Владимирский государственный университет

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ САМОСТОЯТЕЛЬНОЙ ПОДГОТОВКИ СТУДЕНТОВ К РАБОТЕ С ArcView-ГИС и Avenue ПРИ ВЫПОЛНЕНИИ КУРСОВОГО И ДИПЛОМНОГО ПРОЕКТИРОВАНИЯ

Владимир 2002

Министерство образования Российской Федерации
Владимирский государственный университет
Кафедра физики и прикладной математики

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ САМОСТОЯТЕЛЬНОЙ ПОДГОТОВКИ СТУДЕНТОВ К РАБОТЕ С ArcView-ГИС и Avenue ПРИ ВЫПОЛНЕНИИ КУРСОВОГО И ДИПЛОМНОГО ПРОЕКТИРОВАНИЯ

Под редакцией Л.М. Вороновой

Владимир 2002

УДК 621.3

Составители: И.С. Волкова, А.В. Жиров, С.В. Труфанов,
И.С. Крамской, Л.М. Воронова

Рецензент
Начальник Управления главного эколога администрации г. Владимира
П.Е. Широков

Печатается по решению редакционно-издательского совета
Владимирского государственного университета

Методические рекомендации для самостоятельной подготовки студентов к работе с ArcView-ГИС и Avenue при выполнении курсового и дипломного проектирования / Владим. гос. ун-т; Сост.: И.С. Волкова, А.В. Жиров, С.В. Труфанов, И.С. Крамской, Л.М. Воронова; Под ред. Л.М. Вороновой. Владимир, 2002. 76 с.

Приведены основные сведения, необходимые для работы с системой ArcView-ГИС и основные особенности работы с встроенным языком программирования Avenue. Рекомендации имеют целевое назначение – начальное знакомство студентов с указанными системами.

Предназначены для студентов старших курсов специальностей 010200 – прикладная математика и информатика и 072300 – лазерная техника и лазерные технологии при выполнении курсовых работ и дипломных проектов в ГИС-средах.

Ил. 12. Библиогр.: 6 назв.

УДК 621.3

Данные рекомендации по созданию приложений ArcView с использованием программирования на Avenue дают возможность улучшить самопрограммирование, но не являются классическим набором правил, которым должен пользоваться каждый. Они могут применяться, чтобы обеспечить базу данных для развития вашего собственного стиля. Стиль является в значительной степени вопросом вкуса. Однако стандарты полезны для читаемости и согласованности в разработке приложений. Основная цель введения стандартов программирования – сделать программы по возможности понятными для тех, кто будет читать или сопровождать их. В общем, программирование должно следовать принятым правилам и пользователь не должен столкнуться с поведением программы, выходящим из общего ряда.

Код программы должен быть компактным, но понятным. Постарайтесь найти компромисс между чрезмерной вложенностью запросов и излишней упрощенностью. Имена любых компонент должны быть значимыми, но не слишком длинными.

Используйте комментарии во всех ваших программах для объяснения того, что они делают. Организация программы должна быть в виде последовательности операций, каждая из которых сопровождается необходимыми комментариями.

Выражаем благодарность студенту-выпускнику нашей кафедры Д.Г. Вечерову (гр. ПМ-298), который проработал ряд примеров, включенных в приложение.

1. БАЗОВЫЕ ВОЗМОЖНОСТИ ArcView

ArcView – это мощный, простой в использовании инструмент, который выводит географическую, пространственно распределенную и другую информацию на экран монитора. ArcView может быть использован любым человеком, который желает обрабатывать и визуализировать данные, распределенные в пространстве. Прежде всего это относится к данным, имеющим привязку к координатам поверхности Земли.

Краткий словарь используемых терминов

SQL [Structured Query Language]	Язык структурированных запросов (международный стандартный язык для определения и доступа к реляционным базам данных)
Avenue	Встроенный в ArcView объектно-ориентированный язык программирования
Project (Проект)	Класс, объекты которого обеспечивают решение конкретных пользовательских задач и представляют собой файлы (с расширением ".apr"), сохраняющие работу пользователя в ArcView. Конкретный проект обычно содержит Виды, Таблицы, Диаграммы, Макеты и Скрипты, которые вы используете в данной работе. Это компоненты проекта. Например, если вы используете ArcView для нахождения подходящего места для нового офиса, вам лучше сохранить все Виды, Таблицы, Макеты и Скрипты, которые вы используете в этом приложении, в одном Проекте. Чтобы в дальнейшем продолжить работу, нужно лишь запустить файл соответствующего Проекта
View (Вид)	Класс типа интерактивной карты, который позволяет отображать, исследовать, делать запросы и анализировать географические данные в ArcView. Объекты типа View сохранены в проекте ArcView (Project), с которым вы в настоящее время работаете, и могут быть доступны по иконке View компонент Project
Шейпфайлы	Простой, не топологический формат данных для сохранения геометрического местоположения и атрибутивной информации о географических объектах. Это один из географических форматов данных, с которыми можно работать в ArcView. Состоит из файлов с расширениями .shp, .shx, .dbf, .sbn, .sbx, .ain, .aih. Файлы с последними 4 расширениями присутствуют не всегда
Theme (Тема)	Класс, объектом которого может быть множество географических компонент View. Тема представляет источник географических данных на диске или в сети. Источниками данных могут быть: покрытия ArcInfo и шейпфайлы ArcView; рисунки AutoCad; растровые изображения; таблицы, содержащие пары координат *X, Y*. Обычно тема представляет все объекты источника данных, однако они могут фильтроваться согласно установленному критерию. Тема имеет множество параметров, которые можно регулировать. Каждая конкретная тема объекта типа View имеет легенду, отображаемую в таблице содержания для объекта типа Theme

Table (Таблица)	Позволяет работать с табличными данными из большинства табличных источников
Атрибутивная информация об объекте	Обычно это строка таблицы в базе данных, привязанная к объекту
Script (Скрипт)	Компонент проекта ArcView, который содержит код языка программирования Avenue, аналогичен макросу, процедуре или скрипту в других языках программирования, скрипты Avenue сочетают в себе средства для достижения трех основных целей: автоматизировать решение конкретной пользовательской задачи, добавлять новые возможности в ArcView и строить законченные приложения
Chart (Диаграмма)	Визуальное представление данных Таблицы, главным образом, атрибутов географических объектов
Пользовательский интерфейс документа	Предоставляемая пользователю система окон, меню, кнопок и инструментов, позволяющая работать с документом – объектом определенного класса
Пользовательский интерфейс Приложения	Предоставляемая пользователю система окон, меню и других элементов управления, позволяющая общаться с данным приложением
Layout (Макет)	Карта, которая позволяет отображать Виды, Диаграммы, Таблицы, импортированную графику, графические примитивы; используется для компоновки их изображений и вывода на печать. Макет динамически отображает изменения объектов, его составляющих, при обновлении

Существует несколько типовых задач, для решения которых вы можете использовать ArcView:

1. **Отображение географических данных в Виде (View):** вы можете без особого труда создавать карты из существующих источников пространственных данных.

2. **Отображение табличных данных в Виде (View)**: вы можете импортировать табличные данные и затем присоединить их к данным в Виде, чтобы отобразить их географически.

3. **Использование SQL для получения значений из базы данных и отображение их в Виде**: вы можете подсоединиться к базе данных, чтобы получить табличные данные и затем использовать их географически.

4. **Создание и редактирование пространственной информации:** вы можете создать свои собственные пространственные данные, чтобы представить географические объекты, которые вы хотите отобразить и проанализировать с помощью ArcView.

5. **Получение атрибутов каких-либо объектов в Виде:** вы можете щелкнуть на объекте (в том числе и географическом) в Виде, чтобы получить его атрибуты (данные таблицы, характеристики полигона и т.д.).

6. **Отображение Тем (Theme) с помощью различных цветов и символов в соответствии с атрибутами объектов**.

7. **Выбор объектов в соответствии со значениями их атрибутов**: вы можете запросить в компоненте Вид выделить особые объекты.

8. **Создание диаграмм, отображающих атрибуты объектов**: вы можете визуализировать табличные данные с помощью создания диаграмм. Диаграммы дополняют ваши карты.

9. **Суммирование атрибутов объектов:** можно суммировать, например, данные по городам для каждой области или района. Можно также получить статистическую информацию о каком-либо атрибуте.

10. **Выбор объектов по их расположению относительно других объектов.**

11. **Нахождение областей перекрытия объектов.**

12. **Создание Макета (Layout) карты и его распечатка:** вы используете Макеты для компоновки Видов, Таблиц, Диаграмм, картинок и других элементов в качестве карты, готовой для печати.

13. **Создание Макета карты и экспорт его для использования в другой программе:** Макеты могут быть экспортированы в различные форматы.

14. **Кастомизация ArcView для приспособления его к вашим нуждам:** вы можете легко модифицировать пользовательский интерфейс ArcView реорганизуя и удаляя элементы управления. Вы также можете написать скрипты на Avenue, чтобы сделать свои собственные элементы управления.

15. **Создание собственных приложений ArcView для использования другими людьми** (Avenue предоставляет полный комплекс возможностей для разработки приложений, базирующихся на ArcView).

Работа с системой

Загрузите систему ArcView, для чего достаточно ее ярлыка на рабочем столе компьютера, либо знать путь, по которому располагается файл для запуска системы (arcview.exe): откроется главное окно ArcView (рис. 1,*а*). В ArcView вы работаете с Видами (объектами класса Views), Таблицами (класс Tables), Диаграммами (класс Charts) и Скриптами (класс Scripts),

хранящимися в одном файле, который называют Проектом и именуют с расширением .apr (это есть объект класса Project). В ArcView в одно и то же время вы можете работать только с одним Проектом. Поэтому с открытием главного окна на его панели сразу же открывается еще одно окно с запросами: “Открыть новый проект” или “Открыть существующий проект”. Выберите необходимое: в окне ArcView открывается окно проекта – Project (нового или существующего по выбору из списка) (рис. 1,*b*). Проект позволяет вам сохранить вместе все компоненты, необходимые для специфической задачи или приложения.

Окно Проекта

Заголовок окна Проекта представляет собой имя текущего Проекта (рис. 1,*g*). Окно Проекта (рис. 1,*b*) содержит наверху панель кнопок (рис. 1,*n*), слева панель со списком всех компонент (документов) проекта с возможностью манипулировать ими (рис. 1,*o*) и прокручивающийся список документов для выбранной компоненты (рис. 1,*p*).

Работа с компонентами проекта выполняется с помощью панели компонент на левой стороне окна: имеется прокручивающийся список иконок для выбора класса документов (рис. 1,*o*), объекты которого (документы) необходимо отобразить. Двойной щелчок мышью на заголовке компонента открывает его. Например, чтобы увидеть, какие таблицы содержит проект, щелкните мышкой на иконке Tables (Таблицы). Имена таблиц проекта будут представлены в виде списка в поле *p* (рис. 1). Чтобы открыть одну из таблиц в этом списке, щелкните на имени таблицы, для ее выделения, а затем щелкните кнопку проекта Open (Открыть) или сделайте двойной щелчок на имени таблицы. Чтобы открыть сразу несколько таблиц из данного списка, нажмите клавишу Shift, и выделяйте нужные названия, затем нажмите кнопку Open. Чтобы создать новую таблицу, вы можете либо нажать кнопку New (Новая) в меню окна проекта, либо сделать двойной щелчок на иконке Tables.

Каждая иконка в окне Проекта представляет способ отображения ваших данных; каждая иконка представляет пользовательский интерфейс документа. Иконки стандартных пользовательских интерфейсов документа для каждого класса документов (Виды, Таблицы, Диаграммы, Скрипты) отображаются в окне Проекта по умолчанию. Вы можете создать новые способы отображения ваших данных и добавить соответствующие иконки в окно Проекта, а также удалить любые иконки, которые не нужны для вашего приложения. Вы можете переставлять иконки, давать им другие

имена, менять изображение, изменить имена трех кнопок, расположенных вдоль верхней части окна Проекта, и то, что происходит при их нажатии.

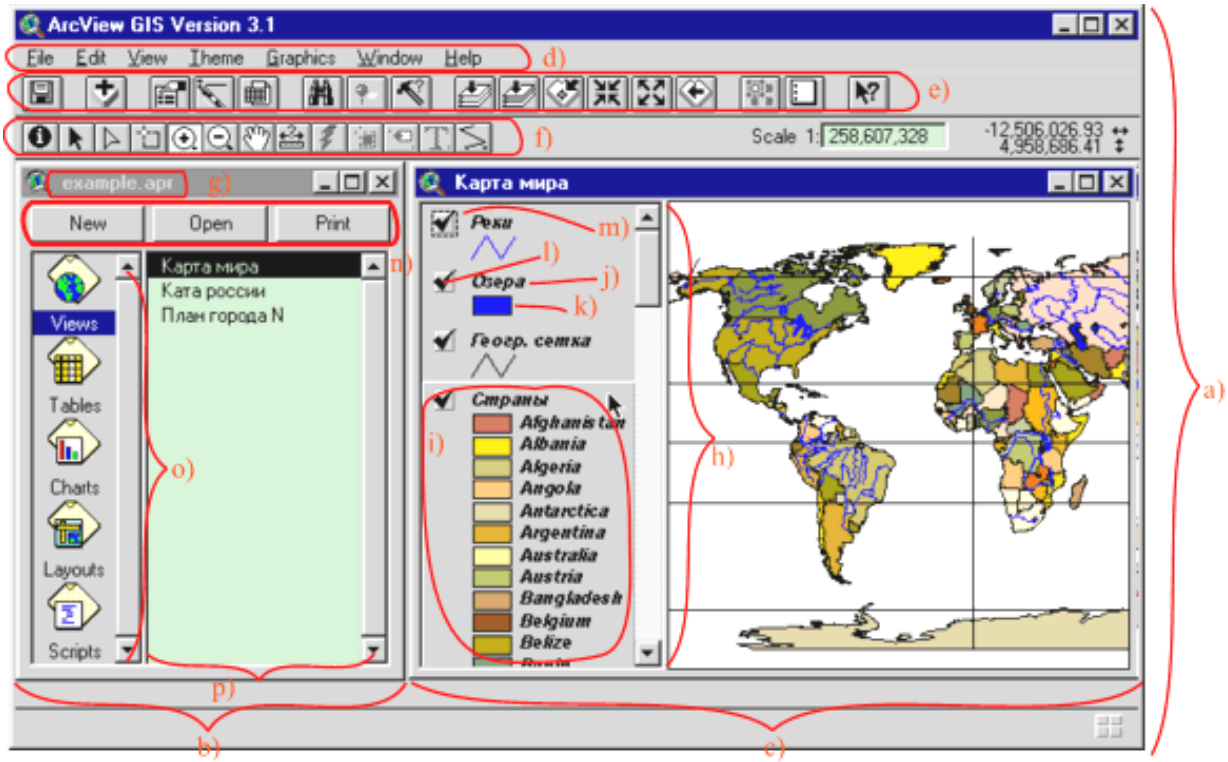


Рис. 1. Пример пользовательского интерфейса ArcView:
a – окно ArcView; b – окно Проекта “example.apr”; c – окно Вида “Карта мира”; d, e, f – панели меню, кнопок, инструментов окна ArcView; g – имя проекта в окне Проекта; o – прокручивающийся список иконок компонент Проекта (классов документов); p – прокручивающийся список документов для выбранного класса документов; n – кнопки окна Проекта для документов класса Вид; h – таблица содержания Вида; i – тема “Стра́ны” активна; j – название темы; k – легенда темы; l – тема включена; m – тема “Реки” редактируется

Чтобы сделать окно Проекта активным, когда окна документов располагаются поверх него, и вы не можете его видеть, используйте пункт главного меню Window (Окно) (рис. 1,*d*). В этом меню окно Проекта всегда на первом месте в списке открытых на данный момент окон. Вы не можете закрыть окно Проекта, если оно оказалось труднодоступным (располагается вне видимой области экрана). Чтобы восстановить ”скрытое” окно Проекта, используйте пункты Tile (Мозаика) или Cascade (Каскад) в Window главного меню. Для отказа от назначенного расположения выполните повторный выбор этого пункта.

Окна Вида, Таблицы, Диаграммы, Макета

Когда вы открываете один из компонентов проекта, он отображается внутри собственного окна. В ArcView одновременно может быть открыто несколько окон, но только одно из них может являться активным. Активное окно – это окно, с которым вы в данный момент работаете.

Все окна, которые в данный момент открыты внутри окна ArcView, перечислены в нижней части меню Window (Окно) главного меню ArcView (рис. 1,*d*). Допустим, часть окна скрыта другими окнами. Чтобы оно было расположено поверх всех окон, его нужно сделать активным, т.е. выбрать его в меню Window.

Когда вы выполняете действия в ArcView, они обычно применяются к активному окну. Интерфейс пользователя ArcView изменяется в связи с тем, какое окно является активным (рис. 2). Например, когда активно окно Проекта, вы увидите кнопки, инструменты и меню для работы с Проектами. На рис. 1 активно окно Вида “Карта мира” – *c* и интерфейс пользователя при этом – *d*, *e*, *f*.

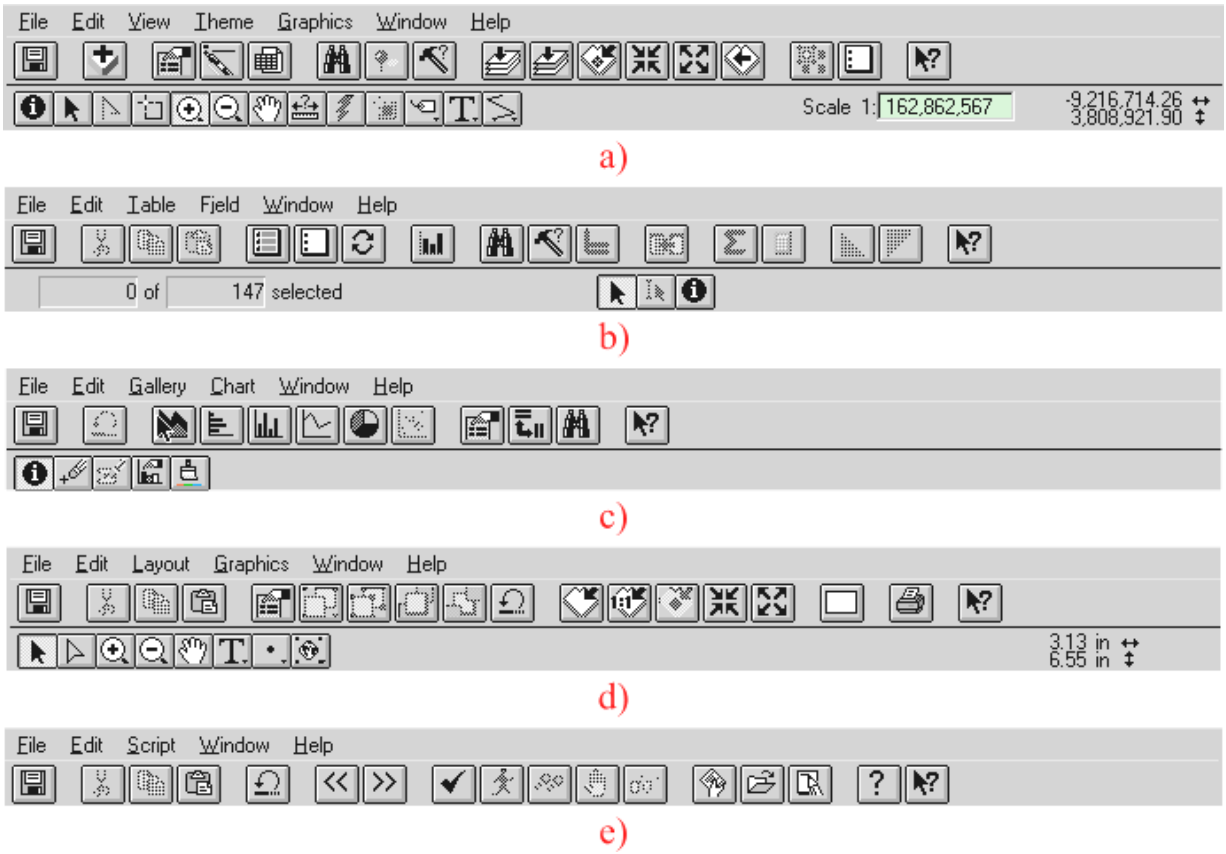


Рис. 2. Графические интерфейсы пользователя документов: a – Вида; b – Таблицы; c – Диаграммы; d – Макета; e – Скрипта

Панель меню (см. рис. 1,d)

Эта панель, расположенная вдоль верхней части окна ArcView, содержит выпадающие меню. Чтобы выбрать пункт из выпадающего меню, вы можете использовать мышь или комбинации клавиш на клавиатуре. Некоторые комбинации клавиш можно увидеть в меню. Другие зависят от графического интерфейса пользователя (ГИП) системы, в которой вы работаете. Содержание панели меню изменяется в соответствии с видом активного окна.

