Глава 1

Знакомство со средой программирования

В этой главе...

- Среда программирования
- Компоненты
- Первая программа
- Подсказки в Delphi
- Примеры программ без написания кода
- Резюме
- Контрольные вопросы

Термин “среда программирования” подразумевает несколько больше, чем только графический интерфейс пользователя, который вы видите при запуске Delphi. Среда программирования предоставляет широкие возможности по разработке, отладке и контрольному запуску программ, при этом предоставляется удобный графический интерфейс пользователя, набор необходимых инструментов и дополнительных компонент, большое количество подсказок, файлы справки и еще много разных 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.

Среда программирования

Среда программирования — это комплекс программ, которые разработаны для того, чтобы создать удобное окружение для реализации концепции быстрой разработки приложений RAD. Среду программирования можно разбить на несколько отдельных частей, среди которых можно выделить графический интерфейс пользователя. Его будем называть рабочим столом Delphi (по аналогии с рабочим столом Windows), и он появляется сразу же после запуска Delphi (рис. 1.1). Затем появляется окно справочной системы (рис. 1.2) и набор вспомогательных окон, которых может быть очень много. Такие из них, как Object Inspector (Инспектор объектов) или Project Manager (Менеджер проектов) располагаются на рабочем столе, а некоторые из вспомогательных окон (например графический редактор) являются самостоятельными компонентами, без которых можно и обойтись.

Рис. 1.1. Рабочий стол Delphi

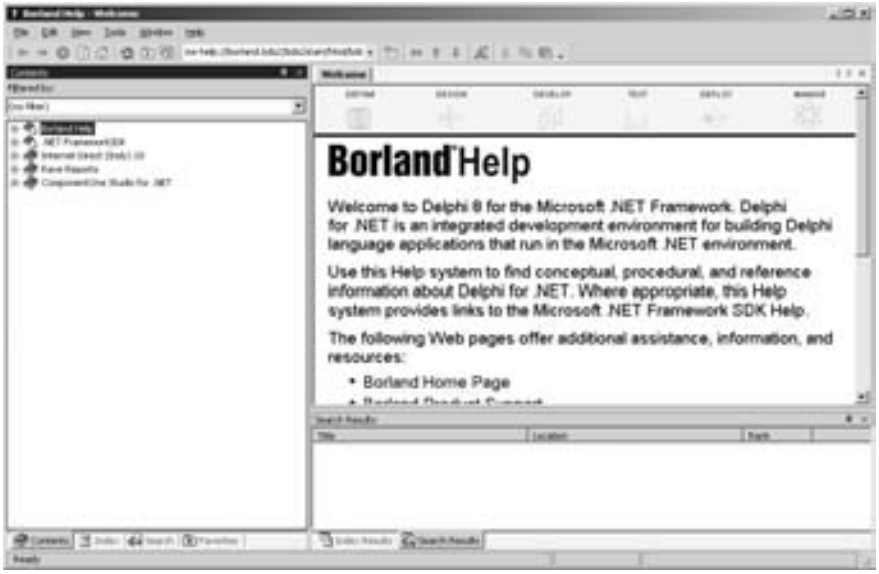

Рис. 1.2. Окно справочной системы

От того, насколько тщательно разработаны графические интерфейсы, зависит удобство использования всей среды разработки. Непосредственно разработка проекта происходит в главном, или рабочем окне, и наиболее часто на первом этапе будут использоваться проекты типа VCL Forms Application (VCL-приложение), Console Application (Консольное приложение) или Library (Библиотека). Продолжим знакомство с рабочим столом, который появляется по умолчанию при запуске Delphi 8.

