Глава 1

Знакомство со средой программирования

В этой главе...

- Знакомство со средой программирования
- Что такое компоненты
- Первая программа
- Подсказки в Delphi
- Примеры создания простейших программ без написания кода

Термин "среда программирования" подразумевает несколько больше, чем только графический интерфейс пользователя, который вы видите при запуске Delphi 7. Среда программирования предоставляет широкие возможности по разработке, отладке и контрольному запуску программ, причем при этом предоставляется удобный графический интерфейс, набор необходимых инструментов и дополнительных компонент, большое количество подсказок, файлы справки и еще много-много необходимых мелочей, которые позволяют программисту работать в очень комфортных условиях.

Но чтобы почувствовать себя комфортно в среде Delphi, необходимо затратить некоторое время на изучение самой среды. Этим сейчас и займемся.

Знакомство со средой программирования

Среда программирования — это комплекс программ, которые разработаны для того, чтобы создать удобное окружение для реализации концепции быстрой разработки приложений RAD (Rapid Application Development). Среду программирования можно разбить на несколько отдельных частей, среди которых основными будут главный графический интерфейс пользователя, или главное окно (рис. 1.1), которое появляется сразу же после запуска Delphi, графический интерфейс справочной системы (рис. 1.2) и набор необходимых компонентов. Дополнительных окон может быть очень много. Такие из них, как Object Inspector (Инспектор объектов) или Object TreView (Дерево объектов) служат дополнением к основному окну, а большинство из дополнитель-

ных окон, таких как графический редактор, являются самостоятельными компонентами, без которых можно и обойтись. От того, насколько тщательно разработаны графические интерфейсы, зависит удобство использования всей среды разработки. Непосредственно разработка проекта происходит в главном окне, и наиболее часто на первом этапе будут использоваться проекты типа Application (Приложение), Console Application (Консольное приложение) или Dialogs (Диалог). По крайней мере так происходит у меня, поскольку я не занимаюсь разработкой библиотек, сервисных программ или специальных приложений. Поэтому начнем знакомство с главным окном, которое появляется по умолчанию при запуске Delphi.

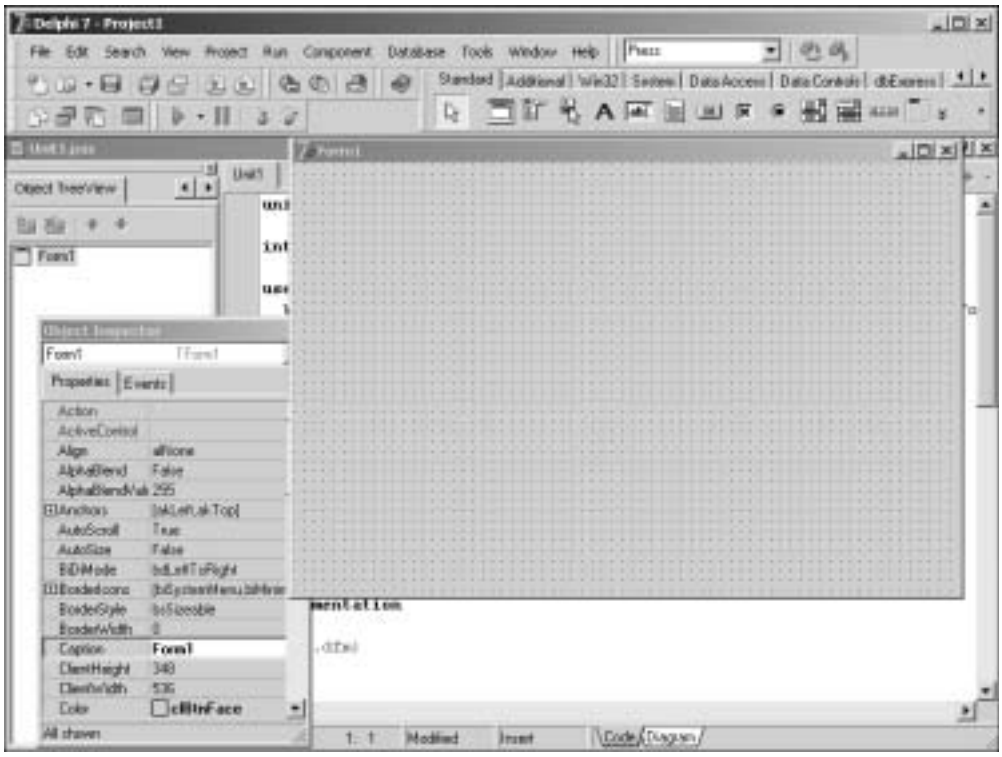

Рис. 1.1. Главное окно

Знакомство с элементами главного интерфейса пользователя

Для того чтобы начать работу в Delphi, дважды щелкните кнопкой мыши на ее пиктограмме на рабочем столе Windows или в меню выбора. При этом на экране сначала появится рекламная заставка, сообщающая о том, что среда разработки Delphi загружается в оперативную память компьютера. Затем Delphi автоматически создаст новый проект (см. рис. 1.1). Это будет уже

полностью работающие приложение, т.е. создается выполняемый файл с расширением `.exe`, пока не имеющее никаких полезных функций. Если не производилось дополнительной настройки, то по умолчанию на переднем плане будет отображаться окно формы с именем Form1, а за ним будет располагаться окно редактора кодов с именем Unit1. Все необходимые процедуры и функции нужно будет добавить в процессе разработки нового приложения. В среде Delphi предусмотрены различные типы проектов. Программист может легко изменить тип проекта по умолчанию. В книге внимание будет сконцентрировано на создании отдельных приложений, поэтому в основном изменять тип проекта по умолчанию не нужно, хотя иногда будут использоваться и другие типы проектов: консольные приложения, приложения Web, библиотеки DLL (Dynamic Link Library — динамически подключаемая библиотека), элементы управления ActiveX, активные формы и др. Если не указано иное, вы будете инициировать новый проект, запустив Delphi с пустым приложением в качестве типа проекта по умолчанию или используя пункты меню File⇨New⇨Application (Файл⇨Новое⇨Приложение).

Рис. 1.2. Окно справки

Для управления рабочим пространством необходимо открывать и закрывать файлы в зависимости от того, с каким фрагментом программы нужно работать. Для того чтобы закрыть окно редактора с соответствующим файлом, поместите его на передний план и затем нажмите клавиши <Alt+F⇨C>. При закрытии файла закрывается и соответствующее окно формы. Написание <Alt+F⇨C> означает, что нужно нажать клавишу <F>, удерживая при этом клавишу <Alt> нажатой , затем отпустить их и нажать клавишу <C>.

Для открытия файла можно использовать пункт меню Open (Открыть) из меню File (Файл) или комбинацию клавиши <Alt+F⇨O>. Аналогичный эф-

фект можно получить и при щелчке мыши на пиктограмме открывающейся папки на панели Standart. Когда открывается файл, который имеет связанное с ним окно формы, нет необходимости дополнительно открывать окно формы, это делается автоматически.

Для повторного открытия файла можно использовать пункт меню Reopen (Повторить открытие) из меню File (Файл) или клавиши <Alt+F⇨R>. Аналогичный эффект можно получить и при щелчке мыши на стрелке рядом с пиктограммой открывающейся папки на панели Standart.

Пункт меню Save (Сохранить) находится в меню File, как и все остальные пункты для работы с файлами. Можно выбрать пункты Save, Save As (Сохранить как), Save Project As (Сохранить проект как) или Save All (Сохранить все) для сохранения работы. Пункт Save используется только для сохранения файла из активного окна, находящегося на переднем плане. При выборе пункта Save As можно сохранить файл из активного окна с новым именем. Пункт Save Project As предназначен для сохранения проекта с другим именем, а с помощью пункта Save All сохраняются все файлы проекта. Когда сохраняется проект, имя проекта сохраняется в файле с расширением `.dpr` и используется при компиляции программы.