Панель кнопок (см. рис. 1,e)

Эта панель, расположенная ниже панели меню, содержит кнопки, дающие быстрый доступ к различным элементам управления. Щелкните на кнопке, чтобы выбрать ее. Содержание панели кнопок изменяется в соответствии с видом активного окна.

Панель инструментов (см. рис. 1,f)

Эта панель, расположенная ниже панели кнопок, содержит различные инструменты, с которыми вы можете работать. Содержание панели инструментов также изменяется в соответствии с видом активного окна. Если вы работаете в окне Проекта или в окне Скрипта, то панель инструментов не появляется вовсе. Вы щелкаете на инструменте, чтобы его использовать. После этого курсор изменяется в соответствии с выбранным инструментом. Инструмент остается выделенным, пока вы не выберите другой или не щелкните на нем снова.

Таблица содержания (см. рис. 1,h)

Каждый Вид имеет свою собственную *таблицу содержания,* которая содержит список Тем и показывает, с помощью каких символов и цветов они отображаются. Вы можете изменить то, как выглядит таблица содержания с помощью пункта TOC Style из View (Вид) главного меню. Вы также используете таблицу содержания для контроля за тем, как Вид отображается.

Таблица содержания показывает:

– **Название каждой темы в Виде**. Темам можно давать любое имя. По умолчанию именем темы является имя источника данных, например, “Cities” (для документа “Cities”) или “Rivers” (для документа“Rivers”). Вы можете давать темам длинные, более описательные имена, можете вместо “Lakes.shp” назвать тему “Озера” (см. рис. 1,*j*);

– **Легенду для каждой Темы**. Легенда Темы показывает символы и цвета, используемые для ее отображения. Тема может быть отображена с использованием одного символа либо с использованием ряда символов и цветов для классификации объектов в Теме (см. рис. 1,*k, i*);

– **Включена тема или нет**. Каждая Тема слева содержит флаговую кнопку, которая показывает, отображается в данный момент Тема в Виде или нет. Вы управляете тем, отображается тема или нет с помощью установки или снятия флажка (см. рис. 1,*l*);

– **Порядок, в котором отображаются темы**. Тема, расположенная в верху таблицы содержания, отображается поверх всех Тем, расположенных ниже. Темы, которые отображаются на заднем плане вида, расположены внизу таблицы содержания. Просто перетаскивайте мышкой Темы вверх или вниз для изменения порядка их отображения;

– **Темы, которые активны**. Когда вы делаете Темы активными, вы выбираете, с какими Темами вы работаете. Когда Тема активна, она выделена флажком в таблице содержания. Просто щелкните на имени Темы или на ее легенде, чтобы сделать ее активной. Чтобы сделать активной более чем одну тему, держите нажатой клавишу Shift, когда вы щелкаете мышкой на Темах (см. рис. 1,*i*). Наибольшее число операций, которые вы выполняете в Виде, применяются к активным Темам (выбор пунктов меню, нажатие на кнопки, использование инструментов);

– **Редактируема ли Тема**. Штриховая линия вокруг флаговой кнопки означает, что вы в данный момент редактируете объекты темы. Только Темы, основанные на шейп-файлах (shapefiles), могут быть редактируемыми (см. рис. 1,*m*).

Сокрытие таблицы содержания. Чтобы скрыть таблицу содержания, перетащите ее правую границу до крайнего левого положения. Вид будет перерисован с учетом освободившегося пространства. Чтобы показать таблицу содержания снова, перетащите ее границу обратно направо.

Сокрытие легенды темы. Таблица содержания обычно показывает легенды каждой из тем в Виде. Тем не менее, вы можете сократить место, занимаемое таблицей содержания с помощью сокрытия легенд Тем. Это особенно полезно, когда Вид содержит много Тем. Чтобы скрыть легенду Темы, сделайте Тему активной и выберите пункт “Hide/Show Legend” (“Скрыть/Показать Легенду”) из Theme (Тема) главного меню. Когда легенда скрыта, название темы и флаговая кнопка остаются видимыми.

Вырезание, копирование и вставка Тем – это стандартные операции в Windows. Они выполняются только с активными темами. Вы можете копировать и вставлять Темы в тот же Вид или в другой. Чтобы удалить тему из вида, просто вырежьте ее из таблицы содержания. Это осуществляется следующим образом. Нужная тема делается активной и выбирается подменю “Cut themes” меню “Edit” панели главного меню ArcView. Вид будет автоматически перерисован без вырезанной темы. Затем эту тему можно вставить в этот Вид или в другой, выбрав подменю “Paste” меню ”Edit” панели меню ArcView (см. рис. 1,*d*).

2. ВСТРОЕННЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ Avenue

2.1. **Правила именования**

Имена переменных, файлов, скриптов, запросов, ключевые слова, используемые в программах, должны отождествлять контекст или цель этих элементов. Использование, например, прописных букв для выделения ключевого слова внутри имени делает код читаемым. Несколько правил, которым рекомендуется следовать при именовании переменных:

– имя переменной может содержать знак подчеркивания, буквенные символы (*a – z, A – Z*), цифры (0 – 9);

– запрещается использовать специальные знаки в имени переменной (%, *, /, +, $, .);

– имя переменной должно начинаться с буквы или знака нижнего подчеркивания “_” (для глобальных переменных). Имя глобальной переменной распространяется на все программы проекта, тогда как область действия локальной переменной ограничена скриптом (так в ArcView называются программы, написанные на языке Avenue), в котором было выполнено присваивание (aVariable – локальная переменная, _aVariable – глобальная переменная);

– язык Avenue не различает буквы по назначению регистра (например *a* и *A* эквивалентны);

– для удобства пользователя первая буква имени переменной делается строчной, первая буква ключевого слова в имени – прописной (myGloVar);

– имена переменных должны быть достаточно (но не слишком) длинными, чтобы описать значение переменной или дать информацию о ее связях. Избегайте использования сокращений и жаргонных терминов.

Какие бы соглашения об именах вы не использовали, придерживайтесь таковых по всей программе. В работе приняты следующие соглашения об именах элементов.

Переменная: первая буква – строчная, первые буквы всех ключевых слов – прописные(_aGlobalConstVariable – “постоянная глобальная переменная ”).

Запрос: первые буквы всех ключевых слов запроса – прописные (GetActiveDoc – “получить активный документ ”).

Оператор языка: записывается строчными буквами (while, return и др.).

Имя скрипта: первые буквы всех ключевых слов – прописные, ключевые слова разделены символом нижнего подчеркивания. Название скрипта должно нести смысловое значение, по которому будет понятно назначение данного скрипта. Допускается использование русского алфавита. Например, Script1_Copy_All_Themes_View1 = = Скрипт1_Выполняет_Резервное_Копирование_Всех_Тем_Вида_View1 (оба имени эквивалентны).

2.2. **Особенности языка**

Avenue – объектно-ориентированный язык программирования, который позволяет посылать запросы к объектам и классам ArcView и включает набор операторов, дающих возможность эффективно управлять этими элементами. Avenue – не рекурсивный язык, не дает возможности создавать свои собственные классы или изменять уже существующие, но позволяет получить доступ и управлять компонентами, составляющими ArcView. Программы на Avenue исполняются только из среды ArcView, которая имеет встроенный компилятор, хотя не исключен импорт программ, написанных на других языках (например на C++ и др.).

Avenue включает посылку запросов к объектам и классам, условный оператор, конструкции циклов и операторы управления ходом программы.

Объекты

Все, с чем вы работаете в ArcView, является объектами (включая числа, переменные). Когда вы открываете View (Вид), вы работаете с объектом типа “Вид”, когда вы добавляете в него Theme (Тему), вы работаете с

объектом типа “Тема”. Когда вы добавляете Button (Кнопку) на диалог, вы работаете с объектом типа “Кнопка”.

Как организованы классы

Объекты со сходными характеристиками относятся к одному типу классов. Класс объединяет действия для объектов своего типа. Далее классы могут быть организованы в иерархию классов, которая определяет отношения между классами. Отношения между классами можно описать как наследование, агрегирование, ассоциация.

Наследование означает отношение “является разновидностью” и дает возможность классам наследовать возможности своих предков. Например: View (Вид) является классом, определяющим один из типов документов, с которым вы работаете в классе Progect (Проект). View наследует некоторые свойства Project, но также определяет и свои собственные свойства.

Агрегирование означает отношение “состоит из…”. Суперкласс группирует классы и дает возможность наследовать их свойства.

Ассоциация описывает “физическую” связь между классами (в класс – через другой). Например, если вы хотите работать с Theme (Темой) в View (Виде), то первоначально вы должны получить доступ к View (Виду), который содержит несколько тем, среди которых вы теперь можете получить доступ к необходимой.

При программировании объектно-ориентированным способом требуется сначала определить класс объекта, а затем послать этому объекту запрос на проведение необходимых действий. Поддерживается посылка запроса непосредственно к классу. Учитывая это, в дальнейшем для упрощения изложения как объекты, так и классы будем называть просто объектами.

Как запросы приводят объекты в действие

Запросы совершают действия, происходящие в ArcView. При помощи запроса можно создать новый объект, например новую Тему (Theme):

myTheme = Theme. Make.

Запрос Make к классу Theme создает новый объект – новую тему с именем myTheme. Пусть для дальнейшей работы с этой темой ее нужно включить в уже существующий объект-вид с именем aView1 (если такой вид не существует, то его нужно создать, используя все тот же запрос Make, но уже для класса View):

aView1.Add (myTheme)

Запрос Add к объекту aView1 класса View добавляет тему myTheme в вид aView1.

Запрос может извлечь информацию об объекте (например о текущем виде). Запрос может изменить свойства объекта (например изменить имя темы). Запросы могут следовать друг за другом, тогда они исполняются слева направо (использование круглых скобок дает преимущество при вычислениях).

Имя запроса обычно складывается из двух слов: первое слово-действие, выполняемое запросом, и второе слово-приемник этого действия. Определив смысл этих слов, можно понять функцию, исполняемую запросом Avenue. Например, запросы GetProject, FindScript, AddDoc, SetValue, где слова Get (взять), Find (найти), Add (добавить), Set (установить) являются действиями, а Project (проект), Script (программа), Doc (документ), Value (значение) воспринимают эти действия. Приведем краткий список наиболее часто используемых ключевых слов-действий:

Add	включает объект в группу объектов;
As	преобразует объект в объект другого класса;
Can	проверяет, может ли объект выполнить указанную фукнцию;
Find	ищет указанный объект, возвращает ссылку на него;
Get	возвращает ссылку на объект;
Has	покажет, есть ли у объекта указанные условия или состояния;
Is	возвращает true или false;
Make	создает новый экземпляр;
Return	создает новый объект и возвращает ссылку на него;
Set	присваивает атрибут объекту.

Нотации запросов

Avenue использует три стиля нотации запроса к объектам: постфиксный, инфиксный и префиксный.

В постфиксной нотации запрос располагается после объекта-получателя, точка разделяет объект-получатель и запрос. Avenue использует постфиксную нотацию для большинства своих запросов:

<объект> . <Запрос>

Примеры:

1. В список-объект myList добавляется еще один элемент – строка “Red”:

myList. Add (“Red”)

2. Запрос Make к классу Table создает объект типа Table с именем newTable:

newTable=Table.Make

3. Постфиксные запросы могут быть связаны цепочкой в следующей форме:

<объект> . <Запрос> . <Запрос> . … . <Запрос>

Например:

theView. FindTheme(“Дороги”). SetVisible (false)

Запрос FindTheme к объекту theView типа View возвращает объект типа Theme (с именем “Дороги”), посылка запроса SetVisible (false) к этому объекту делает найденную Theme с именем “Дороги” невидимой.

В инфиксной нотации запрос располагается между двумя объектами:

<объект> <Запрос> <объект>

Инфиксная форма наиболее часто используется в арифметических операциях и при создании интервала.

Примеры:

1. 5*6 ‘арифметическая операция умножения
2. 1..100 ‘ интервал от 1 до 100

В префиксной нотации запрос появляется перед объектом-получателем:

<Запрос> <объект>

Префиксная нотация используется в операциях отрицания.

Примеры:

1. –5 ‘ арифметическое отрицание
2. not true ‘ логическое отрицание

2.3. **Зарезервированные слова в Avenue**

av

av – это специальное зарезервированное слово, которое объединяет все компоненты открытого (текущего) проекта – объекта класса Project: при запуске ArcView и открытии какого-либо проекта создается некоторый прикладной объект, который существует до тех пор, пока не завершена работа с этим проектом. При написании скриптов **av** очень часто является стартовой точкой для обращения к проекту и его компонентам.

```
myProject = av.GetProject      ‘возвращает текущий проект;
myDoc = av.GetActivDoc         ‘возвращает активный
                               ‘документ текущего проекта.
```

self

self – специальное зарезервированное слово, используемое в Avenue для ссылки на собственный объект внутри скрипта. Этот объект может быть определен как начальный при запуске скрипта. Когда вы связываете один скрипт с другим (из одного скрипта запускается другой), то объект self используется в вызываемом скрипте для обращения к переданным в него параметрам из вызывающего скрипта.

Пример:

```
av.Run (“Script1”, {“Hello”, 1996, av.GetActivDoc})
```

Запрос Run используется для вызова в одном скрипте другого скрипта. В данном примере вызывается скрипт с именем Script1, в который в качестве передаваемого значения сформирован список из трех элементов: первый элемент – строка “Hello”, второй – данное типа “Год”, третий – некоторый текущий объект-документ. При этом Script1 может выглядеть следующим образом:

```
MsgBox.Info (self.Get(0), “ ”)
MsgBox.Info (self.Get(1).AsString, “ ”)
MsgBox.Info (self.Get(2).GetName, “ ”)
```

Script1 при такой реализации может запускаться для любого списка из трех элементов, но типы этих элементов должны соответствовать указанным в предложенном вызове.

2.4. **Комментарии**

Комментарии обеспечивают возможность включить в скрипт любой необходимый вам текст. При компиляции скрипта ArcView игнорирует комментарии (а также пустые строки, которые вы можете использовать для придания скрипту более читаемого вида).

Комментарием считается любой текст справа от апострофа. Комментарии могут располагаться непосредственно в строке или занимать отдельную строку программы.

Пример двух различных способов написания комментариев:

```
‘Получить окно проекта:
projWin = av.GetProject.GetWin
projWin.MoveTo (–500, –500) – ‘перенос окна проекта с указанием
                              ‘координат для его верхнего левого угла.
```

2.5 **Локальные и глобальные переменные**

Переменная создается в момент появления ее слева в операторе присваивания. Именованный объект справа от знака равенства – это тот объект, на который ссылается переменная.

Локальная переменная может использоваться только в том скрипте, в котором она определена.

Признаком глобальной переменной является знак нижнего подчеркивания перед именем переменной. Один раз созданная глобальная переменная существует в течение всего времени работы с проектом до момента ее удаления при редактировании скрипта или выполнения к ней запроса ClearGlobals, аналогичного такому удалению. Для удаления всех глобальных переменных проекта используется запрос ClearAllGlobals. При запуске другого проекта ArcView автоматически удаляет все глобальные переменные предыдущего работающего проекта. Если вы хотите освободить глобальную переменную от ссылки на объект, но не хотите ее удалять, то сделайте ссылку глобальной переменной на пустой объект nil: aVariable=nil.

2.6. **Логические выражения**

Логические выражения – это некоторая последовательность запросов, которая дает в итоге логический результат: объект true – истина или объект false – ложь. Логические выражения используются в условиях операторов if ... then ... else и while в качестве <выражения>.

Простейшие логические выражения можно строить, используя операторы сравнения: =, <>, <=, >= для каких-либо объектов. Класс Boolean (логический) позволяет создавать составные логические выражения посредством использования логических операций: and, or, not, xor с исходными логическими выражениями (A и B), формируя результат (Result) по конкретным правилам для каждой операции:

а) or (или) обеспечивает способ вычисления дизъюнкции (логическое сложение) двух выражений. Результат – “истина”, если хотя бы одно из выражений “истина”.

A	B	Result
T	T	T
T	F	T
F	T	T
F	F	F

б) and (и) обеспечивает способ вычисления конъюнкции (логическое умножение) двух выражений. Результат – “истина”, если оба выражения “истина”.

A	B	Result
T	T	T
T	F	F
F	T	F
F	F	F

в) xor обеспечивает вычисление частного случая логической дизъюнкции: “исключающего или”. Результатом – “истина”, когда только одно из выражения “истина”.

A	B	Result
T	T	F
T	F	T
F	T	T
F	F	F

г) not (не) обеспечивает логическое отрицание логического выражения

A	Result
T	F
F	T

2.7. **Работа с основными элементами**

Работа со строками

Строки ограничены символами двойные кавычки (“ ”). Используя оператор присваивания, можно создать новый строковый объект:

myString = “Hello Word”

Можно объединить две строки, используя операторы + и ++ (++ оставляет пробел между соединяемыми строками).

Строки можно сравнивать, используя операторы: =, <>, < =, >= .

Например:
```
if (myString = “No”) then return nil
  elseif (myString = “Yes”) then
        MsgBox.Info (“Ваш выбор” ++ myString, “ ”)
  end
```

Замечания: 1. Оператор = (проверка на равенство) используется также в качестве оператора присваивания (различие – по контексту вызова).

2. Запрос Info к классу MsgBox выводит на экран информационное окно сообщений. Info требует двух аргументов-строк:

(<передаваемая в окно строка>, <заголовок окна>)

Оба аргумента должны быть строковыми или обращены в строку при необходимости (возможна и пустая строка для передачи).