Рабочий стол

Чтобы начать работу с Delphi, необходимо дважды щелкнуть кнопкой мыши на пиктограмме Delphi на рабочем столе Windows или в меню выбора программ. При этом на экране сначала появится рекламная заставка, сообщающая о том, что среда разработки загружается в оперативную память компьютера. После загрузки Delphi появится рабочий стол Delphi с отображением в главном окне страницы Welcome Page. Эту страницу, как в все другие используемые страницы, можно удалить, если щелкнуть правой кнопкой мыши на корешке вверху окна и в появившемся раскрывающемся меню выбрать команду Close Page. В верхней части рабочего стола находится главное меню с кнопками, на которых перечислены названия команд, и дополнительные панели меню с пиктограммами, где размещаются все основные команды, необходимые при разработке программ. Названия дополнительных панелей меню можно увидеть, если щелкнуть правой кнопкой мыши на поле, где размещаются панели. В появившемся ниспадающем списке будут перечислены следующие пункты: Standard (Стандартная), Debug (Отладка), Desktop (Рабочий стол), Custom (Обычная), Align (Выравнивание), Spacing (Заполнение), Position (Расположение), Browser (Браузер), HTML Design (HTML-эскиз), HTML Format (HTML-формат), HTML Table (HTML-таблица), View (Просмотр), Customize (Настройка). Причем пункт Customize представляет не отдельную панель, а окно настройки. Тот же эффект можно получить, используя команду View⇨ToolBars (Просмотр⇨Панели). Установленная галочка около пункта говорит о том, что соответствующая панель отображается на экране. Галочки устанавливаются простым щелчком мыши. Ниже будет рассказано о каждой из этих панелей.

С левой и правой сторон главного окна находятся окна Object Inspector, Project Manager и Tool Palette (Палитра инструментов).

Для начала работы необходимо выбрать тип приложения, которое будет создаваться. На панели главного меню надо выбрать команду File⇨New и в появившемся списке выбрать команду VCL Forms Application, при этом Delphi автоматически создаст новый проект (рис. 1.3). Это будет уже полностью работающее приложение (т.е. можно будет произвести компиляцию и получить выполняемый файл с расширением `.exe`), но пока не имеющее никаких полезных функций.

Если не производилось дополнительной настройки, то по умолчанию на переднем плане будет отображаться страница с именем Unit1 (Модуль 1), на которой будет располагаться окно, названное Form1 (Форма 1). Эта страница имеет две вкладки — Code (Код) и Design (Эскиз), для переключения между которыми можно использовать корешки с соответствующими названиями, расположенные внизу окна. На вкладке Code будет отображаться рабочий код программы, а на вкладке Design — создаваемый интерфейс пользователя. Вкладку Code обычно называют редактором кодов, а вкладку Design — проектировщиком форм, или формой. Вся страница представляет один программный модуль, который содержит в себе все необходимые конструкции для независимой работы. То есть реализуется концепция модульного программирования. Все необходимые процедуры и функции нужно будет добавить в процессе разработки нового приложения. В среде Delphi предусмотрены различные типы проектов, но в данной книге основное внимание будет уделено проектам типа VCL Forms Application.

В процессе работы необходимо открывать и закрывать различные модули в зависимости от того, с каким фрагментом программы нужно работать. Для того чтобы добавить страницу с необходимым модулем в проект, необходимо в главном меню выбрать пиктограмму с изображением раскрытой папки и знаком “+”, после чего в появившемся окне Add To Project выбрать соответствующий файл. Для удаления модуля из проекта выбирается пиктограмма с изображением открытой папки и знака “–”. Для того чтобы закрыть страницу с соответствующим модулем, не удаляя его из проекта, поместите страницу на передний план и нажмите клавиши <Ctrl+F4>, или щелкните правой кнопкой мыши на корешке страницы и выберите соответствующую команду.

