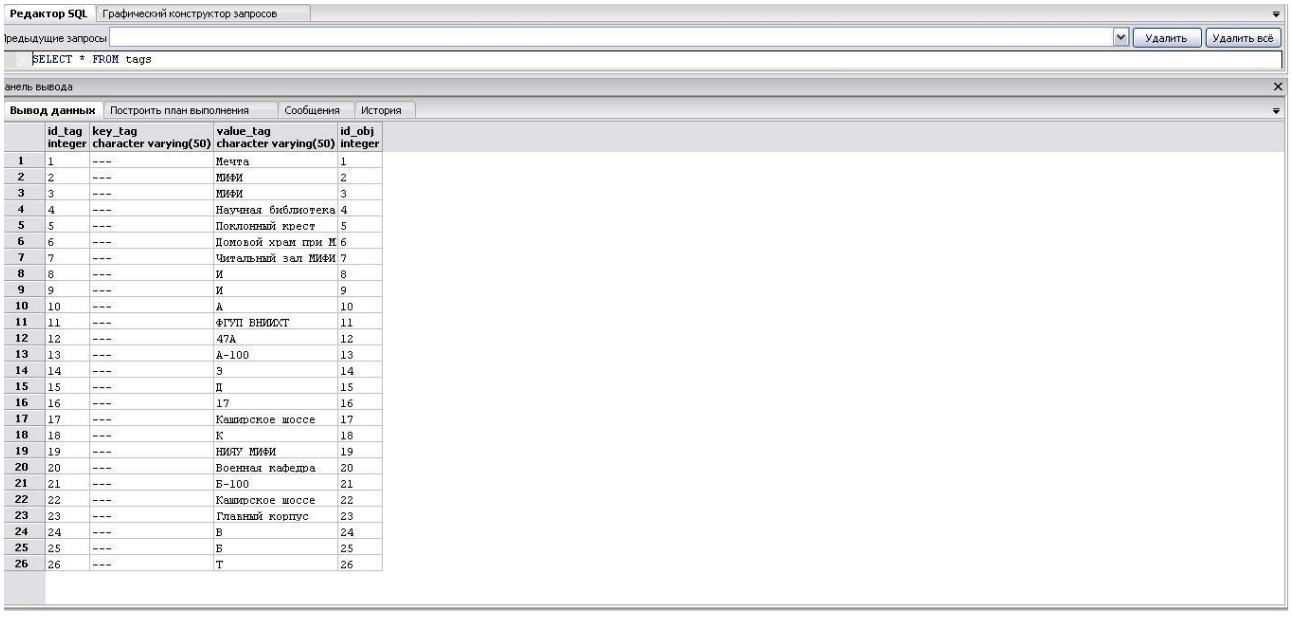
Панель вывода

Вывод данных | Построить план выполнения | Сообщения | История

	id_obj integer	astext text
1	1	POINT(55.648542 37.6621302)
2	2	POINT(55.6490187 37.6626588)
3	3	POINT(55.6494981 37.6625815)
4	4	POINT(55.6503677 37.665085)
5	5	POINT(55.6493967 37.6646639)
6	6	POINT(55.6500725 37.6637064)
7	7	POINT(55.6500057 37.6647085)
8	8	LINESTRING(55.6506109 37.6681111,55.650351 37.6687902,55.6502866 37.6689631,55.6501696 37.6688215,55.6504937 37.6679721,55.6506109 37.6681111,55.6504937 37.6679721,55.6507154 37.6
9	9	LINESTRING(55.6506109 37.6681111,55.650351 37.6687902,55.6502866 37.6689631,55.6501696 37.6688215,55.6504937 37.6679721,55.6506109 37.6681111,55.6504937 37.6679721,55.6507154 37.6
10	10	LINESTRING(55.6507781 37.6638406,55.6508086 37.6638798,55.6508451 37.6639266,55.6507597 37.6641351,55.6507273 37.6640926,55.6507031 37.6640627,55.6502719 37.6635076,55.6501887 37.
11	11	LINESTRING(55.6508107 37.6709013,55.6495862 37.6739012,55.6492472 37.6735579,55.6488355 37.6746737,55.6484965 37.6747595,55.6479395 37.6738154,55.6474214 37.6732227,55.6476017 37.
12	12	LINESTRING(55.6510768 37.6677739,55.6511924 37.6678881,55.6510838 37.6682335,55.6509681 37.6681192,55.6510768 37.6677739)
13	13	LINESTRING(55.6502796 37.6640493,55.6504083 37.6642103,55.6503145 37.6644517,55.6501812 37.6643031,55.6502239 37.6641937,55.650243 37.6641428,55.6502796 37.6640493)
14	14	LINESTRING(55.6497985 37.6665271,55.6497665 37.6664898,55.6497277 37.6664445,55.6496891 37.6663996,55.6492686 37.6675322,55.649378 37.6676598,55.6497985 37.6665271)
15	15	LINESTRING(55.6500681 37.6666249,55.6501159 37.6665006,55.6501411 37.6664351,55.6501559 37.6663968,55.6505984 37.6669315,55.6505106 37.6671596,55.6500681 37.6666249)
