**СОДЕРЖАНИЕ**

1. Постановка и анализ задачи

2. Разработка логической схемы данных

4. Основные алгоритмы

4.1 Описание алгоритма

4.2 Блок-схема алгоритма

5. Программная реализация

5.1 Выбор среды разработки программы

5.2 Работа с таблицами

5.3 Работа с графикой

5.4 Разработка интерфейса

5.5 Некоторые особенности алгоритмов

5.6 Тестирование

6. Описание программы

7. Инструкция по установке

8. Руководство пользователя

8.1 Главное меню

8.2 Панель быстрых кнопок

8.3 Вкладка "Таблица"

8.4 Вкладка "План"

9. Контрольный пример

Заключение

Список литературы

Приложение

##### 1. ПОСТАНОВКА И АНАЛИЗ ЗАДАЧИ

*Задача*: разработать программу информационной поддержки создания плана местности.

*Исходные данные:* Windows98, система описания точек плана местности с возможностью легкой модификации, возможность графического представления плана, построение маршрутов, оценка длины маршрута, вывод информации на экран и печать.

Назначение программы – она позволяет рисовать на карте маршруты, оценивать их длину и расстояния, что может понадобиться пользователям как в личных целях (например, туристам), так и при проведении геодезических работ, работ по прокладке кабеля и трубопровода и т.д. и т.п.

Рассмотрим это задание по пунктам.

Так как мы создаем план местности, подразумевается, что координаты точки плана местности задаются в виде широты и долготы, те в градусах. При этом мы должны учитывать, что широта подразделяется на северную и южную, каждя от 0 до 90 градусов; а долгота – на западную и восточную, от 0 до 180 градусов.

Кроме координат пользователю может понадобиться: номер маршрута, чтобы осуществить рисование нескольких маршрутов, тип объекта, находящегося в данной точке, а также какие-либо дополнительные комментарии.

При таком раскладе систему описания точек плана лучше всего представить в виде таблицы, где каждая запись будет содержать обе координаты, тип объекта, номер маршрута и комментарии. При этом "возможность легкой модификации" по отношению к таблице обеспечить более просто, чем, скажем, по отношению к файлу.

Но для оценки длины маршрута, так как там вычисляется длины отрезков, удобнее будет использовать еще одну таблицу.

И можно будет ввести еще таблицу, в которую будет заноситься перечень типов объектов, чтобы пользователю было удобнее их вносить в главную таблицу.

Возможность графического представление плана означает, что пользователь должен увидеть результат работы программы на экране, либо в распечатанном виде, те должно быть наличие экранного интерфейса, чтобы пользователю было удобнее работать с такой программой.

Так же для удобства работы с графическим представлением плана необходимо реализовать возможность его масштабирования. А для наглядности можно добавить вывод на рисунок названий объектов и расстояний.

Учитывая в качестве операционной системы Windows98 и наличие экранного интерфейса, а также работу с таблицами и графикой, на мой взгляд для разработки программы больше подходит среда С++ Builder 5.

##### 

##### 2. РАЗРАБОТКА ЛОГИЧЕСКОЙ СХЕМЫ ДАННЫХ

В программе для хранения данных используются таблицы: одна основная, с которой работает пользователь, и две вспомогательных, с данными которых оперирует программа.

Все таблицы хранятся в формате таблиц Paradox Database, в файлах с расширением \*.db.

Главная таблица состоит из шести полей со следующими характеристиками:

ID – идентификатор записи. Это поле является ключевым – то есть по нему сортируются данные в таблице. Тип – автоинкремент. Поле ID доступно только для чтения. При создании таблицы его значение равно единице, и будет увеличиваться на единицу при каждом создании новой записи. Удаление записи не изменяет значения поля ID других записей.

Latitude – содержит значение широты текущей точки на карте в формате чисел с плавающей точкой, на которые накладывается ограничение в виде области допустимых значений от –90 до 90. Положительные числа соответствуют северной широте, отрицательные – южной.

Longitude - – содержит значение долготы текущей точки на карте в формате чисел с плавающей точкой, на которые накладывается ограничение в виде области допустимых значений от –180 до 180. Положительные числа соответствуют восточной долготе, отрицательные – западной.

Object Type – поле типа–"строка". В нем указывается тип объекта, находящегося в данной точке.

Course – поле типа "целое число". Содержит номер маршрута, к которому принадлежит данная точка.

Comments – поле типа "строка", отведено для комментариев.

Ввод значений в поля Latitude, Longitude и Course обязательно.

Вспомогательная таблица ObjectTable, хранящаяся в фале objects.db, содержит перечень типов объектов, ее присутствие в каталоге программы обязательно. Данные из этой таблицы заносятся вполе со списком, с помощью которого пользователь легко может занести нужный тип объекта в главную таблицу, не набирая его вручную. В программе возможно добавление типа объекта в таблицу, удалить тип или очистить всю таблицу. В явном виде ObjectTable не отображается.

Она имеет следующие поля:

Object\_ID – идентификатор записи. Тип поля – автоинкремент.

Object Type – тип объекта.

Вспомогательная таблица RasstTable также находится в каталоге программы в файле Rasst.db. Она используется программой в процессе рисования плана: в нее заносится индекс маршрута и расстояния его отрезков. Пользователь может посмотреть содержимое этой таблицы, отключив флажок "Скрыть таблицу расстояний".

Эта таблица содержит поля:

Course – номер маршрута. Тип поля – целое число.

Distance – расстояние, в километрах. Тип поля – число с плавающей точкой.

На рис.2.1 показана схема данных в виде таблиц и связей между ними.

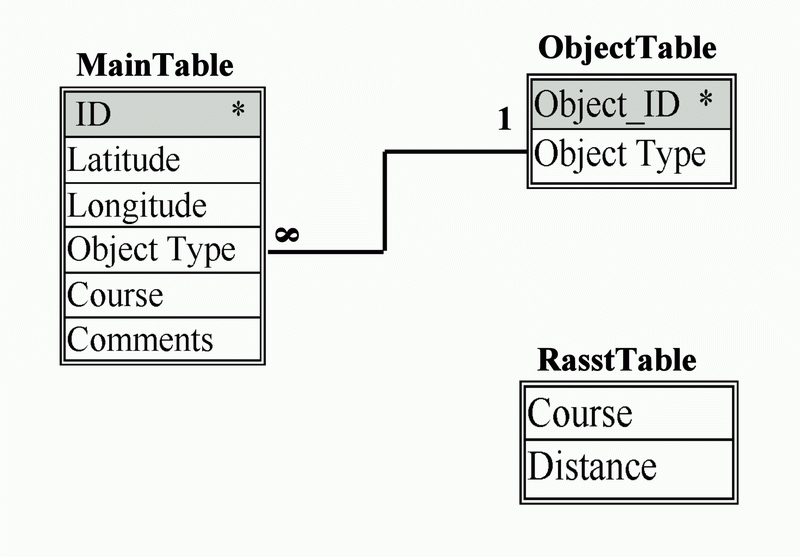


Рис.2.1. Схема данных.

**3. СТРУКТУРНАЯ СХЕМА ПРОГРАММЫ**

Основные функции, которые необходимо реализовать в нашей программе, следуют из постановки и анализа задачи:

1)Отображение данных пользователя в виде таблицы и работа с ней.

2)Добавления и удаления типов объектов.

3)Графическое представление плана с возможностью масштабирования.

4)Работа с файлами и вывод результатов на печать.

5)Удобный пользовательский интерфейс.

Ниже приведена функциональная структурная схема программы, изображенная в виде основных модулей и связей между ними. Она наглядно представляет реализацию вышеперечисленных требований в программе.