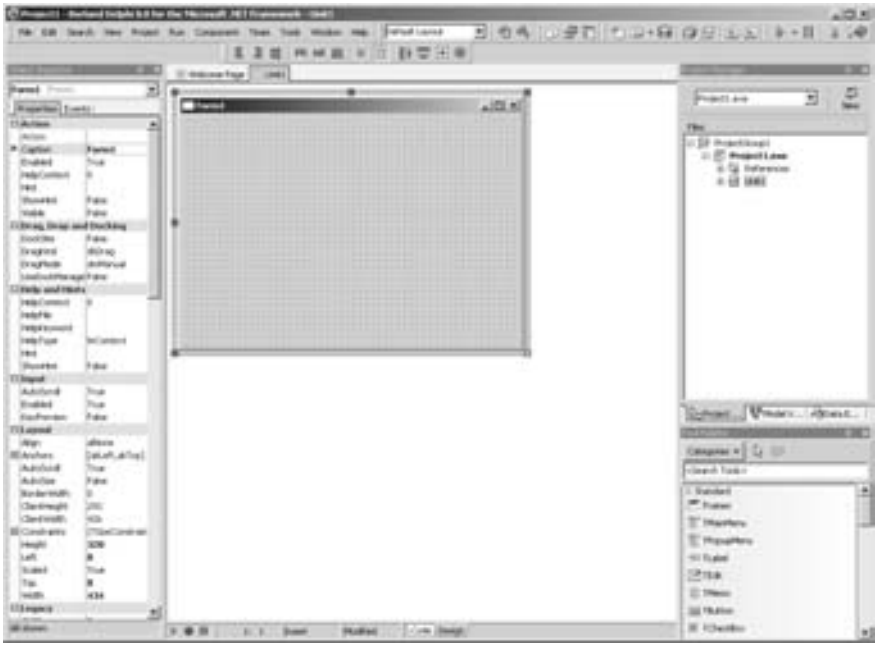

Рис. 1.3. Проект, созданный по умолчанию

Для открытия файла можно выбрать команду File⇨Open. Аналогичный эффект можно получить и при щелчке мыши на пиктограмме открывающейся папки на панели Standard. Когда открывается файл, который связан с формой, нет необходимости дополнительно открывать форму, это делается автоматически.

Для повторного открытия файла можно использовать команды File⇨Reopen. Аналогичный эффект можно получить и при щелчке мыши на стрелке рядом с пиктограммой открывающейся папки на панели Standard.

Команда Save (Сохранить) находится в меню File, как и все остальные команды для работы с файлами. Для сохранения работы можно выбрать команды Save, Save As... (Сохранить как...), Save Project As... (Сохранить проект как...) или Save All (Сохранить все). Команда Save используется только для сохранения файла из активного окна, находящегося на переднем плане. При выборе команды Save As можно сохранить файл из активного окна с новым именем. Команда Save Project As предназначена для сохранения проекта с другим именем, а с помощью команды Save All сохраняются все файлы проекта. Когда сохраняется проект, то создается файл с асширением `.dpr`, который используется при компиляции программы.

Используя пункты главного меню (см. рис. 1.1), можно выполнить все задачи, которые необходимы при разработке проекта. Для вызова некоторых задач удобно использовать 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 меню

Появившийся после запуска рабочий стол Delphi представляет собой набор отдельных панелей и окон, которые наиболее часто используются при разработке приложений. Для того чтобы убрать или отобразить какую-то из панелей меню, щелкни-

те правой кнопкой мыши на поле, где расположены панели, после чего отобразится ниспадающее меню с перечнем всех панелей меню. Тот же эффект можно получить, выбрав команду View⇨ToolBars (Просмотр⇨Панели). Для лучшего ознакомления с панелями меню изучите все пункты меню, последовательно устанавливая или сбрасывая флажки и наблюдая происходящие на экране изменения.

Окна Object Inspector, Project Manager и Tool Palette в этом меню не присутствуют, и для их отображения необходимо выбрать соответствующую команду в меню View. Также можно использовать клавиши быстрого вызова, комбинации которых обычно указываются в пунктах меню. Для инспектора объектов и менеджера проектов это будут клавиши <F11> и <Ctrl+Alt+F11> соответственно. Открывать страницы модулей также можно через команды меню View или с помощью клавиш <Ctrl+F12> и <Shift+F12>.

Чтобы узнать назначение любой пиктограммы, нужно на некоторое время остановить на ней указатель мыши, после чего рядом с указателем появится подсказка, напоминающая о назначении пиктограммы.

Обращаться к пунктам меню можно также с помощью клавиатуры. Например, нажатие клавиши <Alt> приводит к переключению между главным меню и активным окном. После активизации главного меню можно перемещаться по его пунктам с помощью клавиш со стрелками, при этом будет подсвечиваться очередной пункт, а некоторые буквы в названиях пунктов будут подчеркнуты. Нажатие соответствующей буквы на клавиатуре приведет к выполнению команды для этого пункта меню. Например, чтобы обратиться к пункту главного меню View, нужно с помощью клавиши <Alt> перейти к главному меню и затем 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.

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 значительно ускорите процесс проектирования.

Панель Standard

Стандартная панель меню 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 просмотра содержит пиктограммы, предназначенные для создания новых модулей, просмотра форм и редакторов кода, а также для переключения между формой и редактором кода. С помощью панели просмотра можно быстро сделать видимыми разные окна среды разработки Delphi.

Панель Debug

Панель отладки 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

Обычная панель меню по умолчанию содержит только пиктограмму вызова справочной системы Delphi. Через нее удобнее вызывать справочную систему, чем через

главное меню. Эту панель меню, как и все другие панели, можно настроить для отображения других пиктограмм с помощью окна Customize, вызываемого через соответствующую команду.