16	16	LINESTRING(55.65156 37.6667948,55.6516903 37.666473,55.6519212 37.6667667,55.651791 37.6670884,55.65156 37.6667948)
17	17	LINESTRING(55.6484101 37.6689163,55.6485542 37.6685524,55.6490029 37.6691104,55.6488588 37.6694743,55.6484101 37.6689163,55.6506109 37.6681111,55.650351 37.6687902,55.6502866 37.6
18	18	LINESTRING(55.6480709 37.6673216,55.64828 37.6675677,55.6482382 37.6676793,55.6483596 37.6678221,55.6484 37.6677144,55.6484762 37.6678041,55.648556 37.6675914,55.6484632 37.667482
19	19	LINESTRING(55.6477542 37.6673022,55.6479889 37.6666828,55.648327 37.6657902,55.6485746 37.666073,55.6492278 37.6643937,55.6493192 37.6645066,55.6496582 37.6636161,55.6495602 37.66
20	20	LINESTRING(55.6512036 37.6689305,55.6510859 37.6687898,55.6516583 37.667286,55.6517759 37.6674266,55.6512036 37.6689305)
21	21	LINESTRING(55.6498656 37.66509,55.6499997 37.6652456,55.6498937 37.6655085,55.6497663 37.6653553,55.6498656 37.66509)
22	22	LINESTRING(55.6484101 37.6689163,55.6485542 37.6685524,55.6490029 37.6691104,55.6488588 37.6694743,55.6484101 37.6689163,55.6506109 37.6681111,55.650351 37.6687902,55.6502866 37.6
23	23	LINESTRING(55.6494464 37.6653315,55.6493935 37.6652659,55.6496925 37.6645097,55.6498313 37.6641585,55.6500877 37.6635097,55.6501261 37.6635573,55.65021 37.6636614,55.6500904 37.66
24	24	LINESTRING(55.6513354 37.6626112,55.6509002 37.6637245,55.650877 37.6636949,55.6508455 37.6636583,55.6507859 37.6635842,55.6512211 37.6624709,55.6513354 37.6626112)
25	25	LINESTRING(55.6492229 37.6652784,55.6493867 37.6654796,55.6494564 37.6655653,55.6498995 37.6661106,55.6499395 37.6661583,55.6499676 37.6661936,55.6499584 37.6662161,55.6499334 37.
26	26	LINESTRING(55.6522981 37.6635111,55.6521575 37.6638671,55.6520834 37.6637751,55.6521537 37.6635971,55.6520012 37.6634102,55.6517988 37.6631544,55.6517635 37.6631111,55.6516209 37.

