Министерство образования Российской Федерации

Томский государственный университет

факультет информатики

кафедра теоретических основ информатики

ДОПУСТИТЬ К ЗАЩИТЕ В ГАК
зав. кафедрой, к.т.н., доцент:
_____ Костюк Ю. Л.
«_____»_____2002г.

Холкин Вадим Валерьевич

Разработка алгоритмов построения цифровой модели рельефа

Дипломная работа

Научный руководитель:
ст. преподаватель каф. ТОИ
_____ Фукс А. Л.
Автор работы:
студент группы 1471:
_____ Холкин В. В.

Томск 2002

РЕФЕРАТ

Отчет о преддипломной практике на 58 стр., 25 рисунков, 7 таблиц, 6 источников.

ЦИФРОВАЯ МОДЕЛЬ РЕЛЬЕФА, ШЕЙП-ФАЙЛЫ, ИЗОЛИНИИ, ВЫСОТНЫЕ ОТМЕТКИ, РЕДАКТОР КАРТ, ТРИАНГУЛЯЦИЯ ДЕЛОНЕ, ArcView GIS, DELPHI.

(1) Объект исследования – цифровая модель рельефа.

(2) Цель работы: разработка алгоритмов построения цифровой модели рельефа.

(3) Метод исследования: аналитический и экспериментальный на ЭВМ.

(4) Основные результаты: разработана система построения цифровой модели рельефа на основе триангуляции Делоне с ограничениями.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В последнее десятилетие интенсивно развиваются самостоятельные и актуальные направления информатики – геоинформатика и вычислительная геометрия. Важными объектами исследования современных геоинформационных систем являются трехмерные однозначные поверхности, и, прежде всего земной рельеф. Как правило, исследуемая поверхность задается нерегулярным набором точек большого объема. В качестве ограничений дополнительно задаются структурные линии поверхности (например, горизонтали, границы обрывов и оврагов, береговые линии рек).

Важнейшая задача при работе с поверхностью заключается в построении ее цифровой модели, т.е. цифрового представления с помощью прямоугольной или треугольной сетки, в узлах которой заданы значения z (высоты). Цифровая модель позволяет получить производные данные для обработки и анализа поверхности.

Для получения треугольной сетки по проекциям исходных точек на плоскости XOY строится триангуляция. Задание высот в вершинах триангуляции определяет систему пространственных треугольников – простейшую кусочно-линейную поверхность. При переходе от исходных точек к прямоугольной сетке необходимо рассчитать высоты в ее узлах. Для этого обычно применяются различные алгоритмы локальной аппроксимации и интерполяции по значениям высот в нескольких соседних исходных точках.

Основным преимуществом прямоугольной сетки является ее простая и удобная структура, которая требует сохранения только значений высот и позволяет упростить алгоритмы анализа и обработки поверхности. Однако прямоугольная сетка имеет два существенных недостатка:

- в получаемой цифровой модели утрачиваются исходные точки, а они обычно задаются с большой точностью и представляют локальные экстремумы, узлы структурных линий и другие характерные точки;
- при использовании прямоугольной сетки практически невозможно учитывать ограничения, налагаемые структурными линиями поверхности.

В цифровой модели на основе треугольной сетки эти недостатки отсутствуют.

Исходные данные для построения модели рельефа получаются на основе методов дистанционного зондирования или оцифровки картографических материалов. В большинстве случаев эти данные требуют предварительной обработки, которая включает устранение топологических нарушений – недопустимых пересечений линий, вырожденных отрезков, висячих вершин, а также разрывов структурных линий на границах соседних планшетов (участков местности). Очень важной и трудоемкой задачей является проверка правильности задания высот исходных линий и точек.

Однако даже на основе топологически корректных данных не всегда удается построить качественную цифровую модель рельефа. Это связано, прежде всего, с тем, что известные алгоритмы триангуляции с ограничениями не учитывают эти ограничения в полной мере. В результате получаемая кусочно-линейная поверхность искусственно спрямляет рельеф, игнорируя априорно известные изломы.

Настоящая дипломная работа посвящена разработке алгоритмов проверки топологической корректности исходных данных, объединения наборов точек и линий нескольких планшетов для построения единой цифровой модели, а также триангуляции с двумя типами ограничений, позволяющими получить кондиционную модель рельефа.

1. МОДЕЛИ РЕЛЬЕФА

1.1. Входные данные

Любая работа с рельефом требует наличие модели поверхности. Модель строится по входным данным, среди которых, как правило, различают:

точки – замеры высот на поверхности

структурные линии – соответствующие обычно каким-либо изменениям в гладкости или непрерывности поверхности (часто их называют структурными линиями рельефа). Такие линии описывают различия в поведении поверхности по обе стороны от них. Примерами могут служить береговые линии, линии оврагов и обрывов, дамбы, линии инженерных построек и т.д.

изолинии – линии определенного уровня, полученные в результате сечения местности горизонтальной плоскостью с определенным шагом по высоте.

полигоны, определяющие области, вне которых отсутствует достоверная информация по поверхности. Использование таких регионов часто предупреждает генерацию ошибочных данных по рельефу вне региона.

1.2. Цифровая модель рельефа

Цифровая модель рельефа (ЦМР) – это "… средство цифрового представления трехмерных пространственных объектов (поверхностей, рельефов) … в виде совокупности высотных отметок (отметок глубин) и иными значениями аппликат (координат Z) в узлах регулярной сети, нерегулярной треугольной сети или как совокупность записей горизонталей

(изогипс, изобат) или других изолиний…” [1]. ЦМР может использоваться в различных приложениях для обработки и получения производных данных: углов наклона, зон видимости, поперечных сечений, вычисления объемов земляных работ, интерполяции высот, выделения бассейнов и водоразделов и т.д.

ЦМР может быть построена с применением методов дистанционного зондирования или с помощью картографических материалов.

1.2.1. Регулярная сеть

Это формат представления поверхности в виде матрицы равномерно распределенных точек, каждая из которых характеризуется своей высотой. В зависимости от способа вычисления высот поверхности в пространстве между точками различают «решеточную» и «ячеистую» модели. В первой из них такие значения интерполируются по значениям высот в нескольких соседних точках, вторая же модель рассматривает эти точки как центры ячеек с постоянным **z** значением .

Использование «решеточной» регулярной сети имеет смысл в случае представления такой сетью рельефа, самой поверхности. В этом случае используемая интерполяция гарантирует непрерывность ее представления. В случае же, если в качестве **z** значений используются категорийные данные (например, степень озеленения данной местности и т.п.), которые необязательно должны быть непрерывными, разумнее использовать «ячеистую».

1.2.2. Триангуляции

Это одна из форм представления поверхности по нерегулярно заданной системе отсчетов. Образованная совокупностью точек с **x, y, z** координатами

и набором ребер соединяющих эти ребра в треугольники, такая модель часто использует меньшее число точек, чем другие модели. Это объясняется тем, что исходные точки обычно указываются в оптимальных местах (пиках, впадинах), что позволяет «куски» неменяющейся поверхности представлять одним-двумя треугольниками а не разбивать ее на совокупность единообразных маленьких частей (как в случае с регулярной сетью). На плоскости среди всевозможных видов триангуляции часто пользуются триангуляцией **Делоне,** которая удовлетворяет требованию (см. рис. 1):

окружность проведенная через три вершины любого треугольника не должна содержать в себе никаких других точек.

Треугольники в таком случае становятся настолько «равноугольными», насколько возможно. Гарантируется также, что любая точка на плоскости наиболее близка к вершинам треугольника, внутрь которого она попала, чем к другим точкам.

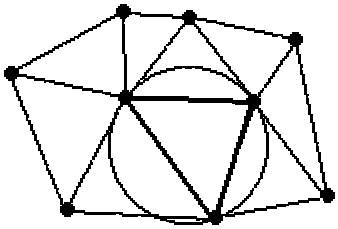


Рис. 1. Триангуляция Делоне

Для построения кусочно-линейной модели поверхности как раз и будем использовать триангуляцию.

1.3. Теоретическое обоснование применения триангуляции в качестве модели рельефа

Обзор существующих моделей рельефа показал, что, выбирая способ представления поверхности, всегда приходится идти на компромисс между точностью и качеством решений с одной стороны и стоимостью вычислений – с другой.

Учитывая тенденцию к увеличению производительности компьютеров и, как следствие, снижению стоимости вычислений, основное внимание при анализе было уделено вопросу построения модели поверхности в виде триангуляции.

Итак, еще раз повторившись, скажем, что триангуляция – это представление поверхности в виде совокупности смежных треугольников. Строится она обычно по исходному множеству точек, которые затем становятся вершинами треугольников. Для точек с пространственными, а не плоскими координатами по существу не определена форма наиболее эффективной триангуляции, но считая, что точки (их **z** координаты) представляют значения однозначной функции поверхности, т.е. каждой паре (**x, y**) соответствует не более одного значения **z**, можно построить эффективную триангуляцию **Делоне** для проекций точек на плоскость **XY**. Восстанавливая, затем **z**-значения у точек получим 3-х мерную модель поверхности (см. рис.2).

Классический вариант построения триангуляции, не рассматривает ни каких других входных данных кроме точек. Тем не менее, как уже говорилось раньше, изолинии и структурные линии, также могут задавать рельеф, довольно сильно корректируя модель, построенную только по точкам.

В данной дипломной работе построение триангуляции было решено проводить по точкам, изолиниям и структурным линиям. После выбора в качестве модели рельефа триангуляции необходимо было отыскать наиболее эффективный способ ее построения. Анализ проводился среди алгоритмов, трудоемкость которых приближалась к оптимальной ($n \cdot \log n$) и построенная триангуляция была бы по возможности «лучшей» (см. п. 3.1 – триангуляция **Делоне**, а также литературу [2]).

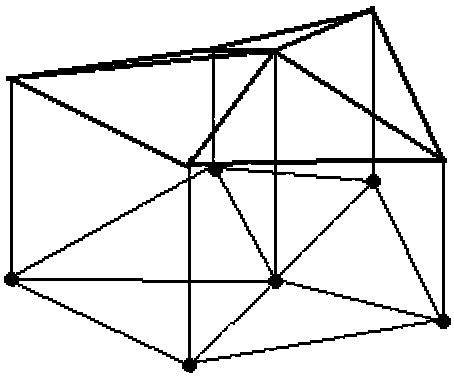


Рис. 2. Трехмерная модель восстановленная по координате Z

Алгоритмы построения триангуляции **Делоне** условно делятся на два класса – итеративные и построенные с использованием принципа «разделяй и властвуй». Их всестороннее и детальное изучение в работе «Отчет о производственной практике» провел А. В. Скворцов (Томск, ТГУ, 1995г.). На основе анализа этой работы был выбран итеративный метод с использованием динамического кеширования, который обладает наиболее предпочтительной среди всех остальных методов трудоемкостью.

Данный алгоритм предполагалось снабдить средством поддержки суперструктуры (фигура – обычно одна из наиболее простых: прямоугольник, треугольник, которая охватывает все множество точек). О ее функциональном назначении (более подробно смотри п. 3.3). Здесь лишь скажем, что она позволяет избежать некоторых вычислительных ошибок при построении.

Таким образом, всесторонне исследовав, с точки зрения представления ею модели рельефа, нами была сформулирована задача: по исходному множеству точек, которые задают отсчеты на местности необходимо построить триангуляцию **Делоне** для проекций точек на плоскость, а затем восстановить значения их **z** координат. Требуется также иметь возможность учета структурных линий рельефа и изолиний. Построение не должно отнимать много времени и требовать слишком много ресурсов.

2. ИСХОДНЫЕ ДАННЫЕ И ИХ ТОПОЛОГИЧЕСКАЯ КОРРЕКТНОСТЬ

2.1. Сшивание линий на границах планшетов.

При оцифровке карт информацию получают в виде отдельных планшетов. Это удобно с точки зрения обработки информации, т.к. объем данных очень велик. Как правило, на границах соседних планшетов линии несколько смещены и не подходят точно друг к другу. Для создания корректной модели рельефа возникает необходимость сшивания линий на границах планшетов. Универсальные ГИС позволяют решить данную задачу, но это решение не является тривиальным, т.к. занимает длительное время и возможно только в интерактивном режиме. Учитывая данную сложность, возникла задача сшивания линий на границах планшетов в полуавтоматическом режиме.

Решение данной задачи представляется следующим образом:

- Выделяется некоторая область, в которой граничат соседние планшеты;
- Заводятся списки граничных точек для каждого планшета;
- Для каждой точки из первого списка ищется ближайшая точка из второго списка, имеющая аналогичную координату **Z**;
- Предлагается изменение координат **X** и **Y** у соответствующей точки.