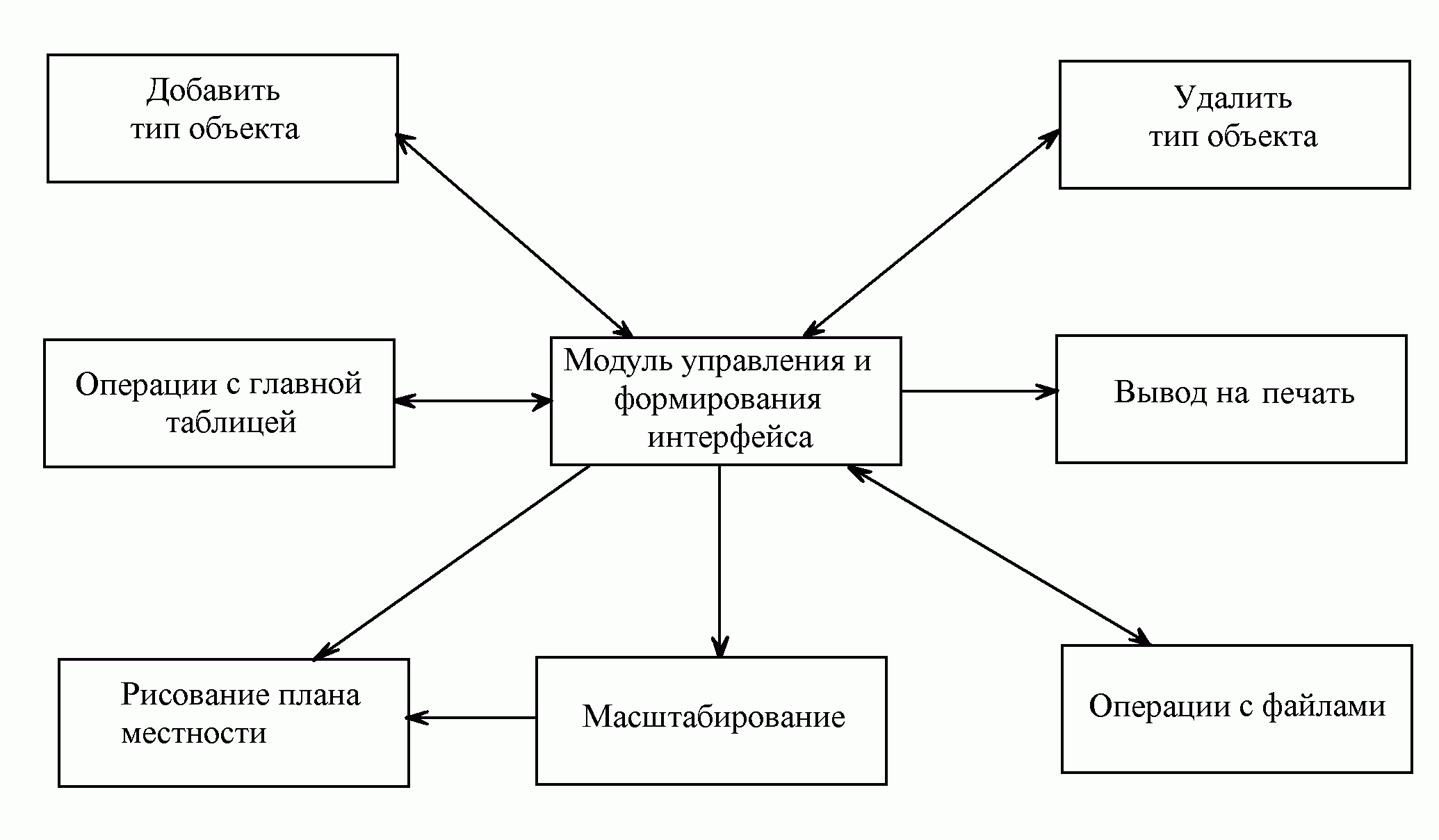


Рис. 3.1. Функциональная структура программы.

Главной частью программы является *модуль управления и формирования интерфейса*. Он представляет собой главную форму, на которой находятся элементы управления, позволяющие осуществлять остальные функции, а также формирующие пользовательский интерфейс.

##### 4. ОСНОВНЫЕ АЛГОРИТМЫ

Основным является алгоритм построения изображения по данным из главной таблицы, реализованный в виде метода plandraw().

Ниже приведены его блок-схема и описание.

# 4.1 Описание алгоритма

Если не активирована вкладка, на которой находится наше изображение плана, мы ее активируем.

Далее активируем кнопки увеличения масштаба и установки масштаба по умолчанию.

Подготавливаем таблицу расстояний, очистив ее от предыдущих записей.

Устанавливаем параметры фона (цвет) и рисуем его, затем устанавливаем параметры пера (толщина линии и стиль), от которых будет зависеть отображение линий маршрута на рисунке. В начале толщина линии равна единице – для рисования координатной сетки.

С помощью свойства таблицы RecordCount находим количество строк в главной таблице.

Устанавливаем указатель текущей записи на первую, и организуем цикл прохода по всем записям таблицы, в котором подсчитываем количество маршрутов.

Далее организуем цикл рисования каждого маршрута, в котором количество повторений определяется ранее подсчитанным количеством маршрутов.

В самом начале цикла осуществляем подготовку для отображения масштаба – переменной масштаба присваиваем коэффициент увеличения (в целых единицах), для обеих полос прокрутки определяем максимальную величину, которая зависит от степени увеличения и размеров рисунка.

Если номер маршрута равен нулю, тогда выполняется условие рисования координатной сетки – меридианов и параллелей. Сначала выполняется цикл рисования меридианов – мы проходим от 0 до 360 градусов с шагом, зависящим от степени увеличения (15, 6, 3, или 1 градусов), причем возле каждого меридиана подписывается соответствующий ему градус долготы (восточная долгота – со знаком "+", западная – со знаком "-"). Нулевой меридиан изображается черным цветом. Аналогичные действия осуществляются и в цикле рисования параллелей, единственное отличие – цикл проходит от 0 до 180 градусов. Со знаком "+" обозначается северная широта, со знаком "-" - южная.

Меняем толщину линии на 2, для изображения линий маршрутов.

Создаем три массива, в которые будем заносить индексы записей текущего маршрута и координаты. Далее организуем цикл, в котором проходим по записям таблицы и для текущего маршрута заполняем эти массивы. Причем в массивы координат заносят уже отмасштабированные величины. В этом же цикле подсчитываем количество точек маршрута.

В следующем цикле сортируем содержимое массива индексов, чтобы потом нарисовать точки маршрутов в том порядке, в котором они находятся в таблице.

Задаем цвет линии в зависимости от номера маршрута. И организуем цикл, рисующий линии.

В цикле рисования линий проделываем следующее: учитывая позиции лент прокрутки, вычисляется положение окна по отношению к карте, и относительно этого положения вычисляются координаты точки в окне. Если у нас первое прохождение, то мы просто перемещаем курсор в точку с вычисленными координатами, если нет, и флажок рисования линий включен, – рисуем линию от предыдущей точки до этой. Если флажок отключен, на карте ставятся только точки.

Затем вычисляется расстояние между точками, находящимися на Земном шаре, проекции которых мы только что изобразили на нашей карте. Расстояния вычисляются в километрах, радиус Земли берется равным 6371км. Вычисление расстояния производится в том случае, если i не равно 0, те не первое прохождение по циклу. Это условие необходимо, так как мы используем координаты предыдущей точки, чтобы найти расстояние до текущей.

Так как поверхность земли имеет шарообразную форму, нам нужно вычислить длину дуги. Основной проблемой здесь является нахождение угла, на который опирается дуга.

Рассматриваются три случая:

1)если точки находятся на одной долготе, угол определить просто - он будет равен разности между большим и меньшим значением широты.

2)если точки находятся на одинаковой широте, то тут его определение тоже не представляет сложности – он равен разности между большим и меньшим значением долготы, умноженной на поправку cos(f), где f – текущая широта.

3)если точки располагаются на разной широте и долготе, этот случай нахождения угла является более сложным. Рассмотрим его подробно.

Сначала находим разность долгот точек, как если бы они находились на одной широте, помножаем на поправку cos(f), и вычисляем линейное расстояние между ними по теореме косинусов (другие две стороны треугольника – радиусы Земли). Таким же образом рассчитываем расстояние между точками, как если бы они находились на одной долготе. Эти расстояния обозначим l1 и l2.

Теперь у нас есть прямоугольный треугольник с катетами l1, l2, гипотенузой которого является расстояние l3. Вычисляем ее по теореме Пифагора. Нашей целью является найти угол **a**. С помощью теоремы косинусов находим косинус этого угла. Вычислив по нему арккосинус, мы получаем угол! Ниже приводится поясняющий рисунок.

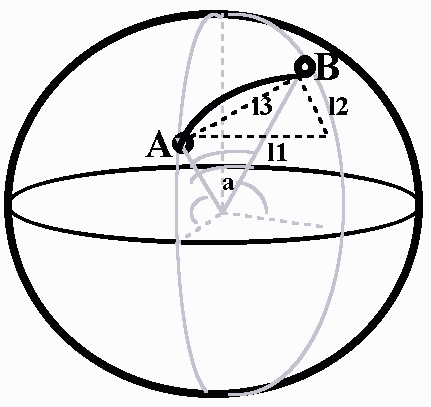


Рис. 4.1. Пояснение к расчету расстояния на различных широте и долготе.