Unix | Строка 1, Колонка 1, Символ 1 | 26 строк. | 62 ms

Содержимое таблицы tags:



Редактор SQL | Графический конструктор запросов

Предыдущие запросы | Удалить | Удалить всё

```
SELECT * FROM tags
```

Панель вывода

Вывод данных | Построить план выполнения | Сообщения | История

	id_tag integer	key_tag character varying(50)	value_tag character varying(50)	id_obj integer
1	1	---	Мечта	1
2	2	---	МИФИ	2
3	3	---	МИФИ	3
4	4	---	Научная библиотека	4
5	5	---	Поклонный крест	5
6	6	---	Домовой храм при М	6
7	7	---	Читальный зал МИФИ	7
8	8	---	И	8
9	9	---	И	9
10	10	---	А	10
11	11	---	ФГУП ВНИИХТ	11
12	12	---	47А	12
13	13	---	А-100	13
14	14	---	3	14
15	15	---	Д	15
16	16	---	17	16
17	17	---	Каширское шоссе	17
18	18	---	К	18
19	19	---	НИЯУ МИФИ	19
20	20	---	Военная кафедра	20
21	21	---	Б-100	21
22	22	---	Каширское шоссе	22
23	23	---	Главный корпус	23
24	24	---	В	24
25	25	---	Б	25
26	26	---	Т	26

Рассмотрим объект 'Главный корпус'. Он представляется в OSM-документе следующим образом:

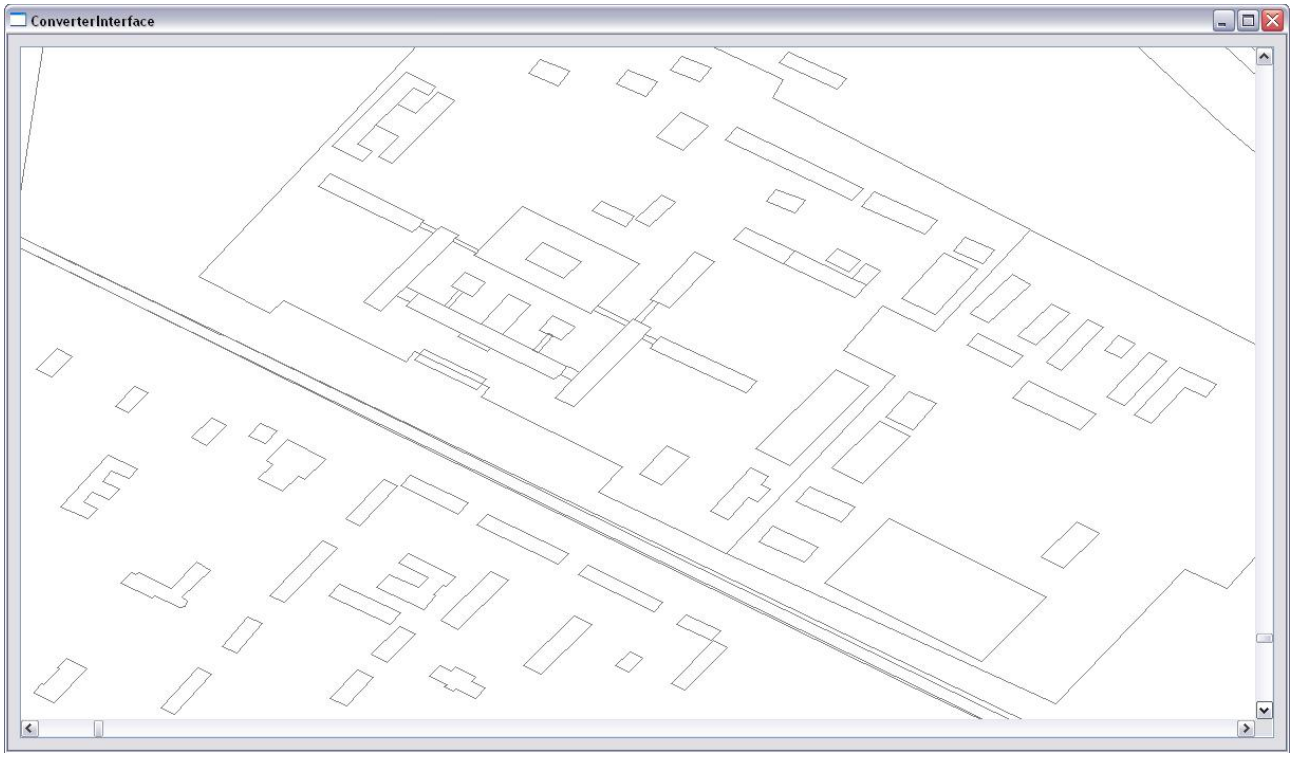
Главный корпус
lat = 55.6494464 lon = 37.6653315
lat = 55.6493935 lon = 37.6652659
lat = 55.6496925 lon = 37.6645097
lat = 55.6498313 lon = 37.6641585
lat = 55.6500877 lon = 37.6635097
lat = 55.6501261 lon = 37.6635573
lat = 55.6502100 lon = 37.6636614
lat = 55.6500904 lon = 37.6639626
lat = 55.6500713 lon = 37.6640135
lat = 55.6499160 lon = 37.6644050
lat = 55.6498174 lon = 37.6646546
lat = 55.6495158 lon = 37.6654177
lat = 55.6494464 lon = 37.6653315