Панель Desktop

Панель рабочего стола 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. В процессе работы программист может манипулировать окнами и панелями рабочего стола, определяющими видимость, размеры, стыковку и расположение окон, а также состав инструментов Delphi, настраивая их наиболее удобным для себя образом, и сохранять настройки с помощью панели рабочего стола.

Панель Align

Панель элементов выравнивания содержит команды для симметричного упорядочения компонентов, расположенных в форме.

Панель Spacing

Панель элементов заполнения содержит команды для установки пространства между компонентами, расположенными в форме.

Панель Position

Панель элементов расположения содержит команды для расположения и взаимного перекрытия компонентов, размещенных в форме.

Панель Browser

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

Панель HTML Design

Панель содержит команды, необходимые при разработке HTML-страниц.

Панель HTML Format

Панель содержит команды, необходимые при форматировании HTML-страниц.

Панель HTML Table

Панель содержит команды, необходимые при разработке таблиц для HTML-страниц.

Окна рабочего стола

По умолчанию на рабочем столе Delphi располагаются помимо главного окна еще три окна: Object Inspector (Инспектор объектов), Project Manager (Менеджер проектов) и Tool Palette (Палитра компонентов). Эти окна являются “плавающими”, т.е. их можно отделить от того места, где они находятся, и разместить в любом другом месте. Для этого нужно навести курсор мыши на панель заголовка окна, нажать левую кнопку мыши и перетащить окно в нужное место. Появляющаяся рамка будет указывать новое расположение окна. На панели заголовка также находятся кнопки закрытия окна с изображением крестика и кнопка свертывания окна с изображением канцелярской кнопки. Если щелкнуть на кнопке свертывания, то окно исчезнет с экрана и от него останется только корешок на краю рабочего стола. Если навести на этот ко-

решок курсор мыши, то окно развернется и можно будет выбрать необходимый элемент окна. Если убрать курсор с окна, то окно опять свернется. Если еще раз щелкнуть на пиктограмме канцелярской кнопки, то окно вернется на свое место. Это очень удобно, когда необходимо иметь как можно больше рабочего пространства.

Окна рабочего стола рассмотрим далее, а сейчас остановимся на главном окне.

Главное окно

Это окно ограничивает пространство, в котором происходит разработка проекта, создается графический интерфейс пользователя и находится рабочий код программы. В главное окно помещаются страницы с модулями, которые обычно состоят из двух вкладок — Code и Design. Вкладка Design содержит графическое изображение создаваемой формы.

Форма

В форме 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. Все эти компоненты приведены в окне Tool Palette. Форма, собственно, и является тем пользовательским интерфейсом, который будет отображаться на экране во время выполнения приложения. Форму можно ассоциировать с создаваемым 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 связанное с формой окно Object Inspector (Инспектор объектов), но об этом поговорим подробнее после знакомства с классами. Пример окна с пустой формой и связанный с ней инспектор объектов показаны на рис. 1.4. Если окна формы не видно

Рис. 1.4. Окно с формой и окно инспектора объектов

Рис. 1.5. Окно Object Inspector

на экране или вы хотите работать с другой формой, выберите команду главного меню View⇨Forms (Просмотр⇨ Формы) или нажмите клавиши <Shift+F12> либо щелкните на пиктограмме View Form на панели View, что будет удобнее всего. В окне View Form (Просмотр форм) 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 команду главного меню File⇨New⇨VCL Form (Файл⇨Новая⇨ VCL-форма).

В среде отладки Delphi используется объектно-ориентированный язык Delphi, 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), Height (Высота), Width (Ширина), Color (Цвет) и множеством других свойств, большинство из которых отражены в окне инспектора 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. Для этого нужно всего лишь щелкнуть на нужном свойстве кнопкой мыши и заменить его текущее значение. На рис. 1.5 показан вид окна Object Inspector для формы Form1. Если окна инспектора объектов на экране не видно, то для его отображения выберите команду View⇨Object Inspector или нажмите клавишу <F11>.

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. Таким образом, изменять состояние объекта можно с помощью свойств.