После нахождения угла, вычисляем длину дуги и заносим полученное расстояние в таблицу расстояний вместе с номером текущего маршрута.

Замечание: так как функции косинуса и арккосинуса оперируют с углами, заданными в радианах, в программе производится пересчет радианов в градусы и наоборот. Все эти вычисления приводят к накоплению погрешности.

Все упомянутые формулы даны в Приложении Д.

Вычислив расстояние, выводим его на план рядом с текущей точкой, если включен соответствующий флажок. Причем в связи с условием оценки длины маршрута на рисунке отображается не длина одного отрезка, а сумма длин отрезков, предшествующих этой точке. Длины же отдельных отрезков пользователь может посмотреть в таблице расстояний.

Также на карте отображаются тип объекта для данной точки, если флажок "показывать тип объекта" установлен.

Если флажок "не показывать таблицу расстояний" отключен, делаем ее видимой.

После того, как все маршрут отрисованы, освобождаем память, отведенную под массивы индексов и координат.

# 4.2 Блок-схема алгоритма

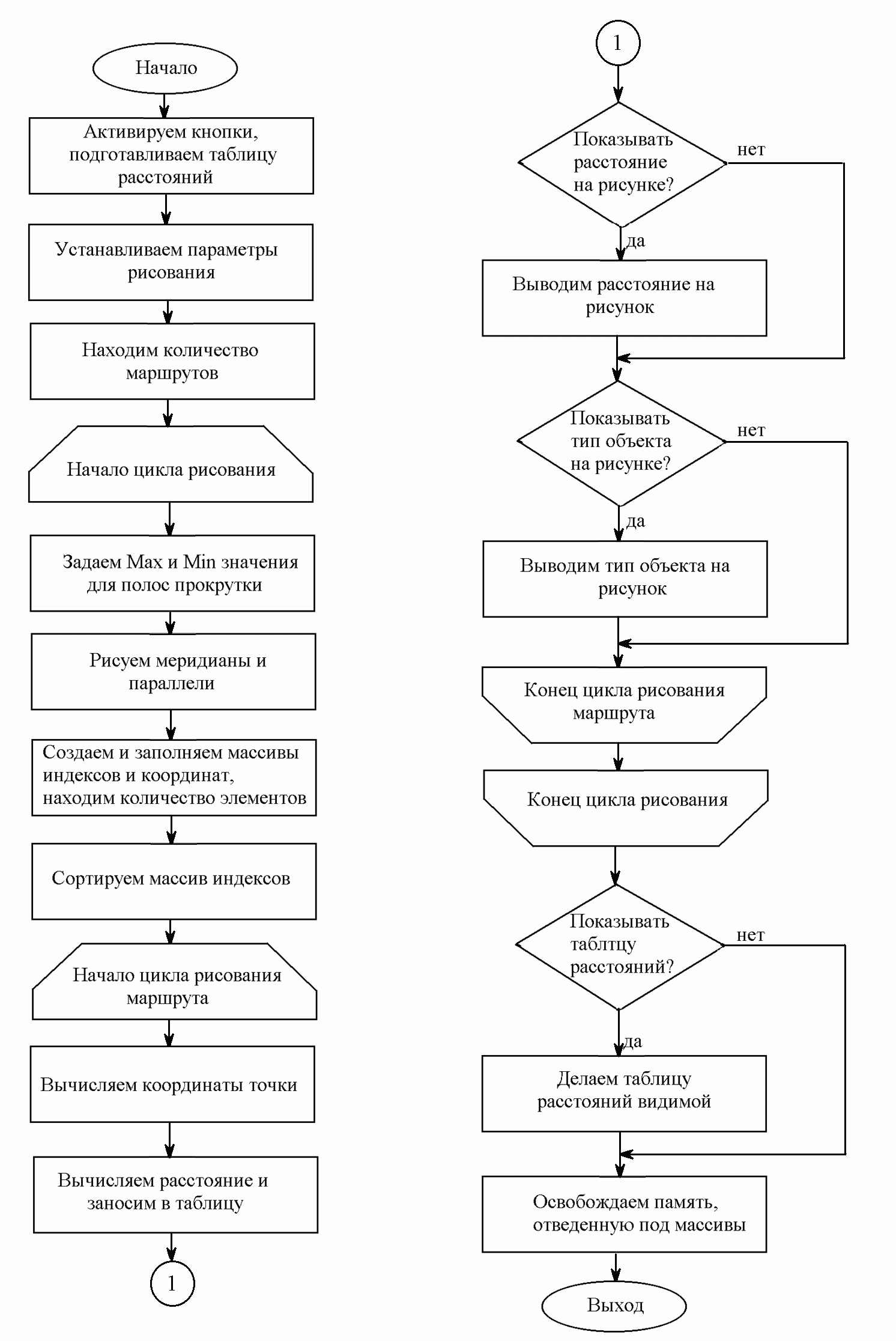


Рис. 4.2. Блок-схема алгоритма рисования плана.

##### 

##### 5. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

# 5.1 Выбор среды разработки программы

Как уже говорилось в "Постановке задачи", для создания этой программы была выбрана среда разработки Borland C++Builder 5. Это решение обуславливается тем, что в ней процесс создания интерфейса не представляет труда даже для программиста, столкнувшегося с Builder’ом впервые, и там хорошо поставлена работа с базами данных и графикой, что как раз и нужно для разработки нашей программы.

Недостатком на мой взгляд является большой исполняемый код программы – чтобы она работала на машине, где Builder не установлен, необходимо включить в него все используемые библиотеки, из-за чего размер программы становиться в несколько раз больше.

Кроме того в Builder хорошо и то, что компоненты обладают множеством свойств, которые можно менять не только при конструировании, но и при выполнении программы, что деалет раоту с ними более гибкой.

# 5.2 Работа с таблицами

В качестве драйвера базы данных используем Paradox. Этот тип базы данных был выбран, во-первых, потому, что Builder имеет встроенные средства для работы с таблицами Paradox, такие как Borland Database Engine, а также с ним поставляется программа Database Desktop. Во-вторых, преимущество Paradox состоит еще и в том, что в качестве имени базы данных можно указывать путь к каталогу, где находится файл таблицы, причем все таблицы хранятся в отдельных файлах. В третьих, они занимают мало места и являются наиболее простыми из локальных таблиц. В-четвертых, именно таблицы Paradox позволяют создавать ключевые поля.

Чтобы обеспечить работу с таблицей мы установили на форме следующие компоненты:

- сетка DBGrid, с помощью которой мы можем вставлять, удалять, или редактировать данные в таблице, или просто отображать их.

- список DBComboBox, с помощью которого мы можем вставлять данные из таблицы объектов в текущую запись главной таблицы. Перед использованием этого списка его нужно заполнить значениями из таблицы объектов. Для этого при запуске программы проходим по всем ее записям и заносим их содержимое в поле Items этого списка.

- навигатор DBNavigator – с помощью него можно удалять, добавлять, редактировать записи в таблице, а также перемещаться по ним. Связываем его с сеткой, в которой будет отображаться наша таблица.

Все эти компоненты связываются с таблицей через DataSourse, в свойствах которого мы указываем ее имя. Причем сама таблица тоже представляется в виде компонента, Table, у которого наибольшее внимание следует уделить трем свойствам:

1)Active – показывает, активна ли таблица. Если попытаться обратиться к ней, если это свойство отключено, то программа выдаст ошибку "Cannot perform this operation on closed dataset". При тестировании все возможные ситуации, которые могли привести к этой ошибке, были отслежены и обработаны.

2)DatabaseName – имя базы данных, в качестве него берется путь к каталогу, из которого открыта или создана таблица.

3)TableName – имя таблицы - файл с расширением .db, в котором храниться таблица.

Все эти свойства меняются в процессе выполнения программы.

Компонент Table нужен нам для того, чтобы избавиться от конкретизации. Те мы можем под видом Table открыть любую конкретную таблицу, и изменить при этом только свойства Table, не затрагивая других компонентов. Например, при открытии таблицы мы закрываем и дезактивируем предыдущую, при этом имя базы данных и имя таблицы в свойствах компонента Table удаляются, и открываем и активируем новую, при открытии записав в DatabaseName и TableName новые имена каталога и файла. (См. методы TBOpenFileClick и TBCloseFileClick в Приложении А).

Компоненты DataSourse и Table располагаются на форме, но они видны только при работе с программой в Builder’е.