Используя пункты главного меню (см. рис. 1.1), можно выполнить большинство задач, которые необходимы при разработке проекта. Для вызова некоторых задач существуют специальные панели и отдельные пиктограммы, но они также дублируются в главном меню. Знакомство с отдельными пунктами главного меню будет происходить на протяжении всей книги, поэтому сейчас не будем их последовательно рассматривать. Также не будем останавливаться на пунктах меню, которые хорошо известны по системе Windows, так как читатель должен быть знаком с этой операционной системой.

Панели инструментов

Появившееся после запуска главное окно представляет собой набор отдельных панелей, которые наиболее часто используются при разработке приложений. Для того чтобы убрать или отобразить какую-то из панелей, щелкните правой кнопкой мыши на поле, где расположены панели главного окна (см. рис. 1.1). При этом появится ниспадающее меню, где перечислены все доступные панели: Standart (Стандартная), View (Просмотр), Debug (Отладка), Custom (Пользовательская), Component Palette (Палитра компонент), Desktop (Рабочий стол) и Internet), а также есть пункт Customize (Настройка). Исследуйте все пункты меню, последовательно устанавливая или сбрасывая флажки и наблюдая происходящие на экране изменения. Тот же эффект можно получить, используя пункты меню View⇨ToolBars (Просмотр⇨Панели).

Окна Object Inspector (Инспектор объектов) и Object TreeView (Дерево объектов) в этом меню не присутствуют, и для их отображения (если их нет на экране) необходимо выбрать соответствующее название в пункте меню View. Также можно использовать клавиши быстрого вызова, комбинации которых

обычно указываются в пунктах меню. Для инспектора объектов и дерева объектов это будут клавиши <F11> и <Shift+Alt+F11> соответственно. Основные окна форм и модулей также можно вызвать, используя пункты меню View или с помощью клавиш <Ctrl+F12> и <Shift+F12>.

Как и в большинстве приложений Windows, панели инструментов Delphi содержат пиктограммы, с помощью которых можно выбирать различные команды. Например, при однократном щелчке кнопкой мыши на пиктограмме, изображающей дискету, компьютер выполняет команду Save (Сохранить). Эту же команду можно выполнить, активизировав соответствующий пункт в меню File. Чтобы узнать назначение любой пиктограммы, нужно на некоторое время установить на ней указатель мыши, после чего рядом с указателем появится этикетка, напоминающая программисту о назначении пиктограммы.

Активизировать пункты меню можно также с помощью клавиатуры. Например, нажатие клавиши <Alt> приводит к переключению между окнами, и при активизации поля стандартного меню будет подсвечен очередной пункт, а некоторые буквы в названиях пунктов будут подчеркнуты. Нажатие такой буквы на клавиатуре приведет к активизации соответствующего пункта меню. Например, чтобы активизировать меню View (Просмотр), нужно с помощью клавиши <Alt> перейти к главному меню и нажать клавишу <V>, что можно обозначить как <Alt+V>. Одновременное нажатие клавиш <Alt> и <V>, которое обозначается как <Alt+V>, приведет к такому же результату.

Обозначение <Alt+V> говорит о том, что нужно нажать и удерживать клавишу <Alt>, а затем нажать клавишу <V>. Обозначение <Alt⇨V> используется, когда нажатие клавиш должно происходить последовательно, исключая одновременное нажатие.

Необходимо отметить, что во многих случаях существенную помощь может оказать контекстное меню, которое вызывается щелчком правой клавиши мыши на соответствующем окне или поле. Поэтому почаще это делайте и старайтесь понять все пункты меню. Изучив их, вы сможете ускорить процесс проектирования.

Панель Standart

Стандартная панель инструментов (Standart) содержит пиктограммы общих задач, таких как открытие, сохранение и создание проектов Delphi, и ассоциированных с ними файлов. Можно добавлять и удалять отдельные файлы из проекта. Делать это удобнее, чем через пункты главного меню, где эти задачи дублируются.

Панель View

Панель инструментов просмотра (View) содержит пиктограммы, предназначенные для создания новых форм, просмотра форм и модулей кода, а также для переключения между формой и модулем кода. С помощью панели просмотра можно быстро сделать видимыми разные окна среды разработки

Delphi. Это необходимо потому, что обычно на экране присутствует довольно много окон, причем они закрывают друг друга.

Панель Debug

Панель инструментов отладки (Debug) используется для интерактивного тестирования и отладки программ. Она обеспечивает быстрый доступ к командам отладчика Delphi, доступным также из пункта главного меню Run (Выполнение). Отладчик Delphi представляет собой утилиту времени разработки, т.е. ее можно использовать непосредственно в среде разработки Delphi при работе над исходным кодом.

Панель Custom

Пользовательская панель инструментов (Custom) по умолчанию содержит только пиктограмму вызова справочной системы Delphi. Через нее удобнее вызывать справочную систему, чем через главное меню. Пользоваться главным меню для вызова справочной системы необходимо только при отсутствии пользовательской панели.

Панель Component Palette

Палитра компонентов (Component Palette) представляет собой панель с вкладками, обеспечивающими быстрый доступ к компонентам VCL (Visual Component Library — библиотека визуальных компонентов) или CLX (Cross-platform Library). Компоненты библиотек организованы в иерархическую структуру. Они служат строительными блоками графического пользовательского интерфейса любого приложения Delphi. Во время выполнения приложения компоненты VCL появляются на экране как элементы управления — кнопки, флажки, списки, поля ввода и т.д. Элементы управления составляют подмножество компонентов VCL: каждый элемент управления является компонентом, но не каждый компонент является элементом управления.

Элемент управления — это графический элемент, который размещается в форме, имеет визуальное отображение и способен обрабатывать сообщения операционной системы Windows. Компонент — это более широкое понятие, которое можно рассматривать как объект, выполняющий определенные функции.

Панель Desktop

Панель рабочего стола (Desktop) содержит список имеющихся раскладок рабочего стола. Можно сохранить текущую раскладку и присвоить ей какое-либо имя. Если одну из раскладок выделить, то при следующем запуске Delphi она будет выведена на экран, и программисту не придется настраивать рабочий стол. С помощью панели рабочего стола программист может манипулировать раскладками рабочего стола, определяющими видимость, размеры, стыковку и расположение окон, а также состав инструментов Delphi.

Панель Internet

Это новая панель помогает создавать страницы для Web и модули для использования безграничных возможностей Internet. В дальнейшем эти возможности будут описаны подробнее.

Форма

В форме (form) размещают все компоненты, которые и будут составлять внутреннее содержание выполняемой программы. В форме можно размещать стандартные компоненты из библиотеки VCL, которая входит в поставку Delphi, или использовать компоненты собственной разработки. На рис. 1.3 показано окно Components (Список компонентов). Чтобы открыть его из главного меню, выберите команду View⇨Component List (Просмотр⇨Список компонентов). В этом окне в алфавитном порядке приведены все компоненты, доступные в текущей версии Delphi и в библиотеке VCL. Большинство компонентов так же отображается на панели компонентов, где они упорядочены по категориям.

Рис. 1.3. Окно списка компонентов

Форма, собственно, и является тем пользовательским интерфейсом, который будет отображаться на экране во время выполнения приложения. Форму можно ассоциировать с окном, которое по умолчанию называется Form1, и ее можно редактировать в процессе разработки, т.е. изменять размер, цвет, размещать на ней компоненты и удалять их из формы и т.д. Для внесения всех этих изменений существует связанный с формой инспектор объектов (об этом поговорим подробнее после знакомства с классами). Пример окна с пустой формой и связанный с ней инспектор объектов показаны на рис. 1.4 и 1.5 соответственно. Если окна формы не видно на экране или вы хотите работать с другой формой, выберите команду View⇨Forms (Просмотр⇨Формы),

или нажмите клавиши <Shift+F12>, или щелкните на пиктограмме формы на панели View, что будет удобнее всего. В окне View Form (Просмотр формы) выделите нужную форму и щелкните на кнопке OK. Если в окне View Form нет ни одной формы, значит, в текущем проекте форм нет. В этом случае при необходимости можно добавить в проект новую форму, выбрав для этого пункты меню File⇨New⇨Form (Файл⇨Новая⇨Форма).