Редактор кода

Редактор кода представляет собой полнофункциональный текстовый редактор, с омощью которого можно просматривать и редактировать исходный код программы. Кроме того, редактор кода содержит многочисленные средства, облегчающие создание исходного кода на языке Delphi. 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 главном окне каждый модуль отображается на отдельной странице, где на соответствующей вкладке находится редактор кода. Чтобы открыть модуль, выберите команду View⇨Units... (Просмотр⇨Модули...) 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 показана вкладка с редактором кодов.

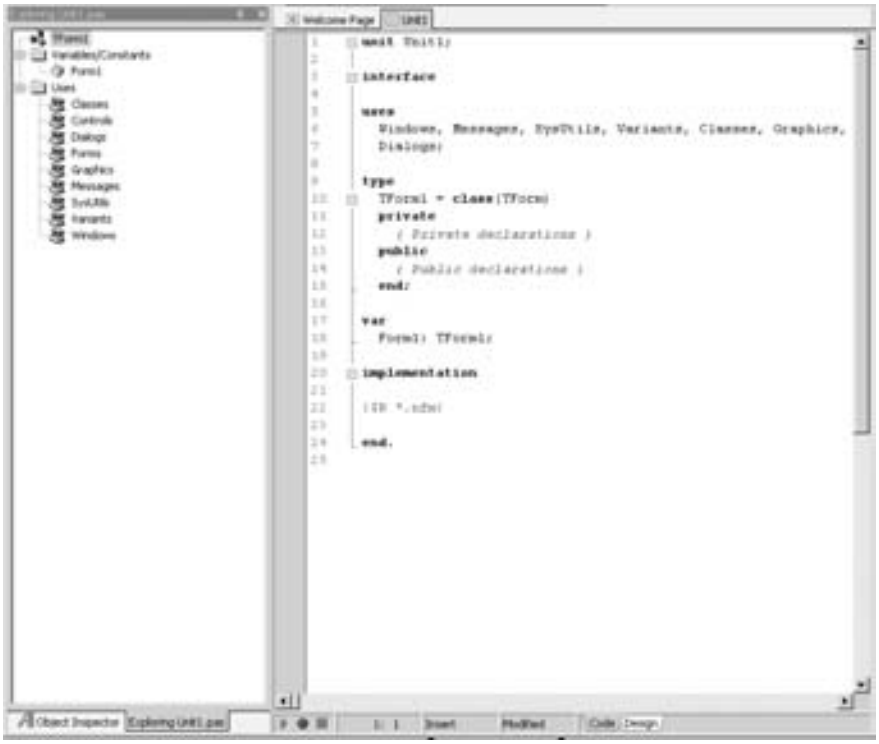

Рис. 1.6. Редактор кода и исследователь кода

Совместно с редактором кода удобно использовать окно Exploring, которое можно открыть с помощью команды главного меню View⇨Code Explorer (Просмотр⇨ Исследователь кода). 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, с которым происходит работа. В исследователе кода перечислены также все модули, подключенные к текущему модулю.

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.

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, если в разделе `var` исправить `Form1` на `Form2`, то после компиляции, для чего необходимо нажать клавиши <Ctrl+F9>, появится окно сообщений с информацией 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 поле редактора кода и в контекстном меню выбрать пункт Message View (Просмотр сообщений). Если дважды щелкнуть кнопкой мыши на каком-либо сообщении, то в редакторе кода подсвечивается строка кода, породившая сообщение. Щелчок правой кнопкой мыши в окне сообщений 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 команды Search⇨Find in Files... (Поиск⇨Найти в файлах...) в окне сообщений выводятся результаты поиска.

Окно Tool Palette

Палитра компонентов представляет собой окно с перечнем доступных компонентов. В этом окне отображаются все компоненты библиотеки VCL. Компоненты библиотеки организованы в иерархическую структуру. Они служат строительными блоками графического пользовательского интерфейса любого приложения Delphi. Во время выполнения приложения компоненты VCL отображаются 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.

Объекты

Сейчас несколько проясним смысл слова объект. Строго говоря, объект — это экземпляр класса. А класс — это структура, в которой объединены поля (т.е. данные) и методы (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 и 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. Таким образом, 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. 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, будет наследовать возможности базового класса. Класс, от которого непосредственно создан производный класс, называют базовым классом, а иногда суперклассом. Но об этом поговорим подробнее, когда обратимся к основам объектно-ориентированного программирования. А 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.