# 5.3 Работа с графикой

Для рисования плана мы используем компонент Image. Этот объект обладает двумя важными свойствами:

1)Picture – представляет собой объект класса TPicture, который является контейнером для графики любого вида. Т.е. этот компонент может хранить bitmap-графику, иконку или другой вид графики, определенный пользователем. В Picture как раз и находится наш рисунок. С его помощью мы сохраняем в файл полученный рисунок. (См Приложение А, метод TBSaveFileClick). Также следует учесть, что размер Picture и Image могут не совпадать. Об этой проблеме более подробно будет рассказано ниже.

2)Canvas – канва. Весь процесс рисования осуществляется именно на канве компонента Image. Canvas позволяет устанавливать параметры пера, кисти, шрифта, выполнять рисование таких объектов, как линии, прямоугольники, эллипсы, а также выводить текст.

В нашей программе мы используем рисование линий с помощью методов канвы MoveTo и LineTo, рисование точек с помощью Ellipse, а также вывод текста методом TextOut. (см. Приложение А, plandraw).

Т.е. канва позволяет нам работать с функциями Windows GDI, не обращаясь к ним непосредственно, что значительно упрощает работу с графикой.

# 5.4 Разработка интерфейса

Разработке интерфейса также следует уделить особое внимание, так как необходимо сделать его удобным для пользователя. Средства Builder'а позволяют легко реализовать интерфейс в стандартах Windows.

Основные элементы стандартного интерфейса Windows-программ:

1) *Меню* - обеспечивают простой путь для выполнения пользователями логически сгруппированных команд.

Главное меню создается с помощью компонента MainMenu, причем его очень удобно редактировать при создании программы, так как для проверки не нужно запускать ее - все его содержимое отображается уже на форме.

2) *Панель инструментов* - содержит кнопки инструментов, которые соответствуют элементам в меню программы и дают пользователю большее количество прямого доступа к ее командам.

Панель инструментов реализуется с помощью компонента ToolBar, позволяющего быстро добавлять и размещать кнопки. Все кнопки инструмента на инструментальной панели имеют одинаковую ширину и высоту.

Для каждой кнопки можно задать иконку, отображающую реализуемое ею действие выбором номера иконки из хранящихся в компоненте ImageList изображений. Это удобно тем, что можно легко сменить пиктограмму, не загружая каждый раз иконку из файла.

В соответствие со стандартом Windows, я создала на панели кнопки, соответствующие пунктам меню "Новый", "Открыть", "Сохранить", "Печать".

3) *Кнопки* – с их помощью пользователь запускает выполнение действия, приписанного этой кнопке.

Обычно по стандарту интерфейса Windows кнопки почти не используются, чтобы не загромождать окно программы. Вместо них применяются панели инструментов и меню.

Но так как наша программа выполняет не очень большое количество действий, можно использовать и кнопки. Это удобно хотя бы тем, что, в отличии от меню, они располагаются рядом с тем элементом, по отношению к которому осуществляется вызываемое ими действие (например, кнопки добавления и удаления объектов, находятся рядом со списком объектов, а кнопки масштабирования – рядом с рисунком), что конечно же будет удобно для пользователя.

К тому же на них можно сделать подписи, объясняющие их назначение, что тоже удобн.

Кнопка создается размещением на форме компонента Button. При разработке программы используется его свойство Caption – надпись на кнопке, а во время выполнения - свойство Enabled, чтобы сделать кнопку неактивной или наоборот активировать ее. (см. Приложение А, HideButtons() и ShowButtons()). Например, при открытии таблицы мы активируем кнопки рисования плана, а при закрытии – делаем их неактивными.

4) *Выпадающие списки* – применяются, чтобы пользователь мог выбрать элемент из списка. Это гораздо удобнее, чем вводить его вручную.

В данной программе поле со списком используется для внесения типа объекта в таблицу (DBComboBox, о нем говорилось выше), и для удаления типа объекта из таблицы объектов (ComboBox).

Отличие DBComboBox от ComboBox заключается в том, что первый связан с полем таблицы, а второй является простым списком. Использование простого списка ComboBox позволяет решить проблемы взаимодействия с таблицей и DBComboBox, которые возникают при удалении объекта из таблицы.

5) *Полосы прокрутки* – реализуются с помощью компонента ScrollBox, в нашем случае они необходимы для прокрутки изображения плана при его увеличении. В программе осуществляется изменение в зависимости от масштаба параметра Max, который определяет максимальную величину прокрутки.

6) *Флажки* (переключатели) – определяют, включена или нет представляемая ими опция.

В Builder’e это компонент CheckBox.

В программе флажки определяют включение или выключение следующих возможностей:

- рисование линий маршрутов;

- отображение расстояния на плане;

- отображение типа объекта на плане;

- показать таблицу расстояний.

Обычно флажки находятся в меню настроек. Но в связи с простотой нашей программы, они вынесены на панель рядом с полем рисунка, что очень удобно для пользователя.

7) *Вкладки* – в данной программе таблица и рисунок плана располагаются на разных вкладках. Я считаю, что так для пользователя будет удобнее, так как вся программа содержится в одном окне, и при этом и область таблицы, и область рисунка располагаются не в ущерб друг другу.

В Builder есть два типа вкладок: TabControl и PageControl.

В программе используется PageControl, так как в отличие от первого, этот компонент может содержать на вкладках разнородные элементы управления. У TabControl все страницы одинаковы.

8) Диалоги – очень удобное средство, помогающее реализовать такие диалоги с пользователем, как используемые в данной программе сохранение, открытие файла, печать и выбор цвета.

Все необходимые диалоги размещаются на форме, но для пользователя остаются невидимыми. Они вызываются программой и реализуют стандартный интерфейс Explorer’а.

С помощью диалогов OpenDialog и SaveDialog программа получает имя файла и каталога, который выбирает пользователь. С помощью диалога печати PrintDialog вызываются настройки печати. А ColorDialog помогает выбрать цвет (он используется для выбора цвета фона рисунка).

Назначение каждого видимого для пользователя компонента интерфейса описано в руководстве пользователя.

Стоит заметить, что у каждого объекта интерфейса есть такое свойство, как Align – выравнивание. Оно позволяет выровнять элемент управления по правой, левой, верхней или нижней стороне формы и панели, причем элемент остается там, даже если размер формы меняется. Например, панель с кнопками возле рисунка выровнена по правому краю, таблица расстояний – по левому, а поле рисунка выровнено по alClient, те занимает все свободное пространство между таблицей и панелью. При растягивании форму эти элементы управления тоже будут растягиваться, оставаясь при этом на своих местах.

Использование свойства Align удобно тем, что уж на этапе проектирования можно расположить компоненты так, как нужно, и чтобы потом не задавать вручную их размеры при каждом изменении формы.

# 5.5 Некоторые особенности алгоритмов

Алгоритм рисования

Так как он был уже довольно подробно описан, добавлю лишь одно замечание касательно его разработки: он рисует карту в *равноугольной* проекции. Те сохраняется равенство углов между направлениями на карте и в натуре. При этом искажаются размеры площадей территорий.

Этот вид проекции был выбран потому, что для него проще всего осуществить рисование координатной сетки, так как она состоит из прямых линий. (Более подробно о картографических проекциях и координатной сетке см. Приложение Е.)

Расстояния же рассчитываются в реальных размерах.

Алгоритм расчета расстояния.

О нем также подробно рассказывается в "Описании алгоритма". Он является частью алгоритма рисования.

Он был выведен автором программы, так как нигде не нашлось его аналогов. Его особенность состоит в том, что необходимо учитывать кривизну земной поверхности, те вычислять длину дуги. Это особенно сложно, если точки располагаются на равной широте и долготе.

В качестве формы земной поверхности берется шар. Это может привести к погрешностям, так как форма земли на самом деле представляет с собой эллипсоид, но значительно упрощает расчеты. Средний радиус земли берется 6371 км. Это дает погрешность около 0.3%.

Также следует отметить, что для вычисления расстояния, особенно в самом сложном случае, применяется несколько последовательных расчетов, в результате чего накапливается погрешность. Также на величину ошибки влияет и пересчет градусной меры угла в радианы и обратно. Но при современной точности вычислений на компьютере эта ошибка будет невелика, к тому же задании не указывалось, что оценка маршрута должна быть точной.

## Создание таблицы

Тут следует сказать, что алгоритм создания таблицы программным способом не был мною найден ни в одной имеющейся у меня книге. Во всей литературе говорилось о создании таблиц в DatabaseDesktop, что для пользователя было бы неудобно – ставить программу еще большего размера, только чтобы решить проблему создания новых таблиц.