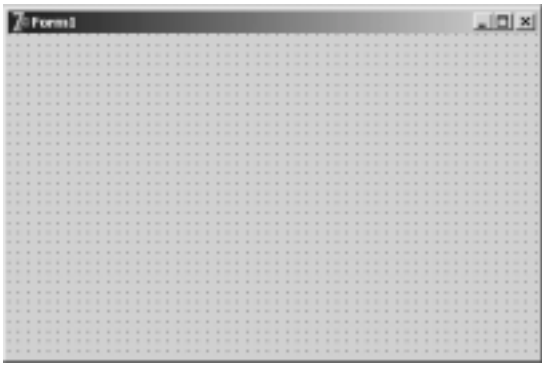

Рис. 1.4. Окно с формой

Класс — это законченный фрагмент программы, в котором объединены поля, определяющие состояние объекта, и методы, определяющие функциональные возможности объекта. Объекты — это экземпляры класса. Образно говоря, класс является шаблоном, или прототипом объекта. На основе одного класса можно создавать сколь угодно много объектов.

Свойства — это управляющие структуры, с помощью которых происходит доступ к полям объекта. То есть изменять состояние объекта можно с помощью свойств.

В среде отладки Delphi используется объектно-ориентированный язык Object Pascal, представляющий собой расширение стандартного Pascal. Все используемые в Delphi компоненты являются объектами. С каждым объектом ассоциирован набор его свойств. Например, форма является объектом с такими свойствами (*Properties*), как `Name` (Имя), `Caption` (Заголовок), `ClientHeight` (Высота), `ClientWidth` (Ширина), `Color` (Цвет) и т.д., что отражено в инспекторе объектов. При создании новой формы Delphi автоматически присваивает каждому свойству значение по умолчанию. При разработке программы свойства компонентов можно изменять с помощью диалогового окна Object Inspector (Инспектор объектов). Для этого нужно всего лишь щелкнуть на нужном свойстве кнопкой мыши и заменить его текущее значение. На рис. 1.5 показан вид окна инспектора объектов для формы Form1. Если окна инспектора объектов на экране не видно, то для его вывода выберите команду View⇨Object Inspector или нажмите клавишу <F11>.

Рис. 1.5. Окно инспектора объектов

Редактор кода

Редактор кода представляет собой полнофункциональный текстовый редактор, с помощью которого можно просматривать и редактировать исходный код программы. Кроме того, редактор кода содержит многочисленные средства, облегчающие создание исходного кода на языке Object Pascal. И хотя редактировать исходный код можно в любом текстовом редакторе, делать это в редакторе кода значительно удобнее, так как встроенные возможности облегчают написание кода, а многочисленные подсказки помогают избежать ошибок. В окне редактора кода каждый модуль кода выводится на отдельной вкладке. Чтобы открыть модуль в редакторе кода, выберите команду View⇨Units (Просмотр⇨Модуль), или нажмите клавиши <Ctrl+F12>, а удобнее всего просто щелкнуть на пиктограмме модуля, расположенной на панели View. Затем в окне View Unit (Просмотр модуля) выделите имя нужного модуля и щелкните на кнопке OK. На рис. 1.6 показано окно редактора кода с двумя вкладками — Unit1 и Project1.

По умолчанию с окном редактора кодов состыковано окно Exploring (Исследователь кода), расположенное слева от редактора кодов (см. рис. 1.6). Можно сделать окно исследователя кодов и самостоятельным окном, для чего необходимо навести указатель мыши на двойную полоску вверху окна и при нажатой левой кнопке мыши перетащить его в любое удобное место. Такие окна, которые могут состыковываться друг с другом, называются “стыкуемые” окна. С помощью исследователя кода программист может легко просматривать файлы модулей. На древовидной диаграмме исследователя кода показаны все типы, классы, свойства, методы, глобальные переменные и глобальные процедуры, определенные в модуле кода, загруженного в данный момент в редактор кода. В исследователе кода перечислены также все модули, используемые текущим модулем редактора кода. Чтобы открыть окно

исследователя кода (если его не видно), необходимо последовательно выбрать пункты меню View⇨Code Explorer (Просмотр⇨Исследователь кода).

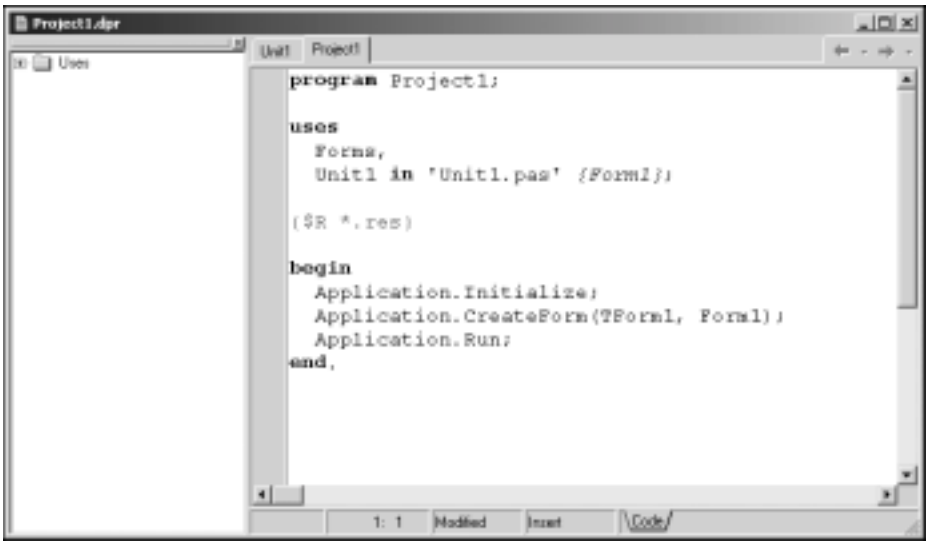

Рис. 1.6. Окно редактора кода с двумя вкладками и состыкованным с ним окном исследователя кодов

Почти все окна, отображаемые на экране во время проектирования, являются стыкуемыми. Для состыковки двух окон необходимо поместить курсор мыши на заголовок окна и при нажатой кнопке мыши переместить его в другое окно. Появляющаяся рамка покажет новое расположение окна. Для расстыковки окон нужно проделать обратную операцию. Удобство состоит в том, что можно объединить несколько окон в одну группу и в дальнейшем обращаться с ней, как с единым окном.

Также по умолчанию с окном редактора кода состыковано окно сообщений. Оно появляется автоматически, если в процессе компиляции программы были сгенерированы сообщения об ошибках или предупреждающие сообщения. Это можно легко проверить, если в стандартный код проекта по умолчанию внести какую-либо ошибку. Например, в разделе `var` исправить `Form1` на `Form2`, то после компиляции, для чего необходимо нажать клавиши <Ctrl+F9>, появится окно сообщений с сообщением об ошибке, а в окне редактора кодов будет подсвечена та строка, где произошла ошибка. Для отображения окна сообщений можно также щелкнуть правой кнопкой мыши на редакторе кода и в контекстном меню выбрать пункт Message View (Просмотр сообщений). Если дважды щелкнуть кнопкой мыши на каком-либо сообщении, то в редакторе кода подсвечиваются строка или строки, породившие сообщение. Щелчок правой кнопкой мыши в окне сообще-

ний активизирует контекстное меню этого окна, с помощью которого можно удалить или сохранить все или отдельные сообщения, а также сделать переход в окно редактора кода на строку с ошибкой. При выборе в главном меню пунктов Search⇨Find in Files... (Поиск⇨Найти в файлах...) в окне сообщений выводятся результаты поиска.