Отображаемое в окне Exploring дерево объектов, так же как и инспектор объектов, необходимо использовать для повышения скорости работы и получения оперативной информации. Щелчок мыши на объекте выделит этот объект в форме, коде и инспекторе объектов.

Инспектор объектов

Вкладки окна Object Inspector автоматически создаются для каждого вновь созданного в форме объекта и являются основным инструментом для настройки объектов. Необходимая вкладка выбирается из списка объектов, находящегося в верхней части окна, по имени объекта. Для этого необходимо щелкнуть на кнопке с изображением небольшой стрелки, указывающей вниз, и в появившемся списке выбрать нужный объект.

После того как необходимый объект выбран, 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.

При этом для каждого объекта в инспекторе 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 команду меню File⇨New⇨VCL Forms 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 New Items, расположенной на панели Standard (Стандартная), и в появившемся окне выберите пиктограмму VCL Forms Application (VCL-приложение).
- Третий способ. Выберите команду меню Project⇨Add New Project... и в появившемся окне выберите пиктограмму VCL Forms Application (VCL-приложение).

Теперь, когда создано рабочее приложение, можно изменять его параметры, что делается с помощью инспектора объектов. Если панель формы скрыта за окном редактора кодов, то переместите ее на передний план, что также можно сделать несколькими способами.

- Если некоторые элементы формы отображаются на экране, то просто щелкните на них мышью, после чего окно полностью отобразится на экране.
- На панели View щелкните на пиктограмме Show Designer (Показать эскиз), что приведет к перестановке отображаемых окон. Нажатие клавиши <F12> приводит к такому же эффекту.
- Выберите команду меню View⇨Forms..., в результате чего появится диалоговое окно View Forms, где можно выбрать нужную форму.

Теперь, когда форма находится на переднем плане, займемся ее 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 ползунком. Не стоит пугаться такого количества свойств, многие из них интуитивно понятны и они повторяются от объекта к объекту. Поэтому после небольшой практики вы хорошо освоите большинство свойств из этого списка.

Все свойства распределены по группам, в которых они расположены в алфавитном порядке. Остановимся только на некоторых из них, например Caption (Заголовок) из

группы Action (Поведение). По умолчанию там будет 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 строку "Мое Окно". Одновременно с набором текста будет меняться название окна формы. Или возьмем, к примеру, два свойства — Height (Высота) и Width (Ширина), которые определяют непосредственные размеры окна. Попробуйте ввести другие значения для них и сразу же увидите изменения на экране. Очень важное свойство Enabled (Доступность). Обычно оно всегда должно быть в состоянии `True`. Попробуйте изменить его на `False` и запустите проект на выполнение (клавиша <F9> или пиктограмма Run на панели Debug). Эффект будет заметен, так как вы не сможете обратиться к окну — оно будет заблокировано. Это удобно использовать в программе, когда необходимо на время заблокировать отдельные элементы интерфейса и запретить к ним доступ пользователя. Далее, с помощью свойства Icon (Пиктограмма) можно изменить пиктограмму, которая будет связана с приложением и будет отображаться рядом с заголовком окна. Пиктограмму можно нарисовать самому или скопировать. Для напоминания функций окна можно использовать свойство Hint (Подсказка). В поле для свойства Hint вводится текст, который будет отображаться на экране как подсказка при наведении на созданное окно указателя мыши. Но для того, чтобы это произошло, нужно разрешить тексту отображаться на экране, для чего существует свойство ShowHint (Показать подсказку), которое должно быть установлено в состояние `True`.

Можно изменять статус окна и вид рамки (свойства BorderStyle и BorderWidth) 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. Остальные свойства рассмотрим, когда еще раз вернемся к проектированию формы.

Компоненты

Компонент — это объект, представляющий собой отдельный строительный блок программы во время проектирования. Компонент является более широким понятием, чем используемое в Windows понятие *элемент управления*. Как и элемент управления, компонент способен принимать и обрабатывать сообщения Windows.

Стандартные компоненты

Стандартными компонентами называются такие компоненты, которые наиболее часто используются для создания графического интерфейса пользователя. Из них составляют довольно сложные комбинации, которые выделяются в отдельные группы графического интерфейса. По своему внутреннему содержанию они могут быть довольно сложными. Форму тоже можно рассматривать как компонент, хотя ее пиктограмма отсутствует в палитре компонентов. Стандартные компоненты выделены в отдельную группу в окне Tool Palette. Рассмотрим несколько стандартных компонентов.