Но все-таки этот алгоритм был найден в помощи Builder’а, хотя там он содержал ошибки.

В программе же представлен его рабочий пример (см. Приложение А, TBNewFileClick).

В качестве особенности этого алгоритма следует отметить то, что до того, как вызвать процедуру создания таблицы CreateTable(), нужно проинициализировать все поля, указав их название, тип, длину (если требуется), и требуемость значения (те обязательно его наличие или нет). После инициализации полей объявляем ключевое поле, и только потом вызываем процедуру создания таблицы. После этого необходимо привязать новую таблицу к компоненту Table, чтобы можно было с ней работать с помощью сетки и навигатора. Имя файла новой таблицы запрашивается с помощью SaveDialog.

## Проблема размера графики

При разработки программы возникла довольно-таки серьезная проблема с рисованием картинки – при изменении размеров формы должен был меняться и размер картинки, но этого не происходило.

Причиной было то, что размер компонента Image все-таки изменялся, а вот размер картинки Picture оставался прежним. В связи с этим была создана процедура ResizeForm (см. Приложение А), которая реагировала на изменение размеров формы и менял размер картинки в соответствии с изменившимся размером компонента Image.

# 

# 5.6 Тестирование

В течение всего времени проектирования программы выполнялись ее тестирования и отладка. Особое внимание уделялось двум моментам – правильности работы с таблицами и правильности рисования плана.

При работе с таблицами в первую очередь нужно было отследить все проблемы, связанные с открытием, созданием таблиц и рисованием. Например, если не дезактивировать кнопки рисования, то нажатие на них при неоткрытой таблице могло привести к ошибке. Для устранения такой ошибки при отсутствии открытой таблицы кнопки делаются неактивными.

Еще одной важной ошибкой, которая была устранена – проблема размера графики (см. выше).

Также нужно было следить, чтобы при отключении/включении флажков и изменении размеров формы не рисовалась картинка, если пользователь еще не нажимал на кнопку "рисовать". Такое отслеживание в программе выполняется флагом draw, который устанавливается, когда пользователь нажимает на кнопку рисования, и снимается, когда нажимает кнопку очистки. В процессе тестирования все ситуации, в которых необходима проверка этого флага, были отслежены и отлажены.

Важнейшей стороной тестирования была проверка работоспособности программы на компьютере, на котором не установлен Builder. Это помогло определить следующее:

1) при компиляции программы необходимо включить в нее все используемые библиотеки. Это достигается с помощью отключения двух опций в настройках компилятора. Исполняемый код программы при этом становится больше, но зато она сможет работать и на машине без Builder’а.

2) для работы программы требуется Borland Database Engine. Если его нет на компьютере, его необходимо установить.

##### 6. ОПИСАНИЕ ПРОГРАММЫ

Программа предназначена для информационной поддержки создания плана местности. Она позволяет хранить данные о точках плана в таблице, осуществлять их графическое отображение, создавать новую таблицу, добавлять и удалять типы объектов, рассчитывать расстояния и оценивать длину маршрута, рисовать несколько маршрутов, сохранять результаты в файл или выводить на печать.

Программа создана для операционной системы Windows, и обладает стандартизованным под нее и удобным для пользователя интерфейсом. Вместе с ней поставляется пример в виде таблицы с маршрутами поездов.

##### 7. ИНСТРУКЦИЯ ПО УСТАНОВКЕ

Для установки программы должны быть выполнены требования: процессор 233МГц и выше, оперативная память от 16Мб, ОС Windows98 и выше.

Чтобы установить программу, необходимо проделать следующие действия:

1) Создать новую папку для программы.

2) С носителя, на котором находится архив программы (дискета или диск), скопировать его в эту папку.

3) Распаковать архив в эту папку.

4) Убедитесь, что фалы objects.db и rasst.db находятся в той же папке, что и программа.

5) Убедитесь, что у этих файлов снят атрибут "Только чтение". Если нет, снимите его.

6) Распакуйте архив с библиотеками в папку Windows.

7) Теперь можно запустить программу и пользоваться ею.

##### 8. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

# 8.1 Главное меню

## Меню "Файл"

Для того, чтобы начать работу в программе, необходимо открыть файл с таблицей или создать новый. Это можно выполнить с помощью пунктов меню "Файл" "Открыть" и "Новый".

"Открыть" - открыть уже существующую таблицу. Вызывает диалог, а котором пользователь должен выбрать файл с расширением .db.

"Новый" - создание новой таблицы. Вызывается диалог, в котором пользователь задает имя новой таблицы.

"Закрыть" - закрывает таблицу.

"Сохранить" - сохраняет картинку в файл с расширением .bmp. Таблица же сохраняется автоматически при утверждении изменений кнопкой "Сохранить изменения" навигатора.

"Печать" - в зависимости от открытой вкладки печатает рисунок или таблицу. При выборе этого пункта открывается диалоговое окно настройки печати, после нажатия кнопки "ОК", задание отправляется на печать.

## Меню "Таблица"

"Построить план" - если таблица открыта, активирует вкладку "План" и рисует план.

"Добавить тип объекта" - вызывает форму добавления типа объекта.

"Удалить тип объекта" - вызывает форму удаления типа объекта.

## Меню "Помощь"

"О программе" - выводит название программы и сведения об авторе.

"Содержание" - выводит файл подсказки.

# 8.2 Панель быстрых кнопок

Действия этих кнопок аналогичны одноименным пунктам меню "Файл".

- "Новый" - создание новой таблицы.



- "Открыть" - открыть уже существующую таблицу.



- "Сохранить" - сохраняет картинку в файл с расширением .bmp.



- "Печать" - вывести на печать рисунок или таблицу.



# 8.3 Вкладка "Таблица"

*Сетка таблицы* – в нее загружается таблица. С помощью этой сетки пользователь может редактировать и просматривать записи таблицы.

*Навигатор* – располагается под сеткой, позволяет редактировать и просматривать таблицу.

### Кнопки навигатора

- "Первая запись" - выполняет переход на первую запись в таблице.



- "Предыдущая запись" - выполняет переход на предыдущую запись в таблице.



- "Следующая запись" - выполняет переход на следующую запись в таблице.



- "Последняя запись" - выполняет переход на последнюю запись в таблице.



- "Добавить запись" - таблица переводится в режим редактирования, перед активной записью вставляется новая пустая запись.



- "Удалить запись" - удаляет текущую запись, предварительно запросив подтверждение.



- "Редактировать" - редактирование текущей записи.



- "Сохранить изменения" - заносит изменения текущей записи в таблицу.



- "Отменить изменения" - отменяет изменения текущей записи, возвращая ее предыдущее значение.



- "Обновить". Обновляет таблицу в сетке.



Под навигатором находятся Список объектов и кнопки удаления и добавления объектов.

*Список объектов* – содержит перечень типов объектов. При выборе типа из списка, он заносится в текущую запись в таблице.

*Добавить тип объекта* – вызывает форму добавления типа. При его добавлении он тут же заносится в список.

*Удалить тип объекта* – вызывает форму удаления типа объекта.

## Форма добавления типа объекта

Содержит *поле редактирования* для ввода текста, в которое заносится название нового типа.

"Добавить" - добавляет тип объекта к списку и в таблицу, не закрывая форму.

"Ок" - если объект не был добавлен, заносит его в список и таблицу, и закрывает форму.

"Отмена" - закрывает форм без добавления.

## Форма удаления типа объекта

*Список объектов* – из него пользователь выбирает, какой объект нужно удалить.

"Удалить" - удаляет выбранный тип объекта из таблицы и списка.

"Удалить все" - полностью очищает таблицу объектов и списки.

"ОК" - закрывает форму.

# 

# 8.4 Вкладка "План"

*Поле рисунка* – область, на которую выводится рисунок плана.

*Полосы прокрутки* – появляются при увеличении изображения нажатием на кнопку "Увеличить". Позволяют прокручивать увеличенное изображение.

*Таблица расстояний* – показывает для каждого маршрута его номер и длины отрезков, в километрах. Таблица видима, если снят флажок "Скрыть таблицу расстояний".

Кнопки:

"Очистка карты" - очищает поле рисунка.

"Нарисовать" - рисует план.

"Цвет фона" - позволяет выбрать цвет фона рисунка. При нажатии на эту кнопку выводится диалоговое окно выбора цвета.

"+ Увеличить" - увеличивает изображение на плане.

"- Уменьшить" - уменьшение изображения на плане. Эта кнопка становиться активной при нажатии кнопки увеличения.

"По умолчанию" - устанавливает изначальный размер изображения.