Дерево объектов

Сейчас несколько проясним смысл слова *объект*. Если подходить строго и формализовано, то объект — это экземпляр класса. А класс — это структура, в которой объединены как поля, т.е. данные, и методы — это процедуры и функции, с помощью которых и реализуется логика работы класса. Можно сказать, что поля определяют состояние объекта, а методы определяют функциональные возможности объекта, созданного на основе этого класса. Основная идея заключается в том, чтобы избежать повторяющегося кода, а использовать уже созданные ранее фрагменты кода для выполнения необходимых функций. Поэтому класс можно рассматривать как шаблон, или прототип, на основе которого создаются объекты. В первом приближении объект, созданный на основе какого-либо класса, — это просто место в памяти, где хранятся только данные, связанные с этим объектом, а весь выполняемый код берется из класса. То есть код, который может быть довольно большим, существует в единственном экземпляре. Именно к этому и стремились разработчики идеологии классов. Но основная мощь классов заключается не в этом. Ведь нельзя предусмотреть нужные классы на все случаи жизни. Всегда что-то должно быть другим. Так вот, на основе одного класса можно создавать другие классы, внося только необходимые изменения. Код исходного класса останется неизменным, а созданный на его основе класс добавит только нужные свойства, не дублируя исходного кода. При этом вновь созданный класс будет обладать всеми возможностями исходного класса, или говоря более строго, наследует возможности базового класса. Класс, от которого создан производный класс, часто называют *суперклассом*. Но об этом поговорим подробнее, когда обратимся к основам объектно-ориентированного программирования. Пока объект будем рассматривать в более широком смысле этого слова как отдельный элемент программы, который может выполнять определенные задачи, и свойства которого можно изменять. Объектом является не только само окно формы, но и все размещаемые в нем компоненты: Button (Кнопка), Label (Надпись), Memo (Область просмотра) и др.

Object TreeView (Дерево объектов), так же как и инспектор объектов, необходимо использовать для повышения скорости работы и получения оперативной информации. Щелчок мыши на объекте в окне дерева объектов выделит этот объект в форме и в инспекторе объектов. Для выделения нескольких объектов необходимо щелкнуть на нужных объектах при нажатой клавише

<Shift>. В дереве объектов также можно группировать отдельные объекты, просто перемещая их при нажатой кнопке мыши.

Если в тексте говорится о кнопке мыши, то имеется в виду левая кнопка мыши. Нажатие правой кнопки мыши всегда специально оговаривается.

Инспектор объектов

Object Inspector (Инспектор объектов) является основным инструментом для настройки объектов. Для каждого вновь созданного объекта в окне инспектора объектов создается относящаяся к нему вкладка, к которой можно обратиться, если выбрать в поле выбора, находящегося в верхней части окна, имя объекта. Для этого необходимо щелкнуть на кнопке с изображением небольшой стрелки, указывающей вниз, и в появившемся списке имен выбрать нужный объект.

После того как выбран необходимый объект, можно не только изменять и настраивать все его свойства, но и создавать обработчики событий для данного объекта, которые будут определенным образом реагировать на такие события, как, например, щелчок кнопки мыши, перемещение мыши или нажатие клавиш клавиатуры.

Для этого на каждой страничке инспектора объектов существуют две вкладки: Properties (Свойства) и Events (События). Кратко остановимся на некоторых свойствах для такого объекта, как форма. Для того чтобы создать проект по умолчанию, существует несколько способов.

Первый способ. Выберите последовательно пункты меню File⇨New⇨ Application. Проект по умолчанию с формой и редактором кода будет создан. Это полностью работающее приложение, и результат его работы можно увидеть, если нажать клавишу <F9>.

Второй способ. Щелкните на пиктограмме проекта, расположенной на панели Standart (Стандартная) и в появившемся окне выберите вкладку New, а в ней щелкните на пиктограмме Application (Приложение).

Третий способ. Последовательно выберите пункты меню Project⇨Add New Project... Как и во втором случае, появится окно New Items, где выбирается пиктограмма Application из вкладки New.

Названия панелей не отображаются на экране. Поэтому необходимо запомнить расположение панелей, для чего можно щелкнуть правой кнопкой мыши на поле, где расположены панели, и открыть список всех панелей. Последовательно подключая и отключая отдельные панели, можно увидеть их расположение.

Теперь, когда создано рабочее приложение, можно изменять его параметры, что делается с помощью инспектора объектов.

Если окно формы скрыто за окном редактора кодов, то переместите его на передний план, что также можно сделать несколькими способами.

1. Если некоторые элементы окна формы отображаются на экране, то просто щелкните на них мышью, после чего окно полностью отобразится на экране.
2. На панели обзора (View) щелкните на пиктограмме Toggle Form/Unit (Переключение форм/модулей), что приведет к перестановке отображаемых окон. Нажатие клавиши <F12> приводит к такому же эффекту.
3. Выберите последовательно пункты меню View⇨Forms, в результате чего появится диалоговое окно View Forms, где можно выбрать нужную форму.

Теперь, когда окно формы находится на переднем плане, займемся его редактированием с помощью инспектора объектов. Так как во вновь созданном проекте нет никаких объектов, кроме формы, то именно она и будет отображаться в инспекторе объектов. Выберите вкладку Properties (Свойства) и посмотрите на довольно большой перечень всех свойств. Разумеется, весь перечень не помещается на экране, и для доступа к отдельным свойствам нужно воспользоваться бегунком. Не стоит пугаться такого количества свойств, многие из них интуитивно понятны и они повторяются от объекта к объекту. Поэтому после небольшой практики вы будете хорошо понимать большинство свойств этого списка.

Все свойства расположены в алфавитном порядке, именно так и начнем их рассматривать. Остановимся только на некоторых из них, например Caption (Заголовок). По умолчанию там записано Form1, что и отображается в заголовке окна. Попробуйте изменить это свойство, для чего щелкните на поле, где записано Form1 и введите любое подходящее название, например “Мое Окно”. Одновременно с набором текста будет меняться название окна формы. Дальше идут два свойства ClientHeight и ClientWidht, которые определяют непосредственные размеры окна. Попробуйте ввести другие значения для них и сразу же увидите изменения на экране. Очень важное свойство Enabled (Блокировка). Обычно оно всегда должно быть в состоянии `True`. Попробуйте изменить его на `False` и запустите проект на выполнение (клавиша <F9> или пиктограмма на панели Debug). Эффект будет заметен. Далее есть свойство Icon (Пиктограмма). С помощью его можно изменить пиктограмму, которая будет связана с приложением и будет отображаться рядом с заголовком окна. Пиктограмму можно нарисовать самому или скопировать. Для дополнительного описания функций окна можно использовать свойство Hint (Подсказка). В окно вводится текст, который будет отображаться на экране в виде этикетки при наведении на созданное окно указателя мыши. Но для того, что бы это произошло, нужно разрешить этому тексту отображаться на экране, для чего существует свойство ShowHint (Показать подсказку), которое должно быть установлено в состояние `True`.

Можно изменять размеры и вид рамки (BorderStyle и BorderWidht) или цвет окна (Color). Можно изменить вид курсора (Cursor), который появится при наведении курсора мыши на созданное окно.

Обычно все перечисленные свойства присутствуют в любом объекте, и ими чаще всего приходится пользоваться. Об остальных свойствах поговорим, когда еще раз вернемся к проектированию формы.