2.2. Возможные топологические ошибки

Кондиционную модель рельефа невозможно построить без устранения возможных топологических нарушений, к которым относятся:

1. Разрывы линий на границах смежных планшетов карт, связанные с

неточностями самих карт и/или их координатной привязки при векторизации. Данное нарушение приводит к скачкообразному изменению высот на границах планшетов.

2. Вырожденные отрезки (расстояние между соседними вершинами линии меньше заданного порогового значения).
3. Самопересечение линий.
4. Полное или частичное совпадение линий. Эта ошибка может появиться во время векторизации и не обнаруживается визуально.
5. Примыкание или пересечение изолиний в точках, имеющих разную высоту. Недопустимо на однозначной поверхности.
6. Наличие висячих вершин. Если основная изолиния незамкнута, то ее конечные точки должны лежать либо на границе планшета, либо на некоторой трехмерной линии. Промежуточная изолиния может быть любой. В конечных точках незамкнутой трехмерной линии должна быть определена высота, поэтому они должны либо лежать на изолиниях, либо совпадать с высотными отметками.
7. Резкое изменение направления линии (угол поворота превышает заданный порог). Это потенциальное нарушение, которое может быть связано с ошибками векторизации (например, с включением берг-штрихов в оцифрованную линию).

Указанные нарушения можно легко обнаружить с помощью простых проверок взаимного расположения точек и/или отрезков. Однако эта задача трудоемкая, т.к. общее число точек n обычно весьма велико, и $0(n^2)$ сравнений «каждого с каждым» могут потребовать слишком большого времени. Для построения триангуляции существует эффективный алгоритм, поэтому предлагается выделять нарушения на основе триангуляции набора всех исходных точек (вершин линий и высотных отметок), а. Коррекцию ошибок в большинстве случаев следует проводить в интерактивном режиме.

Геометрические проверки будут корректными только для невырожденных объектов:

вершины отрезка или треугольника не могут совпадать, три вершины треугольника не могут лежать на одной линии и т.д. Для выполнения этих условий необходимо заранее определить пороговое значение *eps,* задающее минимальное расстояние между отдельными объектами. Например, точки, расстояние между которыми меньше *eps,* считаются совпадающими, они не могут быть вершинами одного отрезка и, следовательно, только одна их них может быть вершиной триангуляции. Точка, расположенная на расстоянии менее *eps* от отрезка, считается лежащей на данном отрезке, и, следовательно, разбивает его на два. Очевидно, что величина *eps* должна быть на несколько порядков больше величины предельной точности вычислений.

3. РАЗРАБОТКА И ИССЛЕДОВАНИЕ АЛГОРИТМОВ ТРИАНГУЛЯЦИИ ДЕЛОНЕ

3.1. Задача построения триангуляции Делоне

Еще раз остановимся на задаче построения триангуляции **Делоне**.

Итак, дано **n** точек на плоскости. Требуется соединить все эти точки непересекающимися отрезками так, чтобы образовалось планарное разбиение плоскости со следующими свойствами:

- *Триангулярность*. Разбиение должно состоять из **m** фигур, из которых **m-1** – треугольники и еще одна – внешняя бесконечная фигура.
- *Выпуклость*. Нельзя построить ни одного нового отрезка между данными точками без пересечения с уже существующими.
- *Условие* **Делоне**. Внутрь окружности, описанной вокруг любого построенного треугольника, не должна попасть ни одна из заданных точек.

3.2.Базовые алгоритмы, применяемые при построении триангуляции

3.2.1. Уравнение прямой, проходящей через заданную пару точек

Пусть даны точки A и B. Требуется записать уравнение прямой, проходящей через эти две точки.

В аналитической геометрии показывается, что если даны точки A(x,y) и B(x,y), то прямая проходящая через них, представляется уравнением:

$$\begin{vmatrix} B.x - A.x & B.y - A.y \\ x - A.x & y - A.y \end{vmatrix} = 0$$

Уравнение прямой при построении триангуляции само по себе обычно не используется, а применяется в основном для решения других задач, в основном вопросов определения взаимного расположения геометрических фигур относительно прямой.

3.2.2. Взаимное расположение двух точек относительно заданной прямой

Пусть прямая задана точками ***A***, ***B*** и заданы также точки ***C*** и ***D***. Требуется определить, расположены ли точки ***C***, ***D*** по одну сторону относительно прямой ***AB***, или по разные.

Для этого запишем уравнение прямой, проходящей через A,B в виде:

$$f(x,y) = \begin{vmatrix} B.x - A.x & B.y - A.y \\ x - A.x & y - A.y \end{vmatrix}.$$

Как известно из аналитической геометрии, если знаки чисел ***f(C.x, C.y)*** и ***f(D.x, D.y)*** разные то эти точки ***C***, ***D*** лежат по разные стороны от прямой ***AB***. Если их знаки совпадают, значит точки ***C***, ***D*** лежат по одну сторону от ***AB***. Если хотя бы одно из этих чисел равно нулю, то что соответствующая точка лежит на прямой ***AB***.

3.2.3. Определение факта пересечения двух заданных отрезков

Пусть даны два отрезка ***AB*** и ***CD*** (см. рис. 3). Требуется определить пересекаются ли они.

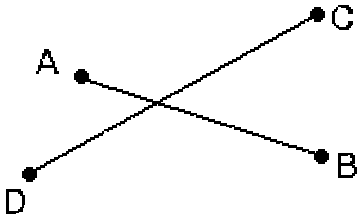


Рис. 3. Пересечение отрезков

Условия пересечения отрезков ***AB*** и ***CD*** эквивалентно следующему: точки ***C***,***D*** находятся не на одной стороне от прямой ***AB*** и точки ***A***,***B*** находятся не на одной стороне от прямой ***CD***. Заметим, что в соответствии с таким определением пересечения касание отрезков рассматривается как пересечение.

3.2.4. Определение попадания точки в треугольник

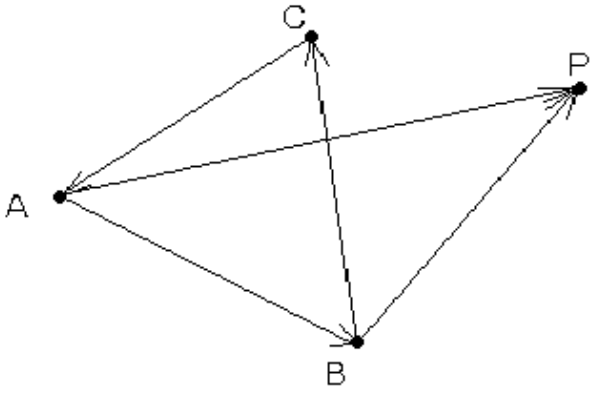
Пусть на плоскости даны треугольник ***ABC*** и некоторая точка ***P***. Необходимо определить попадает ли точка ***P*** в треугольник ***ABC*** (см. рис. 4, 5). Договоримся, что обход треугольника мы производим против часовой стрелки. Теперь рассмотрим векторное произведение векторов ***AB*** и ***AP***. В

$$[AB,\ AP] = \bar{k} \cdot ((B.x - A.x) \cdot (P.y - A.y) - (P.x - A.x) \cdot (B.y - A.y),$$

декартовых координатах это будет:

где k – вектор декартового базиса: перпендикулярный плоскости ABC. При этом, как известно, вектора ***AB***, ***AP***, ***[AB, AP]*** образуют правую тройку векторов. Поэтому, если $\angle BAP$ обходим против часовой стрелки, то вектор ***[AB, AP]*** торчит “вверх из листа бумаги” и величина

d=(B.x – A.x)*(P.y – A.y) – (P.x – A.x)*(B.y – A.y) будет больше нуля. Иначе если $\angle BAP$ обходим по часовой стрелке, то вектор ***[AB, AP]*** торчит “вниз” и $d<0$. Если $\angle BAP=0$, или $\angle BAP=180°$, то $d=0$.



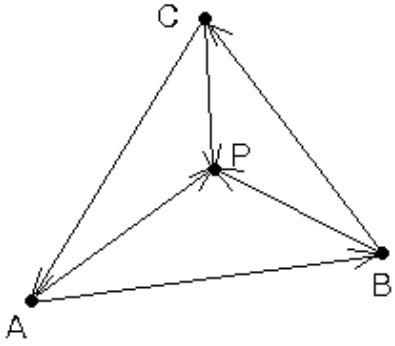


Рис. 4. Точка P вне треугольника ABC **Рис. 5**. Точка P в ABC

3.2.5. Определение факта попадания заданной точки внутрь окружности, описанной вокруг заданного треугольника

Пусть дан треугольник ***ABC*** и точка ***D***. Требуется определить, попадает ли точка ***D*** строго внутрь окружности описанной вокруг треугольника ***ABC***.

Данная задача непосредственно используется для проверки построенной триангуляции на условие **Делоне,** в котором требуется, чтобы никакая точка ***D*** не попадала внутрь никакой описанной вокруг треугольников триангуляции окружности (см. рис. 6).

Данная задача имеет два разных решения. Первое, когда строится уравнение окружности, проходящее через данные точки ***ABC***, и делается проверка на принадлежность точки ***D*** соответствующему кругу.

Второе решение основывается на другой эквивалентной формулировке условия **Делоне**. Условие **Делоне** верно тогда и только тогда, когда ***max(min(∠ABC, ∠BCA, ∠CAB), min(∠ADC, ∠DCA, ∠CAD))≤ max(min(∠ABD, ∠BDA, ∠DAB), min(∠BDC, ∠DCB, ∠CBD))***.

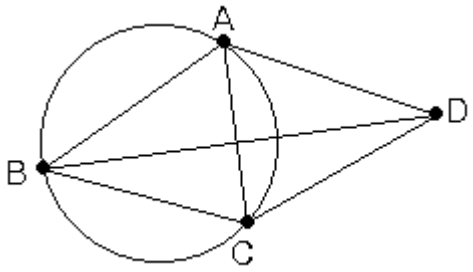


Рис. 6. Условие Делоне

При написании программы использовался алгоритм на базе проверки минимума углов. По сути дела считались не все углы, а только лишь углы противолежащие минимальным сторонам.

3.3.Построение модели поверхности на основе триангуляции Делоне

На вход задачи построения рельефа поверхности подавались точки – отсчеты высот на поверхности изолинии и структурные линии рельефа. Требовалось построить модель рельефа. Точки и линии были заданы в виде shape-файлов, их высоты хранились в одном из атрибутов.

После небольшой предварительной обработки, заключающейся в подсчете границ охватывающего прямоугольника, из исходных точек и линий составлялся массив в памяти, который подавался затем на вход алгоритма построения триангуляции **Делоне**.

Как уже упоминалось выше, для построения триангуляции был использован итеративный алгоритм с использованием динамического кеширования. В двух словах опишем основные принципы этого алгоритма (более подробно его описание смотри в работе «Отчет о производственной практике» А.В. Скворцова).

Его основой являлся обычный итеративный алгоритм, в котором была улучшена процедура поиска треугольника куда попадает очередная точка (идея итеративного алгоритма: пусть имеем триангуляцию, построенную на

некотором подмножестве исходного множества точек – частичную триангуляцию. Первую частичную триангуляцию можно получить, соединив между собой любые три исходные точки. Добавляем в нее по одной точке из оставшихся, изменяя, возможно, в соответствии с условием **Делоне** связи между вершинами. Одна из задач, которую приходится решать при этом – это определение треугольника, в который попадает новая добавляемая точка).

Улучшение процедуры поиска треугольника, куда попадает очередная точка, заключалось в добавлении особой структуры – *динамического кэша*, которая хранила ссылки на ближайшие к любой точке плоскости треугольники.

3.4. Триангуляция с ограничениями

Если искомый рельеф задан только набором исходных точек, то его кусочно-линейную модель можно построить с помощью обычной триангуляции Делоне проекций точек на *XOY*. Структурные линии задают дополнительные ограничения рельефа, и для их учета необходимо строить триангуляцию с *ограничениями*. Будем считать, что структурные линии являются ломаными, и выделим два типа ограничений триангуляции.

1. **Слабые.** Каждый отрезок структурной линии должён совпадать с одним из ребер триангуляции. Это позволяет включать линии в модель поверхности.
2. **Сильные.** Выполняются слабые ограничения, и в пространственной триангуляции отсутствуют горизонтальные ребра, лежащие на уровнях изолиний и не совпадающие с отрезками структурных линий. Такие ограничения гарантируют отсутствие плоских горизонтальных участков, по крайней мере, на заданных уровнях.