Флажки:

"Рисовать линии маршрутов" - если он установлен, на плане будут рисоваться маршруты, если нет – будут обозначаться только точки.

"Показывать расстояния на карте" - если флажок установлен, рядом с каждой точкой маршрута выводится расстояние в виде сумм длин отрезков, предшествующих этой точке. Возле первой точки маршрута ставится "0".

"Показывать тип объекта на карте" - если флажок установлен, рядом с каждой точкой выводится тип объекта.

"Скрыть таблицу расстояний" - если флажок установлен, таблица расстояний невидима. Если снят, таблица появится справа от поля рисунка.

Все флажки по умолчанию установлены.

##### 9. КОНТРОЛЬНЫЙ ПРИМЕР

В качестве контрольного примера изобразим маршруты нескольких поездов Горьковской железной дороги.

Возьмем следующие поезда:

N 497Г Горький-Моск - Адлер

N 471Г Горький-Моск - Новороссийск

N 431Г Горький-Моск - Адлер

N 367Г Горький-Моск - Самара

N 059A Горький-Моск - Санкт-Петербург-Главн

N 039Г Горький-Моск - Москва Курская

Занесем в таблицу в качестве точек каждого маршрута координаты наиболее крупных населенных пунктов. В качестве объектов будем использовать названия городов.

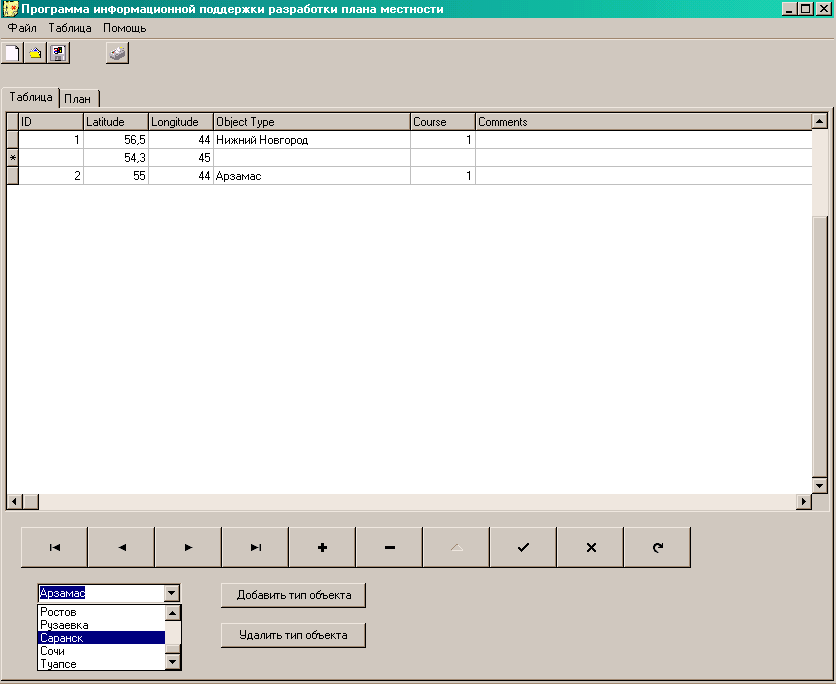


Рис. 9.1. Общий вид программы на вкладке "Таблица", ввод в запись названия объекта с помощью списка.

Ранее мы занесли в таблицу объектов станцию Красный Узел, но так как она нам не понадобиться, мы ее удалим.

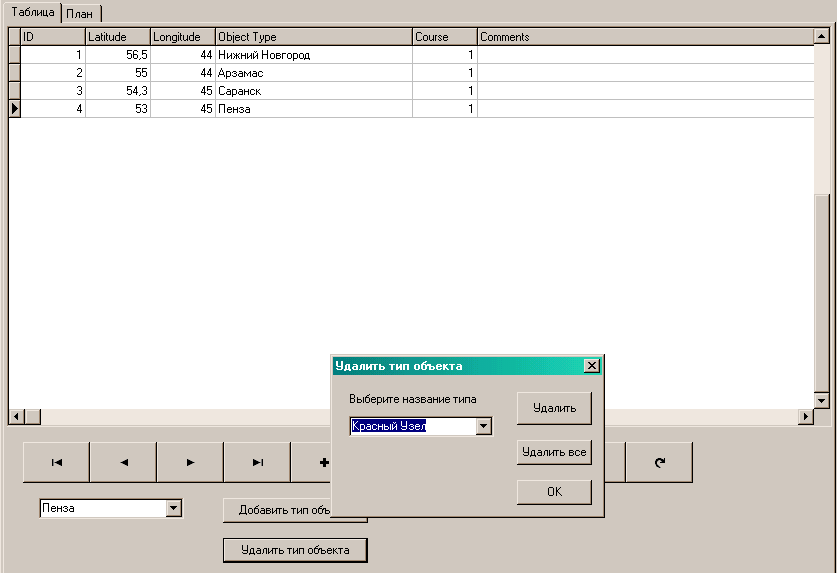


Рис. 9.2. Удаление типа объекта.

При наборе маршрута поезда 431 нам понадобиться Владимир. Так как его нет в списке, нужно его добавить.

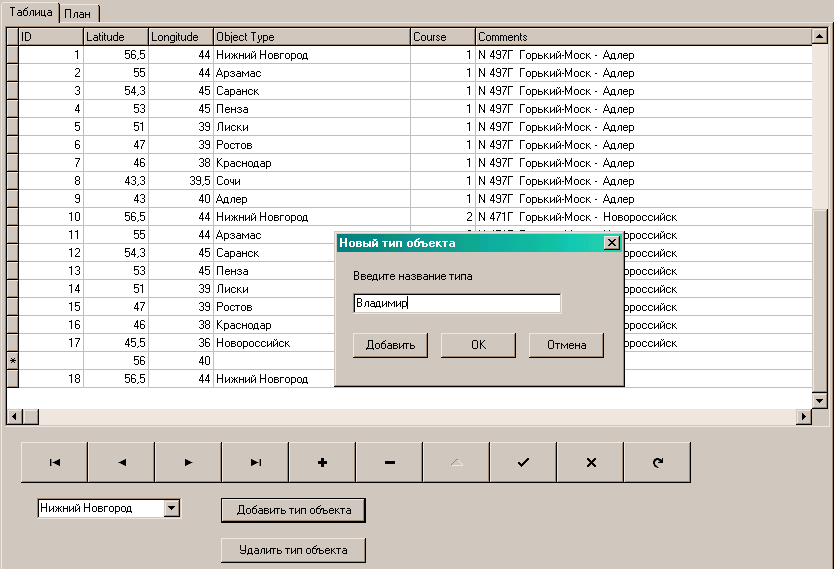


Рис. 9.3. Добавление типа объекта.

Итак, мы набрали все маршруты, какие хотели. В итоге у нас получилась такая таблица.

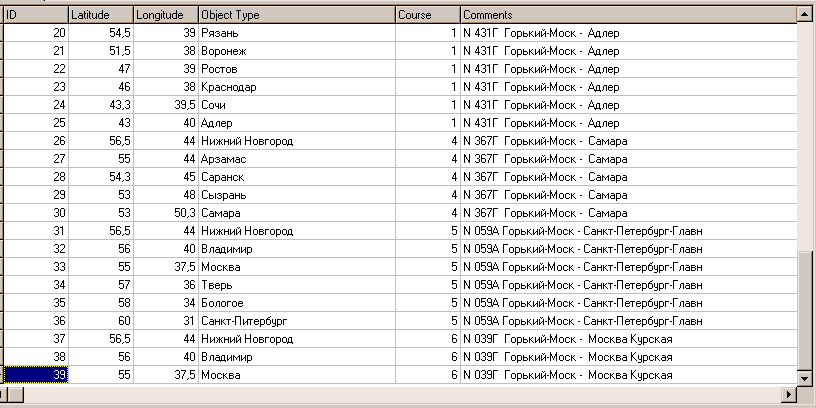


Рис. 9.4. Таблица

Теперь по нашей табличке построим план.