Что такое компоненты

Компонент — это объект, который является отдельным строительным блоком программы во время проектирования. Компонент является более широким понятием, чем широко используемое в Windows понятие *элемент управления* (*control*). Как и элемент управления, компонент способен принимать и обрабатывать сообщения Windows.

Простые компоненты

Простыми компонентами называются такие компоненты, которые наиболее часто используются для создания графического интерфейса пользователя. Из них составляют довольные сложные комбинации, которые выделяются в отдельные группы графического интерфейса. По своему внутреннему содержанию они могут быть и не очень простые, поэтому удобнее было бы назвать их *элементарными*, или *базовыми*, но обычно говорят *простые компоненты*. Форму тоже можно рассматривать как компонент, хотя ее пиктограмма и отсутствует в палитре компонентов. Рассмотрим несколько компонентов.

Компонент TLabel

Компонент `TLabel` (Надпись) используется для отображения текста на форме, который нельзя изменять непосредственно через графический интерфейс пользователя, хотя в программе, при необходимости, может быть предусмотрено изменение надписи. Рассмотрим методику работы с надписями. Выполните следующие действия.

1. Создайте новый проект типа `Application`.
2. Поместите надпись в форму. Для этого на палитре компонент выберите вкладку Standart (обычно она открыта по умолчанию) и дважды щелкните кнопкой мыши на пиктограмме надписи (буква **A**). Другой способ размещения надписи в форме — один раз щелкнуть кнопкой мыши на пиктограмме надписи на палитре компонентов, а затем щелкнуть в нужном месте формы. При этом можно, не отпуская кнопки мыши, задать необходимые размеры. Чтобы удалить надпись из формы, выделите ее (щелкните на ней мышью, при этом она выделится черными квадратиками) и нажмите клавишу <Delete>. Также удаление можно выпол-

нить с помощью контекстного меню, если щелкнуть правой кнопкой мыши на объекте. Выделение можно сделать и с помощью дерева объектов, щелкнув на пиктограмме надписи. Чтобы отменить выделение, щелкните кнопкой мыши в любом месте за пределами надписи. Поэкспериментируйте с размещением и удалением надписей.

3. Переместите надпись в другое место формы методом перетаскивания. Для этого установите указатель мыши на надписи, щелкните кнопкой мыши и, удерживая ее нажатой, передвиньте надпись в другое место. Когда надпись займет нужное положение, отпустите кнопку мыши. Обратите внимание на то, что при перетаскивании объекта границы надписи будут привязаны к разметочной сетке формы.
4. Измените свойство надписи Name (Имя) на `MyFirstLabel` (по умолчанию она называлась `Label1`). Для этого в инспекторе объектов щелкните на свойстве Name и введите слово `MyFirstLabel`. Убедитесь, что вы изменяете свойство надписи, а не формы (это типичная ошибка новичков). Для этого надпись в форме должна быть выделена, а в поле выбора в верхней части инспектора объектов должно быть написано `Label1: TLabel` (когда вы измените имя надписи, там будет написано `MyFirstLabel: TLabel`). После ввода нужного имени надписи зафиксируйте его, нажав клавишу <Enter>.
5. Измените саму надпись. Для этого выберите в инспекторе объектов свойство Caption (в данном случае можно перевести как *надпись*), введите новую надпись: “Моя первая надпись!” и нажмите клавишу <Enter>. Введенный текст появится в форме. Обратите внимание, как изменяются границы надписи. Это связано со свойством AutoSize. Если установить свойство AutoSize в состояние `False`, то автоматического изменения границ происходить не будет. Границы можно изменять вручную, для чего выделите объект и наведите курсор мыши на одну из черных меток границы Когда он примет вид двунаправленной стрелки, нажмите кнопку мыши и задайте необходимые размеры.
6. Измените цвет фона надписи. Для этого выберите свойство Color (Цвет), щелкните на стрелке, выберите в раскрывшемся списке желтый цвет и щелкните на нем.
7. Измените шрифт и цвет текста надписи. Для этого выберите свойство Font (Шрифт) и щелкните на трех точках. В окне Font измените шрифт на `Arial`, стиль на `Bold Italic`, а размер — на `20`. В раскрывающемся списке выберите красный цвет и щелкните на кнопке OK.
8. Добавьте к форме еще одну надпись. На этот раз воспользуйтесь другим методом — щелкните на пиктограмме надписи на палитре компонентов, переместите указатель мыши в произвольное место формы и еще раз щелкните кнопкой мыши. При этом в форме в указанном вами месте

должна появиться новая надпись. Можете не отпускать кнопку мыши и задать необходимые размеры.

9. Измените свойство Name новой надписи на `MySecondLabel`, а свойство Caption — на "Моя вторая надпись!".
10. Выберите пункты меню View⇨Alignment Palette (Просмотр⇨Палитра способов выравнивания). Выделите в форме надписи, что можно сделать несколькими способами. В окне дерева объектов при нажатой клавише <Shift> щелкните на тех объектах, которые необходимо выделить, в нашем случае это `MyFirstLabel` и `MySecondLabel`. Или обведите их на форме указателем мыши при нажатой клавише. Используйте различные пункты из палитры способов выравнивания и оцените результат.
11. Теперь выделите форму. Это можно сделать двумя способами: щелкнуть в любом месте формы за пределами надписей, или щелкнуть на пиктограмме формы на панели View. Если форма видна, то первый способ, конечно, удобнее, однако если в проекте есть много форм, причем нужная форма закрыта другими окнами, то более удобен второй способ.
12. Измените свойства формы: свойству Name задайте значение "LabelExample", а свойству Caption — значение "Надпись".
13. Итак, вы создали простое приложение, которое, правда, пока что ничего полезного не делает. Выполните его. Это можно сделать одним из трех способов: щелкнув на пиктограмме Run (Выполнить) на панели отладки, выбрав в главном меню команду Run⇨Run или нажав клавишу <F9>. При этом на экране должна появиться форма, как показано на рис. 1.7.
14. Щелкнув на кнопке закрытия окна в верхнем правом углу, завершите приложение. Это же можно сделать и в среде Delphi. Для этого запустите приложение еще раз, активизируйте любое окно Delphi (однократно щелкнув на нем кнопкой мыши) и выберите Run⇨Program Reset (Выполнить⇨Переустановка программы), или нажмите клавиши <Ctrl+F2>.

Все слова и тексты, которые отображаются на экране, можно вводить на русском языке. Все идентификаторы, которые используются непосредственно в программе, должны использовать только английский алфавит.

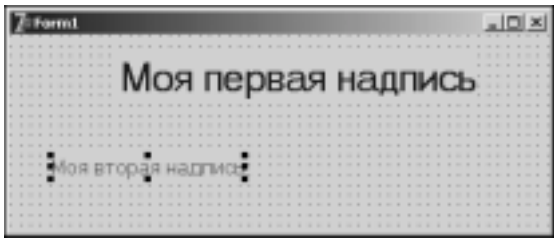

Рис. 1.7. Форма с надписью

Компонент TEdit

В компоненте `TEdit` (Поле ввода) хранится текст, который можно помещать в данный компонент как во время разработки, так и во время выполнения. Текст, видимый в поле ввода, находится в свойстве Text. Свойство MaxLength определяет максимальное количество символов в поле ввода. Если значение свойства MaxLength равно `0`, то количество символов ничем не ограничено. С помощью свойства Font можно устанавливать шрифт текста. Если свойство Readonly (Только чтение) установить в `True`, то во время выполнения программы пользователь не сможет изменять текст поля ввода. Для лучшего понимания работы поля ввода выполните следующее.