Другие операции со строками:

– извлечение из строки подстроки:

```
myString= “Красный, желтый, зеленый”   ‘разделитель подстрок в
MsgBox.Info (myString.Extract(2), “ ”)   ‘строке – пробел, поэтому
                                        ‘в окне появится: “желтый,”
```

– замена одной подстроки другой подстрокой (в том числе замена всей строки):

```
newString = myString.Substitute (“желтый”, “синий”)
MsgBox.Info (newString, “ ”)
```

– обращение строкового объекта в другой объект использованием постфиксных запросов:

```
AsFileName – в имя файла
AsList     – в список
AsNumber   – в число
AsDate     – в дату
AsTime     – в объект времени
```

Примеры:

```
1. aFile= “Test.cpp”.AsFileName    ‘Запрос AsFileName к строковому
                ‘объекту “Test.cpp” обращает его в объект типа “имя файла”.
2. aNum = “100” ‘создается строка aNum из цифровых символов;
newNum=aNum.AsNumber ‘строковый объект обращается в числовой.
3. myString = “Красный желтый зеленый”
   myList = myString.AsList
```

Строка myString, состоящая из трех слов (разделитель – пробел), после выполнения запроса AsList обратиться в список из трех элементов-строк (разделитель элементов списка – запятая):

myList = {“Красный”, “желтый”, “зеленый”}

Приоритеты в Avenue

В Avenue все запросы прочитываются последовательно слева направо. Так как арифметические операторы – это простейшие запросы (инфиксные или префиксные), то они также прочитываются и исполняются слева направо; традиционные приоритеты арифметических операций не выполняются. Поэтому необходимо использовать скобки для установления нужного порядка действия и получения правильного результата. Будьте особенно внимательны, когда вы смешиваете записи различных запросов.

Примеры:

1. If (not(myString = “Hello”)) then … ‘скобки повышают приоритет ‘операции сравнения над впереди идущей операцией отрицания.
2. for aCount = 1..(myList.Count – 1) … ‘скобки используются для ‘верного вычисления величины интервала (его верхней границы).

Работа с числами

Числовое значение в Avenue поддерживается классом Number. Не поддерживается традиционный приоритет арифметических операций, они выполняются слева направо в порядке следования.

Например: *x* = 5 + 3 * 3 ‘ *x* получит значение 24, а не 14.

Для получения правильного результата используйте скобки:

x = 5 + (3 * 3).

Числовой объект может быть обращен в другой объект при использовании постфиксных запросов:

AsString – в строку
AsDays – в объект “день”
AsYears – в объект “год”
AsHours – в объект “час”
AsMinutes – в объект “минута”
AsDegrees – в объект “градус”
AsRadians – в объект “радиан”

Работа со списками и словарями

Списки и словари обеспечивают механизм сбора в одно целое различных объектов и ссылочных компонент (ключей в словарях).

Список – это упорядоченное собрание разнотипных объектов, записанных в фигурных скобках ({} – признак списка). Элементы списка – некоторые объекты, разделенные запятыми. Каждый элемент списка имеет

индексный номер, который определяет его положение в списке. Индексирование в списке начинается с нуля, т.е. первый элемент списка имеет индекс 0, а последний равен <Число элементов списка –1>. На каждый элемент можно сослаться, используя его индекс. Списки нужны для эффективного последовательного доступа или для случайного доступа к элементам.

Некоторые операции со списками:

– Создание списка возможно двумя способами.

1. По связывающей записи:

myList ={“Red”, 201, myView, anotherList}

Создается список, состоящий из следующих элементов: первый элемент – строка “Red”, второй – число 201, третий – объект типа View, четвертый – какой-то другой список.

2. По запросу Make к классу List:

```
myList = List.Make            ′ создается пустой список;
myList.Add (“Red”)            ′ добавляется первый элемент;
myList.Add (201)              ′ добавляется второй элемент;
myList.Add (myView)           ′ добавляется третий элемент;
myList.Add (anotherList)      ′ добавляется четвертый элемент.
```

– Добавление элемента в начало существующего списка происходит при помощи запроса Insert:

today = Date.Now ′объекту today типа Date присваивается текущая дата;

myList.Insert(today) ′ получим список {today, “Red”, 201, ′ myView, anotherList}, в котором элемент today имеет индекс 0.

– Замена элемента списка другим элементом происходит посредством запроса Set :

myList.Set (1, “Blue”) ′ в списке myList элемент с индексом 1 ′ будет заменен на строковый “Blue”, получим список {today, “Blue”, 201, ′ myView, anotherList}.

– Для получения элемента из списка используется запрос Get :

myElement = myList.Get(1) ′ переменная myElement получит ′ значение элемента списка с индексом 1: myElement = “Blue”.

– Для получения последовательного доступа к каждому элементу списка используется цикл for each. Пусть требуется для каждого элемента-числа из списка вычислить новое число и вывести их в окно-строку. Новое число на 1 больше очередного элемента из списка.

```
myList = {1, 2, 3, 5, 8}
for each t in myList
        t1 = t + 1
        MsgBox.Info (t1.AsString," ")
end
```

Словарь – это неупорядоченное (непоследовательное) собрание связанных между собой кодов и значений. Каждое объединение – <код + значение> – элемент словаря. И код, и значение – произвольные объекты. Код уникально определяет значение, поэтому он не может быть использован в объединении более, чем с одним значением. Словарь используется для случайного доступа к нескольким объектам: нахождение значения по коду и наоборот. При создании словаря определяется его начальный размер, который обеспечивает грубую оценку числа элементов, вводимых в словарь. Можно вводить больше элементов, чем определено размером, но когда число элементов превышает существующий размер, то требуется больше времени, чтобы найти нужный элемент, так как он расположен где-то вне отведенного места.

Для получения доступа к элементам словаря используется цикл for each. Заметим, что порядок, в котором будет идти просмотр элементов, неизвестен, но каждый элемент будет просмотрен один раз.

Некоторые операции со словарями.

– Создание словаря происходит по запросу Make :

```
aDict = Dictionary.Make (<размер>)
```

– Добавление элемента в словарь – по запросу Add :

```
aDict.Add (<код>, <значение>)
```

– Нахождение значения по коду (ключу) – запрос Get :

```
myVariable = aDict.Get (<код>).
```

Приведем пример, иллюстрирующий основные операции со словарями:

```
aDict = Dictionary. Make (11) ′создается словарь на 11 элементов;
aDict.Add (“List1”, {“a”, “b”, “d”}) ′в созданный словарь добавляется
′ 1-й элемент, код которого – строковый “List 1”, а значение – список из
′ трех строковых элементов;
aDict.Add (“List2”, {“What”, “Is”, “This”}) ′ добавляется 2-й элемент;
aDict.Add (“List3”, {1, 2, 3}) ′ добавляется 3-й элемент со значением –
списком чисел;
aList = aDict.Get (“List 1”)      ′из словаря извлекается элемент: по коду
′ “List 1” будет извлечен список {“a”, “b”, “d”}.
```

2.8 **Операторы языка**

Оператор присваивания. Основание оператора присваивания – это переменная, ориентированная на связь с объектом. Переменная создается, когда она появляется с левой стороны оператора присваивания.

Синтаксис: Переменная = Объект

Имя переменной должно быть единственным в своем роде, уникальным. Если имя переменной конфликтует с именем класса или зарезервированным словом, то Avenue при компиляции выдаст сообщение об ошибке.

Примеры.

1. Создать переменную aVariable1, которая будет эквивалентом активного документа ArcView типа View с именем “Документ1” (активных документов может быть несколько):

aVariable1 = av.GetActiveDoc.FindDoc (“Документ1”).

Эту сложную операцию присваивания можно разбить на две простые:

aVariable1 = av.GetActiveDoc

aVariable2 = aVariable1.FindDoc (“Документ1”).

2. Создать локальную переменную aView, которая в текущем (“работающем”) проекте будет эквивалентом найденного документа с именем “View1” (не обязательно активного). Если переменная не пуста (если документ найден), то установить другое имя документа – “States”, то есть переименовать документ:

```
aView = av.GetProject.FindDoc (“View1”)
if (aView<>nil) then
    aView.SetName (“States”).
end
```

Замечание: Оператор присваивания (=) также участвует в логических выражениях операторов условия и цикла (проверка на равенство); различие – по контексту вызова.

Оператор if … then … else – разветвленный оператор условия.

Синтаксис:

```
1.  if (<Выражение1>) then
        <Блок запросов1>
    end
2.  if (<Выражение1>) then
        <Блок запросов1>
    else
```

```
        <Блок запросов2>
    end
3.  if (<Выражение1>) then
        <Блок запросов1>
    elseif (<Выражение2>) then
        <Блок запросов2>
    [else
        <Блок запросов3>]
    end
```

Во всех трех формах синтаксиса выполняется <Блок запросов1>, если <Выражение1> – истина.

Если в случае 1 <Выражение1> – ложь, то ArcView игнорирует <Блок запросов1>.

Если в случае 2 <Выражение1> – ложь, то ArcView выполняет <Блок запросов2>.

Если в случае 3 <Выражение1> – ложь, то ArcView проверяет на истинность <Выражение2>. Если оно – истина, то выполняется <Блок запросов2>, если оно – ложь, то выполняется <Блок запросов3>.

Вы можете многократно использовать оператор elseif в случае 3.

Примеры.

1. Простейшая проверка одиночного условия. Если переменная aResult не пуста, то выполнить действия – блок запросов.

```
…
if (aResult <> O) then
   aResult1 = aResult-10
   MsgBox.Info (aResult1AsString, " ")
end
…
```

2. Двойная проверка условия. Если число Тем в Виде aView равно 0, то печатать сообщение *"Темы в виде отсутствуют"*, иначе, если число тем в виде не равно нулю, а число активных тем равно 0, то печатать сообщение *"Нет активных тем"*.

```
If (aView.GetThemes.Count = 0) then
    MsgBox.Info ("Темы в виде отсутствуют", " ")
Elseif (aView.GetActiveThemes.Count = 0) then
    MsgBox.Info ("Нет активных тем", " ")
end
```

3. Если активный документ типа View, то размер окна максимизировать, иначе, если активный документ типа Project, то вывести сообщение “Пожалуйста, откройте View”, иначе (если активный документ не View и не Project) закрыть все.

```
if (av.GetActiveDoc.Is (View)) then
    av.GetActiveDoc.GetWin.Maximize
elseif (av.GetActiveDoc.Is (Project)) then
        MsgBox.Info (“Пожалуйста, откройте View”, “ ”)
    else
        av.GetActiveDoc.GetWin.Close
end
```

Оператор for each применяется для посыла запроса на каждый элемент из совокупности элементов, представленной, например интервалом.

Синтаксис: for each <переменная> in <совокупность>

```
            [by <шаг>]      ‘запрос by работает только с интервалами.
                <блок запросов>
            end
```

<переменная> – это повторяющаяся переменная или переменная цикла. Каждый раз при переходе через цикл переменная получает следующее значение из <совокупности> с учетом <шага>.

Примеры.

1. Простейший цикл for each: для каждого элемента *i* из интервала от 1 до 10 выводится окно сообщений, в котором печатается значение этого элемента (при умолчании шага он равен единице).

```
for each i in 1..10
    MsgBox (“Элемент – “ + + i.AsString, ” ”)
end
```

В примере переменная *i* – переменная цикла, она принимает значения 1, 2, 3, …, 10, изменяясь на 1 при новом прохождении цикла. Цикл проработает 10 раз. При каждом повторении цикла ArcView вычисляет и присваивает ей следующее значение из совокупности – интервала. Совокупность – это обычно интервал или список. Элементы в совокупности не обязательно должны быть упорядочены. Если элементы упорядочены, то порядок учитывается. Если элементы неупорядочены (для списка), то цикл возьмет каждый элемент по очереди.

Замечание. Переменные, которые используются в цикле, остаются определенными только до конца работы цикла. Поэтому нельзя использовать

какое-либо значение переменной цикла после завершения работы с циклом.

2. Начальное значение дается объекту в цикле и от него отсчитывается следующее с шагом 2; выводится окно сообщений, в котором печатается значение этого объекта:

```
for each n in 1..10 by 2
    MsgBox (n.AsString, ” ”)
end
```

Использование инфиксного запроса ”..” создает интервал значений для объекта, а использование запроса by устанавливает указанный шаг между значениями. Можно менять значения и по убыванию, используя отрицательные числа для шага.

3. В переменную aView заносится эквивалент активного документа текущего проекта ArcView (предполагается, что он один). В цикле для каждой активной темы этого документа редактируется таблица этой темы:

```
aView = av.GetActiveDoc
for each t in aView.GetActiveThemes
    t.EditTable
end
```

Оператор while повторяет исполнение блока запросов так долго, пока <Выражение> – истина. При отладке убедитесь, что цикл while дает конечную проверку условия, иначе цикл будет бесконечным, т.е. нужно проверить правильность выполнения условия выхода из цикла.

Синтаксис: while (<Выражение>)
<Блок запросов>
end

<Блок запросов> может быть выполнен один раз, несколько раз или может не выполниться ни разу.

Примеры.

1. Происходит проверка выполнения запроса *”Запустить скрипт еще раз?”*. Если пользователь ответит положительно, то скрипт с именем “TestScript”, созданный в том же проекте, запустится на выполнение, после чего снова будет сделан запрос *”Запустить скрипт еще раз?”*, и так до тех пор, пока пользователь не даст отрицательный ответ; в этом случае произойдет автоматический выход из цикла while.

```
while (MsgBox.YesNo (“Запустить скрипт еще раз”, “ ”, true))
av.Run (“TestScript”, nil)
end
```

Запрос YesNo к классу MsgBox имеет 3 аргумента и выводит на экран окно сообщений, в котором будет строка сообщений – первый аргумент запроса и две кнопки Yes и No. Второй аргумент запроса YesNo – заголовок окна сообщений. Третий аргумент определяет, по какому исходу (по какому ответу пользователя) запрос вернет истину и последует выполнение <блока запросов> в цикле while (аргумент true – по ответу Yes, аргумент false – по ответу No).

2. При нажатии кнопки No выполнить досрочный выход из бесконечного цикла (для досрочного выхода можно использовать оператор break).

```
while (true)
    aVariable = MsgBox.YesNo (“ ”, ” ”, true)
    if (not aVariable) then
        break
    end
end
```

Оператор break досрочно завершает выполнение циклов for each, while и применим только внутри этих циклов.

Пример.

Когда счетчик aNum в цикле for each достигнет значения 5, выполнить оператор break: происходит выход из цикла и следующий за break оператор (вывод сообщения) не выполнится более ни разу.

```
for each aNum in 1…10
    if (aNum = 5) then
        break
    end
    MsgBox.Info (“Число =” + + aNum.AsString, “ ”)
end
MsgBox.Info (“ ”, “ ”)
```

Так как цикл for each выполняется до тех пор, пока переменная цикла не получит значение 5, то сообщение “Число = … ” выведется только для первых четырех значений. Как только переменная aNum станет равна 5, произойдет досрочный переход за пределы второго end (т.е. за пределы ближайшего цикла, а не за пределы блока if), т.е. цикл for each не выполнится указанное число раз.

Оператор continue может использоваться, когда возникает необходимость осуществить переход к следующей итерации цикла, пропустив оставшиеся операторы в теле цикла.

Синтаксис: continue

Пример.

Скрипт проверяет каждое поле в таблице каждой темы активного вида и, если это поле нечисловое, делает его невидимым. Оператор continue использовать для перехода к следующему полю в списке полей таблицы в случае, если оно числовое.

```
aView = av.GetActiveDoc
for each t in aView.GetThemes          ‘ в качестве <совокупности> здесь
                                       ‘ выступает список тем вида aView;
    for each f in t.GetFTab.GetFields  ‘ в качестве <совокупности> здесь
                                       ‘ выступает список полей таблицы темы;
        if (f.IsTypeNumber) then       ‘ если поле числовое – перейти к
            continue                   ‘ следующему полю f таблицы
        end                            ‘ (к следующей итерации внутреннего цикла)
    f.SetVisible (false)               ‘ иначе (поле нечисловое)
                                       ‘ сделать его невидимым;
    end                                ‘ конец внутреннего цикла по полям;
t.GetFTab.Refresh      ‘ выполненные обновления таблицы сохранить;
end                                    ‘ конец внешнего цикла по темам.
```

Оператор return используется для завершения работы данного скрипта с возвратом управления и аргументов в точку его вызова.

Синтаксис: return <объект>

Примеры.

1. В текущем проекте найти документ с именем “View1”; если документ не найден, то вывести сообщение об ошибке: “Невозможно найти вид с именем View1”, после чего завершить выполнение скрипта. Оператор return использовать именно для завершения работы скрипта.

```
aView = av.GetProject.FindDoc (“View1”)
if (aView = nil) then
    MsgBox.Error (“Невозможно найти вид с именем View1”, “ ”)
    return nil
end
```

2. В проекте есть два скрипта: Script1 и Script2. Script2 вызывается из “Script1” с одновременной передачей в него собственного аргумента –

строки “Green”. После его работы управление возвращается в Script1 с передачей в точку вызова строки-аргумента (результата работы Script2), которую Script1 выведет в окно сообщений.

```
‘ Текст скрипта Script1:
    aResult = av.Run (“Script2”, “Green”)
    MsgBox.Info (aResult, “ ”)

‘ Текст скрипта Script2 :
    if (self = “Red”) then
        return “Stop”
    elseif (self = “Green”) then
        return “Go”
    else
        return “Use attention”
```

Результатом совместной работы этих скриптов будет окно сообщений, в котором выведется “Go”. Если в качестве передаваемого аргумента в “Script2” отправить строку “Red”, то в окне сообщений выведется “Stop”. В любом другом случае выведется “Use attention”(“Будьте внимательны”).

Замечания 1. Старайтесь использовать оператор return вместо exit для досрочного завершения текущей программы.

2. Оператор return обязательно должен содержать аргумент. Для завершения работы скрипта без передачи конкретного результата используйте в качестве аргумента nil (см. пример 1).

Оператор exit прерывает выполнение текущего скрипта, завершая работу всей программы.

Синтаксис: exit

Используйте оператор exit, только когда вы хотите завершить выполнение всех проходов всех скриптов, в других случаях используйте оператор return.

Пример.

Работу программы завершить, если имя указателя на строку не объявлено или строка ввода имени оказалась пустой; в противном случае введенное имя считать именем каталога. Если в списке указателей на каталоги такого имени нет, то также завершить работу программы.

Используется запрос Input к классу MsgBox, где в качестве первого аргумента выступает строка сообщений, второй аргумент – это заголовок

окна сообщений, третий аргумент – строка для ввода имени, по которому будет вестись поиск:

```
newDir = MsgBox.Input (“Spesify directory: ”, “ ”, “ ”)
    if (newDir = nil) then
        exit
    elseif (newDir.AsFileName.IsDir.Not) then
Msg.Box.Error(newDir.AsFileName.GetFullName++”is not directory”,” ”)
        exit
    else
    . . .
    end
```

3. РАБОТА СО СКРИПТАМИ

3.1. **Что такое скрипт**

Скрипт – это компонент какого-либо проекта ArcView, который содержит код Avenue. Точно также, как макросы, процедуры и сценарии в других языках программирования, скрипты ArcView группируют вместе средства, чтобы выполнить три основные цели:

- автоматизировать решение задачи;
- добавить новые возможности к ArcView;
- построить новые запросы.