Компонент TLabel

Компонент TLabel (Надпись) используется для отображения текста в форме, причем этот текст нельзя изменять непосредственно через графический интерфейс пользователя, хотя в программе при необходимости можно предусмотреть изменение надписи. Рассмотрим методику работы с надписями. Выполните следующие действия.

1. Создайте новый проект типа VCL Forms Application.
2. Поместите надпись в форму. Для этого в окне Tool Palette найдите группу Standard и дважды щелкните кнопкой мыши на компоненте TLabel. Можно

щелкнуть один раз, а затем щелкнуть в нужном месте формы. При этом сразу можно задать необходимые размеры, наводя указатель мыши на ограничивающие квадраты и перемещая их при нажатой левой кнопке мыши на нужное расстояние. Чтобы удалить надпись из формы, выделите ее (щелкните на ней мышью, при этом она выделится черными квадратиками) и нажмите клавишу <Delete>. Удаление можно выполнить также с помощью контекстного меню, если щелкнуть правой кнопкой мыши на объекте. 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 значение свойства надписи Name (Имя) на `MyFirstLabel` (по умолчанию она называлась `Label1`). Для этого в инспекторе объектов щелкните на свойстве Name (Имя) и введите строку "MyFirstLabel". Убедитесь, что вы изменяете свойство надписи, а не формы (это типичная ошибка новичков). Для этого надпись в форме должна быть выделена, а в раскрывающемся списке в верхней части инспектора объектов должен быть выбран объект `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 строку "Моя первая надпись!" и нажмите клавишу <Enter>. Введенный текст появится в форме. Обратите внимание, как изменяются границы надписи. Это связано со свойством AutoSize (Подгонка размера). Если установить свойство AutoSize в состояние `False`, то автоматического изменения границ происходить не будет. Границы можно изменять вручную. Для этого 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 кнопке с тремя точками. В поле Font измените шрифт на `Arial`, стиль на `Bold Italic`, а размер на 32. 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 компоненте TLabel, переместите указатель мыши в произвольное место формы и еще раз щелкните кнопкой мыши. При этом в форме в том месте, где находился указатель мыши, должна появиться новая надпись.
9. Измените значение свойства Name новой надписи на `MySecondLabel`, а значение свойства Caption — на "Моя вторая надпись!".
10. Сделайте видимой панель Align и выделите в форме надписи, для чего обведите их на форме указателем мыши при нажатой левой кнопке. Используйте различные способы выравнивания из панели Align и оцените результат.
11. Теперь выделите форму, для чего щелкните в любом месте формы за пределами надписей.
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) из 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. Щелкните на кнопке закрытия окна в верхнем правом углу и завершите приложение. Это же можно сделать и в среде Delphi, используя команду Run⇨Program Reset (Выполнить⇨Переустановка программы) или нажав комбинацию клавиш <Ctrl+F2>.

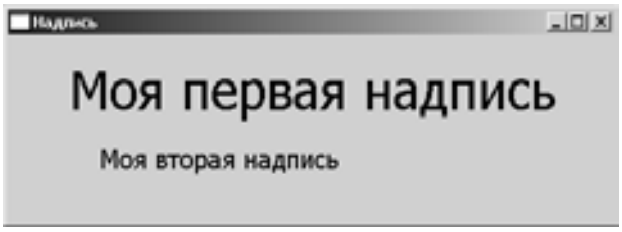

Рис. 1.7. Форма с надписями

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

Компонент TEdit

В компоненте `TEdit` (Поле ввода) хранится текст, который можно помещать в анный компонент как во время разработки, так и во время выполнения. Текст, видимый в поле ввода, является значением свойства Text. Свойство MaxLength (Максимальная длина) 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 VCL Forms Application.
2. Разместите компонент `TEdit` в форме. Компонент `TEdit` находится в группе Standard, как и надпись. Размещение также 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, так же, как это делалось с надписью.
5. Установите значение свойства Name на `MyText`. Для этого в инспекторе объектов щелкните на свойстве Name и введите строку “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 `Editl: 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, который в данном случае сам используется как свойство. 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 появится 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 True для свойства Readonly изменять содержимое поля ввода через графический интерфейс пользователя уже будет нельзя. Значение свойства Readonly можно менять из программы, запрещая или разрешая таким образом пользователю вводить данные.
10. Завершите приложение.