1. Создайте новый проект типа `Application`.
2. Разместите поле ввода в форме. Пиктограмма компонента "Поле ввода" находится рядом с пиктограммой компонента "Надпись" на вкладке Standart. Как и в случае надписи, размещение можно сделать несколькими способами.
3. Измените размер поля ввода. Для этого установите указатель мыши на одном из черных квадратиков и, удерживая кнопку мыши нажатой, переместите черный квадратик (а с ним и границу поля ввода) в нужном направлении. Установив необходимый размер, отпустите кнопку мыши.
4. Переместите поле ввода в нужное место методом перетаскивания.
5. Установите значение свойства Name в `MyText`. Для этого в инспекторе объектов щелкните на свойстве Name и введите `MyText`. Как и в случае с надписью, убедитесь, что вы изменяете свойство поля ввода, а не формы. Для этого в заголовке поля выбора в верхней части инспектора объектов должно быть написано `Edit1: TEdit`.
6. Выберите в инспекторе объектов свойство Text и введите его новое значение. Нажав клавишу <Enter>, зафиксируйте введенный текст. Обратите внимание: во время ввода изменяется текст в поле ввода формы.
7. Измените цвет текста в поле ввода на синий. Для этого в инспекторе объектов щелкните на значке (+) рядом со свойством Font. При этом значок (+) изменяется на (–), и появляется список свойств объекта Font. Выберите свойство Color и щелкните на стрелке, расположенной в этом поле. При этом раскрывается список доступных цветов. Найдите в нем синий цвет и щелкните на нем.
8. Выделите форму. Измените значение свойства Name формы на "EditBoxExample", а значение свойства Caption — на "Поле ввода".
9. Нажав клавишу <F9>, запустите разработанную программу. При этом на экране появляется изображение, показанное на рис. 1.8. В отличие от надписи, текст в поле ввода можно изменять, запоминать и извлекать из буфера обмена. Но после установки значения свойства Readonly в "True" изменять содержимое поля ввода через графический интерфейс пользо-

вателя уже будет нельзя. Значение свойства Readonly можно менять из программы, запрещая или разрешая таким образом пользователю вводить данные.

10. Завершите приложение.

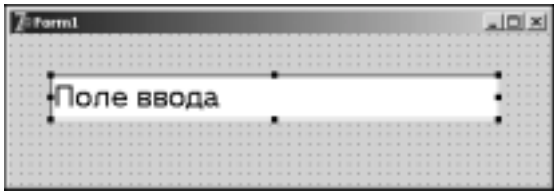

Рис. 1.8. Поле ввода

Компонент TMemo

Компонент `TMemo` (Область просмотра) предназначен для вывода на экран нескольких строк текста. Свойства MaxLength, Font и Readonly области просмотра аналогичны соответствующим свойствам поля ввода. Свойство Text содержит весь текст области просмотра, однако это свойство доступно только по время выполнения. Свойство Lines содержит отдельные строки текста области просмотра, оно доступно как во время разработки, так и во время выполнения. Свойство WordWarp определяет, будут ли переноситься строки, выходящие за пределы области просмотра, или они останутся невидимыми.

Если вместо русского текста на экране появились произвольные символы, то нужно изменить значение свойства Charset (Набор символов) объекта Font (Шрифт). Для большинства шрифтов подходящими значениями свойства Charset являются `DEFAULT_CHARSET` или `RUSSIAN_CHARSET`.

Для лучшего знакомства с областью просмотра выполните следующие действия.

1. Создайте новый проект типа `Application`.

2. Разместите область просмотра в форме, так же как делали ранее для надписи или поля ввода.

3. Установите подходящий размер области просмотра и переместите область просмотра в удобное место.

4. Измените знвчение свойства Name области просмотра на “memSample”, для чего в инспекторе объектов щелкните на свойстве Name и введите “memSample”. Как и в случае надписи или поля ввода, убедитесь, что вы изменили свойство области просмотра, а не формы. В поле выбора в верхней части инспектора объектов должно быть написано `Memo1: TMemo` (после изменения имени там будет `memSample: TMemo`).

5. Выберите свойство Lines и щелкните на кнопке с тремя точками. При этом появляется окно редактора строк String List Editor. Введите текст, показанный на рис. 1.9. Закончив ввод текста, щелкните на кнопке OK.
6. Выделите форму. Для этого щелкните на ней левой кнопкой мыши, или щелкните на имени формы в раскрывающемся списке инспектора объектов. Измените значение свойства Name на "MemoBoxExample", а свойства Caption — на "Область просмотра".
7. Запустите программу на выполнение. На экране должно появиться изображение, показанное на рис. 1.10. Попробуйте вводить тексты различной длины. Попробуйте режимы выделения текста, сохранения и извлечения из буфера обмена.
8. Завершите работу программы.
9. В инспекторе объектов измените значение свойства WordWarp области просмотра memSample на "False", а значение свойства ScrollBars — на "ssBoth" (это свойство определяет наличие или отсутствие полос прокрутки).
10. Нажав клавишу <F9>, запустите программу еще раз. Вводите текст в области просмотра до тех пор, пока он не выйдет за правую границу. Попробуйте также добавлять новые строки, пока они не выйдут за нижнюю границу.
11. Завершите работу программы.
12. Для лучшего понимания работы области просмотра поэкспериментируйте с различными установками свойств WordWarp и ScrollBars.

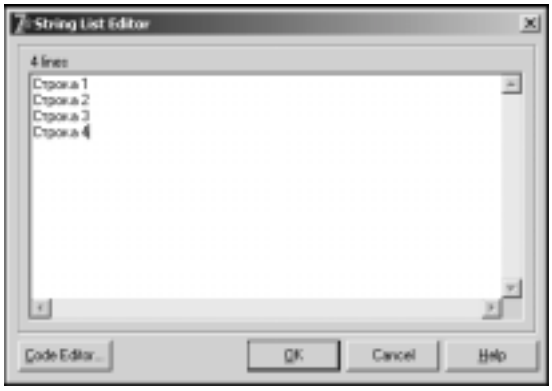

Рис. 1.9. Текст, вводимый в редактор строк

Компонент TButton

Обычно с помощью компонента `TButton` (Кнопка) пользователь инициирует выполнение какого-либо фрагмента кода или целой программы. Други-

ми словами, если щелкнуть на элементе управления TButton, то программа выполняет определенное действие. При этом кнопка принимает такой вид, будто она нажата.

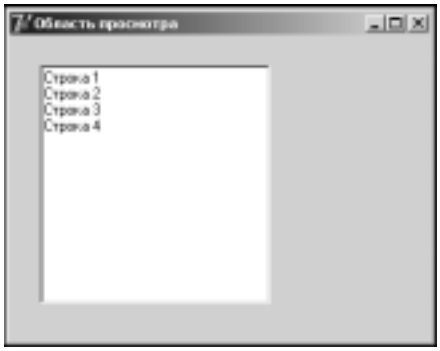

Рис. 1.10. Окно с областью просмотра

Кнопкам можно присваивать комбинации быстрых клавиш. Во время выполнения нажатие такой комбинации клавиш эквивалентно щелчку мыши на кнопке. Выполните следующие действия.

1. Создайте новый проект типа `Application`
2. В инспекторе объектов измените значение свойства формы Name на "ButtonExample", а свойства Caption — на "Кнопка".
3. Поместите кнопку в форму. Пиктограмма кнопки находится на вкладке Standart из палитры компонентов, и она похожа на кнопку OK.
4. Измените значение свойства Name кнопки на "MyButton".
5. Измените значение свойства Caption кнопки на "&Щелкать здесь". Обратите внимание: в надписи на кнопке буква, перед которой стоит символ "`&`", будет подчеркнутой. В данном случае это буква `Щ`. Это означает, что теперь кнопке присвоена комбинация клавиш быстрого вызова <Щ>.
6. Нажав клавишу <F9>, запустите программу. При этом на экране появляется изображение, показанное на рис. 1.11.
7. Щелкните на кнопке. При этом кнопка принимает такой вид, будто она нажата. Пока еще с кнопкой не связан какой-либо код, поэтому никакой реакции на нажатие кнопки не происходит.
8. Завершите работу программы.