ArcView принимается как собрание системных скриптов, т.е. каждая операция, выполняемая в ArcView, связана с работой того или иного системного скрипта. Вы можете использовать системные скрипты в качестве основы для своих скриптов и, редактируя системный скрипт, достигнуть необходимого результата.

3.2. **Создание скрипта**

При создании скрипта создается компонент проекта типа Script, который может быть привязан к некоторому инструменту (например, пункту меню, кнопке диалогового окна и т.д.) и/или нацелен на конкретный исход (например на поиск нужного элемента в списке). Созданный скрипт появляется в списке скриптов проектного окна (при выборе строки Script про-

екта) и одновременно появляется в списке Script Manager (Редактор скриптов). Есть два пути создания скриптов.

1. Создание скрипта в проектном окне:

а) сделайте окно проекта активным либо запустите проект;

б) на левой панели выберите Script в списке компонент проекта (появится окно с новым меню);

в) щелкните на кнопке New меню: создастся и откроется новый скрипт (имя скрипта задается автоматически: Script<номер>);

г) для задания желаемого имени скрипта выберите тот скрипт, имя которого хотите изменить, нажмите комбинацию клавиш Ctrl+R. В появившемся диалоговом окне наберите желаемое имя и нажмите OK (так переименовываются и другие компоненты объекта).

2. Создание скрипта в Customize dialog box:

а) сделайте окно проекта активным;

б) в пункте меню Project выберите Customize;

в) создайте новое управление (кнопку, панель инструментов) или выберете уже существующее управление;

г) двойной щелчок на исходе Apply (эквивалентно нажатию клавиши <Ввод>), Click (эквивалентно щелчку мыши) или Update (эквивалентно изменению состояния управления) в списке исходов и ArcView выведет Script Manager (список системных скриптов);

д) для создания нового скрипта щелкните на кнопке <New>. ArcView выведет диалоговое окно, в котором будет запрашиваться имя нового скрипта, затем откроется окно нового скрипта. Если вы хотите отредактировать какой-либо системный скрипт, выберите его в списке скриптов Script Manager. Двойной щелчок на имени выбранного скрипта или нажатие кнопки OK откроет скрипт для редактирования.

3.3. **Загрузка текстового файла в скрипт**

1. Сделайте окно Script активным.

2. Щелкните на кнопке <Load Text File>, расположенной на верхней панели инструментов.

3. В представленном окне появится список файлов, выберите нужный.

4. Двойной щелчок на имени выбранного файла или один щелчок + кнопка OK диалогового окна и ArcView вставит содержание файла с точки ввода (с последнего положения курсора).

3.4. **Загрузка системного скрипта**

Вы можете использовать все или часть системных скриптов в скрипте, который пишите самостоятельно (или возможно использование системного скрипта в качестве основы, фундамента вашего скрипта).

Для загрузки системного скрипта:

1. Активизируйте пункт Script в окне проекта.

2. Если вы хотите загрузить системный скрипт в новый скрипт, создайте его (кнопка <New>). Если хотите загрузить в какой-либо ваш уже существующий скрипт, то задайте точку ввода, которая будет определяться положением курсора.

3. Щелкните на кнопке Load Script на панели инструментов.

4. В окне Script Manager появится список скриптов; выберите название того системного скрипта, который вы хотите загрузить.

5. Двойной щелчок на имени выбранного скрипта или один щелчок на имени + кнопка OK и ArcView вставит текст системного скрипта в точке ввода.

3.5. **Редактирование скрипта**

Для редактирования скрипта сделайте его окно активным.

Выбор нужного слова в скрипте: двойной щелчок на слове.

Выбор строки: переместите курсор влево до крайнего левого символа в строке, которую вы хотите выбрать, далее один щелчок на этой строке.

Выбор нескольких строк: переместите курсор влево до крайнего левого символа в первой строке из тех, что хотите выбрать. Далее щелкните один раз и, не отпуская мышки, выделите необходимое количество строк.

Выбор всех строк срипта: в меню окна скрипта выберите пункт Edit-Select All.

Удаление текста от позиции курсора до крайнего левого символа: можно использовать комбинацию клавиш <Ctrl+U> или в меню окна скрипта пункт Edit-Delete Left.

Отмена предыдущего удаления: используется комбинация клавиш <Ctrl+Z> или пункт меню Edit-Undo.

3.6. **Отладка скрипта**

Перед тем как запустить скрипт ArcView на выполнение, необходимо его откомпилировать (отладить). В процессе компиляции ArcView прове-

ряет текст скрипта на наличие синтаксических ошибок: правильность написания ключевых слов, операторов и т.д.

Для компиляции скрипта: щелкните по кнопке Compile на верхней панели инструментов окна скрипта. Если при компиляции будут обнаружены ошибки, система выдаст сообщение об этом. Проверьте при этом правильность написания всех переменных и запросов, согласованность всех циклов. После завершения процесса компиляции скрипт можно запустить на выполнение или прогнать его пошаговое выполнение.

3.7. **Запуск скрипта**

Как только при компиляции не получено сообщение об ошибках, скрипт можно запускать на выполнение: нажать кнопку Run на той же панели инструментов окна скрипта. В течение выполнения программы могут обнаружиться ошибки периода исполнения (система выдаст соответствующее сообщение). Чтобы избежать этого или определить шаг появления ошибки, используйте пошаговое выполнение скрипта (кнопка Step на каждый шаг), тогда существующие ошибки выйдут на конкретном шаге.

3.8. **Размещение и удаление точек разрыва**

Точка разрыва – это команда, которая прерывает нормальное выполнение программы. Вы можете разместить точки разрыва в нужных местах скрипта, там, например, где по вашему предположению может возникнуть ошибка. Как только точка разрыва становится ненужной, вы можете ее удалить.

Размещение точки разрыва.

1. Откомпилируйте скрипт.

2. Выберите местоположение точки разрыва – переместите курсор в то место, где вы хотите выполнить прерывание.

3. В пункте меню Script окна скрипта выберите пункт Toggle Breakpoint.

ArcView добавляет точку разрыва и подсвечивает строку. Выполнение скрипта прерывается, как только система встречает точку разрыва. При этом можно просмотреть промежуточные результаты, после чего запустить скрипт на довыполнение или осуществить дальнейшее пошаговое выполнение скрипта от точки разрыва (кнопка Step на верхней панели инструментов, одно нажатие которой соответствует одному шагу выполнения программы).

Удаление точки разрыва.

1. Установите курсор в месте расположения точки разрыва.

2. В пункте меню Script выберите Toggle Breakpoint или нажмите F9.

3. Для удаления всех точек разрыва, размещенных в скрипте, в пункте Script меню окна скрипта выберите Clear All Breakpoint.

Замечание: при повторной компиляции (перекомпиляции) все точки разрыва удаляются.

3.9. **Запись скрипта в текстовый файл**

Вы можете записать весь скрипт или его часть в текстовый файл, который может быть использован в другом проекте или может служить резервной копией вне проекта.

Запись всего или части скрипта в текстовый файл.

1. Если вы хотите записать часть скрипта, сначала выделите тот текст, который вы хотите записать, затем нажмите кнопку Write Script на верхней панели инструментов (для записи всего скрипта достаточно просто нажать эту кнопку).

2. Выберите путь, по которому будет размещаться текстовый файл, в появившемся диалоговом окне.

3. В строке File Name дайте имя текстовому файлу.

4. Нажмите OK.

3.10. **Сохранение работы**

При работе в ArcView вы можете сохранять как компоненты проекта (таблицы, виды), так и весь проект полностью. При возобновлении работы с компонентом проекта он будет в том состоянии, в котором был при последнем сохранении. Сохранение можно делать в любой момент работы с системой. При сохранении всего проекта сохраняется вся работа, проделанная со всеми компонентами. Если вы не сохраните проект при выходе из ArcView, то система сделает запрос о том, сохранять или нет изменения в проекте.

Для сохранения вашей работы в пункте меню File окна проекта выберите Save Project или воспользуйтесь комбинацией <Ctrl+S>.

Для сохранения проекта с другим именем:

1. Сделайте активным проектное окно.

2. В пункте File меню окна проекта выберите Save Project As.

3. В появившемся диалоговом окне укажите при необходимости путь, новое имя проекта.

3.11. **Примеры простейших сеансов начальной работы**

1. Пусть вы начинаете свой первый сеанс работы с ArcView. Загружаете систему ArcView, для чего достаточно ее ярлыка на рабочем столе компьютера либо знать путь, по которому располагается файл для запуска системы (arcview.exe).

В ArcView вы работаете с проектом, поэтому с открытием главного окна системы на его панели сразу же открывается еще одно окно с запросами: “Открыть новый проект” или “Открыть существующий проект”. Выбираем пункт: “Открыть новый проект”. Открывается окно проекта – Project, а проект автоматически получает имя “proj1.apr”.

Из первого раздела настоящих рекомендаций вам известно, что окно Project содержит верхнюю панель кнопок и окно слева со списком возможных компонент проекта. Для написания скрипта выбираем компоненту “Script”: откроется окно скриптов, которое, в свою очередь, содержит собственное верхнее меню и панель кнопок.

Для создания нового скрипта выбираем кнопку New: откроется окно редактирования, в котором можно писать текст программы, а скрипт автоматически получает имя Script1. Пусть для Script1 набираем следующий текст:

‘ Это строка-комментарий для Script1

MsgBox.Info(“Вы начали работу с проектом”, ” ”)

Следующий шаг работы – компиляция (кнопка Compile) и, если система не обнаружит ошибок, запуск на выполнение (Run): на экране увидите окно с сообщением “Вы начали работу с проектом” и кнопку ОК. Нажав на кнопку, вы завершите работу Script1.

Завершить сеанс работы можно двумя способами.

I. Вы можете закрыть окно проекта: на любом окне есть соответствующий значок. Последует запрос: “Вы хотите сохранить изменения в proj.apr?” При утвердительном ответе выполняется сохранение и проект закрывается (останется только окно ArcView). При отрицательном ответе проект закроется без сохранения: будут потеряны все его изменения, внесенные в текущем сеансе (т.е. созданный нами Script1).

II. Можно использовать в меню ArcView пункт File, затем в выпадающем подменю – пункт Save (этой последовательности операций аналогична комбинация клавиш Ctrl+S). Если при этом хотите изменить имя проекта, в подменю надо выбрать Save As…, система запросит новое имя и новый путь для проекта.

Для повторной работы с proj1.apr надо использовать следующую последовательность действий: запустить ArcView; в окне запроса выбрать пункт “Открыть существующий проект”; по запросу системы указать путь и имя нужного проекта. Но при наличии ярлыка ArcView перед именем файла-проекта вы можете найти проект непосредственно (по его пути) и двойным щелчком на его имени загрузить сразу и ArcView, и сам проект. При любом способе открытия проекта его окно будет таким, каким было при последнем сохранении.

Для запуска Script1 есть два пути:

I. В списке компонент проекта выбрать пункт Script; в окне Script в списке скриптов выбрать имя Script1 (в этом списке пока будет только одно имя); открыть текст скрипта (двойной щелчок мышки или один щелчок и пункт меню Open); запустить скрипт на исполнение (Run);

II. В списке компонент проекта выбрать пункт Script; в окне Script в списке скриптов выбрать имя Script1; не открывая текста скрипта, в окне Script нажать кнопку Run.

Скрипт выполнится, а затем в первом случае управление вернется в окно с текстом скрипта, во втором – в окно Script.

2. Совершенно аналогично создадим и сохраним еще один скрипт Script2, который будет вызывать уже созданный нами Script1. Его текст может выглядеть следующим образом:

```
‘ Это строка-комментарий для Script2
aResult = MsgBox.YesNo(“Начать работу с проектом?”, “ ”, true)
    if (aResult) then
        av.Run(“Script1”, {})
        aDialogBox1 = av.GetProject.FindDialog(“Dialog1”)
        aDialogBox1.Open
    else
        CloseAll
end
```

При исполнении Script2 появляется окно-запрос с вопросом “Начать работу с проектом?” и двумя кнопками Yes и No. При ответе Yes открыва-

ется окно с сообщением “Вы начали работу с проектом” и кнопкой ОК (это проработал вызванный Script1). После нажатия на кнопку откроется диалог с именем Dialog1 (будем считать, что он существует). При ответе No проект закрывается полностью.

Сохранение и повторное использование проекта аналогично рассмотренному ранее.

Для углубления знаний по работе с системой рекомендуем ознакомиться с разделом 4 настоящих рекомендаций.

4. СОЗДАНИЕ БАЗ ДАННЫХ

Повторим, ГИС целесообразно использовать для решения задач с пространственно распределенной (в том числе географической) информацией. Перед началом работы над проектом, который объединит все компоненты для решения задачи, необходимо сформулировать цели проекта, чтобы определить его масштабность и пути реализации. Как правило, проект ГИС может быть организован в виде последовательности логических шагов, каждый из которых основан на предыдущем. Не следует забывать, что на создание и выполнение отдельного проекта влияют многие факторы: и выбранные пути решения задачи, и форма представления конечных результатов, и то, как часто необходимо их выдавать, кому предназначен окончательный продукт, возможны ли другие сферы использования данных и какие дополнительные требования могут предъявляться к ним.

Самая критичная и зачастую самая трудоемкая часть проекта – это **создание базы данных** (БД). Разработка цифровой БД включает в себя следующие шаги:

– Проектирование БД: определение границ исследуемой территории; используемых систем координат; необходимых слоев (покрытий); объектов в каждом слое; атрибутов, необходимых для каждого типа объектов; способов кодирования и организации атрибутов.

– Ввод данных в компьютер: помещение пространственных данных в базу данных (оцифровка или преобразование данных из других форматов); приведение пространственных данных в пригодный вид (проверка и исправление ошибок, а затем создание топологии); помещение атрибутивных данных в БД (ввод атрибутивных данных и связывание атрибутов с пространственными данными); управление БД (преобразование пространст-

венных данных в реальные координаты, стыковка смежных покрытий и манипулирование БД).

ГИС в сочетании со знаниями о целях проекта и разрабатываемой БД предоставляет прикладным задачам множество возможностей для анализа данных, для составления специализированных карт и отчетов. С помощью ГИС весьма эффективно можно решать крайне трудоемкие или вообще не решаемые вручную задачи, можно легко проверять альтернативные сценарии, внося незначительные изменения в методы анализа. На конечном шаге анализа ключевым является умение обобщать и представлять результаты как в графическом (карта, диаграмма и т.д.), так и в табличном виде, что обеспечит реальное влияние анализа на процесс принятия решения.

4.1. **Проектирование БД**

Первый шаг к созданию цифровой БД – это определение ее содержимого, обеспечение наличия в ней всех необходимых объектов, их атрибутов, покрытий к моменту выполнения анализа. Хорошо спроектированная БД может также облегчить использование данных в последующих проектах.

Как было отмечено, на проектирование БД влияют многие факторы, поэтому предлагаемая методика является самой общей и упрощенной. Важную роль в проектировании играют доступные источники данных (как карты в твердых копиях, так и цифровые данные). Поэтому предварительно нужно выяснить, какие источники данных по изучаемой территории доступны. Проектирование БД включает три основные стадии:

– определение содержимого БД (определение необходимых географических объектов и их свойств (атрибутов), организация слоев (покрытий) данных, подлежащих вводу в компьютер);

– определение подлежащих хранению значений и типов каждого атрибута;

– обеспечение привязки к реальным координатам (см. раздел 4.2).

Определение географических объектов и их свойств

Сначала рекомендуется определить необходимые для БД географические объекты и связанные с ними атрибуты. Это непосредственно зависит от поставленной задачи: какой анализ нужно выполнить и какой картогра-

фический продукт или систему на основе картографического продукта создать. В зависимости от критериев анализа и от создаваемых карт могут потребоваться определенные атрибуты для каждого объекта.

1. Пусть для некоторой территории важны следующие критерии:

– найти пригодные для строительства почвы;

– по кодам землепользования выбрать территорию, покрытую кустарником (предполагается, что тип землепользования (назначение земли) кодируется);

– оценить стоимость приобретения участка по площади и коду землепользования.

Исходя из этих критериев, следует выбрать один географический объект: *почвенные* полигоны с атрибутом – степень пригодности, другой географический объект: полигоны *землепользования* с атрибутами – код землепользования и стоимость единицы площади. Перечень объектов и их атрибутов будет выглядеть следующим образом:

Географический объект	**Класс объекта**	**Атрибуты объекта**
Почвы	Полигоны	Пригодность
Землепользование	Полигоны	Код землепользования Стоимость гектара

2. Если на результирующей карте дополнительно должны быть отображены полигоны землепользования, *подписанные* в соответствии с типом землепользования (специальным термином-строкой), дороги с обозначением типа покрытия и основные водотоки, то потребуется расширить список географических объектов и их атрибутов:

Географический объект	**Класс объекта**	**Атрибуты объекта**
Почвы	Полигоны	Пригодность
Землепользование	Полигоны	Код землепользования Стоимость гектара Тип землепользования
Дороги	Линии	Код дороги
Гидрография	Линии	Порядок водотока

Организация слоев данных

Определив необходимые географические объекты, их атрибуты, можно приступать к организации объектов в слои данных. На организацию

слоя в географической БД влияет несколько факторов, различающихся в каждом конкретном случае. Имеется два наиболее общих принципа организации слоев: по классам географических объектов (точка, линия и полигон) и тематическая (семантическая) группировка объектов.

Обычно слои организованы таким образом, что точки, линии и полигоны хранятся в разных слоях. Например, скважины, представленные точками, могут храниться в одном слое, а дороги, представленные линиями, – в другом.

Кроме того, объекты могут быть организованы тематически в соответствии с тем, что они представляют. Например, гидрографическая сеть (водотоки) может составлять один слой, а дороги – другой. Хотя и водотоки, и дороги являются линейными объектами, есть смысл хранить их порознь. Например, атрибутами водотоков могут быть название, класс водотока и скорость течения, а атрибутами дороги могут быть название, тип покрытия и число полос. Из-за различия в атрибутах водотоки и дороги следует хранить в разных слоях, описывающих одну и ту же территорию. Слои для приведенного выше примера могут выглядеть следующим образом (рис. 3):

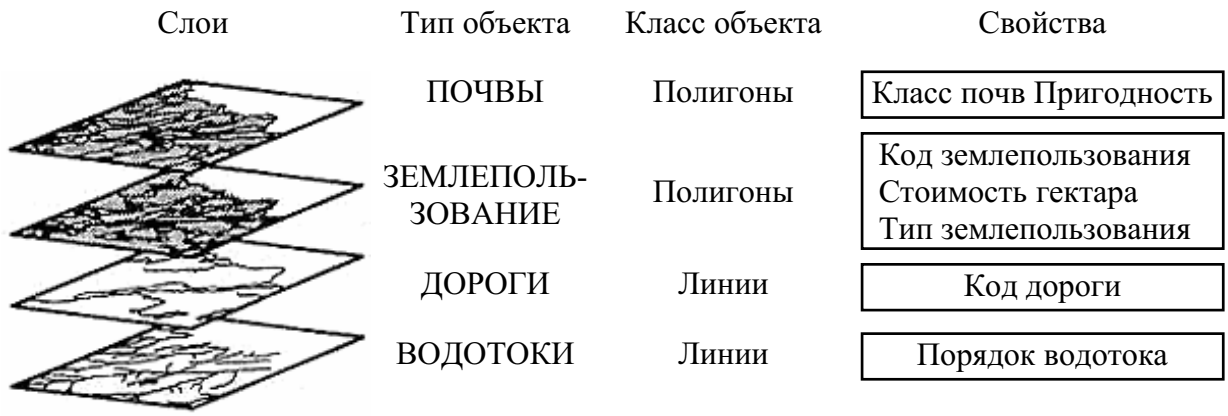


Рис. 3

Состав покрытий проектируемой цифровой БД определяется в процессе выделения географических объектов и их атрибутов и организации этих данных по слоям. В некоторых случаях слои данных могут существовать на отдельных картах или уже иметься на компьютере в цифровом виде (например карта, показывающая только границы землевладений по типам землепользования). В других случаях придется вводить данные слоя с единой базовой карты.