3.4.1. Построение триангуляции со слабыми ограничениями

Слабые ограничения используются в большинстве известных систем построения моделей рельефа.

Алгоритм триангуляции, предложенный в пункте 3.1. настоящей работы позволяет быстро перестроить треугольную сетку при добавлении новой точки, используя динамическое кеширование поиска. При построении триангуляции с ограничениями требуются дополнительные структуры данных. Чтобы сохранить информацию о линиях, достаточно для каждой точки хранить информацию о предыдущей и последующей точке или **nil.** Если линии соприкасаются, пересекаются, или высотная отметка расположена на линии, то в исходном наборе обязательно окажутся совпадающие точки. При этом только одна из них может быть включена в триангуляцию, следовательно, чтобы не потерять информацию об отрезках, необходимо для каждой вершины триангуляции хранить списки совпадающих с ней точек. Кроме того, нужно сохранять связи между обработанными точками и треугольниками.

Предварительная обработка линий исключает случаи пересечения отрезков линий или попадания точек внутрь отрезка.

Задача построения триангуляции со слабыми ограничениями решается следующим способом. Если при включении в триангуляцию точки ***C*** и последующей проверкой условия **Делоне** выяснилось, что необходимо перестроение, то просматриваем ребро, которое необходимо перестроить. Если это ребро является отрезком ***AB*** изолинии (т.е. точки ***A*** и ***B*** являются соседними в изолинии), то находим центр данного отрезка, в этот центр ставим новую точку ***D***, тем самым, разбивая отрезок на два отрезка. Ссылки на соседство между точками изменяются, а дальше просто производится перестроение, не обращая внимание на то, что отрезок изолинии не вошел в ребра триангуляции (см. рис. 7).

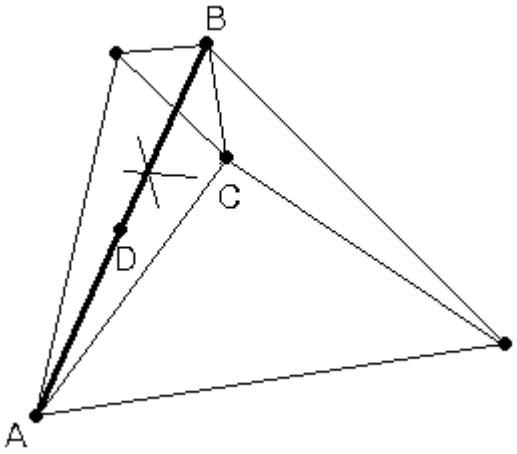


Рис. 7. Добавление новой точки D

В какой-то момент времени мы включаем в триангуляцию точку ***C*** и в зависимости от условия **Делоне** перестраиваем или нет. Если при перестроении мы видим, что отрезок линии, связывающий данную точку с предыдущей в линии или последующей в ней, не вошел в ребра триангуляции, то в центр данного отрезка добавляем новую точку ***E***, разбивая его на два отрезка (см. рис.8).

Далее, процесс повторяется, пока все отрезки изолиний не войдут в ребра триангуляции (см. рис. 9).

На практике видно, что хоть и приходится добавлять новые точек в отрезки линий, но их на самом деле не так уж и много.

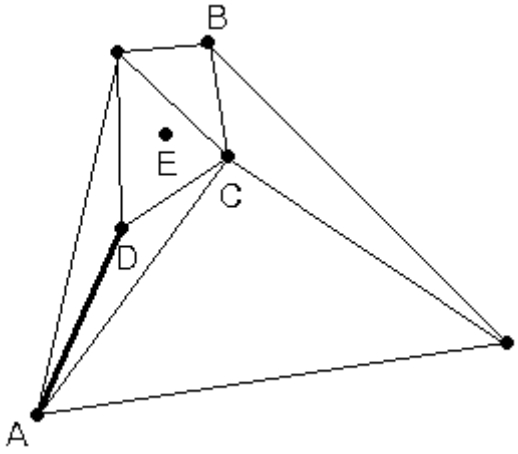


Рис. 8. Добавление новой точки E

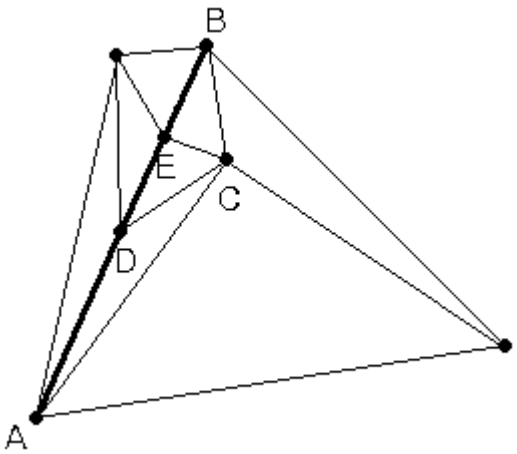
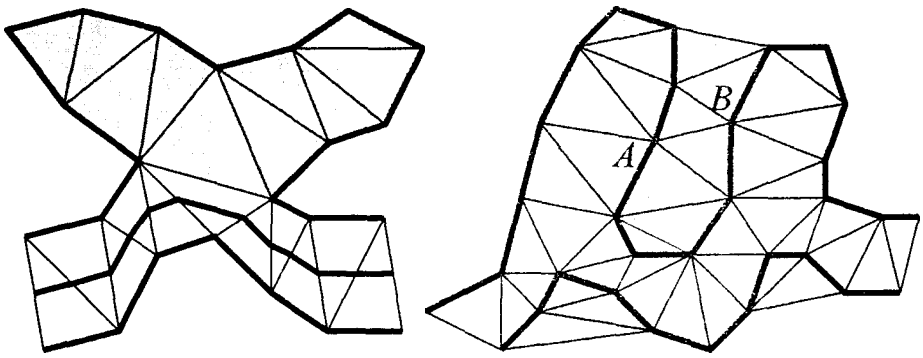


Рис. 9. Отрезок линии вошел в триангуляцию

3.4.2. Построение триангуляции с сильными ограничениями

Учет слабых ограничений при построении модели поверхности является необходимым, но недостаточным. В этом легко убедиться, если на полученной после триангуляции кусочно-линейной поверхности рассчитать изолинии промежуточных уровней (рис. 10а). По сравнению с исходными, эти линии окажутся очень спрямленными, что обусловлено наличием недопустимых горизонтальных ребер и треугольников, т.е. нарушением сильных ограничений. Отметим сразу, что устранить этот недостаток на основе сглаживания поверхности в общем случае не удастся, т.к. во многих точках будет невозможно правильно определить положение касательных плоскостей (например, в точках *A* и *B* на рис. 10б касательные плоскости окажутся горизонтальными).



(а) (б)

Рис. 10. Недостатки слабых ограничений (выделены горизонтальные треугольники): (а) - спрямление промежуточных расчетных изолиний, (б) - невозможность определения касательных в точках *A* и *B*

Необходимость учета сильных ограничений декларируется явно только в системе ArcInfo [3]: пользователь в процессе построения модели рельефа получает возможность выделить на триангуляции участки нарушений, а затем задать дополнительные точки и/или линии. Попытаемся автоматизировать данный процесс, определяя при этом минимальное число новых точек.

Назовем недопустимыми ребра, на которых сильные ограничения не выполняются, т.е. горизонтальные ребра, лежащие на уровнях изолиний, но не совпадающие с отрезками изолиний. Треугольная сетка может содержать треугольники с одним, двумя или тремя недопустимыми ребрами (рис. 11). Треугольник *ABC* с тремя недопустимыми ребрами на уровне z_0 представляет участок поверхности, который реально близок к плоскому. Будем считать, что центр *ABC* (точка *O)* отклоняется от z_0 по вертикали на фиксированную величину Δz. Для определения знака отклонения необходимо проверить треугольники в окрестности *ABC* (алгоритм проверки приводится в п. 4.5). Задав в *O* высоту $z_0 \pm \Delta z$, можно заменить *ABC* на три треугольника *ABO, BCO, CAO,* содержащих по одному недопустимому ребру.

Перестроим все треугольники с тремя недопустимыми ребрами. После этого в триангуляции останутся цепочки смежных треугольников с двумя недопустимыми ребрами, причем данные цепочки начинаются и заканчиваются треугольниками с одним недопустимым ребром. Пусть некоторая цепочка начинается треугольником *ABC,* а заканчивается треугольником *UVW,* причем *BC* и *VW* - недопустимые ребра, т.е. $z_B = z_C = z_V = z_W = z_0$. Предположим, что ломаная, проходящая из *A* в *U* через центры недопустимых ребер, имеет длину *L,* а длина самого короткого недопустимого ребра цепочки равна *l*. Вычислим на ломаной новую точку *M* по следующим правилам:

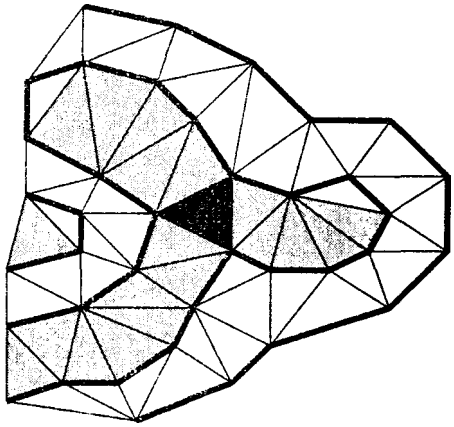
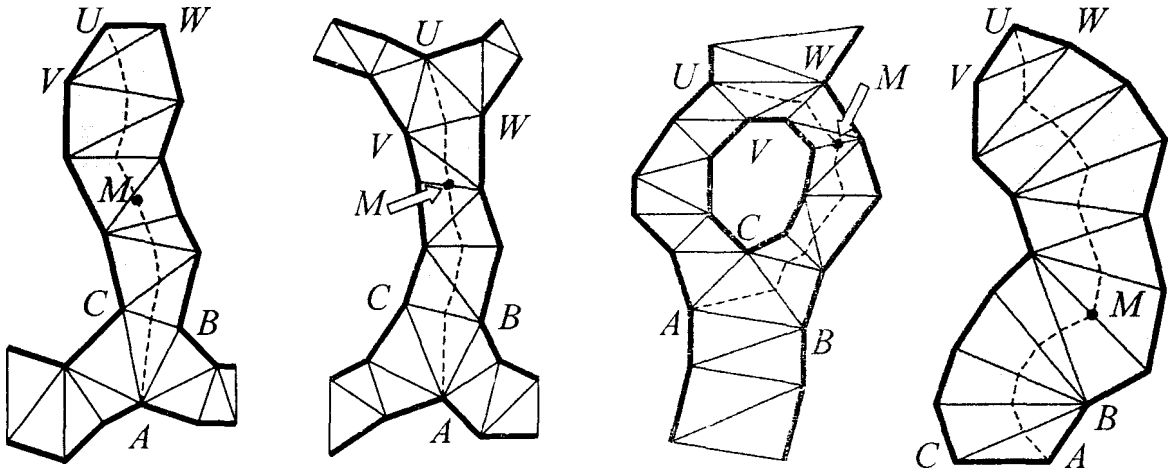


Рис. 11. Треугольники с одним (светлая закраска), двумя (средняя) и тремя (темная) недопустимыми ребрами

1. Если $z_A \neq z_U$, то полагаем, что высота вдоль ломаной *от A* до *U* изменяется линейно. Тогда *M*- это равноудаленная от *A* и *U* точка ломаной (рис. 12а) и $z_M=(z_A+z_U)/2$.

2. Если $z_A=z_U\neq z_0$, то поверхность имеет седловую точку. Будем считать, что это точка *M,* которая лежит на ломаной в центре самого короткого недопустимого ребра (рис. 12б и 12в), при этом $z_M=(z_0\cdot L+z_A\cdot l)/(L+l)$.
3. Если $z_A=z_U=z_0$, то цепочку образуют треугольники внутри замкнутой изолинии, которые представляют почти плоский участок поверхности. Будем считать, что наибольшее отклонение от плоскости достигается в точке *M* - центре самого длинного недопустимого ребра (рис. 12г). Высота $zM=z0\pm\Delta z$, где знак отклонения определяется так же, как и для треугольников с тремя недопустимыми ребрами.



(а) (б) (в) (г)

Рис. 12. Проверка цепочек треугольников с двумя недопустимыми ребрами и добавление новой точки *M:* (а) - между изолиниями разных уровней, (б) и (в) - на участках с седловой точкой, (г) - внутри замкнутой изолинии

После этого точка *M* включается в триангуляцию, причем по условию Делоне перестраиваются *только недопустимые ребра*. Полученная система треугольников уже не будет триангуляцией Делоне, зато в ней не будут нарушены слабые ограничения. После добавления точки *M* цепочка разбивается на две более короткие, а общее число недопустимых ребер уменьшается. Процесс заканчивается, когда недопустимых ребер в триангуляции не останется. В целом проверку сильных ограничений выполняет следующий алгоритм.