Если в названии кнопки, которое отображается на экране, одна из букв подчеркнута, то это значит, что кнопке присвоена комбинация быстрых клавиш. Нажатие клавиши с подчеркнутой буквой приведет к активизации кнопки, аналогично щелчку мыши на ней. Но при этом необходимо учитывать регистр и раскладку клавиатуры.

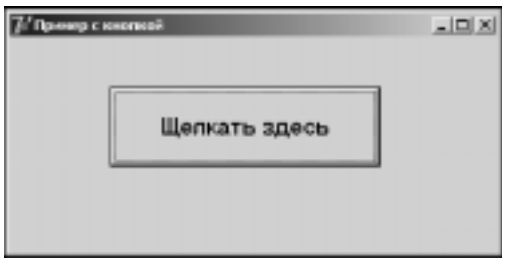

Рис. 1.11. Пример формы с кнопкой

В данном примере клавиатура должна быть переключена на русский язык (ведь буква Щ — русская). Кроме того, должен использоваться верхний регистр (для оперативного получения верхнего регистра необходимо нажать клавишу <Shift> или нажать клавишу <Caps Lock>, при этом постоянно будет включен верхний регистр, о чем говорит подсвеченная лампочка CapsLock).

Что делать, если в названии кнопки должен отображаться символ "&"? Ведь если поместить его в название, то он сделает следующую букву подчеркнутой, а сам виден не будет. Чтобы решить эту проблему, используется следующее правило: символ "&" отображается в названии кнопки, если в свойстве Caption записаны два стоящих подряд символа — "&&". Например, чтобы название кнопки имело вид `This & That`, в свойство Caption необходимо записать `This && That`. При этом никакая комбинация клавиш быстрого вызова кнопке не присваивается.

Не бойтесь экспериментировать с этими компонентами. Попробуйте изменять другие их свойства. Худшее, что можно сделать, — это нарушить рабу Delphi (что крайне маловероятно). Тогда придется всего лишь перезапустить ее заново, а в крайнем случае — переустановить.

Первая программа

Теперь вы уже имеете необходимые знания для того, чтобы создать свою первую программу. Это будет простенькая программа, имеющая интерфейс с двумя кнопками и надписью, которую можно как выводить на экран, так и убирать с экрана. Постарайтесь как можно лучше оформить этот незамысловатый графический интерфейс. Образно можно сказать, что программу встречают по одежке, т.е. по виду интерфейса. Хорошо и гармонично оформленный интерфейс, где интуитивно понятно назначение всех его частей, "внушает доверие" к программе. Понятно, что невозможно только средствами графики выразить все функциональные особенности программы, поэтому должны быть и поясняющие надписи, и появляющиеся этикетки, и файл справки. Только в самых простейших случаях, например, первая программа, можно этого не делать.

Сейчас приступим к разработке первой программы, это не займет много времени.

1. Создайте проект типа `Application`.
2. Разместите на нем две кнопки и надпись. Используя палитру способов выравнивания, или вручную постарайтесь расположить их симметрично, выбрав подходящие размеры.
3. Назовите первую кнопку "Вывод надписи", а вторую — "Очистка" (это свойство Caption).
4. Для надписи выберите размер шрифта 14 пунктов (выше говорилось, как это сделать).
5. Создайте обработчики событий для нажатия кнопки. Этого вы еще не делали, поэтому остановимся на этом подробнее.
6. Для создания обработчика события нажатия кнопки необходимо в инспекторе объектов выбрать вкладку Events (События), затем выбрать событие OnClick (Щелчок). Разумеется, инспектор объектов должен отображать настройки нужного объекта. В нашем случае это будет кнопка `Button1` с названием "Вывод надписи".
7. Щелкните мышкой на появившемся справа от события OnClick белом поле, и Delphi моментально перенесет вас в редактор кода, где сделает заготовку для обработчика события и установит курсор на том месте, где нужно ввести необходимый код.
8. Вручную введите следующий код.

```
Label1.Caption := 'Моя первая программа';
```

9. Проделайте то же самое со второй кнопкой, для которой введите такой код.

```
Label1.Caption := '';
```

10. Запустите программу и пощелкайте на кнопках. На рис. 1.12 показан графический интерфейс созданной программы.
11. Завершите работу программы.

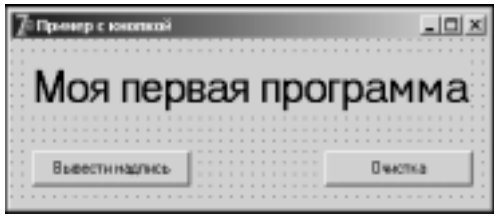

Рис. 1.12. Графический интерфейс первой программы

Полностью листинг вашей первой программы приведен ниже.