Определение значений всех атрибутов

После определения атрибутов, необходимых для каждого слоя, рекомендуется выяснить, каковы возможные значения каждого атрибута и каковы типы этих значений. Если это сделать сейчас, в дальнейшем будет легче создавать файлы данных для хранения значений атрибутов.

Кодирование. Атрибуты хранятся в памяти в виде чисел или символьных строк. Необходимо решить, какие атрибуты будут храниться в виде чисел, а какие – в виде символов. Например, если вы хотите подписать названия улиц, их надо хранить именно в том виде, в каком они будут выводиться. Точно также следует хранить фактические значения атрибутов, представляемые числами, например ширину или длину улиц.

Некоторые атрибуты, описываемые символьными строками, лучше представлять кодами. Если код описывает классификацию, легче и эффективнее хранить во всех таблицах коды, а не описания. Это облегчает выбор и отображение объектов определенного класса. Пусть имеется *Поисковая (справочная) таблица* с номерами внутренних условных обозначений (SYMBOL) типов землепользования (LABEL), которым сопоставим (отдельным столбцом LUCODE) коды классификации этих типов. Тогда конкретный код (а не строка-тип) может быть использован для поиска в другой таблице (например в *Таблице атрибутов*) номеров всех соответствующих типу участков (и их площадей). Внутреннее условное обозначение каждого объекта-участка с таким кодом, используемое при поиске, будет в точности такое же, как и в справочно-поисковой таблице.

Поисковая (справочная) таблица

LUCODE	SYMBOL	LABEL (тип)
100	16	Городская
200	45	Сельскохозяйственная
300	24	Кустарниковая

Таблица атрибутов

AREA	ID	LUCODE
4322	10	200
3901	11	300
5200	12	300
1698	13	100
2004	14	200

С целью уменьшения размера БД также выгоднее хранить в виде кодов значения атрибутов, если в конкретных таблицах они многократно повторяются (см. *Таблицу атрибутов*).

Удобно кодировать и числовые значения, описывающие интервалы. Например, атрибут полигонного покрытия, представляющий классы укло-

нов 0 – 10 %, 11 – 30 %, 31 – 45 % и более 45 %, можно представить в компьютере кодами 1, 2, 3 и 4 соответственно:

УКЛОН	КОД
0 – 10	1
11 – 30	2
31 – 45	3
45+	4

Проектируя свою БД, помните, что объекты всегда могут быть сгруппированы в более крупные классы. Гораздо сложнее разделить классы на более мелкие, если это не было предусмотрено заранее.

Распределение памяти. Кроме определения способа хранения каждого атрибута, необходимо решить, какой объем памяти ему выделить. Например, сколько символов зарезервировать для хранения названия улицы (определяется по самому длинному названию). Для числовых значений определяется требуемое количество значащих цифр и десятичных знаков. Решающее значение имеют накладные расходы при хранении большого количества данных. Чем меньше памяти выделяется каждому атрибуту, тем меньше получаются файлы данных, тем меньше места потребуется для них на диске и тем быстрее будет выполняться обработка.

Создание словаря данных. Словарь данных – это список названий атрибутов и описание их значений (включая при необходимости каждый код) по всем покрытиям (слоям) для каждого объекта. Словарь данных по вашей БД будет очень полезен как для справок при выполнении проекта или модернизации проекта, так и при переносе информации в другие проекты.

Пример словаря данных:

СЛОЙ	ОБЪЕКТ	КЛАСС	АТРИБУТЫ	ЗНАЧЕНИЕ	ОПИСАНИЕ
Почвы	SOILS	Полигоны	SOILCODE (коды и типы почв)	Сокращения, принятые для всех типов почв	
			SUIT (пригодность)	0	Непригодные
				1	Малопригодные
				2	Умеренно пригодные
				3	Пригодные

СЛОЙ	ОБЪЕКТ	КЛАСС	АТРИБУТЫ	ЗНА-ЧЕНИЕ	ОПИСАНИЕ
Землепользо-вание	LANDUSE	Полиго-ны	LUCODE (коды и типы землепользования)	100 200 300 400 500 600 700	Городские Сельскохозяйственные Кустарники Лесные Водные Переувлажненные Пустоши
			COSTHA (стоимость гектара)	Фактическая цена	
Водотоки	STREAMS	Линии	STRMCODE (код водотока)	1 2	Основной водоток Малый водоток
Канализация	SEWERS	Линии	STRCODE (код водотока)	1 77	Труба 60 см Труба 45 см
Дороги	ROADS	Линии	RDCODE (код дороги)	1 2	Улучшенные Полуулучшенные

4.2. **Ввод данных**

После того как проведено тщательное проектирование БД (определены все слои и атрибуты), можно приступать к ее созданию в компьютере. Ввод данных предполагает: помещение пространственных данных в БД (оцифровкой или преобразованием существующих компьютерных данных из других форматов в формат ArcView); ввод в БД атрибутивных данных и их связывание с пространственными; и самое важное – преобразование пространственных данных в реальные координаты, стыковка смежных покрытий. Нужно собрать слои, которые уже существуют в цифровом виде, в виде шейпов ArcView или в формате, который может быть преобразован в формат ArcView. Если же необходимо выполнить оцифровку некоторой базовой карты, возможности создания электронной карты описаны ниже.

Общему обзору всех перечисленных вопросов посвящаются последующие материалы. Часть их носит чисто информационный характер. Часть при первом прочтении может быть пропущена (если уже имеете готовую электронную карту с пространственной БД или пространственные данные вашей задачи не нуждаются в строгой географической привязке, отображая лишь план расположения объектов). Выбор за вами, однако эти вопросы когда-то предстоит решать самостоятельно.

Методы создания электронных карт

Цифровая карта (ЦК) в общем случае состоит из графической (точки, линии, полигоны) и атрибутивной (связанные с графическими объектами данные) составляющих.

В настоящее время применяются две технологии создания электронных карт – это дигитайзерный ввод и цифрование по растру, которые существуют параллельно и дополняют друг друга. Практика показывает, что сейчас нельзя говорить о преимуществе какой-то одной технологии. Так, с помощью дигитайзера лучше цифровать сложные, насыщенные многоцветные карты, так как оператору лучше понятна ситуация. Полуавтоматическая векторизация дает хорошие результаты при цифровании четких контуров на растре хорошего качества, например расчлененных оригиналов рельефа на пластике.

При дигитайзерном вводе основной объем работ по вводу цифровых карт выполняется оператором в ручном режиме, т.е. для ввода объекта оператор наводит курсор на каждую выбранную точку и нажимает кнопку.

Существует еще полуавтоматический режим ввода, когда фиксируется пара координат *X, Y* через заданный интервал времени или через определенное расстояние. Полуавтоматический режим, возможно, экономит время, но для точного ввода не годится, и далее мы будем рассматривать только ручной режим. Точность ввода при цифровании в огромной степени зависит от квалификации оператора. Если при создании традиционных карт пером очень сложно прорисовывать линии и передавать форму объектов, то что говорить о цифровании, где непрерывную кривую надо аппроксимировать отрезками без потери формы. Большое влияние оказывают и индивидуальные качества оператора.

При векторизации растра субъективные факторы влияют меньше, так как растровая подложка позволяет все время корректировать ввод. Программы векторизации растровых изображений условно можно разделить на три группы: ориентированные на ручную векторизацию, полуавтоматическую и автоматическую. Алгоритмы автоматической векторизации для ввода картографической информации на данный момент не используются для массового ввода картографического материала, поэтому мы их рассматривать не будем.

Подразумевается, что алгоритмы полуавтоматического векторизатора или квалификация оператора при ручном вводе обеспечивают проведение векторной линии в пределах толщины трассируемой растровой линии. Иногда, конечно, встречаются карты, где векторные линии лишь в среднем

совпадают с линиями исходного растрового материала, однако плохие программы и неквалифицированных операторов мы обсуждать не будем. Точность ввода информации у опытного оператора при ручной векторизации выше, так как при полуавтоматической на передачу формы влияет качество растра и при изрезанных краях растровой линии начинают появляться изгибы проводимой векторной линии, которые вызваны не общей формой линии, а локальными нарушениями растра. Оператор же в таких и подобных случаях форму объекта передает более точно, ориентируясь на дополнительные материалы (источник получения растра) и анализируя ситуацию.

Нужно отметить, что при векторизации растра точность ввода значительно выше, чем при цифровании дигитайзером, и в основном зависит от качества исходного растра. Используемое оборудование также оказывает влияние на качество ЦК.

К а р т а

Карта – математически определенное, уменьшенное, генерализованное изображение поверхности Земли, другого небесного тела или космического пространства, показывающее расположенные или спроецированные на них объекты в принятой системе условных знаков. Карта рассматривается как образно-знаковая модель, обладающая высокой информативностью, пространственно-временным подобием относительно оригинала, метричностью, особой обзорностью и наглядностью, что делает ее важнейшим средством познания в науках о Земле и социально-экономических науках. По масштабу различают крупномасштабные карты [1: 100 000 и крупнее], среднемасштабные карты [1: 200 000 – 1: 1 000 000] и мелкомасштабные карты [мельче 1 : 1 000 000]. В соответствии с содержанием различают следующие группы (виды) карт: общегеографические карты, тематические карты, в том числе карты природы, социально-экономические карты, карты взаимодействия природы и общества, а также специальные карты. Все они могут быть аналитическими, комплексными или синтетическими картами. По практической специализации различают несколько типов карт: инвентаризационные карты, показывающие наличие и локализацию объектов; оценочные карты, характеризующие объекты (например, природные ресурсы) по их пригодности для каких-либо видов хозяйственной деятельности; рекомендательные карты, показывающие размещение мероприятий, предлагаемых для охраны, улучшения природных условий и

оптимального использования ресурсов; прогнозные карты, содержащие научное предвидение явлений, не существующих или неизвестных в настоящее время.

Система координат, используемая при создании карты: широта и долгота

Наиболее привычной системой описания положения в пространстве является система широт и долгот. Эта система может использоваться для определения местоположения точек в любом месте на земной поверхности.

Широта и долгота являются угловыми величинами, измеренными от центра земли до точки на земной поверхности. Широта может быть северная и южная, долгота – восточная или западная. Картографическая сетка (сетка широт и долгот) может быть наложена на земную поверхность, чтобы географически описывать местоположения. Линии долготы, иногда называемые меридианами, начинаются и заканчиваются на Северном и Южном полюсах. Линии широты, называемые параллелями, охватывают земной шар параллельными кольцами.

Широта и долгота традиционно измеряются в градусах, минутах и секундах. Широта, равная 0 град., располагается на экваторе, 90° – на Северном полюсе, –90° – на Южном. На долготе, равной 0 град., расположен начальный меридиан, который начинается на Северном полюсе, проходит через Гринвич в Англии и заканчивается на Южном полюсе. Долгота положительна до 180°, если двигаться к востоку от Гринвича, и отрицательна до –180°, если двигаться на запад от Гринвича. Например, Австралия, которая находится южнее экватора и восточнее Гринвича, имеет положительную долготу и отрицательную широту.

Однако широта и долгота являются географической описательной системой, а не двумерной (плоской) системой координат. Отметим, как меридианы сходятся на полюсах, но разделяются или расходятся при приближении к экватору. Таким образом, длина одного градуса долготы будет различна в зависимости от широты, на которой она измерена. Например, один градус долготы на экваторе составляет 111 км, а длина одного градуса долготы на Южном или Северном полюсах приближается к нулю. Так как эти единицы измерения не связаны со стандартной длиной, они не могут быть использованы для точного измерения расстояний. И так как в этой системе измеряются углы от центра земли, а не расстояния на земной поверхности, она не является плоской системой координат.

Плоские системы координат, используемые при создании карты

Плоские системы координат (часто называемые Декартовыми системами координат) обладают некоторыми свойствами, которые делают их пригодными для представления реальных географических координат на карте:

– имеется два измерения: *x* измеряет расстояние в горизонтальном направлении, а *y* измеряет расстояние в вертикальном направлении;

– меры длин, углов и площадей остаются постоянными по всем измерениям;

– существуют различные математические формулы для отображения сферической поверхности земли на двумерную поверхность.

ГИС, как и плоские карты, используют различные плоские координатные системы для картографирования земной поверхности. Каждая из используемых координатных систем базируется на определенной картографической проекции.

Картографические проекции

Поскольку поверхность земли является сфероидом, для создания плоской карты необходимо использовать математические преобразования. Эти математические преобразования называются *картографическими проекциями.*

Картографическая проекция – математически определенный способ изображения поверхности земного шара или эллипсоида (планеты) на плоскости. Общее уравнение картографической проекции связывает геодезические широты (B) и долготы (L) с прямоугольными координатами x и y на плоскости: $x = f_1(B, L)$; $y = f_2(B, L)$, где f_1 и f_2 – независимые, однозначные и конечные функции. Все картографические проекции обладают теми или иными искажениями, возникающими при переходе от сферической поверхности к плоскости. По характеру искажений картографические проекции подразделяют на *равноугольные* проекции (не имеющие искажений углов и направлений), *равновеликие* (не содержащие искажений площадей), *равнопромежуточные*, сохраняющие без искажений какое-либо одно направление (меридианы или параллели) и *произвольные* (в той или иной степени содержащие искажения углов и площадей).

Главный масштаб карты показывает степень уменьшения линейных размеров эллипсоида (шара) при его изображении на карте. Искажения масштаба проявляются в наличии частного масштаба карты в любой ее точке. Под этим понимается отношение длины бесконечно малого отрезка

на карте к длине бесконечно малого отрезка на поверхности эллипсоида (шара). Мерой искажений в картографической проекции в каждой точке карты служит бесконечно малый эллипс искажений. Существуют специальные карты, иллюстрирующие распределение искажений разных видов посредством изокол – изолиний равных искажений. В зависимости от положения сферических координат картографические проекции делят на *нормальные* проекции, в которых ось сферических координат совпадает с осью вращения Земли, *поперечные,* в которых ось сферических координат лежит в плоскости экватора, и *косые,* когда ось сферических координат расположена под углом к земной оси. Различие требований к картам разного пространственного охвата, тематики и назначения, а также сами особенности конфигурации картографируемой территории и ее положение на земном шаре привели к огромному многообразию картографических проекций.

Различаются картографические проекции по виду меридианов и параллелей нормальной сетки. *Цилиндрические* проекции, в которых меридианы изображены равноотстоящими параллельными прямыми, а параллели – прямыми, перпендикулярными к ним. *Конические* проекции с прямыми меридианами, исходящими из одной точки, и параллелями, представленными дугами концентрических окружностей. *Азимутальные* проекции, в которых параллели изображаются концентрическими окружностями, а меридианы – радиусами, проведенными из общего центра этих окружностей. *Псевдоцилиндрические* проекции, где параллели представлены параллельными прямыми, а меридианы – в виде кривых, увеличивающих свою кривизну по мере удаления от прямого центрального меридиана. *Псевдоконические* проекции, в которых параллели представлены дугами концентрических окружностей, средний меридиан – прямой, а остальные меридианы – кривые. *Поликонические* проекции, в которых параллели изображены эксцентрическими окружностями, центры которых лежат на прямом центральном меридиане, а все остальные – кривыми линиями, увеличивающими кривизну с удалением от центрального меридиана. *Условные* проекции, в которых меридианы и параллели на карте могут иметь самую разную форму. Для карт, создаваемых в виде серий листов, используют *многогранные* проекции, параметры которых могут меняться от листа к листу или группе листов. Компьютерные технологии позволяют рассчитывать картографические проекции любого вида и с заранее заданным рас-

пределением искажений. Иногда картографической проекцией ошибочно называют сетку меридианов и параллелей на карте.

Каждая базовая карта хранится в конкретной проекции. Поэтому необходимо определить проекцию вашей базовой карты перед вводом ее в систему. Понимание использования картографической проекции и широтно-долготных реперных точек (о реперных точках см. ниже) является очень важным при создании БД картографической информации. Имеется в виду следующее:

– любое двумерное представление земной поверхности всегда вносит искажения в некоторые параметры либо в форму, площадь, расстояние или направление;

– различные проекции вносят различные искажения;

– характеристики каждой проекции делают ее удобной для некоторых приложений и непригодной для других.

Хотя детальное знание картографических проекций не является необходимым для того, чтобы убедиться, что значения координат в вашей базе данных выражены в системе реальных географических координат, оно становится важным, когда вы имеете дело с большим объемом картографических данных, исходные проекции которых могут отличаться.

Координаты и другие параметры, используемые в картографических проекциях

Координаты – числа, заданием которых определяется положение точки на плоскости, поверхности или в пространстве. *Прямоугольные*, или *декартовы координаты*, точки на плоскости – это снабженные знаками ”+” или ” – “ расстояния *x* и *y* (абсцисса и ордината) этой точки от двух взаимно перпендикулярных прямых *X* и *У*, являющихся координатными осями, которые пересекаются в некоторой точке, называемой началом координат. *Прямоугольные координаты* точки в пространстве – три числа *x*, *y* и *z* (абсцисса, ордината, аппликата), определяющие положение точки относительно трех взаимно перпендикулярных плоскостей. Плоскости пересекаются в начале координат и по координатным осям *X*, *У* и *Z*. *Полярные координаты* на плоскости (на поверхности) – два числа: полярное расстояние точки от фиксированного начала и полярный угол между выбранной полярной осью и направлением на точку. В качестве полярной оси на плоскости часто принимают направление, параллельное оси абсцисс, а на эллипсоиде – северное направление меридиана. В первом случае полярным

углом будет дирекционный угол, во втором – азимут. В пространстве в качестве полярных координат используют радиус-вектор (расстояние от начала координат до заданной точки), вертикальный угол и азимут. *Сферические координаты* – три числа: радиус-вектор, геоцентрические широта и долгота. *Эллипсоидальные координаты* – три числа: геодезические широта, долгота и высота; определяют положение точки земной поверхности относительно земного эллипсоида.

Измерениями на физической поверхности определяют астрономические широты и долготы. Различия геодезических и астрономических координат обусловлены уклонениями отвесных линий, зависят от формы Земли (земного эллипсоида) и являются особым предметом изучения геодезии. В мелкомасштабном картографировании различием геодезических и астрономических широт и долгот пренебрегают и их именуют географическими координатами – названием, исторически сложившимся по отношению к шарообразной и однородной по строению Земле. Часто ошибочно геодезические координаты называют географическими. Координаты с началом на земной поверхности или в околоземном пространстве называют топоцентрическими координатами, с началом в центре масс – геоцентрическими координатами, около центра масс Земли – квазигеоцентрическими координатами. Различают: координаты экваториальные – одной из координатных плоскостей является плоскость экватора, координаты горизонтные – координатной плоскостью служит плоскость горизонта. На эллипсоиде, шаре и на картах применяют криволинейные координаты – сетку меридианов и параллелей. Трансформирование координат – преобразования, осуществляющие сдвиг, вращение и масштабирование координат при пересчете из одной системы в другую.