В базу данных этот объект занесен в таблицу tags с идентификатором '23' с атрибутом 'name = Главный корпус' и ссылкой на запись таблицы objects с идентификатором '23'. Под этим идентификатором в таблице objects хранятся координаты объекта 'Главный корпус', а именно:

'LINESTRING(55.6494464 37.6653315,55.6493935 37.6652659, 55.6496925 37.6645097,55.6498313 37.6641585,55.6500877 37.6635097, 55.6501261 37.6635573,55.65021 37.6636614,...)'

В базе данных объект хранится в виде, соответствующем спецификации OpenGIS.

Графическое представление извлеченных данных.



5 Выводы

При выполнении учебно-исследовательской работы пройдены основные этапы разработки специализированного прикладного программного обеспечения преобразования геоданных из формата OpenStreetMap в формат OpenGIS:

- формализация задачи;
- сбор и обобщение необходимых исходных данных, используемых в программе;
- составление блок-схемы алгоритма решения задачи;
- написание и редакция программного кода.

В процессе выполнения работы выполнены следующие шаги:

1. Изучение структуры документа OSM XML и анализ геоданных, хранимых в данном документе.
2. Изучение синтаксиса и возможностей языка XPath. Формирование XPath-запросов для извлечения геоданных об объектах указанного типа.
3. Изучение особенностей СУБД PostgreSQL и её расширения PostGiS.
4. Изучение языка SQL. Формирование SQL-запросов для занесения в базу данных необходимой геоинформации.
5. Разработка модуля на языке C++ с помощью библиотеки Qt.

Разработанный программный модуль :

- может являться основой для разработки геоинформационных систем различных областей применения;
- позволяет оптимизировать работу разработчика ГИС путем получения исходных данных в стандартном формате OpenGIS.

Исходный код модуля написан на языке Visual C++ и представлен в практической части.

Программа разработана в кросс-платформенной среде разработки - Qt Creator 4.7.4.

Пояснительная записка набрана в LaTeXe при помощи программы WinEdt и оттранслирована в формат '.pdf' при помощи пакета MikTeX.

6 Приложение

6.1 Приложение А. Список осей и функций XPath

Оси XPath.

- ancestor:: — Возвращает множество предков.
- ancestor-or-self:: — Возвращает множество предков и текущий элемент.
- attribute:: — Возвращает множество атрибутов текущего элемента.
- child:: — Возвращает множество потомков на один уровень ниже.
- descendant:: — Возвращает полное множество потомков.
- descendant-or-self:: — Возвращает полное множество потомков и текущий элемент.
- following:: — Возвращает необработанное множество, ниже текущего элемента.
- following-sibling:: — Возвращает множество элементов на том же уровне, следующих за текущим.
- namespace:: — Возвращает множество имеющее пространство имён (то есть присутствует атрибут xmlns).
- parent:: — Возвращает предка на один уровень назад.
- preceding:: — Возвращает множество обработанных элементов исключая множество предков.
- preceding-sibling:: — Возвращает множество элементов на том же уровне, предшествующих текущему.
- self:: — Возвращает текущий элемент.

Системные функции.

- node-set document(object, node-set?) - возвращает документ, указанный в параметре object.
- string format-number(number, string, string?) - форматирует число согласно образцу, указанному во втором параметре, третий параметр указывает именованный формат числа, который должен быть учтён.
- string generate-id(node-set?) - возвращает строку, являющуюся уникальным идентификатором.
- node-set key(string, object) - возвращает множество с указанным ключом (аналогично функции id для идентификаторов).