Алгоритм проверки сильных ограничений триангуляции.

перестроение всех треугольников с тремя недопустимыми ребрами;
пока *есть цепочки треугольников с недопустимыми ребрами* **цикл**
    начальный просмотр очередной цепочки;
    | выделение конечных треугольников,
    | расчет длины ломаной вдоль цепочки,

| поиск самого длинного и самого короткого недопустимого ребра вычисление новой точки *M* и включение ее в триангуляцию;

| M выбирается в нужном треугольнике - локализация *M* не нужна

кц

Конец алгоритма.

Приведенный алгоритм во многом напоминает алгоритм быстрой сортировки Хоара : включение новой точки *M* и разбиение цепочки на две более короткие - это полный аналог разделения сортируемого массива опорным элементом. Поэтому совпадут и оценки трудоемкости: если в цепочку входят *m* треугольников, то для ее полного перестроения потребуется в среднем *0(mlogm)* операций. Общая трудоемкость будет вполне приемлемой даже в том случае, когда исходная триангуляция содержит 0(n) треугольников с недопустимыми ребрами.

Только триангуляция с сильными ограничениями позволяет получить кусочно-линейную интерполирующую поверхность, которую можно использовать как начальную модель рельефа для необходимых расчетов или дальнейшего сглаживания.

4. БЫСТРАЯ ПРОВЕРКА ТОПОЛОГИЧЕСКОЙ КОРРЕКТНОСТИ И ПРАВИЛЬНОСТИ ЗАДАНИЯ ВЫСОТ

Проверка и корректировка исходных данных является необходимым начальным этапом при построении цифровой модели рельефа. Покажем, как можно упростить и ускорить эту проверку с помощью предложенных алгоритмов.

4.1. Поиск точек пересечения отрезков линий и точек, лежащих на отрезках

Для выделения этих точек можно использовать модифицированный алгоритм триангуляции со слабыми ограничениями: при обработке каждого отрезка линии дополнительно проверять все ребра треугольников, пересекаемые данным отрезком. Если вершина пересекаемого ребра расположена на расстоянии менее *eps* от отрезка или пересекаемое ребро совпадает с другим отрезком, то в соответствующие линии необходимо добавить новые вершины. Такое перестроение линий легко провести в автоматическом режиме.

4.2. Проверка топологии

Разрывы линий на границах планшетов лучше всего устранять вручную, *точки резкого изменения направления линии* вычисляются элементарно. Все остальные нарушения быстро выделяются с помощью

триангуляции Делоне (без ограничений):

• *вырожденные отрезки и другие случаи совпадения точек* обнаруживаются при локализации новых точек во время триангуляции, все совпадающие точки нужно объединять в списки;

• при *самопересечении и полном или частичном совпадении линий* некоторые их вершины не будут вершинами треугольников, но попадут в списки совпадающих точек; проверка этих списков позволит быстро обнаружить нарушение;

• *примыкание или пересечение изолиний на разной высоте, высотная отметка на изолинии* обнаруживаются элементарно при проверке списков совпадающих вершин (для трехмерных линий эти случаи не являются нарушениями);

• наконец, *висячие вершины* - это просто концы линий, не совпадающие с другими точками. Причины однотипных топологических нарушений могут быть самыми разными, поэтому наилучшим способом проверки является сочетание автоматизированного выделения и интерактивной коррекции ошибок.

4.3. Интерполяция высот в вершинах трехмерных линий

После построения триангуляции со слабыми ограничениями по спискам совпадающих вершин можно установить, какие изолинии примыкают или пересекают данную трехмерную линию, и какие высотные отметки расположены на ней. В соответствующих вершинах линии будут, тем самым, определены высоты. Во всех вершинах, расположенных между парой точек с известной высотой, значения Z можно рассчитать с помощью линейной интерполяции. Необходимым условием правильной интерполяции является задание высот в конечных точках незамкнутой линии и хотя бы в одной точке замкнутой линии (в этом случае значения Z

во всех вершинах будут одинаковыми, но линия будет считаться трехмерной). После завершения обработки трехмерных линий в триангуляции не должно остаться вершин, в которых значения Z не заданы.

4.4.Начальная проверка перепадов высот

Ошибки в задании высот линий и точек характерны тем, что их практически невозможно обнаружить визуально или выделить все с помощью только автоматизированных проверок. Обработка высот должна включать несколько шагов и выполняться в интерактивном режиме. Начальным шагом, на котором выявляются самые очевидные ошибки, является проверка перепадов высот. Для ее проведения необходимо построить триангуляцию со слабыми ограничениями и вычислить высоты во всех вершинах трехмерных линий. Очевидно, что при правильном задании исходных данных перепад высот на любом ребре триангуляции не может превышать значения шага изменения высот для основных изолиний. Нарушение этого условия свидетельствует о неправильном задании высот или пропуске изолиний при векторизации. Но существуют и два исключения:

1. Если концы незамкнутых линий не выводятся точно на границу планшета, то ребра на границе триангуляции могут принадлежать очень вытянутым треугольникам и соединять точки, расположенные на большом расстоянии друг от друга. Необходимо либо вывести концы линий на границу, либо исключить из триангуляции граничные треугольники с недопустимым перепадом высот.
2. На участках с почти вертикальными обрывами большой высоты основные изолинии практически сливаются, поэтому некоторые из них могут быть пропущены. Такая ситуация не является ошибочной, более того, попытка восстановить все изолинии может привести к тому, что из-за совпадения их проекций будет невозможно построить триангуляцию с

ограничениями.

4.5. Определение знака отклонения высоты

При проверке сильных ограничений триангуляции необходимо определять знак отклонения высоты в центрах треугольников с недопустимыми горизонтальными ребрами. Одновременно с этим можно провести и проверку правильности задания высот изолиний, основанную на простом правиле: точки изолинии не могут быть локальными минимумами или максимумами поверхности. Для триангуляции со слабыми ограничениями это можно сформулировать следующим образом: если *AB* - отрезок изолинии ($z_A = z_B$), который является общим ребром треугольников *ABC* и *BAD,* причем $z_C \neq z_A$ и $z_D \neq z_A$, тогда $(z_C - z_A)\cdot(z_D - z_A) < 0$. Другими словами, если $z_C < z_A$, то $z_D > z_A$, и наоборот.

Определение знака приращения для горизонтального треугольника *ABC* состоит в проверке окрестности *ABC* с переходом к смежным треугольникам *по горизонтальным ребрам* до тех пор, пока не будет найден треугольник с такой вершиной *D,* что $z_D \neq z_A$. Если переход к треугольнику с вершиной *D* проходил *только по недопустимым ребрам,* то знак приращения в центре *ABC* такой же, как в точке *D.* При каждом пересечении отрезка изолинии знак приращения меняется на противоположный. Если высоты изолиний заданы правильно, то знак приращения не зависит от порядка проверки окрестности *ABC,* а убедиться в этом можно, проверив все треугольники на границе горизонтального участка, в который входит *ABC.* (рис. 13 иллюстрирует данную проверку): знаки приращений для пар точек *D, E* и *G, H* должны быть разными.

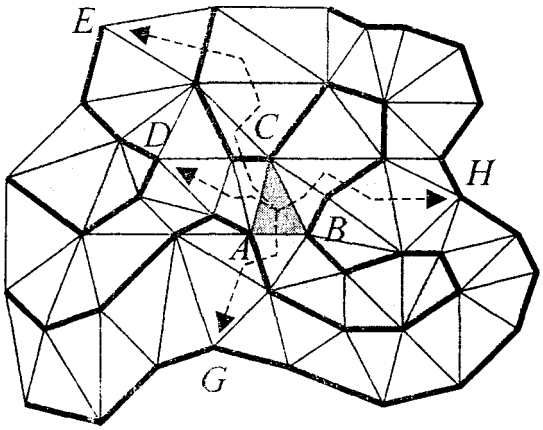


Рис. 13. Проверка высот относительно ABC:
знаки приращений для пар точек D,E и G,H
должны быть различны

ЗАКЛЮЧЕНИЕ

Настоящая дипломная работа посвящена разработке алгоритмов построения цифровой модели рельефа, заданного нерегулярными наборами структурных линий и высотных отметок. В процессе выполнения работы были решены следующие задачи:

1. Разработаны алгоритмы проверки топологической корректности исходных данных и полуавтоматического сшивания линий на границах смежных векторизованных картографических планшетов.
2. Разработан простой графический редактор, позволяющий обрабатывать наборы линий и точек, представленных в формате шейп-файлов системы ArcInfo.
3. Разработан быстрый алгоритм проверки правильности задания высот в узлах линий и высотных отметках на основе триангуляции Делоне.
4. Разработаны алгоритмы построения триангуляции со слабыми и сильными ограничениями, позволяющие построить кондиционную модель поверхности, заданной исходным набором горизонталей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Жихарев С.А., Скворцов А.В. Моделирование рельефа в системе ГрафИн // Геоинформатика: Теория и практика. Выпуск 1. – Томск: Изд-во Томск. ун-та, 1998, с. 194-205.
2. Скворцов А.В., Костюк Ю.Л. Эффективные алгоритмы построения триангуляции Делоне // Геоинформатика: Теория и практика. Выпуск 1. – Томск: Изд-во Томск. ун-та. 1998, с. 22-47.
3. ARC/INFO User's Guide: ARC/INFO Data Model, Concepts & Key Terms. Environmental Systems Research Institute.Inc., NY, 1992. – 298 p.
4. ARC/INFO User's Guide: Surface Modeling with TIN. Environmental Systems Research Institute.Inc., NY, 1992. – 240 p.
5. ArcView GIS. Environmental Systems Research Institute.Inc., NY, 1997. – 350 p.
6. ESRI Shapefile: A Technical Description. Environmental System Research Institute. Inc., USA, 1997. – 23 p.

ПРИЛОЖЕНИЕ 1. РУКОВОДСТВО ПРОГРАММИСТА

Прилагаемая к отчету программа представляет собой инструмент для построения цифровой модели рельефа. Данная программа позволяет обрабатывать наборы линий и точек, представленных в формате шейп-файлов системы ArcInfo, осуществляет проверку топологической корректности исходных данных, их редактирование и полуавтоматическое сшивание линий на границах смежных векторизованных картографических планшетов.

Программа состоит из модулей написанных на Delphi 6.0. Ниже приведен список модулей с описанием процедур и функций.

В модуле MainUnit описан класс TLayer который используется для создания слоя, а так же набор процедур и функций, позволяющих производить элементарные операции над ним.

```
TLayer = class
 private
  FName: string;
  FShapeFile: TShapeFile;
  FVisible: boolean;
  FLayers: TLayers;
  FNew: boolean;
  FModified: boolean;
  procedure SetVisible(Value: boolean);
 public
  constructor Create;
  destructor Destroy;
  property Name: string read FName write Fname;
  property ShapeFile: TShapeFile read FShapeFile;
  property Visible: boolean read FVisible write SetVisible;
```

```
    property Layers: TLayers read FLayers;
    property IsNew: boolean read FNew;
    property Modified: boolean read FModified;
  end;
```

Переменные, описанные в классе TLayer имеют следующий смысл:

FName – хранит имя Шейп-файла;

FShapeFile – содержит в себе Шейп-файл который отображается в данном слое;

FVisible – признак видимость слоя;

FLayers – указывает на массив слоев, в котором содержится данный слой;

FNew – признак того, что слой только что создан;

FModified – признак того что слой был изменен;

процедура SetVisible(Value: boolean) – устанавливает видимость слоя.