Как известно, почти для каждого типа картографической проекции (задающего на принципиальном уровне сам тип проектирования) могут использоваться многочисленные дополнительные параметры, конкретизирующие способ формирования изображения проекции и систему прямоугольных координат, используемую в проекции. Так, например, для поперечной проекции Меркатора задаются: центральный меридиан проектирования, масштабный коэффициент на центральном меридиане, широта точки от начала отсчета координат, смещение начала отсчета координат.

Надо учитывать, что проекция может быть определена для сферы или для какого-либо референц-эллипсоида. Референц-эллипсоид определяется длинами двух полуосей или длиной одной из них и коэффициентом сжа-

тия, определяющим отношение между осями. Обычно конкретные модели референц-эллипсоидов имеют для удобства использования имена собственные, как, например, эллипсоид Красовского или эллипсоид Кларка 1866 года.

Наконец, для точных работ необходимо учитывать и положение конкретного эллипсоида по отношению к геоиду (определение геоида см. ниже), что задается на практике смещением центра эллипсоида по отношению к центру масс Земли по трем координатам и поворот его относительно этого центра по трем углам.

Геоид – геометрическое тело, точнее всего передающее форму планеты Земля, имеет неправильную, нерегулярную форму и определяется с помощью измерений гравитационного поля. За поверхность геоида принимается поверхность, перпендикулярная к направлению вектора силы тяжести, на которой абсолютное значение этого вектора равно значению силы тяжести на поверхности открытого океана. Иначе говоря, это поверхность нулевого уровня моря. Она совпадает со средним значением уровня моря на морях и океанах, а на суше она проводится путем вычислений по измеренным значениям гравитационного поля. Поверхность геоида не совпадает с поверхностью рельефа суши, обычно отклоняясь от нее в ту или другую сторону. Форма геоида математически описывается достаточно сложно, с помощью разложения в ряд по сферическим функциям, коэффициенты которого описывают потенциал гравитационного поля Земли. Использовать в практических расчетах такую математическую модель затруднительно. В силу такой сложности формы геоида его нельзя с точностью, достаточной для построения топографических карт, аппроксимировать каким-то одним эллипсоидом для всей поверхности нашей планеты. Однако для конкретных, достаточно обширных участков поверхности это возможно.

Национальные картографические агентства разных стран, имея задачу обеспечения точного картографирования территорий своих стран, в итоге, путем длительных исследований, подобрали для своих территорий те модели референц-эллипсоидов и тот способ их пространственной ориентации по отношению к геоиду, который обеспечивал бы наиболее точное соответствие измерений, проводимых на топографической карте, измерениям на местности.

Это касается в первую очередь измерений, связанных с геодезическими координатами X и Y на карте. Что касается измерений высот (координа-

ты *Z*), то здесь ситуация более сложная. Без использования формального математического описания геоида, которое достаточно сложно, построить универсальную систему учета отклонений по вертикали используемого референц-эллипсоида от геоида затруднительно. На практике с помощью построения сетей нивелирования и триангуляции определенная базовая высотная отметка транслируется (переносится или, как часто говорят, передается) на определенные опорные точки, называемые пунктами государственной геодезической сети. Именно так используется, например, средний уровень Балтийского моря, определенный в конкретной точке в так называемой Балтийской системе высот. Для всех опорных пунктов в специальных каталогах зафиксированы значения геодезических координат *X* и *Y*, а также значение высоты рельефа местности в данной точке по отношению к нулевой отметке (уровню моря в определенной точке). При этом расхождение этой высоты с геоидом обычно оставалось неизвестным. Сегодня, с ростом потребности в высокоточных геодезических измерениях в пределах больших территорий, с ростом использования спутниковой геодезии, данных дистанционного зондирования, особенно радарных съемок, вопрос о так называемом “вертикальном датуме”, то есть системе высотных координат, встает заново, и он уже находит практическое решение благодаря возросшей мощности современных компьютеров.

Таким образом, национальные картографические службы используют определенные типы картографических проекций, конкретные их параметры, референц-эллипсоиды и определенную их ориентировку для построения национальных опорных геодезических сетей и базирующихся на них систем топографических карт. В русском языке всю эту совокупность используемых положений и параметров можно назвать *“системой геодезических координат”*, что в английском языке принято называть словом *Datum*. Именно эти параметры определяют специфику используемых в различных странах систем координат (иногда также и различных для различных масштабов карт).

Как правило, типы проекций, используемые при этом, доступны, все же остальное может быть задано в виде значений параметров. Можно вводить параметры интерактивно для разового применения, но для удобства повторного использования лучше определить весь набор параметров или целую совокупность таких наборов как стандартную национальную систему (например систему геодезических прямоугольных координат 1942 года, используемую в России для топографических карт в проекции Гаусса – Крюгера, что и будет предметом дальнейшего описания).

Координаты Гаусса – Крюгера

Координаты Гаусса – Крюгера – это система плоских прямоугольных координат, вводимая при помощи одноименной равноугольной картографической проекции (проекции Гаусса – Крюгера). Математически она описывается как поперечная проекция Меркатора и имеет следующую конструкцию. Вся Земля делится по долготе на шестиградусные зоны, которые нумеруются к востоку, начиная от Гринвичского меридиана, как от нулевого. В этих координатах земной эллипсоид отображается на плоскости именно такими зонами, ограниченными меридианами с разностью долгот 6°. В каждой зоне имеется свой центральный меридиан. Например, для первой зоны (от 0 до 6 град. восточной долготы) центральный меридиан располагается по 3-му градусу восточной долготы. В каждой из таких шестиградусных зон строится своя, отличная от соседних зон система координат; отличие по параметру “центральный меридиан проекции” (“центральный, осевой меридиан зоны”). Осью *X* (абсцисс) является изображение среднего или осевого меридиана зоны, осью *Y* (ординат) – изображение экватора. Восточная долгота осевого меридиана в первой шестиградусной зоне равна 3°, во второй – 9° и т.д. Начало координат зоны – точка пересечения экватора и осевого меридиана; для первой зоны эта точка имеет $x = 0$ м, $y = 500\,000$ м (от меридиана Гринвича). Номер зоны указывается перед *y*. Значение *x* для точек на осевом меридиане равно длине дуги меридиана эллипсоида от экватора до заданной параллели. При топографических съемках масштабов 1:5 000 и крупнее применяют трехградусные зоны, для которых осевые меридианы совпадают с осевыми и граничными меридианами шестиградусных зон.

Система разбиения на шестиградусные зоны тесно связана с построением системы разграфки и номенклатуры листов топографических карт разных масштабов. Каждой шестиградусной зоне соответствует одна колонна листов карты 1 : 1 000 000. Набор листов карты, отвечающий по долготе одной зоне, имеет одну цифру в номенклатуре, но отличается буквой, обозначающей пояс по широте. Лист карты масштаба 1 : 1 000 000 разбивается регулярной сеткой на равные в градусном выражении листы карт более крупного масштаба – 1: 500 000, 1: 200 000, 1: 100 000 и т.д. Для всей колонки листов карт используется одна и та же проекция с одними параметрами и одной системой прямоугольных координат. Для крайних в колонне листов, помимо сетки прямоугольных координат “своей” зоны, приводятся также подписанные по рамке значения координат в системе “соседней” зоны.

В России принято, что прямоугольные координаты для проекции Гаусса – Крюгера имеют нестандартные именования осей – ось *X* направлена вдоль центрального меридиана зоны, а ось *Y* – по широте параллельно экватору с Запада на Восток. В принятой у нас системе координат для того, чтобы избежать использования отрицательных значений координат по широте, к значениям координаты *Y* добавляется постоянная величина 500 000 метров (трехградусная полуширина зоны даже на экваторе составляет 500 000 метров). Для определенности, чтобы только по численному значению координаты *Y* можно было определить, к какой зоне относятся эти значения, к ним слева приписывается номер зоны. Важно понимать, что этим не создается единая непрерывная прямоугольная система координат для всего земного шара, и эти приписанные слева номера зон не являются значениями координат, а служат как бы индексным префиксом к ним.

Таким образом, мы получаем двухуровневую систему координат, сначала дискретную, использующую деление территории Земли на равные области (шестиградусные зоны), а затем обычную прямоугольную систему числовых координат, свою в пределах каждой зоны. Для каждой зоны описывается, собственно говоря, отдельная проекция, так как в каждой зоне свое расположение осевого меридиана и свое начало координат. Описания для всех этих зон вместе помещаются в новый раздел (категорию) библиотеки стандартных проекций под общим названием “Russian National Mapping System (GK 1942)”.

Реперные точки

Представление графической составляющей во многом определяется процессом преобразования пространственных данных в реальное изображение (в действительных координатах). Для этого, перед тем как начать оцифровку территории, всегда устанавливают регистрационные точки (реперы) – точки, для которых известны реальные координаты. Эти общие регистрационные точки (реперы) должны быть на каждом листе карты, чтобы обеспечить общую позиционную привязку каждого покрытия. Как только карта оцифрована, эти точки становятся регистрационными или контрольными точками данного покрытия, позволяющими описывать все объекты покрытия в общей системе координат. Кроме того, другие покрытия, например, соседние или другие слои данного покрытия могут быть сопоставлены с использованием тех же регистрационных точек и тех же картографических проекций, гарантируя географический контроль.

В процессе оцифровки необходимо сделать информацию о координатах реперов осмысленной. Чтобы ввести масштаб, необходимо преобразовать заданные величины в реальные координаты в той же проекции, в которой была исходная карта. Этот процесс называется преобразованием.

Использование проекции Гаусса – Крюгера в ArcView-ГИС для оцифровки территории

Для выполнения всех нижеперечисленных операций на компьютере должны быть установлены кроме ArcView 3.1: Spatial Analyst, ImageWarp, программа ERDAS IMAGINE (с координатным калькулятором Coordinate Calculator).

Описания всех используемых проекций ArcView хранит в файле $HOME/default.prj. Поскольку корректная структура и правильное содержание файла default.prj критически важны для функционирования, перед любым редактированием этого файла необходимо сделать его резервную копию. Пусть создали эту копию в той же директории $HOME/ под именем standart_default.prj. Теперь необходимо ввести в default.prj описание проекции координат Гаусса – Крюгера 1942 года с определенными параметрами проекции, например для зоны 7 (Москва) с назначением:

– долгота точки начала координат, т.е. центрального меридиана седьмой шестиградусной зоны (.SetCentralMeridian) = 39 град. восточной долготы;

– широта точки начала координат (.SetReferenceLatitude) = 0 град. северной широты;

– масштабный коэффициент в точке начала координат (.SetScale)= = 1.00000;

– сдвиг значений координат по долготе (.SetFalseEasting)= 7 500 000, где 7 – приписанный слева номер зоны, 500 000 – стандартный сдвиг для получения только положительных значений координат);

– сдвиг значений координат по широте (.SetFalseNorthing) = 0;

– используемый сфероид – сфероид Красовского 1940 г. (#SPHEROID_KRASOVSKY);

– единицы, используемые для значений координат, – метры.

Для карт с другими зонами нужно изменить соответствующие параметры, иначе привязка будет невозможна.

Для ввода в default.prj описания этой проекции с параметрами был создан в поле скриптов в ArcView на встроенном в ArcView языке Avenue следующий скрипт:

```
' Create a CoordSys object and get its list of projections
```

```
c = CoordSys.Make
c.SetName("Pulkovo 1942")
projections = c.GetProjections
' Create a projection
r = Rect.Make(103@"-56".AsNumber,164@"-7".AsNumber)
projection1 = TrnMerc.Make(r)
projection1.SetDescription("AMG ; Zone 7")
projection1.SetCentralMeridian(39)
projection1.SetReferenceLatitude(0)
projection1.SetScale(1.00000)
projection1.SetFalseEasting(7500000)
projection1.SetFalseNorthing(0)
projection1.SetSpheroid(#SPHEROID_KRASOVSKY)
' Add the new projections to the CoordSys projections
projections.Add(projection1)
' Create a default.prj file and add the CoordSys object
defprj = ODB.Make("$HOME/default.prj".AsFilename)
defprj.Add(c)
defprj.Commit
```

Этот скрипт заносят в ArcView всего один раз, если он еще не был внесен. Для этого нужно выбрать в основном окне Scripts, добавить этот скрипт в появившееся окно, включить компиляцию (Compile ) и затем запустить скрипт (Run ).

Теперь можно постоянно пользоваться сформированным описанием проекции Гаусса – Крюгера для 7-й зоны как стандартным (закладка Standart) в диалоге выбор/описания проекций (View/Preferences/Projections – Вид/Установки(категории)/Проекции). Достаточно выбрать из меню категорий “Russian National Mapping System (GK 1942)” и из меню проекций строку с соответствующим номером зоны.

Привязка подложки к реальным координатам

Подложкой назовем картинку-изображение пространственных объектов для решаемой задачи. Ее привязка к реальным координатам осуществляется последовательным выполнением следующих действий:

1. Загрузить ArcView и создать проект (File → New Project) с именем proj2.apr.

2. Выбрать доступные расширения (File → Extensions): ImageWarp, Spatial Analyst, JPEG, TIFF или другие.

3. Создать вид (Views), который можно переименовать (View → Properties ).

4. Перейти на вид и добавить новую тему (View → Add Theme ). В появившемся диалоговом окне (рис. 4) выбрать Image Data Source, после

чего по соответствующему маршруту выбрать картинку (пусть исходный источник карт-изображений – primer.jpg) и загрузить ее (“Карта”, рис. 6).

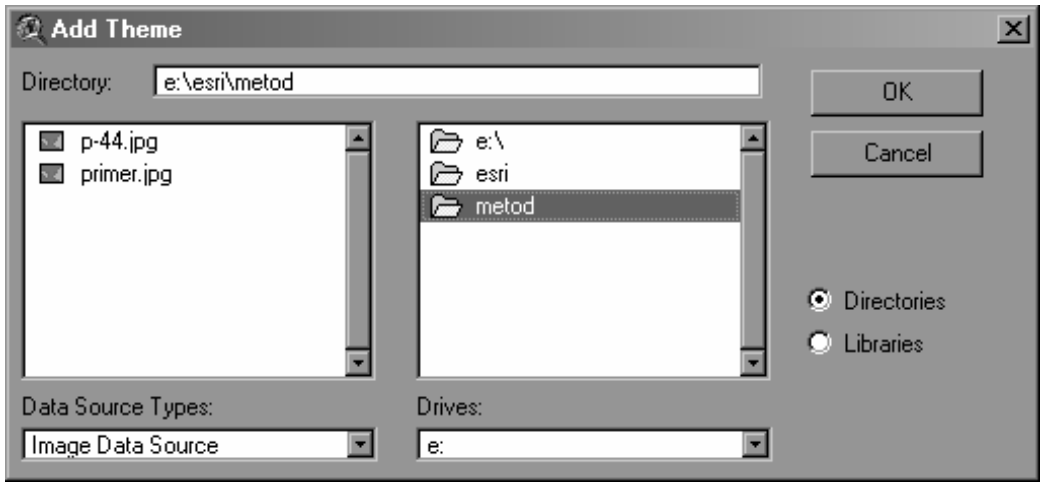


Рис. 4. Диалоговое окно «Добавление темы»

5. После загрузки карты необходимо создать новый слой (View → New Theme), в котором будут находиться реперные точки (тип – Point) и сохранить его (лучше указывать имя файла, соответствующее назначению слоя, например rep-points.shp).

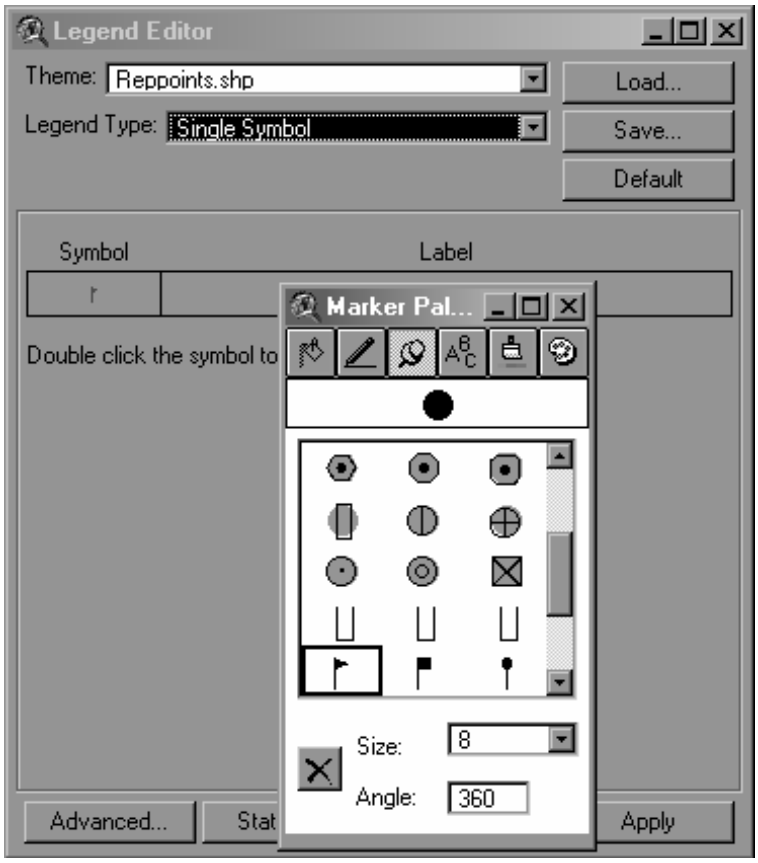


Рис. 5. Редактор легенды

6. Используя редактор легенды Legent Editor (двойной щелчок на тему), выбрать вид реперных точек (рис. 5).

7. Вставить реперные точки (Draw Point) по точкам с известными координатами с Карты (рис. 6). Таких точек должно быть не менее трех, четырех (и больше – для более точного расчета). При этом должно быть разрешено редактирование в редакторе легенды для указанной темы (Theme → Start Editing).

8. Выделить очередную реперную точку (рис. 6) и перейти к редактированию таблицы (Theme – Table). В появившуюся таблицу добавить два новых поля (Edit → Add Field) (рис. 7,*а*)): *X* и *Y*, после чего занести уже известные координаты (Edit), пересчитанные на координатном калькуляторе (см. приложение). И так по всем реперным точкам (рис. 7,*б*).