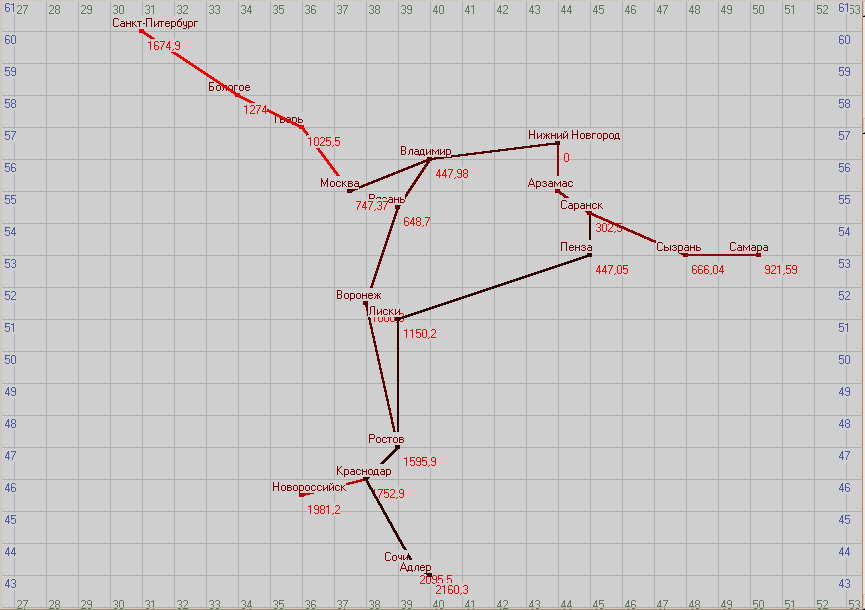


Рис. 9.5. Изображение плана со всеми надписями.

По умолчанию у нас отображаются и линии, и расстояния, и названия городов.

Посмотрим, как будут выглядеть маршруты без подписей.

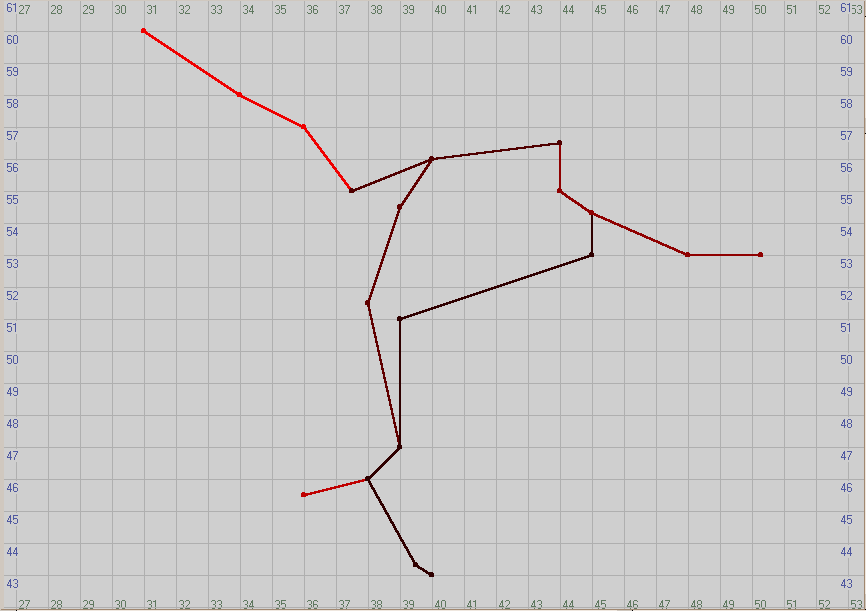


Рис. 9.6. Изображение плана без всех надписей.

Теперь посмотрим только расстояния.

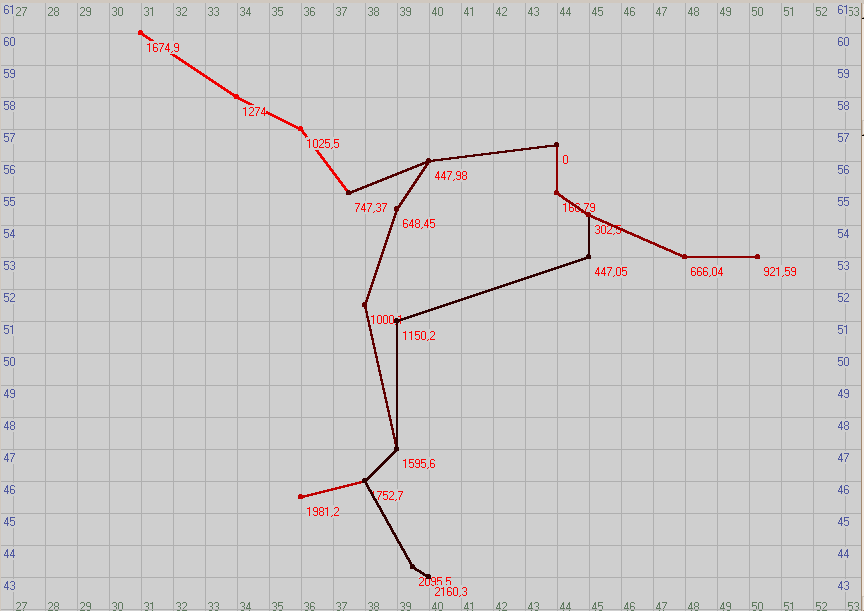


Рис. 9.7. Изображение плана только с расстояниями.

На самом деле они рассчитаны не очень точно, но это оттого, что мы учли не все пункты этих маршрутов, а также координаты взяты не совсем верные - они были определены по карте "на глаз".

Теперь посмотрим наш план только с названиями городов.

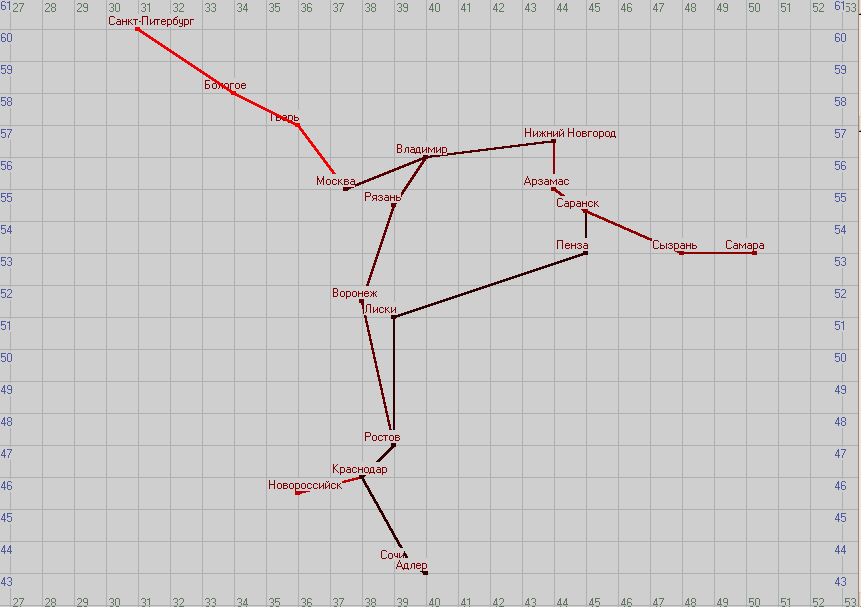


Рис. 9.8. Изображение плана с выводом только типов объектов.

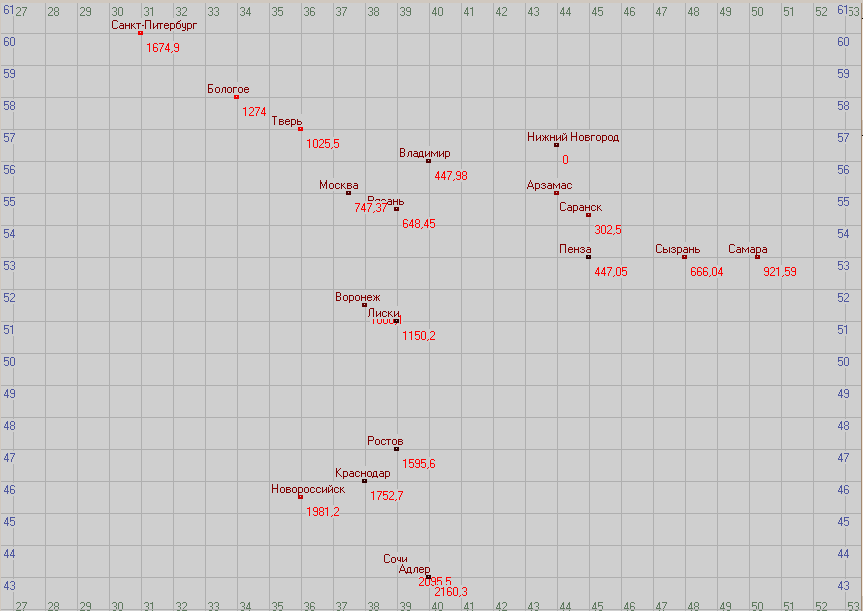


Рис. 9.9. Изображение плана в виде точек со всеми подписями.