Описан класс TLayers который используется для создания массива слоев, а так же набор процедур и функций, позволяющих производить элементарные операции над ним.

```
TLayers = class
    FLayers: TList;
    FScale: double;
    FCanvas: TCanvas;
    FImage: TImage;
    FOwner: TMainForm;
    FWndOrg: TMapPoint;
    FBoundsRect: TMapRect;
    FEditLine: boolean;
    FTrianVisible: boolean;
    FNumberVisible: boolean;
    FNewPointVisible: boolean;
    function GetLayer(Index: integer): TLayer;
```

```
    procedure SetScale(AScale: double);
    function GetLayerCount: integer;
  public
    constructor Create;
    destructor Destroy;
    procedure UpdateAll;
    procedure AddLayer(AFileName: String);
    procedure DeleteLayer(ALayerName: string); overload;
    procedure DeleteLayer(ALayer: TLayer); overload;
    function GetLayerByName(Name: string): TLayer;
    procedure FlatToScreen(MP: TMapPoint; var DP: TPoint);
    procedure ScreenToFlat(DP: TPoint; var MP: TMapPoint);
    function MapDelta(SrcDist: double): double;
    property Layers[index:integer]: TLayer read GetLayer;
    property Scale: double read FScale write SetScale;
    property LayerCount: integer read GetLayerCount;
  end;
```

Переменные описанные в классе TLayers имеют следующий смысл:

FLayers – список указателей на слои;

FScale – масштаб соотношения ед.файла к ед.экрана;

FCanvas – объект, в котором отображается изображение;

FImage - указатель на объект, в котором отображается изображение;

FOwner - указатель на рабочую форму;

FWndOrg - положение центра экрана;

FBoundsRect – охватывающий прямоугольник;

FEditLine - признак того, что линия выделена;

FTrianVisible – признак видимости треугольной сетки;

FNumberVisible – признак видимости типа треугольников;

FNewPointVisible – признак видимости нововведенных точек;

функция GetLayer(Index: integer): TLayer – позволяет получить слой по его номеру

процедура SetScale(AScale: double) – позволяет установить масштаб;

функция GetLayerCount: integer – возвращает число слоев;

процедура AddLayer(AFileName: String); - добавляет новый шейп с заданным именем файла;

процедура DeleteLayer(ALayerName: string); overload; - удалить слой с данным именем;

процедура DeleteLayer(ALayer: TLayer); overload; - удаляет слой по указателю на слой;

функция GetLayerByName(Name: string): TLayer; - позволяет получить слой по имени;

процедура FlatToScreen(MP: TMapPoint; var DP: TPoint); - переводит координаты файла в экранные координаты;

процедура ScreenToFlat(DP: TPoint; var MP: TMapPoint); - переводит экранные координаты в координаты файла;

функция MapDelta(SrcDist: double): double; - расстояние на экране переводит в расстояние на карте;

Структура хранения точек представлена в следующем виде:

```
PPoints = ^TPoints;
  TPoints = object
  private
    X1: TCoordinate;
    Y1: TCoordinate;
    Z1: TCoordinate;
    Next: PPoints;
    Last: PPoints;
    OnToTri: Boolean;
    Eq: TList;
end;
```

где X1, Y1, Z1 – координаты;

Next, Last – указатель на следующую, предыдущую для линий;

OnToTri – признак того, что точка включена в триангуляцию;

Eq – список совпадающих точек;

Структура триангуляции представлена в следующем виде:

```
PTriangle = ^TTriangle;
  TTriangle = object
    P: array[0..2] of PPoints;
    T: array[0..2] of PTriangle;
    Rod: byte;
  end;
```

где P – массив указателей на точки треугольника;

T – массив указателей на соседние треугольники;

Rod – тип треугольника;

Структура организации динамического кеша представлена в следующем виде:

```
PCash = ^TCash;
  TCash = object
    C: TList;
  end;
```

процедура Search(i : integer; Var Curt: PTriangle; Var Side: integer) - производит поиск треугольника внутрь которого попадает текущая точка

процедура Intersect(i,Vspom: integer; Curt: PTriangle;

var sign,InSide: Boolean;

Var Side:integer) – определяет какой следующий треугольник пути;

процедура InsertPoint(i: Integer; Curt: PTriangle; tip: integer) – включает очередную точку в триангуляцию}

процедура Rebuild(TRule: PTriangle; tip: integer) – производит проверку относительно надобности перестроения;

процедура CheckRuleDelone(n1,n2: PTriangle) – проверяет условие Делоне;

процедура Perestr(n1,n2: PTriangle) – производит перестроение двух треугольников;

процедура RefreshCash(Trian: PTriangle) – производит обновление кеша;

процедуры Strong_Limitation_3() и Strong_Limitation_2() – удовлетворяют сильные ограничения;

процедура Del_3(Curt: PTriangle) – избавляет от плоских треугольников;

процедура NextWay(pt1,pt2,pt: integer) – запоминает путь по треугольникам;

процедура Rib_Rule(Tri: PTriangle) – выявляет горизонтальные ребра;

функция OneLine(Tri: PTriangle): Boolean – проверяет треугольники на вырожденность;

процедура CoordVect(Tri:PTriangle; p0,p1,p2: integer; Var ax,ay,bx,by:Double) – используется для вычисления координат вектора;

процедура MediumPointWay(Tri:PTriangle; p0,p1,p2: integer; Var Point_X, Point_Y:Double) – находит координаты средней точки пути через треугольники.

ПРИЛОЖЕНИЕ 2. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Запуск программы осуществляется в среде Windows. Запускаемый файл называется SHPWORK.exe. После запуска файла появится окно, в которое загружаются шейп-файлы.

Программа позволяет обрабатывать несколько векторизованных картографических планшетов. Каждый планшет задан двумя шейп-файлами: файлом линий и файлом высотных отметок. Файлы с расширением .dbf должны содержать атрибут с высотой (для изолиний это одна высота, а для трехмерных линий она восстанавливается).

Приложение работает в оконном режиме. Рабочий стол приложения состоит из меню, панели инструментов, панели для вывода и редактирования изображения, легенды и окошка для выбора масштаба отображения. Открытие проекта (см. рис. 14)

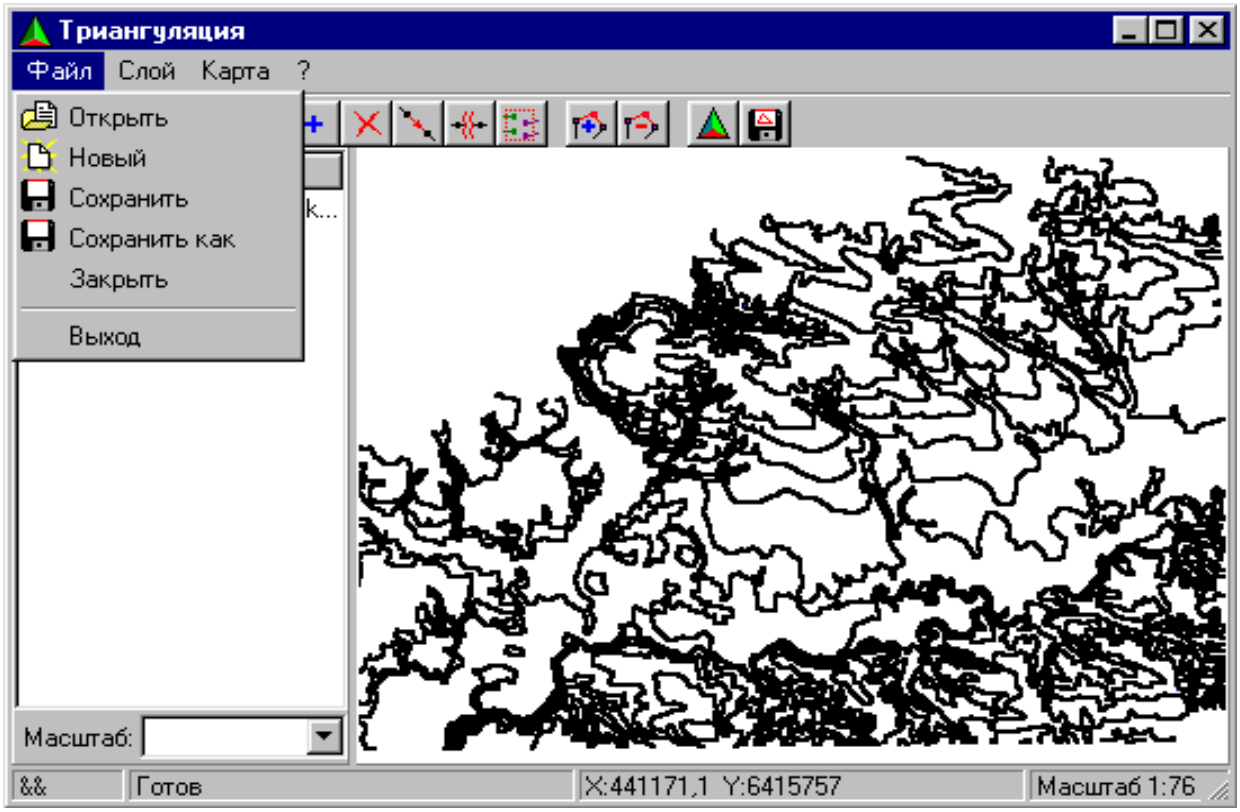


Рис. 14. Открытие проекта

В окне имеются меню с пунктами Файл/ Слой/ Карта/ ?.

Опция ***Файл*** позволяет открывать шейп-файл, создавать новый, сохранять, сохранять с новым именем, закрывать файл, а так же выйти из программы.

Опция ***Слой*** позволяет вписать в окно выделенные слои. Данная операция производится путем выбора в легенде файлов, которые необходимо вписать в окно отображения. При добавлении новых слоев все слои вписываются автоматически с сохранением пропорций (см. рис.15)

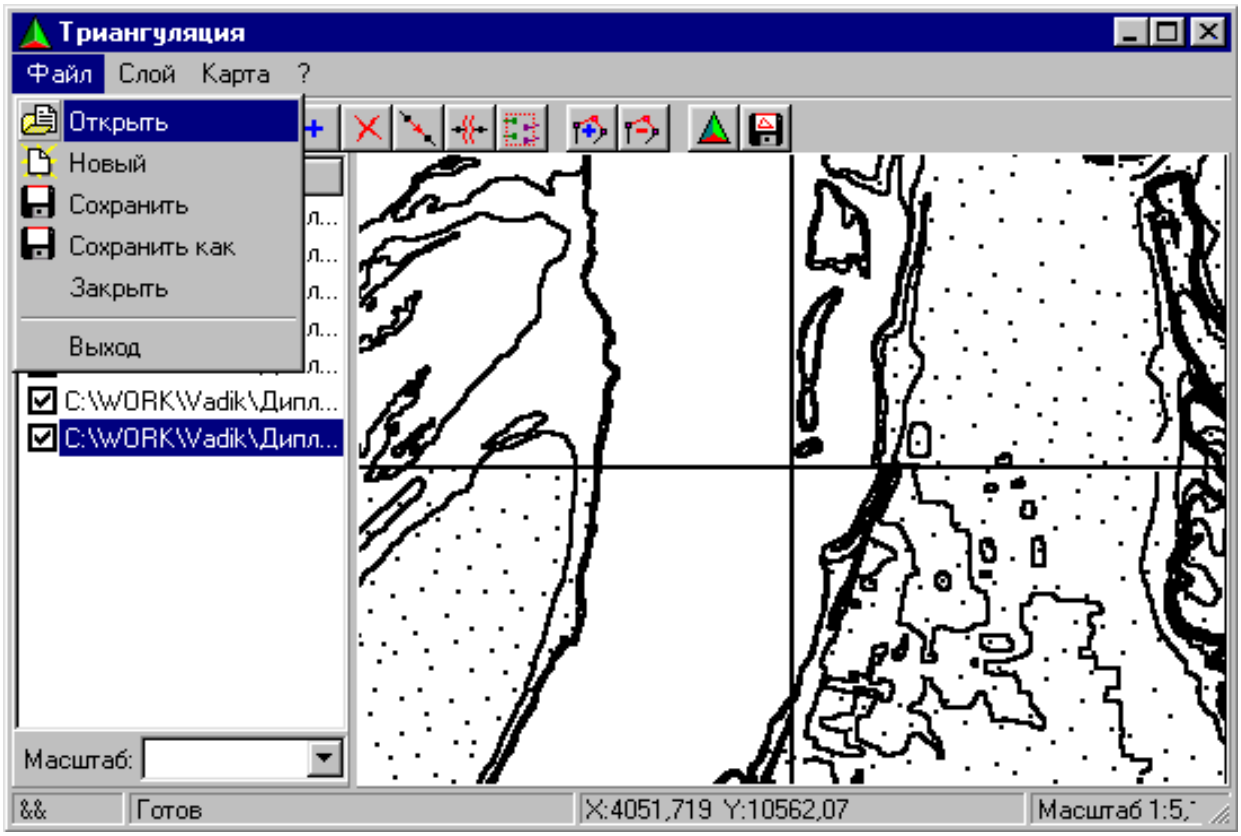


Рис. 15. Открытие нескольких векторизованных картографических планшетов

Опция ***Карта*** позволяет изменить масштаб отображения. Его так же можно изменить в окне масштаб, либо произвести увеличение графического изображения загруженных векторизованных картографических планшетов нажав правую кнопку мыши и выделив необходимую область в окне

отображения (см. рис. 16). Для уменьшения проделывается та же самая операция с удержанием клавиши Alt.