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.Caption := 'Моя первая программа';
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Label1.Caption := '';
end;
end.
```

Посмотрев на него, вы увидите имена созданных вами объектов и выводимых надписей. Больше об этом листинге говорить не будем, пока не познакомимся с основами объектно-ориентированного языка Pascal. Хочу только отметить, что несмотря на огромную работу, которую должен проделать компьютер по выводу на экран созданного вами окна, обработке событий (для чего необходимо взаимодействие с системой Windows по передаче сообщений), созданию кода по обработке критических ситуаций и т.д., программа, с которой работает пользователь, очень короткая. В ней показаны только те фрагменты, в которых необходимо делать изменения. Вся остальная часть огромной программы скрыта в недрах Delphi, и вам с ней на первом этапе не

придется работать. Это одно из преимуществ Delphi. В других языках программирования выводится гораздо больше кода, с которым труднее работать.

Теперь несколько слов об оформлении интерфейса. Когда у вас всего три объекта, то нетрудно их располагать вручную, перемещая элементы на нужные места и задавая необходимые размеры. Но когда интерфейс достаточно загружен, и при этом установлена достаточно мелкая сетка на форме, то удобнее пользоваться палитрой способов выравнивания. Размеры сетки, как и многие другие настройки среды Delphi, можно задать в окне Environment Options, которое можно открыть, используя пункты меню Tools⇨Environment Options... В окне выбирается вкладка Designer, на которой находится настройка Grid Size, с помощью которой и устанавливаются размеры сетки для формы.

В этой же группе находятся такие настройки, как Display Drid (Показать сетку) и Snap to Grid (Привязка к сетке). Привязка к сетке означает, что границы всех объектов будут проходить только по сетке, что удобно для разработки аккуратно смотрящихся интерфейсов (рис. 1.13).

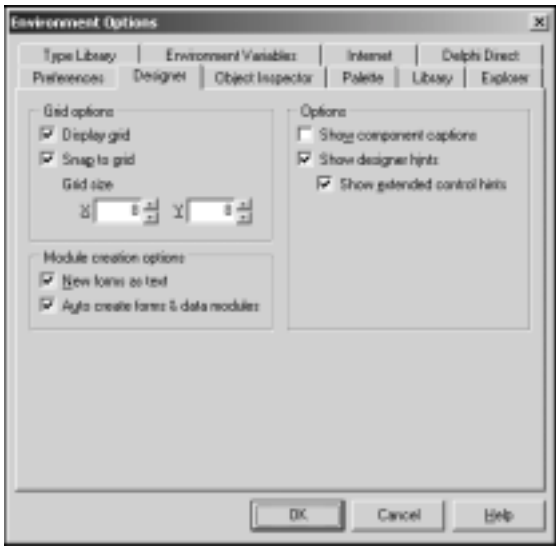

Рис. 1.13. Окно настроек Environment Options

Об остальных настройках поговорим в тех случаях, когда будем обсуждать соответствующие темы. Хотя можете и поэкспериментировать, многие из них интуитивно понятны и не требуют дополнительного пояснения. После экспериментов нужно будет вернуть все установки в исходное состояние, так как в дальнейшем при описании работы Delphi предполагается, что все установки сделаны по умолчанию.

И наконец, разберемся, что же делать с разработанной программой. Она нам может пригодиться в дальнейшем, поэтому сохраним ее. Для этого выберите пункты меню File⇨Save Project as... и сохраните проект и исходные фай-

лы в отдельном каталоге (я предполагаю, что вы знакомы с Windows и знаете, как это сделать). При этом сохранятся файл проекта (расширение .dpr), файл формы (расширение .dfm), исходный файл (расширение .pas) и несколько других нужных для проекта файлов. После того как вы запустите проект, в этом каталоге появятся файлы с расширением .exe и .dcu. Это будут выполняемый файл и файл, созданный компилятором. Можете запустить выполняемый файл отдельно и убедиться, что он работает так же, как и в среде Delphi.

Подсказки в Delphi

Справочная система

Система Delphi содержит гипертекстовую справочную систему, с помощью которой программист может легко и быстро получить необходимую информацию о среде разработки Delphi и объектно-ориентированном языке Object Pascal. Для активизации справочной системы выберите команду Help⇨Delphi Help (Справка⇨Справка Delphi), или щелкните на пиктограмме Help contents (Содержание справочной системы) на пользовательской панели инструментов. При этом появится окно справочной системы Delphi, показанное на рис. 1.2.

Кроме того, в справочной системе Delphi есть информация об инструментах разработки Delphi (пункты меню Help⇨Delphi Tools) и о программировании в среде Windows (пункты меню Help⇨Windows SDK). К сожалению, перевода справочной системы Delphi на русский язык пока что нет. Однако, чтобы пользоваться ею, совсем не обязательно владеть английским языком в совершенстве. Даже если вы слабо знаете английский язык, все равно справочная система во многих случаях будет вам полезной.

Подсказки редактора кодов

Во-первых, выделение ключевых слов в редакторе кода уже является хорошей подсказкой. Так что если вы набираете ключевое слово, а оно не выделилось жирным шрифтом, значит, оно набрано с ошибкой. Наличие отступов и выделение отдельных фрагментов кода также необходимо для лучшего восприятия программы.

Во-вторых, несмотря на наличие дерева объектов и инспектора объектов, где можно получить подробную информацию об объектах, в редакторе кодов есть тоже достаточно много средств, помогающих получить информацию об объектах или найти описание некоторого объекта. Например, щелкните правой кнопкой мыши на поле редактора кода и в появившемся контекстном меню выберите пункт Browse Symbol at Cursor (Исследовать идентификатор под курсором). Введите имя нужного объекта и получите его местоположе-

ние — имя модуля и номер строки. Щелкнув еще раз на имени, перейдите непосредственно в этот модуль.

Подсказки могут происходить и непосредственно во время написания кода. Например, когда вы вводили код для вашей первой программы, то обратили внимание на появляющийся перечень всех доступных методов и свойств для данного объекта после того, как поставили точку после имени объекта. Если не обратили на это внимание, то повторите ввод еще раз и сделайте задержку после того, как набрали “Label1.”. Появится перечень доступных свойств и методов.

Об остальных подсказках поговорим в дальнейшем, сейчас необходимо уяснить, что подсказок достаточно много, и они приносят ощутимую пользу. Еще раз вернемся к простейшим программам и создадим несколько полезных приложений.

Примеры создания простейших программ без написания кода

Сначала создадим календарь, разработка которого займет не более минуты.

1. Создайте проект типа `Application`.
2. Разместите на нем компонент MonthCalendar (Календарь), который находится на вкладке Win32 из палитры компонент.
3. Разместите календарь в углу формы и установите подходящие размеры формы (рис. 1.14).
4. Запустите программу и попробуйте работу компонента MonthCalendar.

Завершите работу программы, сохраните ее и можете ею пользоваться.

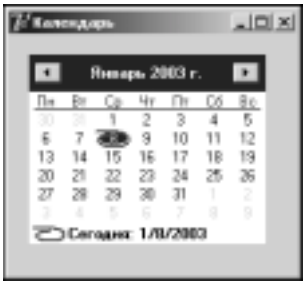

Рис. 1.14. Интерфейс программы “Календарь”

Теперь создайте простейший навигатор Windows (Просмотр каталогов), для чего используйте компонент ShellTreeView (Просмотр каталогов), который находится на вкладке Samples (Образцы). Сделайте все то же самое, как и при создании календаря, и получите навигатор Windows (рис. 1.15).

Рис. 1.15. Интерфейс программы "Навигатор"

В заключение еще раз хочу подчеркнуть преимущества Delphi. Вы уже могли убедиться, что с помощью нескольких щелчком мышью можно создать довольно сложное приложение, такое как календарь или навигатор. На разработку подобного приложения "с нуля" уйдет достаточно много времени. В Delphi таких заготовок существует достаточно много и на все случаи жизни. Можно очень быстро создать доступ к базам данных или использовать уже готовый модуль для прогулок по Internet. Наличие готовых модулей позволяет быстро создавать распределенные системы и обеспечить работу целого предприятия.

В дальнейшем мы познакомимся в некоторыми из таких модулей и будем создавать программы, складывая их из отдельных компонентов, как из кубиков. Но чтобы объединить работу отдельных компонентов, приходится вручную писать необходимый код. Для того чтобы это делать, а самое главное, чтобы самому разрабатывать отдельные компоненты, необходимо знать язык программирования Pascal, на котором написана и сама Delphi. Поэтому все программы, разработанные в Delphi, хорошо интегрируются со средой Delphi. В следующей главе займемся изучением языка программирования Pascal.

Резюме

В данной главе произошло ваше первое знакомство со средой программирования Delphi. Вы познакомились с главным окном и почувствовали, что создавать графический интерфейс пользователя с помощью Delphi не такая уж и сложная задача. Большую часть работы среда программирования делает за вас, необходимо только грамотно указывать ей, что нужно сделать. Наверное, пока не все понятно, но мы будем возвращаться к этим вопросам на протяжении всей книги.

Контрольные вопросы

Как уже говорилось, среди свойств, используемых в редакторе объектов, есть свойства, задающие размеры окна и положение окна. Это свойства Heigh

(Ширина), Wight (Высота), свойства ClientHeigh, ClientWight (Ширина, Высота клиентской области) и свойства Top, Left (Верхний, Левый), задающие положение верхнего левого угла окна. Размеры окна несколько больше размеров клиентской области, так как сюда входят заголовок и рамка. Все значения устанавливаются в пикселях (точки экрана). Обычно большинство современных экранов имеет разрешение 800*600 пикселей, поэтому прежде, чем сделать проверку, попробуйте ответить на следующие вопросы.

1. Где будут размещаться окна со следующими значениями свойств.

```
а) Heigh = 100; Wight = 150; Top = 0; Left = 0
б) Heigh = 100; Wight = 150; Top = 500; Left = 650
в) Heigh = 100; Wight = 150; Top = 225; Left = 365
```

2. Напишите формулы для установки значений свойств Top и Left (при известных значениях Heigh и Wight) таким образом, чтобы окно размещалось точно в центре экрана. Эти формулы будут использоваться довольно часто.
3. Разместите на форме кнопку размером `100*20`, которая должна быть расположена точно в середине окна и иметь размеры `400*300`. Все размеры для кнопки задаются относительно создаваемого окна, а не относительно экрана. Какие значения должны иметь свойства кнопки Top и Left?
4. Установите размеры шрифта для кнопки из предыдущего вопроса в `20` пунктов. В результате высота шрифта будет совпадать с высотой кнопки (`20` пикселей). Проведите небольшой эксперимент, меняя размеры и вид шрифта и кнопки, и ответьте: это случайное совпадение или нет?