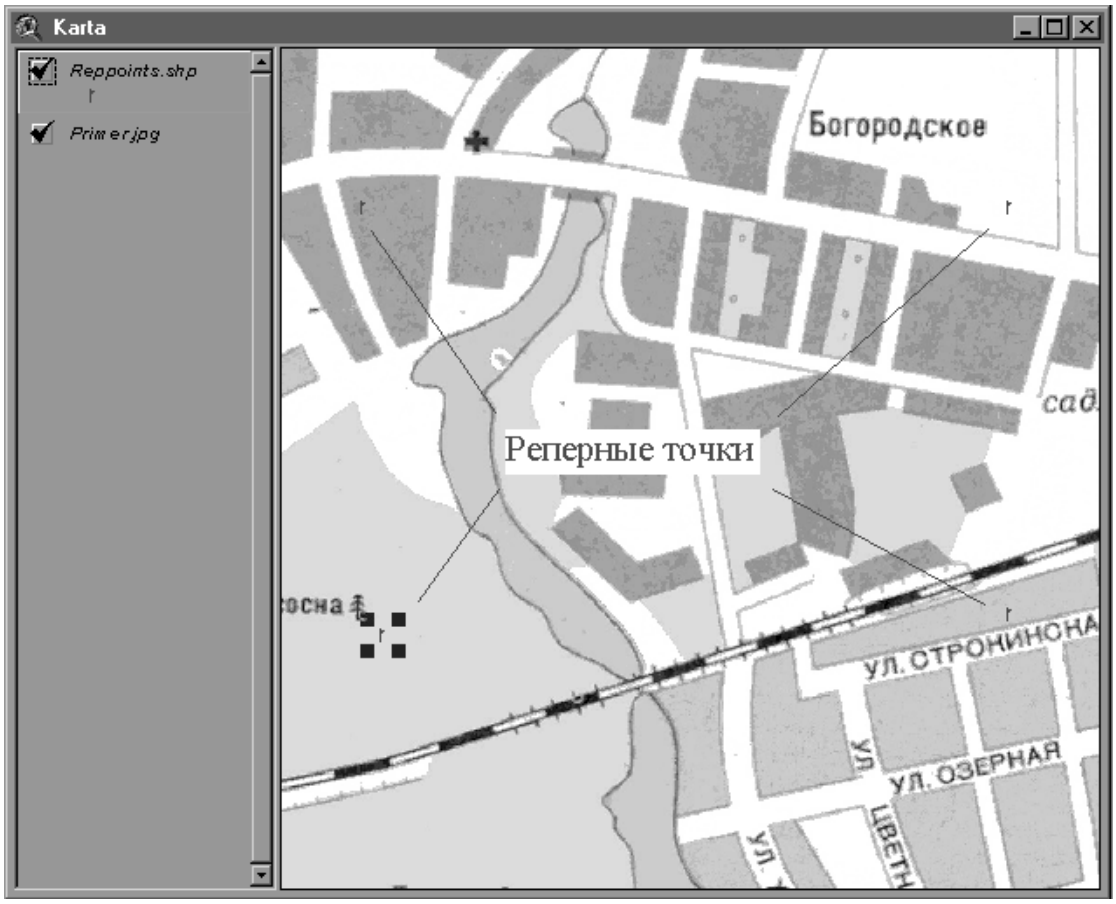


Рис. 6. Вставка реперных точек по точкам с известными координатами

9. По окончанию ввода нужно сохранить таблицу атрибутов (Table → Save Edits).

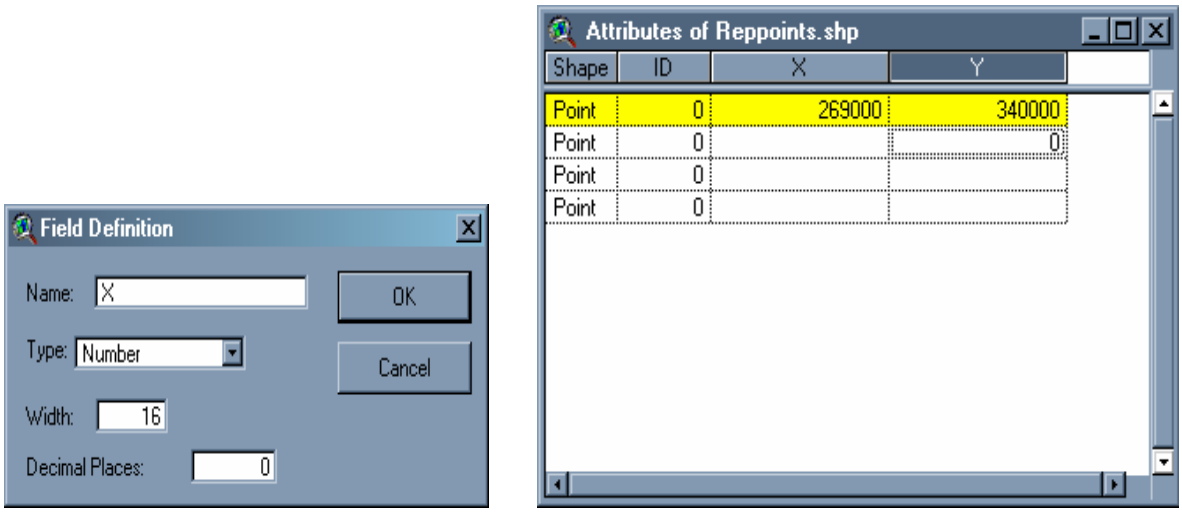
10. По окончанию вышеперечисленных операций сохранить редактирование Карты (Theme → Save Edit).

11. Остановить редактирование (Theme → Stop Editing).

12. Перейти на окно управления проектом (рис. 8).

13. Выполнить привязку изображения к выбранной системе координат. Для этого запустить ImageWarp → Image Warp Session и исполнить следующую последовательность шагов:

– в появившемся диалоговом окне указать картинку (Карту), которую нужно “привязывать” и указать тему (слой реперных точек), по которой привязка будет происходить;



а) *б)*

Рис. 7. а – диалоговое окно добавления нового поля в таблицу атрибутов; б – таблица атрибутов

– выбрать проекцию Гаусса – Крюгера (как указывалось ранее);

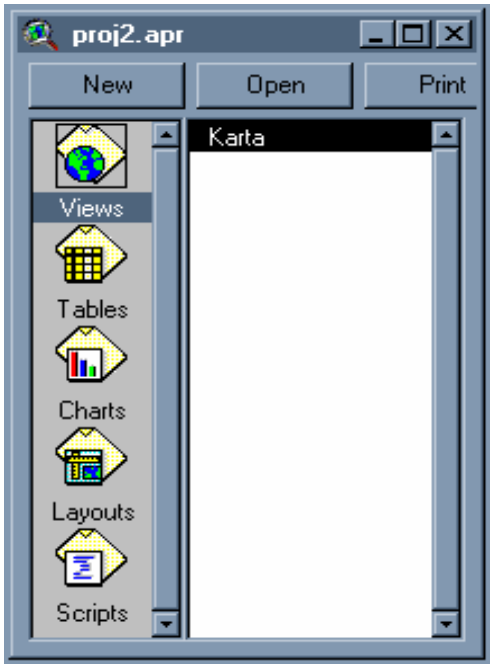


Рис. 8. Окно управления проектом

– если уже была создана таблица привязки, то указать ее, иначе создать новую таблицу;

– выбрать GCP Pick Tool (на панели инструментов ImageWarp) и сопоставить все точки на реперном слое (TO) и на картинке (FROM) поочередно. Результат выводится в окне TO *** ROAM (рис. 9);

– затем необходимо выполнить следующие вычисления:

1. Calculate RMS . Выбор степени полинома зависит от количества реперных точек (в нашем случае – 1).
2. Calculate from GCP . В результате файл картинки (Карты) будет привязан к текущей системе координат.

– записать измененный вид картинки (Карты) (Go Write the new ImageFile) (рис. 10).

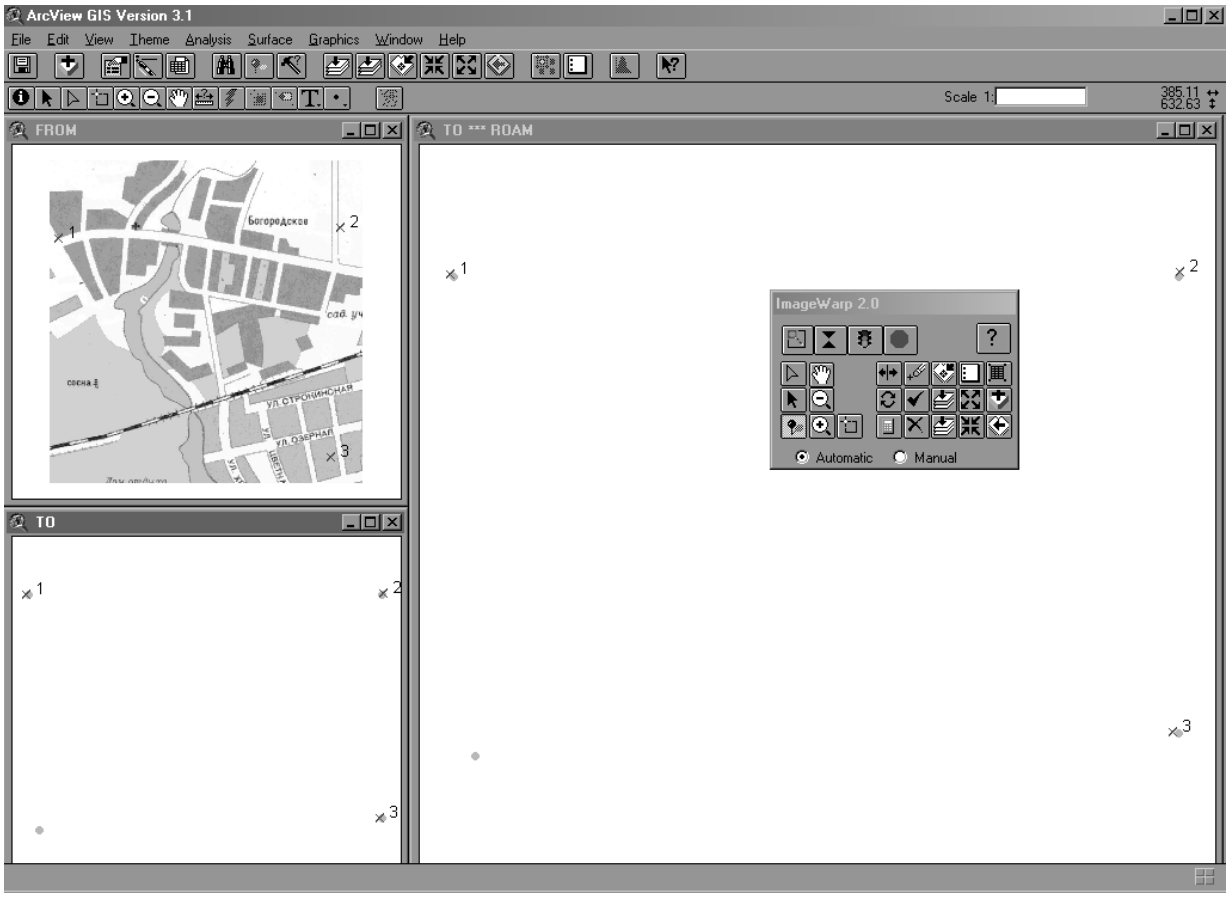


Рис. 9. Сопоставление реперных точек с точками на карте в ImageWarp

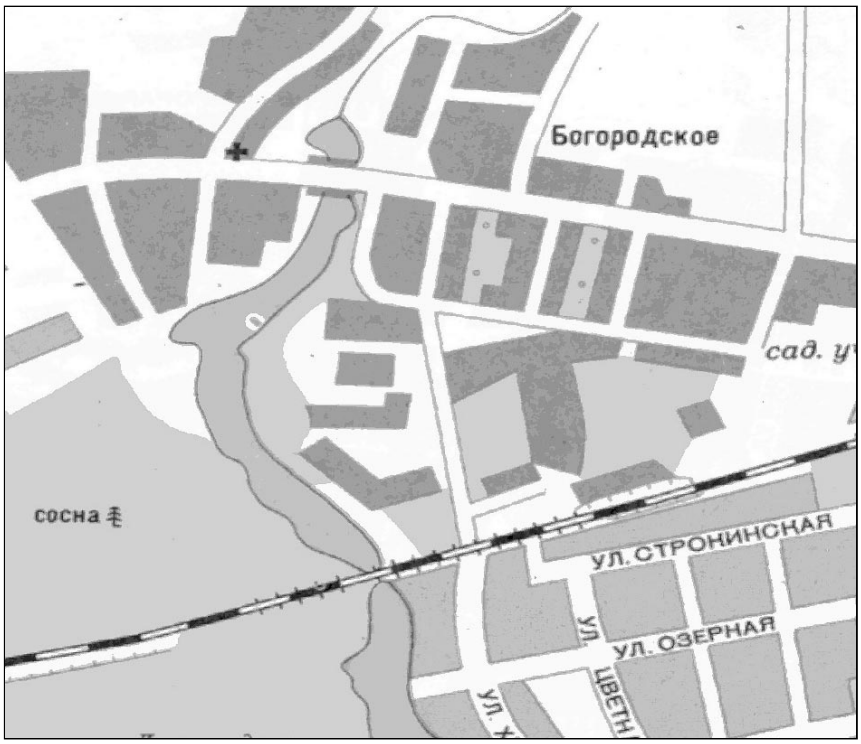


Рис. 10. Картинка (Карта) после привязки

Оцифровка

Добавьте уже привязанное к системе координат изображение (карту) в проект (в том случае, если она была записана под другим именем). Создайте новые темы с полигонами, точками, линиями (в зависимости от необходимости) и, используя полигоны, точки, линии, создайте соответствующие объекты на карте. Так до полной оцифровки карты (рис. 11). При этом следует включить редактирование (Theme → Start Editing), если оно было отключено.

Рис. 11. Оцифровка «привязанной» карты

Для оцифровки слоев, которых нет в цифровом виде, нужно получить наилучшие из доступных оригиналов карт.

Вы сэкономите много времени и избежите многих неприятностей, если будете ежедневно копировать результаты работы на архивные носители. Для окончательных покрытий, с которыми ведется более интенсивная работа, желательно делать резервные копии чаще. Это застрахует вас от потерь данных при сбоях или случайном удалении. Выполняйте всю текущую обработку только на копиях имеющихся покрытий. После завершения обработки покрытия можно стереть уже ненужный оригинал этого покрытия. Такой подход позволяет возвращаться при необходимости к предшествующей стадии с пригодным покрытием.

Заполнение БД

База данных заполняется редактированием таблицы атрибутов по каждому объекту (подобно редактированию таблицы атрибутов для реперных точек).

Рассмотрим подробно заполнение атрибутов для слоя “Почвы”, который проектировался в разделе 4.1, а его описание воспроизведено в нижеследующей таблице:

СЛОЙ	ОБЪЕКТ	КЛАСС	АТРИБУТЫ	ЗНАЧЕНИЕ	ОПИСАНИЕ
Почвы	SOILS	Полигоны	SOILCODE (типы почв)	Сокращения, принятые для всех типов почв	
			SUIT (пригодность)	0	Непригодные
				1	Малопригодные
				2	Умеренно пригодные
				3	Пригодные

1. Выделить соответствующий слой (Soils) и перейти к редактированию таблицы (Theme → Table ). Разрешить редактирование таблицы (Table → Start Editing). В появившуюся таблицу добавить шесть новых полей (Edit → Add Field…): [name: Layer (слой), type: String], [name: Object (объект), type: String], [name: Class (класс), type: String], [name: Attributes (атрибуты), type: String], [name: Value (значение), type: Number], [name: Description (описание), type: String].

2. Выбрать редактирование записей таблицы (Edit ) и заполнить поля соответствующими значениями (рис. 12):

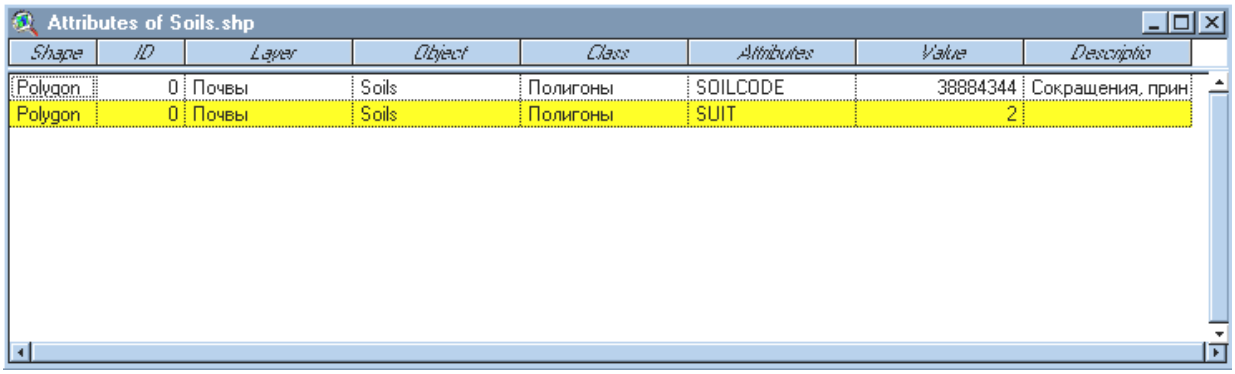


Рис. 12. Таблица атрибутов

3. По окончанию ввода сохранить таблицу атрибутов (Table → Save Edits). Аналогично можно заполнить поля и записи таблиц атрибутов для других объектов и слоев.

ПРИЛОЖЕНИЕ

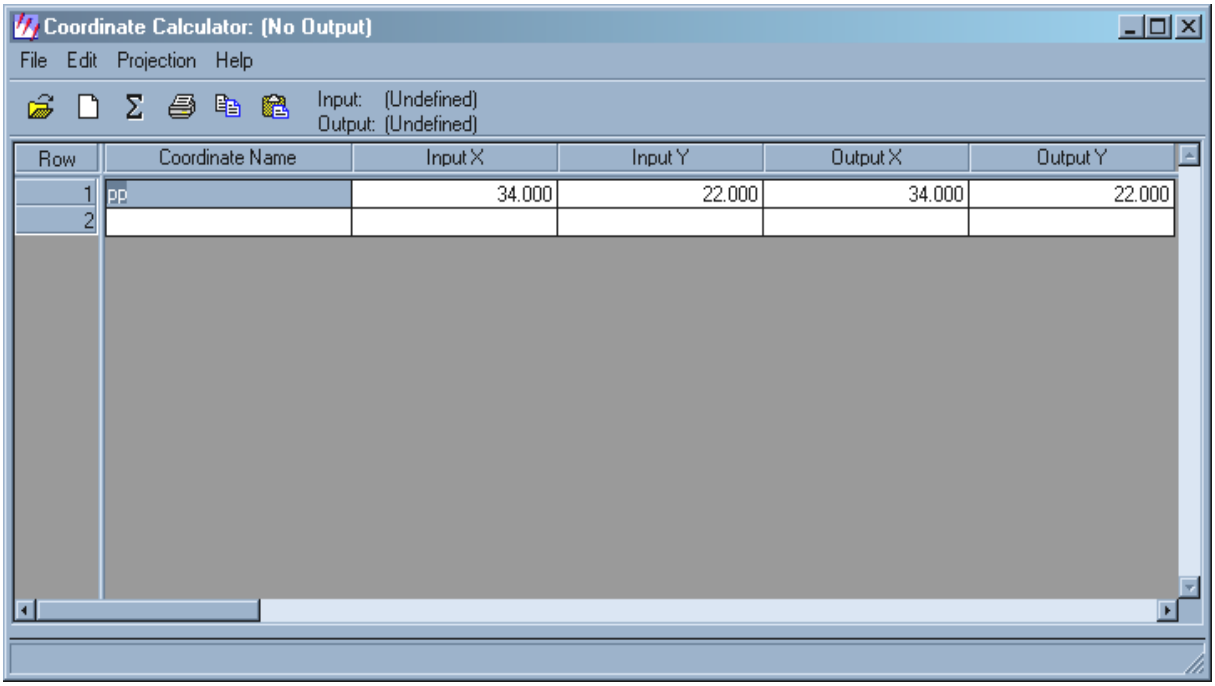
Преобразование координат

Так как на карте координаты всех реперных точек указаны в градусах, то их необходимо преобразовать из угловой системы координат в метрическую. Так как в ArcView нет соответствующей программы для преобразования координат, то можно воспользоваться координатным калькулятором программы ERDAS.