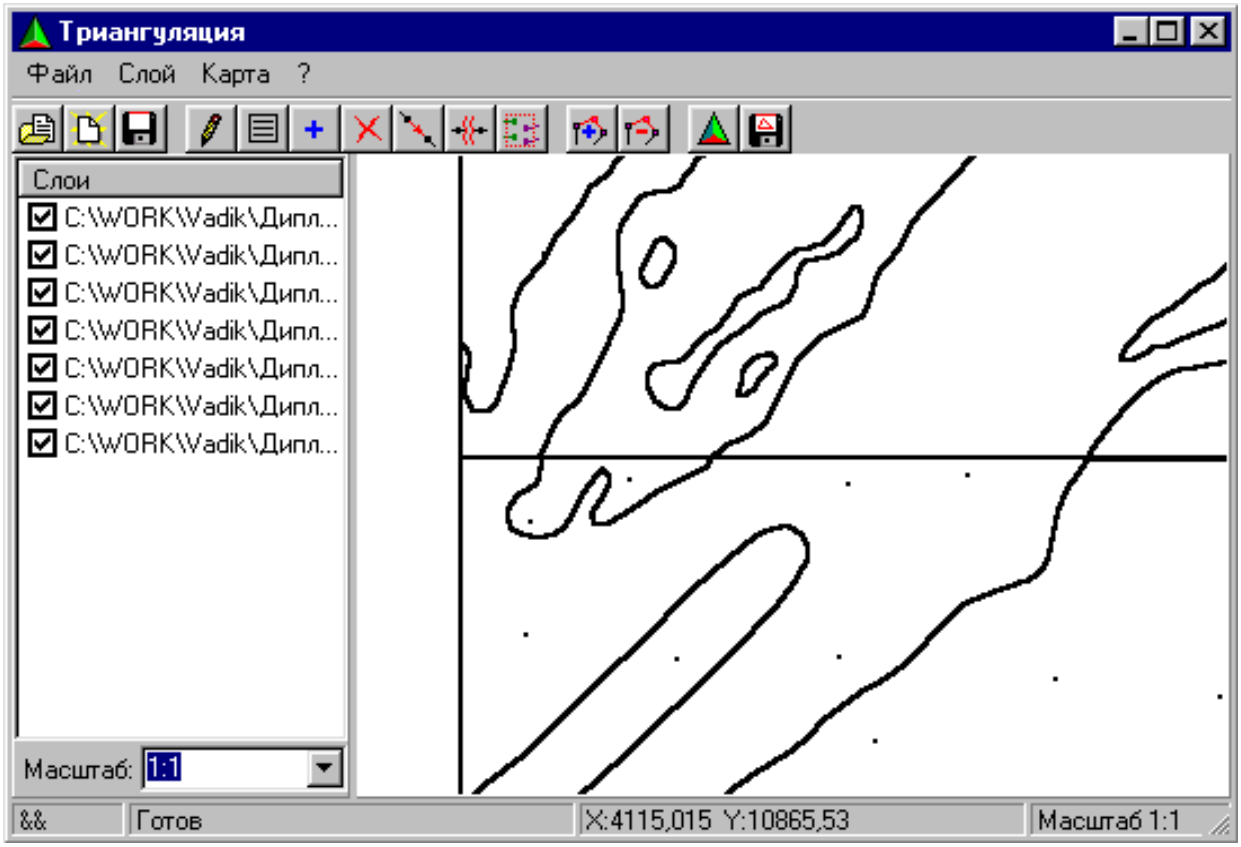


Рис. 16. Изменение масштаба изображения

Опция ***?*** содержит информацию об авторе.

Так же имеется панель инструментов. На ней находятся кнопки, выполняющие функции:

 открытие файла,

 создание файла,

 сохранение файла,

 сохранение дополнительно введенных точек в отдельный файл.

А так же кнопки дающие возможность редактирования точечных и линейных шейп-файлов:

 позволяет перейти в режим редактирования, т.е. передвигать на новую позицию точки в линейных или точечных шейп-файлах в зависимости от того, какой слой выбран. Для этого необходимо выбрать соответствующую точку нажатием левой кнопки мыши и не отпуская ее переместить на желаемую позицию. При редактировании линий нажатием левой кнопки на линии линия активизируется для редактирования, для дезактивации линии необходимо нажать на ней правой кнопкой. Для выхода из режима редактирования нажать кнопку, приводящую систему в данный режим.

 позволяет получить информацию о точках или линиях. Для точек – это координаты *X, Y, Z*, а для линий – количество точек в линии и ее уровень. При активизации информационного окна, имеется возможность изменить уровень, для линий и высоту и координаты для точек (см. рис. 17).

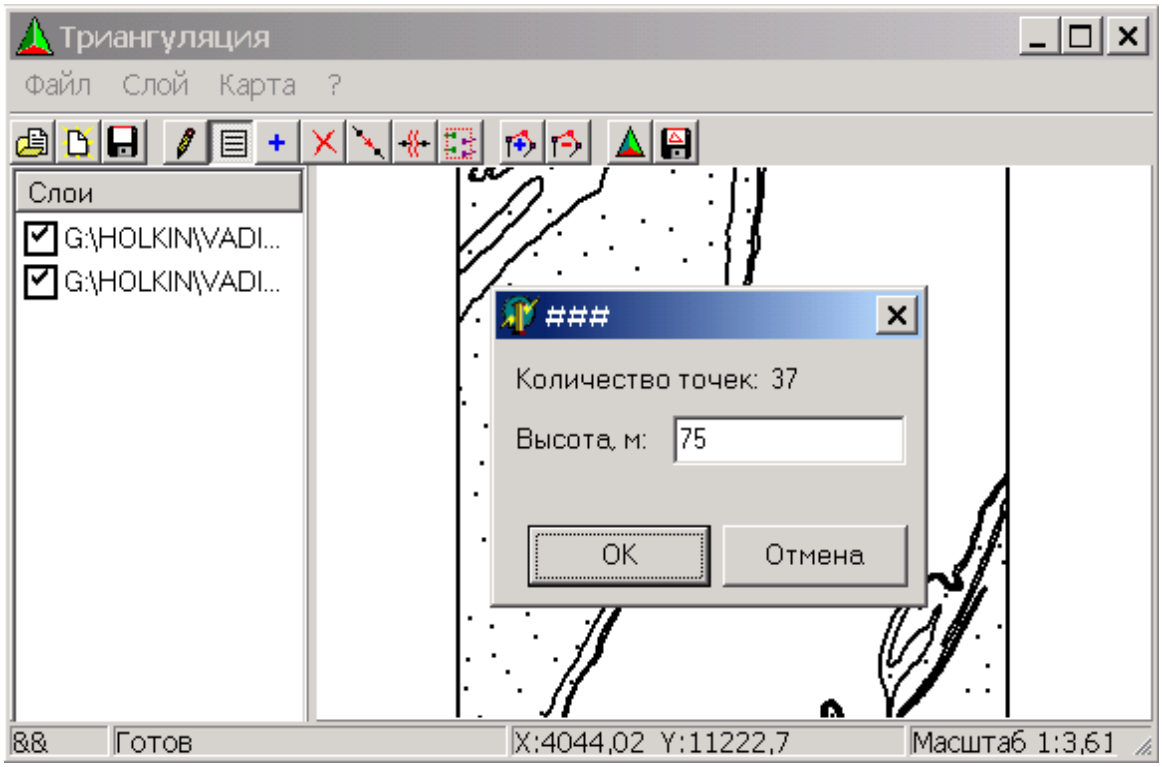


Рис. 17. Информация, полученная о линии

Активизация данной кнопки позволяет рисовать новые точки для файлов с точками и линии для файлов с линиями.

Активизация данной кнопки позволяет удалять точки в файлах с точками и линии в файлах с линиями.

Активизация данной кнопки позволяет склеить концы разных линий, либо замкнуть линию. Для этого нужно последовательно выбрать левой кнопкой мыши склеиваемые концы.

Активизация данной кнопки позволяет разорвать линию на две, либо разомкнуть линию, путем указания ребра, которое будет удалено.

Активизация данной кнопки позволяет сшивать линии на границах векторизованных картографических планшетов. Для этого в легенде выделяется слой, к которому будем подшивать, и нажимается данная кнопка. Система предложит выбрать подшиваемый файл (см. рис18).

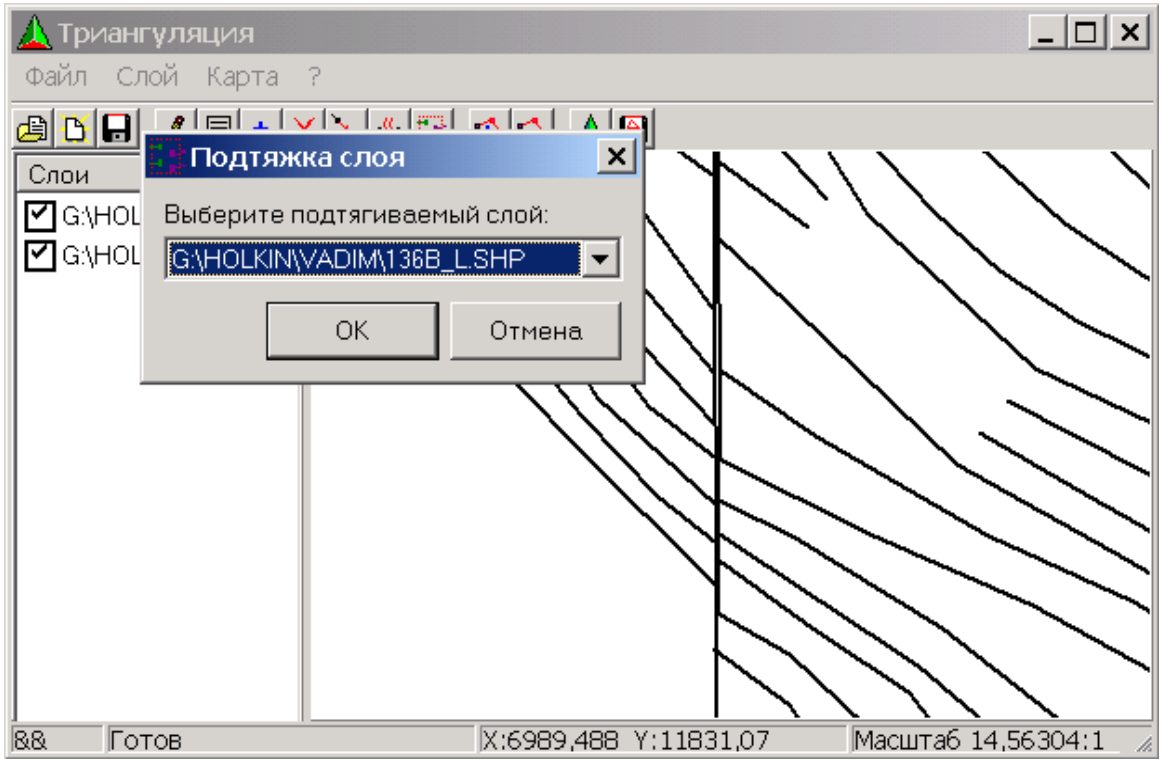


Рис. 18. Выбор подшиваемого слоя

После выбора подшиваемого слоя необходимо нажав левую кнопку мыши выбрать на границе сшиваемых картографических планшетов область в которой будет осуществляться сшивание (см. рис.19).