- string unparsed-entity-uri(string) - возвращает непроанализированный URI, если такового нет, возвращает пустую строку.
- boolean element-available(string) - проверяет, доступен ли элемент или множество, указанное в параметре. Параметр рассматривается как XPath.
- boolean function-available(string) - проверяет, доступна ли функция, указанная в параметре. Параметр рассматривается как XPath.
- object system-property(string) - параметры, возвращающие системные переменные, могут быть: - xsl:version — возвращает версию XSLT процессора.

  - xsl:vendor — возвращает производителя XSLT процессора.

  - xsl:vendor-url — возвращает URL, идентифицирующий производителя.

  Если используется неизвестный параметр, функция возвращает пустую строку.
- boolean lang(string) - возвращает истину, если у текущего тега имеется атрибут xml:lang, либо родитель тега имеет атрибут xml:lang и в нем указан совпадающий строке символ.

Функции с множествами.

* — обозначает любое имя или набор символов, @* — любой атрибут.

$name - обращение к переменной, где name — имя переменной или параметра.

[] — дополнительные условия выборки.

{} — если применяется внутри тега другого языка (например HTML), то XSLT процессор рассматривает содержимое фигурных скобок как XPath.

/ — определяет уровень дерева.

- node-set node() - возвращает все узлы. Для этой функции часто используют заменитель '*', но в отличие от звездочки — node() возвращает и текстовые узлы.
- string text() - возвращает набор текстовых узлов
- node-set current() - возвращает множество из одного элемента, который является текущим. Если мы делаем обработку множества с условиями, то единственным способом дотянуться из этого условия до текущего элемента будет данная функция.
- number position() - возвращает позицию элемента в множестве. Корректно работает только в цикле <xsl:for-each/>
- number last() - возвращает номер последнего элемента в множестве. Корректно работает только в цикле <xsl:for-each/>
- number count(node-set) - возвращает количество элементов в node-set.

- string name(node-set?) - возвращает полное имя первого тега в множестве.
- string namespace-uri(node-set?) - возвращает ссылку на url определяющий пространство имён.
- string local-name(node-set?) - возвращает имя первого тега в множестве, без пространства имён.
- node-set id(object) - находит элемент с уникальным идентификатором

Строковые функции.

- string string(object?) - возвращает текстовое содержимое элемента. По сути возвращает объединенное множество текстовых элементов на один уровень ниже.
- string concat(string, string, string*) - объединяет две или более строк
- number string-length(string?) - возвращает длину строки.
- boolean contains(string, string) - возвращает истину, если первая строка содержит вторую, иначе возвращает ложь.
- string substring(string, number, number?) - возвращает строку вырезанную из строки начиная с указанного номера, и если указан второй номер — количество символов.
- string substring-before(string, string) - если найдена вторая строка в первой, возвращает строку до первого вхождения второй строки.
- string substring-after(string, string) - если найдена вторая строка в первой, возвращает строку после первого вхождения второй строки.
- boolean starts-with(string, string) - возвращает истину, если вторая строка входит в начало первой, иначе возвращает ложь.
- boolean ends-with(string, string) - возвращает истину, если вторая строка входит в конец первой, иначе возвращает ложь.
- string normalize-space(string?) - убирает лишние и повторные пробелы, а также управляющие символы, заменяя их пробелами.
- string translate(string, string, string) - заменяет символы первой строки, которые встречаются во второй строке, на соответствующие позиции символам из второй строки символы из третьей строки.

Логические функции

- or — логическое «или»

- and — логическое «и»
- = — логическое «равно»
- < (<) — логическое "меньше"
- > (>) — логическое "больше"
- <= (<=) — логическое "меньше либо равно"
- >= (>=) — логическое "больше либо равно"

Числовые функции

- + — сложение
- - — вычитание
- * — умножение
- div — обычное деление (не деление нацело!)
- mod — остаток от деления

6.2 Приложение В. Программный код модуля.