Вызов координатного калькулятора (Coordinate Calculator):

1. Запустить программу ERDAS IMAGINE.

2. Зайти в меню “Tools” главной панели управления, выбрать подпункт “Coordinate Calculator…” для запуска утилиты калькулятора (см. рисунок).



Координатный калькулятор

Рабочее поле программы делится на две секции. В первой выводится число колонок соответственно преобразованным координатам, во второй происходит само преобразование.

В меню “Projection” выбираем входную проекцию преобразования “Input Projection and Units Setup” и выходную “Output Projection and Units Setup”. В них устанавливаем систему преобразования координат (Гаусса – Крюгера) и единицы преобразования.

Вводим название очередного пересчета координат в колонку “Coordinate Name”, соответствующие входные координаты для *X*, *Y* и получаем их пересчитанные значения в колонках Output *X* и Output *Y*.

Работа с таблицами

В ArcView файлы проектов хранят абсолютные пути к файлам, так что если проект раньше находился в каталоге r:\folder1\folder2\ folder3, то вам придется на своем компьютере создать такой же каталог, скопировать в него все файлы проекта и работать с ними из того же места. В связи с этим при разработке своего проекта можно ассоциировать с папкой, в которой этот проект будет располагаться, имя какого-нибудь диска и в дальнейшем работать с этим диском. Такая ассоциация существенно облегчит перенос проекта с одной машины на другую. Само же ассоциирование можно произвести с помощью команды **SUBST** (Windows 95, 98) или с помощью подключения сетевого (виртуального) диска (NT, Win2000, XP).

Рассмотрим несколько приемов работы с таблицами.

Чтобы открыть таблицу необходимо:

а) в главном меню ArcView выбрать пункт Window, в подменю выбрать имя окна-проекта (файла с расширением .apr);

в) в появившемся окне проекта на левой панели выбрать Tables. Появится список всех таблиц;

с) двойным щелчком открыть нужную таблицу.

Редактирование данных:

а) по умолчанию таблица не находится в режиме редактирования. Перевести в этот режим ее можно с помощью изменившегося главного меню Table(таблица) → Start Editing;

в) затем, выбрав на панели инструментов Edit (), можно изменять данные.

Изменение имен столбцов и имени таблицы:

а) данное изменение можно сделать, открыв свойства таблицы: Table → Properties;

в) в поле Title можно изменить название таблицы (но не название файла на диске, который ее содержит). ArcView теперь будет воспринимать таблицу под новым именем;

с) полям тоже можно присвоить новые имена, записывая их в поле Alias. Это не приведет к изменению имен столбцов в файле на диске, но ArcView будет воспринимать столбцы под этими именами.

Для работы с таблицами в других приложениях (например в Excel) необходимо сделать следующее:

а) открыть таблицу и перейти в режим редактирования;

в) выбрать пункт Tables → Save Edits As… и указать имя выходного файла;

с) файл сохранится на диске в формате Dbase, т.е. имена столбцов “обрежутся” до 10 символов, пробелы в этих именах заменятся на знак подчеркивания, знаки подчеркивания заменятся на заглавную литеру “Z”;

д) если имена столбцов записаны кириллицей, то корректно прочитать их можно будет только в DOS-приложениях. Открыть такие файлы, например в Excel, с сохранением кириллических имен файлов не получается, приходится корректировать вручную.

Отсюда некоторые выводы. Если нужно использовать полученные таблицы не только в ArcView, то имена столбцов записывают только латинскими буквами. Старайтесь не употреблять длинных имен столбцов (более 10 символов). Если после обработки в других приложениях таблицы предполагается снова использовать в ArcView, то не используйте в именах столбцов пробелы и символы подчеркивания.

Чтобы узнать тип столбца таблицы:

а) выбрать таблицу и перейти в режим редактирования;

в) перейти в режим калькулятора, нажав кнопку Calculate ( );

с) выделить нужный столбец, выбрав его название в поле Field;

д) в поле Type будет показан тип данных в этом столбце.

Взаимодействие Avenue и DLL, написанных на C++

Подготовительный этап

1) Необходимо иметь в распоряжении файлы avexec32.h и avexec32.lib. Обычно эти файлы располагаются на диске с установленной ArcView, путь к ним примерно следующий: …\esri\av_gis30\acrview\lib32\.

2) Завести новую папку для разрабатываемых DLL (например j:\new) и скопировать в нее эти файлы.

3) Обязательно конвертировать файл avexec32.lib с помощью утилиты coff2omf.exe. Эта утилита входит в поставку Borland C++ Builder 5.0 и располагается в каталоге Bin. Путь к утилите такой: … \ Borland\ Cbuilder5\Bin\:

– скопировать файл coff2omf.exe в созданную директорию;

– конвертировать файл avexec32.lib, набрав в командной строке следующее: … >coff2omf.exe avexec32.lib avexec.lib;

– удалить файл avexec32.lib (его размер 1976 байт);

– переименовать файл avexec.lib в avexec32.lib. Размер нового avexec32.lib равен 512 байт. В дальнейшем речь будет идти только про него.

Написание DLL в среде Borland C++ 5.02

4) Запустить C++ 5.02.

5) Создать новый проект: File → New → Project.

6) В окне “New Target” указать имя проекта (например MyDll), его расположение (j:\new\MyDll.ide) и параметры, флажки, переключатели (Target Type: Dynamic Library [.dll] ; Platform : Win32 ; Target Model : GUI ; Dynamic ; Myltithread).

7) Нажать кнопку “Advanced” и установить флажок .cpp Node. Затем “OK”. В окне “New Target” также нажать “OK”.

8) Вывести окно Project: View → Project. В окне щелкнуть на имени файла-проекта mydll.dll .

9) Присоединить библиотеку avexec32.lib: при выделенном имени файла проекта в окне Project нажать клавишу “Insert” и указать тип присоединяемого к проекту файла (Libraries [*.lib]) и его местоположение (j:\new\avexec32.lib).

10) Затем ввести текст программы: в окне проекта щелкнуть на имени cpp-файла. В появившемся окне ввести текст программы и скомпилировать DLL: Project → Build all.

11) Библиотека mydll.dll появится в папке j:\new\.

Приведем пример текста функций, составляющих простейшую библиотеку.

```
#include <windows.h> // – обеспечение доступа к стандартным функциям Windows.
#include "avexec32.h" // – обеспечение доступа к стандартным функциям ArcView.
// функция входа в DLL обеспечивает работоспособность DLL в среде Windows:
int WINAPI DllEntryPoint(HINSTANCE hinst, unsigned long reason, void* lpReserved)
{ return 1; }
// определены 2 неэкспортируемые функции (внутренние функции DLL): функция min,
// возвращающая минимальное из двух чисел типа float и функция max,
// возвращающая максимальное из двух чисел типа float.
float min(float val1, float val2)
{ return (val1<=val2) ? val1 : val2; }
float max(float val1, float val2)
{ return (val1>val2) ? val1 : val2; }
// определена экспортируемая (доступная для других приложений, использующих эту
```

```
//DLL) функция minmax, возвращающая минимальное или максимальное из двух чисел
// типа float в зависимости от переключателя – SwitchOp. Стоит обратить внимание на
// префикс имен таких функций: extern "C"__stdcall _declspec(dllexport)
extern "C"_stdcall _declspec(dllexport) float minmax(float value1, float value2, int swhichOp)
{ return swhichOp ? max(value1, value2) : min(value1, value2); }
// определена экспортируемая функция sf ; демонстрирует взаимодействие C++ с
// встроенным в ArcView языком программирования Avenue; префикс имени тот же :
extern "C" _stdcall _declspec(dllexport) void sf(void)
{char *myAvenueStr = "av.GetActiveDoc";     // запись команды Avenue, как строки;
char *resultStr=AVExec(myAvenueStr);        // возвращение строки-результата
                                            // выполнение команды Avenue;
MessageBox(GetFocus(), resultStr, "Result of AVExec", MB_OK); // выполнение команды
}                                           // Avenue по выводу результата на экран.
```

Отметим, что среда Borland C++ for Windows (version 3.1) не подходит для создания DLL, взаимодействующих с ArcView, так как позволяет создавать только 16-разрядные приложения.

Работа с DLL в Avenue

Приведем пример Avenue-скрипта, работающего с только что написанной DLL.

```
'Подключить библиотеку mydll.dll
aDLL=DLL.Make("j:\new\mydll.dll".asFileName)
'Объявить функцию minmax этой библиотеки, взаимодействующую с Avenue
'Функция требует два числа типа float; и одного параметра типа integer(переключателя)
'Возвращает минимальное из чисел, если значение переключателя равно 0
'Возвращает максимальное из чисел в противном случае
MinMax=DLLProc.Make(aDLL, "minmax", #DLLPROC_TYPE_FLOAT,{#DLLPROC_TYPE
_FLOAT,#DLLPROC_TYPE_FLOAT,#DLLPROC_TYPE_INT32})
' Объявить функцию Библиотеки sf, взаимодействующую с Avenue
' функция не требует и не возвращает значений
SF=DLLProc.Make(aDLL, "sf", #DLLPROC_TYPE_VOID,{#DLLPROC_TYPE_VOID})
sf.call({})          ‘ вызов этой функции
'Определить два любых числа
a=1.2
b=12.3
'Найти минимальное из них, вызвав нужную функцию
x=MinMax.call({a, b, 0})
'Отобразить результат
msgbox.info("Минимум равен" ++x.asString, "Числа" ++a.AsString++" и "++b.AsString)
```

```
'Найти максимальное из тех же чисел
x=MinMax.call({a, b, 1})
'Отобразить результат
msgbox.info("Максимум равен"++x.asString,"Числа "++a.AsString++" и "++b.AsString)
```

Приведем список соответствия типов в Avenue и типов в C++ при работе с DLL. При написании DLL для Avenue в C++ следует использовать именно эти типы принимаемых и возвращаемых значений экспортируемых функций.

Тип переменной в Avenue	Эквивалент в C++
#DLLPROC_TYPE_VOID	Void
#DLLPROC_TYPE_INT16	Short int
#DLLPROC_TYPE_INT32	Int
#DLLPROC_TYPE_FLOAT	Float
#DLLPROC_TYPE_PINT16	* short int
#DLLPROC_TYPE_PINT32	* int
#DLLPROC_TYPE_PFLOAT	* float
#DLLPROC_TYPE_POINTER	* void

Другие примеры программирования на Avenue

1) Задать стартовый скрипт (запускающийся при старте проекта)

Window → «имя_проекта.apr»

Project → Properties

В поле StartUp указать имя скрипта.

2) Заставка при загрузке проекта

В стартовый скрипт вставить строку:

MsgBox.Banner("j:\graph\заставка.gif".AsFileName,3,"")

Здесь 3 – число секунд, в течение которых видна заставка.

3) При старте проекта открыть какой-либо документ

В стартовый скрипт добавить:

```
theView=av.FindDoc("Карта земельных участков")
win=theView.GetWin
win.open
win.MoveTo(0,0)
win.Maximize
```

В данном случае откроется вид “Карта земельных участков”.

Замечание. Никогда не давайте одинаковые имена различным документам (таблицам, видам, темам и т.д.). Это значительно усложнит их поиск.

4) Выборка из таблицы

```
‘ ищем таблицу по имени
theTable1=av.GetProject.FindDoc("Продуктивность культур на нулевом фоне")
theVtab1=theTable1.GetVtab        ‘ получаем ее виртуальную копию
‘ получаем битовую карту таблицы (список выделений)
theBitmap1=theVTab1.GetSelection
theBitmap1.ClearAll               ‘ снимаем выделение
‘ запишем строку-запрос (выбрать все записи, у которых
‘ в столбце “Tableno” стоит “1”
expr="([Tableno]="++1.asString++")"
‘ в данном случае столбец “Tableno” имеет тип Number
‘ если бы он имел тип String, то строку пришлось бы переписать так:
‘expr="([Tableno]="""+1.asString+""")"
([Tableno]= 1)                    ‘ так выглядит в первом случае запрос
‘ ([Tableno]= "1")                  а так во втором
‘ произведем выборку и обновим выделение
theVTab1.query(expr, theBitmap1, #VTAB_SELTYPE_NEW)
theVTab1.UpdateSelection
rec= –1                           ‘ пройдемся по выделенным записям
while (true)
    rec=theBitmap1.GetNextSet(rec)
        if (rec=–1) then break
            else                  ‘ что-то делаем
end
```

Замечание. Не работайте одновременно с несколькими битовыми картами (картами выделений) одной и той же таблицы. Это невозможно. Различные битовые карты будут синонимами.

5) Запуск скриптов

```
av.run("Рассчет базовых урожайностей",{"ur0.dbf",0})
```

Запустится скрипт "Рассчет базовых урожайностей" с передачей ему двух аргументов.

6) Проверка параметров при вызове скрипта. Скрипт может проверить: с какими параметрами он вызван и, если нужно, прекратить свое выполнение.

```
if (self.Is(list).Not) then return nil   ‘ скрипт не будет выполняться, если
elseif (self.count<2) then return nil ‘ он вызван менее чем с двумя
                                      ‘ параметрами.
```

```
else
the File=self.Get(0)          ‘ параметры можно получить через self
fon=self.Get(1)               ‘ методом Get.
end
```

7) Пример по созданию скрипта, объединяющего выбранные темы (схожие по полям) в одну новую. Перед запуском скрипта необходимо активизировать соответствующий темам Вид и выбрать в нем темы для объединения. Скрипт иллюстрирует объединение *n*-го количества выбранных тем с одинаковыми полями (Shape и Height) в их таблицах. Файл, в котором будет храниться новая тема с “объединенной” таблицей, назван Summa_Themes.shp.

```
‘ Создание по выбранным темам новой темы, таблица которой содержит
‘ лишь одинаковые поля выбранных тем, объединяя (“сливая”) их данные
‘ ************************************************************
‘ Поиск активного вида (если нет, то выход)
theView = av.GetActiveDoc
if (theView.AsString.Contains(".apr")) then ‘Contains – содержать, вмещать
        MsgBox.Info("Выберите какой-нибудь Вид","Нет активного вида")
        return nil
end
‘ Поиск активных тем (если нет, то выход)
theActiveThemesList = theView.GetActiveThemes
if (theActiveThemesList.Count=0) then
        MsgBox.Info ("Сделайте нужные темы активными", "Нет активной темы")
        return nil
end
‘ Подсчет количества активных тем
aCountoftheThemes=theActiveThemesList.Count
‘ Название файла, в котором будут храниться объединенные темы
aFileOfNewTheme="Summa_Themes.shp"
‘ Создание таблицы (точек)
myFTab = FTab.MakeNew (aFileOfNewTheme.AsFileName, Point)
k=0             ‘ счетчик активных тем
‘ Перебор всех активных тем и поиск в них общих полей (Shape и Height)
for each theTheme in theActiveThemesList
        k=k+1
‘ Получение объекта-таблицы активной темы,
‘ поиск в ней полей Shape и Height
        aFtab=theTheme.GetFtab
        aShape = aFTab.FindField("Shape")
```

```
        aHeight = aFTab.FindField("Height")
‘ Проверка, является ли запись типа Point (точка)
        aSrcName=aFtab.GetSrcName.GetSubName.AsString
        if (aSrcName<>"Point") then
‘ если нет, то переход к следующей записи
                continue
        end
‘ Создаем поля Shape и Height, если они еще не созданы
        if (theActiveThemesList.FindByValue(theTheme)=0) then
                myFTab.AddFields ({aHeight.clone})
                ShapeF = myFTab.FindField ("Shape")
                HeightF=myFTab.FindField ("Height")
        end
        NumRecords=aFtab.GetNumRecords     ‘число записей таблицы текущей темы
        av.ShowStopButton                  ‘вывод кнопки остановки процесса
‘…и заносим в них значения, полученные из других тем
        for each record in aFTab
                P=aFTab.ReturnValue(aShape, record)
                H=aFTab.ReturnValue(aHeight, record)
                rec = myFTab.AddRecord
‘ Занесение записей в новую таблицу атрибутов
                myFTab.SetValue(ShapeF, rec, P)
                myFTab.SetValue(HeightF, rec, H)
‘ Вывод процесса выполнения задачи
                progress = 100.0*record/(NumRecords-1)
                progress = progress*k/aCountoftheThemes
                doMore = av.SetStatus (progress)
                if (not doMore) then
                        break
                end
        end                        ' конец цикла for для записей темы
end                                ' конец цикла for по темам
‘ Добавляем новую (объединенную) тему
theView.AddTheme (FTheme.Make (myFTab))
myFtab.Flush                       ' обновление данных
```

РЕКОМЕНДАТЕЛЬНЫЙ БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Avenue. Руководство пользователя. GIS by ESRI / Printed in the United States of America: Contracts Manager, Environmental Systems Research Inctitute, Copyright Thomas G. Lane, 1996. – 280 с.
2. Введение в ARC/INFO версии 7.1.1. – М., 1998. – 164 с.
3. Как быстрее освоить ARCVIEW GIS // ARCVIEW. – 1998. –№ 1.
4. Что такое ГИС? // ARCVIEW. – 1998. – № 4.
5. Выбор подходящей ГИС / ARCVIEW. – 1999. – № 1.
6. Коновалова Н.В., Капралов Е.Г. Введение в ГИС: Учеб. пособие. – Петрозаводск, 1995. – 148 с.

ОГЛАВЛЕНИЕ

1. БАЗОВЫЕ ВОЗМОЖНОСТИ ArcView..... 3
2. ВСТРОЕННЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ Avenue..... 12
3. РАБОТА СО СКРИПТАМИ..... 31
4. СОЗДАНИЕ БАЗ ДАННЫХ..... 38
Приложение 64
Рекомендательный библиографический список..... 73

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ САМОСТОЯТЕЛЬНОЙ ПОДГОТОВКИ СТУДЕНТОВ К РАБОТЕ С ArcView-ГИС и Avenue ПРИ ВЫПОЛНЕНИИ КУРСОВОГО И ДИПЛОМНОГО ПРОЕКТИРОВАНИЯ

Составители:
ВОЛКОВА Ирина Сергеевна
ЖИРОВ Андрей Владимирович
ТРУФАНОВ Сергей Викторович и др.

Ответственный за выпуск — зав. кафедрой профессор С.М. Аракелян

Редактор Е.А. Амирсейидова
Корректор И.А. Арефьева
Компьютерная верстка К.Г. Солнцев

ЛР № 020275. Подписано в печать 28.11.02.
Формат 60x84/16. Бумага для множит. техники. Гарнитура Таймс.
Печать офсетная. Усл. печ. л. 4,42. Уч.-изд. л. 4,77. Тираж 100 экз.
Заказ
Редакционно-издательский комплекс
Владимирского государственного университета.
600000, Владимир, ул. Горького, 87.