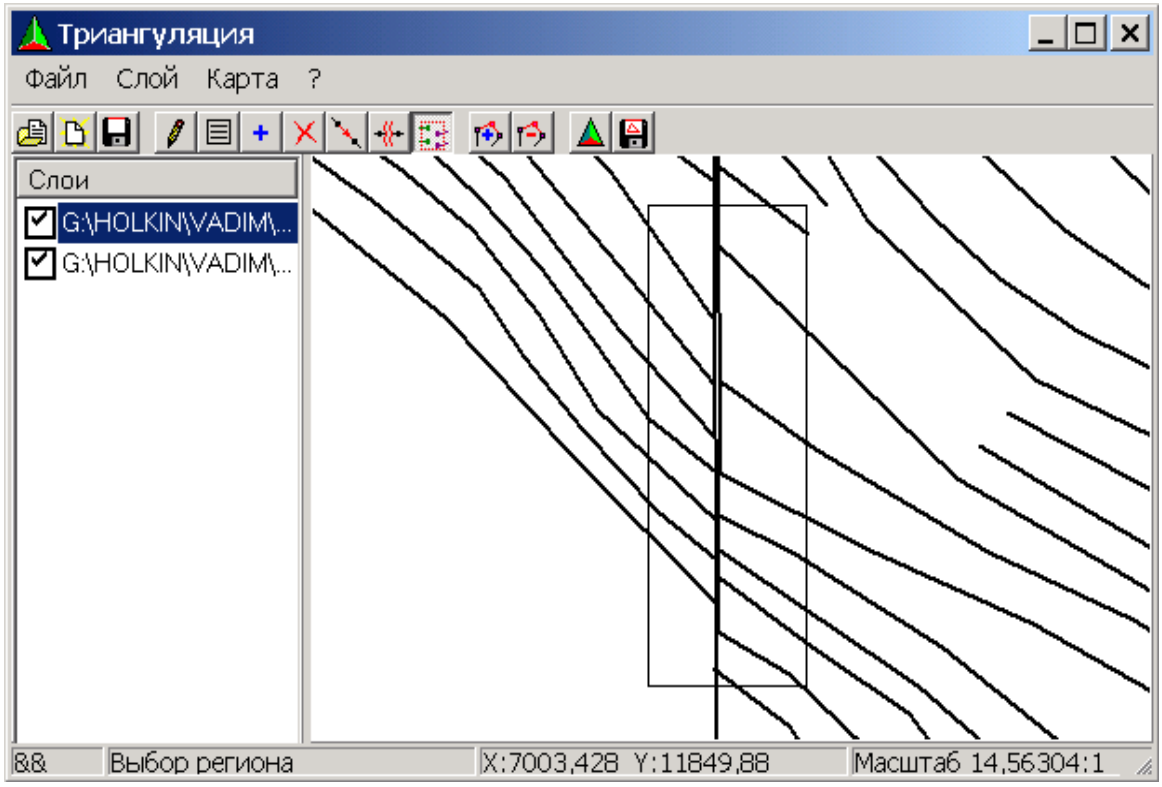


Рис.19. Выделение области, в которой будет осуществляться сшивание

После того как выбрана область сшивания, системой в некотором порядке будет предложено сшить линии. Предложение можно принять, либо отказаться от него (см. рис. 20).

В результате соответствующие концы линий в выделенной области на границе картографических планшетов точно совпадают (см. рис. 21).

Активизация данной кнопки позволяет добавить дополнительную точку на ребро в линии. Необходимо левой кнопкой мыши указать место вставки.

Активизация данной кнопки позволяет удалить точку из линии. Необходимо левой кнопкой мыши указать соответствующую точку.

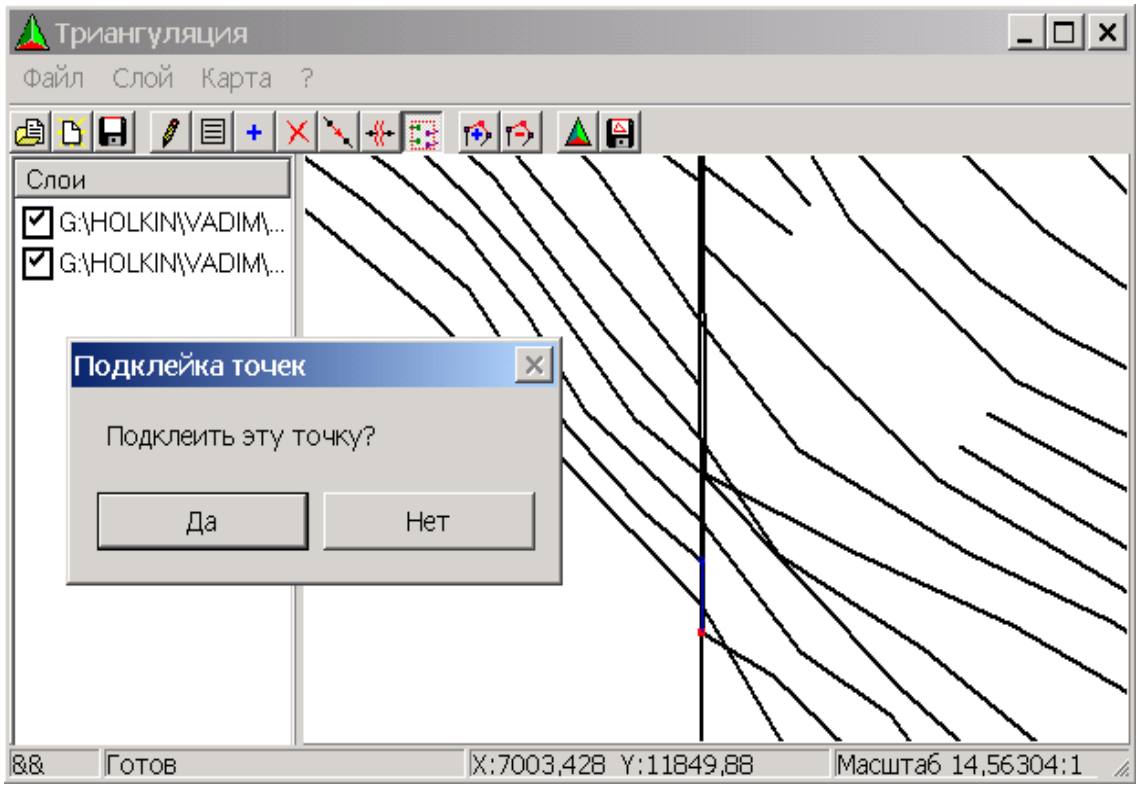


Рис. 20. Запрос подтверждения на сшивание линий

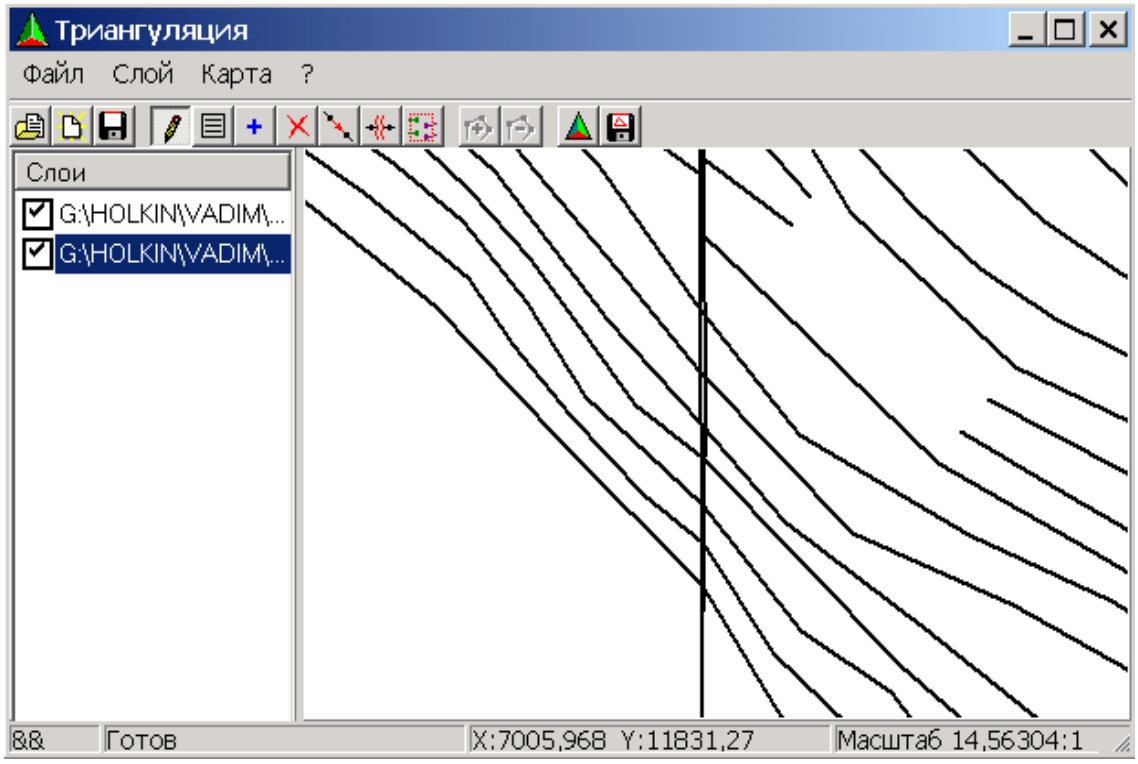


Рис. 21. Результат сшивания линий в заданной области

построение триангуляции Делоне со слабыми и сильными ограничениями (см. рис.22).

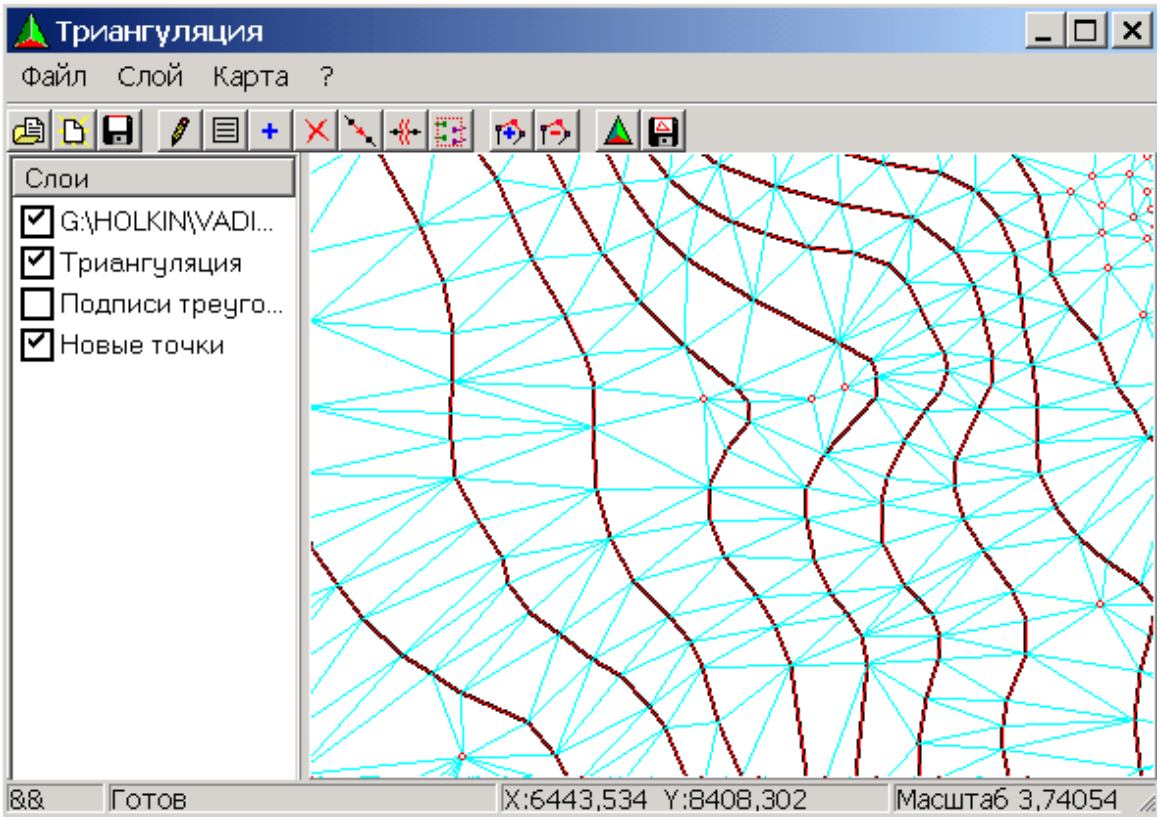


Рис.22. Триангуляция Делоне с ограничениями

ПРИЛОЖЕНИЕ 3. ФОРМАТ ШЕЙП-ФАЙЛОВ

Исходные данные, используемые в данной работе, берутся из шэйп-файлов. Ниже приводится описание формата шэйп-файлов для точек и изолиний.

Информация получена из технического описания шейп -файлов фирмы ESRI Inc. (ESRI® Shapefile Technical Description), а также из описания формата таблиц dBase III фирмы Borland Inc [6].

В шейп-файлах хранится нетопологическая геометрическая и атрибутная информация пространственных объектов в базах данных. Геометрия составляющих фигур представляется как набор плоских координат. В связи с тем, что шейп-файлы не имеют топологических ограничений на взаимные связи, они имеют определённые преимущества над другими источниками данных, что в первую очередь проявляется в высокой скорости отрисовки и простоте редактирования. Все фигуры шейп–файла полностью независимы друг от друга, в частности, они могут произвольно пересекаться между собой и даже совпадать. Кроме того, нетопологический формат шейп-файлов обычно требует меньше дискового пространства по сравнению с топологическими форматами, а также он проще для чтения и записи.