Можно посмотреть и таблицу расстояний.

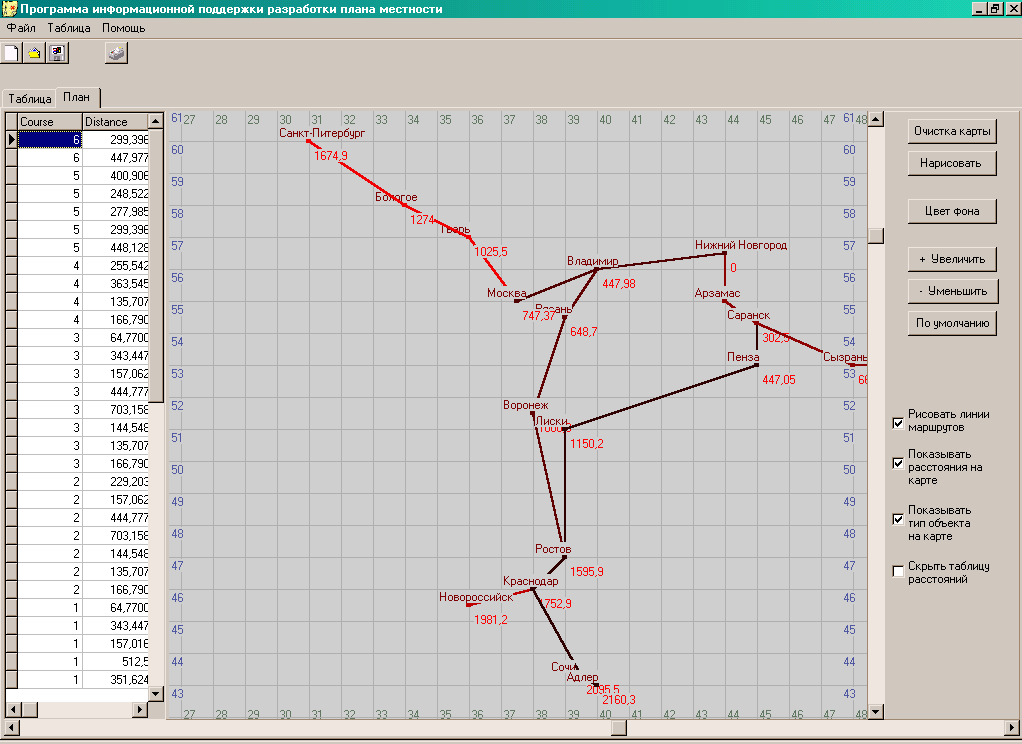


Рис. 9.11. Общий вид программы на вкладке "План" с видимой таблицей расстояний.

Еще мы можем сохранить картинку в файл или распечатать результаты. Тогда мы будем обладать своей собственной схемой маршрутов поездов.

##### ЗАКЛЮЧЕНИЕ

Задача, поставленная в задании бакалаврской работы, была успешно выполнена. Разработанная программа полностью удовлетворяет начальным условиям, описанным в постановке задачи. В частности в программе были реализованы работа с таблицами, графикой и файлами.

Работа пользователя программы осуществляется с помощью несложного интерфейса, выполненного по стандартам интерфейса программ Windows.

В качестве дальнейших улучшений и расширения возможностей данного проекта можно указать добавление расчета кратчайшего маршрута, нанесение на рисунок пиктограмм объектов, изображение рельефа, а также возможность загрузки в качестве фона изображения карты.

##### СПИСОК ЛИТЕРАТУРЫ

1. *П. Густавсон, М. Кэшмэн, Б. Сворт, Дж. Холингворт*. Borland C++ Builder 6. Руководство разработчика. – Вильямс, 2004.

2. *А. Архангельский.* Программирование в C++ Builder 6. – Бином, 2002.

3. *Т.А. Павловская*. С/С++. Программирование на языке высокого уровня. – Питер, 2001.

##### ПРИЛОЖЕНИЕ

# ПРИЛОЖЕНИЕ Е. Картографические проекции и координатная сетка

Сферическую поверхность развернуть на плоскости без разрывов и складок невозможно, то есть ее плановое изображение на плоскости нельзя представить без искажений, с полным геометрическим подобием всех ее очертаний. Полного подобия спроектированных на уровненную поверхность очертаний островов, материков и различных объектов можно добиться лишь на шаре (глобусе). Изображение поверхности Земли на шаре (глобусе) обладает равномасштабностью, равноугольностью и равновеликостью.

Эти геометрические свойства одновременно и полностью сохранить на карте невозможно. Построенная на плоскости географическая сетка, изображающая меридианы и параллели, будет иметь определенные искажения, поэтому будут искажены изображения всех объектов земной поверхности. Характер и размеры искажений зависят от способа построения картографической сетки, на основе которой составляется карта.

Отображение поверхности эллипсоида или шара на плоскости называется*картографической проекцией.* Существуют различные виды картографических проекций. Каждому из них соответствуют определенная картографическая сетка и присущие ей искажения. В одном виде проекции искажаются размеры площадей, в другом - углы, в третьем - площади и углы. При этом во всех проекциях без исключения искажаются длины линий.

Картографические проекции классифицируют по характеру искажений, виду изображения меридианов и параллелей (географической сетке) и некоторым другим признакам. По характеру искажений различают следующие картографические проекции:

**-** *равноугольные***,** сохраняющие равенство углов, между направлениями на карте и в натуре. На рис.Е.1 показана карта мира, на которой картографическая сетка сохраняет свойство равноугольности. На карте сохранено подобие углов, но искажены размеры площадей. Например, площади Гренландии и Африки на карте почти одинаковы, а в действительности площадь Африки примерно в 15 раз больше площади Гренландии.

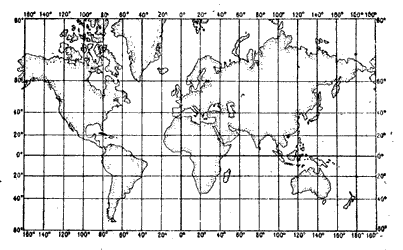


Рис.Е.1 Карта мира в равноугольной проекции.

**-** *равновеликие***,** сохраняющие пропорциональность площадей на карте соответствующим площадям на земном эллипсоиде. На рис Е.2 показана карта мира, составленная в равновеликой проекции. На ней сохранена пропорциональность всех площадей, но искажено подобие фигур, то есть отсутствует равноугольность. Взаимная перпендикулярность меридианов и параллелей на такой карте сохраняется только по среднему меридиану.

- *равнопромежуточные*, сохраняющие постоянство масштаба по какому-либо направлению;

- *произвольные***,** не сохраняющие ни равенства углов, ни пропорциональности площадей, ни постоянства масштаба. Смысл применения произвольных проекций заключается в более равномерном распределении искажений на карте и удобстве решения некоторых практических задач.

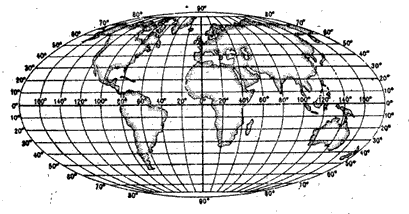


Рис. Е. 2 Карта мира в равновеликой проекции.

По виду изображения сетки меридианов и параллелейкартографической проекции подразделяются на *конические, цилиндрические, азимутальные и др***.** Причем в пределах каждой из этих групп могут быть разные по характеру искажений проекции (равноугольные, равновеликие и т. д.). Геометрическая сущность конических и цилиндрических проекций заключается в том, что сетка меридианов и параллелей проектируется на боковую поверхность конуса или цилиндра с последующим развертыванием этих поверхностей в плоскость. Геометрическая сущность азимутальных проекций заключается в том, что сетка меридианов и параллелей проектируется на плоскость, касательную к шару в одном из полюсов или секущую по какой-либо параллели. Картографическую проекцию, наиболее подходящую по характеру, величине и распределению искажений для той или иной карты, выбирают в зависимости от назначения, содержания карты, а также от размеров, конфигурации и географического положения картографируемой территории. Благодаря картографической сетке все искажения, как бы велики они ни были, сами по себе не влияют на точность определения по карте географического положения (координат) изображаемых на ней объектов. В то же время картографическая сетка, являясь графическим выражением проекции, позволяет при измерениях по карте учитывать характер, величину и распределение искажений. Поэтому любая географическая карта представляет собой математически определенное изображение земной поверхности.