Шейп-файлы могут хранить точечные, линейные и площадные фигуры. Смешение разных типов фигур в одном файле не допускается. Атрибуты файлов содержаться в формате файла таблиц dBase ® . Каждая атрибутная запись хранится в отношении «*один -к -одному »* с соответствующей записью геометрии фигуры.

В настоящее время формат шейп-файлов широко поддерживается многими программными продуктами фирмы ESRI и других производителей

Формат шейп-файла ESRI состоит из главного файла, файла индекса, а также таблицы dBase. Все эти 3 файла должны иметь одинаковое имя, но

различные расширения: .shp для главного, .shx для индексного и .dbf для файла атрибутов.

Организация главного файла

Главный файл (с расширением .shp) состоит из заголовка фиксированной длины, за которым идут записи переменной длины, описывающие геометрию фигур (на рис. 23 условно представлена структура главного файла).

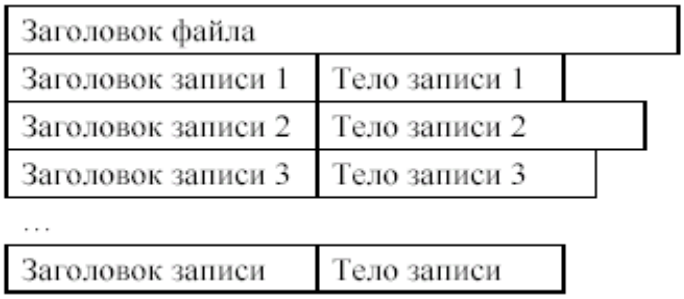


Рис. 23.Структура главного файла

Заголовок главного файла

Заголовок главного файла состоит из 100 байтов. В табл. 1 описаны поля заголовка вместе с их позицией относительно начала файла, значением, типом и порядком байтов.

Если тип шейп-файла не содержит значений меры или высоты, то значения диапазонов изменения этих величин устанавливаются равными нулю. Указанная в таблице длина файла определена как общая длина файла в 16-разрядных словах, включая 50 16-битовых слов заголовка файла.

Все фигуры в шейп-файле должны быть одного типа. Значения кода типа файла могут быть следующие:

1 – точки;

3– полиполилинии;

Таблица 1. Описание заголовка главного файла шейп-файла

Позиция	Поле	Значение	Тип поля	Порядок байтов
Байт 0	Код файла	9994	Целый	Обратный
Байт 4	Зарезервировано	0	Целый	Обратный
Байт 8	Зарезервировано	0	Целый	Обратный
Байт 12	Зарезервировано	0	Целый	Обратный
Байт 16	Зарезервировано	0	Целый	Обратный
Байт 20	Зарезервировано	0	Целый	Обратный
Байт 24	Длина файла	Длина файла в 16-разрядных словах	Целый	Обратный
Байт 28	Версия	1000	Целый	Прямой
Байт 32	Тип файла	Код типа фигур шейп-файла	Целый	Прямой
Байт 36	Минимальный ограничивающий все фигуры прямоугольник	X_{min} прямоугольника	Вещественный	Прямой
Байт 44		Y_{min} прямоугольника	Вещественный	Прямой
Байт 52		X_{max} прямоугольника	Вещественный	Прямой
Байт 60		Y_{max} прямоугольника	Вещественный	Прямой
Байт 68	Диапазон значений высот Z	Z_{min} высот	Вещественный	Прямой
Байт 76		Z_{max} высот	Вещественный	Прямой
Байт 84	Диапазон значений меры M	M_{min} меры	Вещественный	Прямой
Байт 92		M_{max} меры	Вещественный	Прямой

Заголовок записей

Заголовок каждой записи содержит порядковый номер записи, а также длину тела записи. В табл. 2 описаны поля заголовка.

Таблица 2. Описание заголовка записей

Позиция	Поле	Значение	Тип поля	Порядок байтов
Байт 0	Номер записи	Порядковый номер; нумерация с 1	Целый	Обратный
Байт 4	Зарезервировано	Длина тела записи без заголовка	Целый	Обратный

Длина тела измеряется в 16-разрядных словах. Таким образом, каждая запись занимает (4+длина тела) 16-битовых слов.

Рассмотрим теперь формат тел записей.

Тело записей

Тело записей шейп-файла состоит из кода типа фигуры, за которым идут геометрические данные фигур. Общая длина тела записи от количества частей и точек в фигуре. Ниже описаны тела записей для разных типов фигур.

Точки

Точки в шейп-файлах описываются парой вещественных координат *(*X, Y), а также дополнительно в зависимости от типа шейп-файла высотой *Z* и мерой M. В табл. 3 приведено описание тела записи точек.

Таблица 3. Описание тела записи точек

Позиция	Поле	Значение	Тип поля	Порядок байтов
Байт 0	Тип фигуры	1	Целый	Прямой
Байт 4	Координата X	X	Вещественный	Прямой
Байт 12	Координата Y	Y	Вещественный	Прямой

Полиполилинии

Полиполилинии состоят из полностью независимых полилиний, каждая из которых является ломаной, соединяющей некоторое количество точек. В табл. 4 приведено описание тела записи полиполилинии.

Таблица 4. Описание тела записи полиполилинии

Позиция	Поле	Значение	Тип поля	Порядок байтов
Байт 0	Тип фигуры	8	Целый	Прямой
Байт 4	Прямоугольник	Мин. объемлющий прямо-угольник	4 вещественных	Прямой
Байт 36	P (число частей)	Число частей в фигуре	Целый	Прямой
Байт 40	N (число точек)	Число точек в фигуре	Целый	Прямой
Байт 44	Части	Число точек в частях	P целых	Прямой
Байт p	Точки	Координаты пар точек	N точек	Прямой
Байт n	Z_{min}, Z_{max} высот	Диапазон значений высот	2 вещественных	Прямой
Байт n+16	Высоты точек	Координаты (Z) точек	N вещественных	Прямой
Байт k	M_{min}, M_{max} меры	Диапазон значений меры	2 вещественных	Прямой
Байт k+16	Меры точек	Меры (M) точек	N вещественных	Прямой

Организация файла индекса

Файл индекса (.shx) состоит из 100-байтового заголовка, за которым идут 8-байтовые записи, соответствующие фигурам шейп-файла (на рис. 24 представлена общая структура файла индекса).

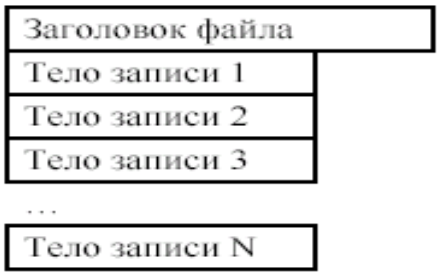


Рис. 24. Структура файла индекса

Заголовок файла индекса

Заголовок файла индекса полностью идентичен по структуре заголовку главного файла.

Тело записей

Каждая i-я запись в индексном файле хранит смещение начала и длину соответствующей i-й записи в главном файле. Смещение записи главного файла измеряется от начала этого файла в 16-битовых словах. Например, смещение первой записи в главном файле равно 50, так как она идёт сразу после 100-байтового заголовка. Длина тела измеряется в 16-разрядных словах. Это значение равно значению поля длины записи в заголовке записи главного файла. В табл. 5 описаны поля тела записей индекса.

Таблица 5. Описание тела записей индекса

Позиция	Поле	Значение	Тип поля	Порядок байтов
Байт 0	Смещение записи	Смещение записи от начала файла	Целый	Обратный
Байт 4	Длина записи	Длина тела записи без заголовка.	Целый	Обратный

Организация файла атрибутов

Файл атрибутов (.dbf) может содержать любые желаемые атрибуты или ключевые поля, по которым могут быть присоединены другие таблицы с атрибутами. Формат этого файла является стандартным DBF-файлом, используемым во многих программных продуктах. В файле атрибутов может содержаться любой набор атрибутов. Таблица должна содержать по одной записи на каждую фигуру шейп-файла в порядке, соответствующем порядку фигур в главном файле шейп-файла (на рис. 25 схематично представлена структура файла атрибутов).

Рис. 25. Структура файла атрибутов

Заголовок файла атрибутов

В табл. 6 описаны поля заголовка файла атрибутов. Общая длина заголовка составляет 32 байта.

Таблица 6. Описание заголовка файла атрибутов

Позиция	Поле	Значение	Тип поля	Порядок байтов
Байт 0	Код типа файла	3	Байт	Прямой
Байт 1	Дата последнего изменения файла	Номер года – 1900 (0..255)	3 байта	Прямой
Байт 2		Номер месяца (1-12)	3 байта	Прямой
Байт 3		Номер дня (1-31)	3 байта	Прямой
Байт 4	Число записей	Количество записей в файле	Целый	Прямой
Байт 8	Первая запись	Смещение первой записи данных	2 байта	Прямой
Байт 10	Длина записи	Длина одной записи с данными	2 байта	Прямой
Байт 12	Зарезервировано	0	16 байт	Прямой
Байт 28	Наличие индекса	0 (шейп-файл его не поддерживает)	Байт	Прямой
Байт 29	Зарезервировано	0	3 байта	Прямой

Количество записей в файле должно совпадать с соответствующим полем в заголовке главного файла. Смещение первой записи данных

определяется как смещение первой записи с атрибутами от начала файла атрибутов в байтах. Длина записи определена также в байтах и учитывает байт признака удаления перед началом записи. Заметим, что этот признак имеется в файле и всегда равен коду 32, хотя он не имеет смысла в шейп-файлах. Так как в шейп-файлах не предусмотрено наличие индекса по атрибутам, то признак наличия индекса всегда должен быть равен 0. После заголовка файла идут записи, описывающие имеющиеся в файле атрибуты. Признаком завершения описания полей является байт с кодом 1.

Запись описания атрибута.

В табл. 7 представлена структура записи, описывающей каждый атрибут файла.

Таблица 7. Описание записи поля в файле атрибутов

Позиция	Поле	Значение	Тип поля	Порядок байтов
Байт 0	Название атрибута	Название, дополненное кодами 0	11 байт	Прямой
Байт 11	Тип атрибута	Символ C, L, N, F или D	Байт	Прямой
Байт 12	Смещение атрибута	Смещение атрибута в записи	Целый	Прямой
Байт 16	Длина атрибута	Занимаемое атрибутом место	Байт	Прямой
Байт 17	Точность чисел	Число цифр после запятой	Байт	Прямой
Байт 18	Зарезервировано	0	14 байтов	Прямой

Название атрибута может быть максимально 10 символов, при этом справа оно дополняется до длины 11 байтов пустым символом с кодом 0 Тип атрибута представляется в виде символа и может быть «C» для символьных атрибутов, «L» – для логических, «D» – для хранения даты, а также «N» и «F» – для числовых полей (целых и вещественных).

Смещение атрибута внутри записи определено как смещение первого байта атрибута от начала соответствующей записи с данными. Длина атрибута может быть максимально 254 для символьных полей, 20 – для числовых. Для логического поля длина всегда равна 1, а для атрибута даты – 8 байтов. Для нечисловых полей поле *Точность чисел* всегда равно 0. Максимальное число полей в файле может быть 255, а максимальное число символов в одной записи – 4000.

Записи с данными

Данные в файле начинаются с позиции, указанной в записи заголовка файла атрибутов под именем *Первая запись*. Все записи начинаются с байта – признака удаления. Так как в шейп-файле понятие удалённой записи не имеет смысла, то в этой позиции находится пробел (код 32). За этим байтом подряд без разделителей идут данные атрибутов в порядке, в котором они были объявлены в заголовке файла.

Значения строковых атрибутов при хранении дополняются справа до длины атрибута пустыми символами (код 0).

Значения чисел хранятся в формате чисел с фиксированной точкой прижатыми к правому краю записи и дополненные слева пробелами. Количество цифр после запятой всегда равно значению, указанному в поле *Точность чисел*. Разделителем дробной и целой части чисел служит десятичная точка.

Логические поля могут быть равны символам «Y» и «N» соответственно для значений *лжи* и *правды*.

Значения дат хранятся в формате «ДД .ММ .ГГ » и занимают 8 байт. Если значение какой-то компоненты даты меньше 10, то оно дополняется слева символом нуля. Нумерация годов отсчитывается от 1900 года. Если значение какого-либо не строкового атрибута в файле атрибутов не задано или не верно, то оно считается имеющим специальное «пустое значение ».