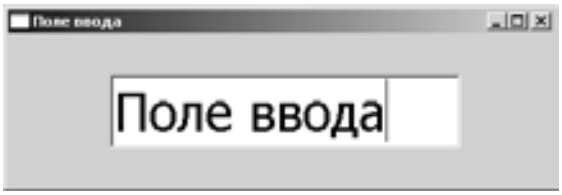

Рис. 1.8. Поле ввода

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 во время выполнения. Свойство Lines содержит отдельные строки текста области просмотра, оно доступно как во время разработки, так и во время выполнения. Свойство WordWrap 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.

Чтобы лучше изучить область просмотра, выполните следующие действия.

1. Создайте новый проект типа VCL Forms 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 значение свойства Name области просмотра на MemSample, для чего в инспекторе объектов щелкните на свойстве Name и введите строку "MemSample". Как и в случае надписи или поля ввода, убедитесь, что вы изменили свойство

области просмотра, а не формы. В раскрывающемся списке 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 — появится окно редактора строк String List Editor. Введите текст, показанный на рис. 1.9. Закончив ввод текста, щелкните на кнопке OK.

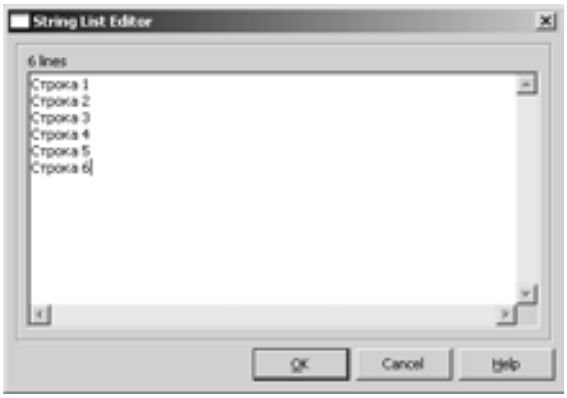

Рис. 1.9. Текст, вводимый в редакторе строк

6. Выделите форму. Для этого щелкните левой кнопкой мыши на форме или 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. Поэкспериментируйте с режимами выделения текста, сохранения и извлечения из буфера обмена.

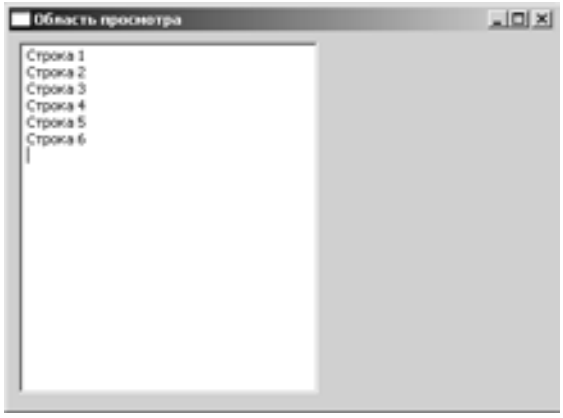

Рис. 1.10. Окно с компонентом `TMemo`

8. Завершите работу программы.
9. В инспекторе объектов измените значение свойства WordWrap 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. Чтобы лучше понять работу области просмотра, поэкспериментируйте с различными установками свойств WordWrap и ScrollBars.

Компонент TButton

Обычно с помощью компонента TButton (Кнопка) пользователь инициирует выполнение какого-либо фрагмента кода или целой программы. Другими словами, если щелкнуть на элементе управления TButton, то программа выполняет определенное действие. При этом кнопка принимает такой вид, будто она нажата.

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 VCL Forms Application.
2. В инспекторе объектов измените значение свойства формы Name на ButtonExample, а свойства Caption — на Кнопка.
3. Поместите кнопку в форму.
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. Так как с кнопкой еще не связан какой-либо код, то 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, что не всегда удобно.

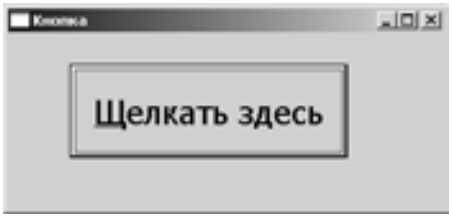

Рис. 1.11. Пример формы с кнопкой

В данном примере для использования комбинации быстрых клавиш клавиатура должна быть переключена на русский язык. Кроме того, должен использоваться верхний регистр (для оперативного переключения на верхний регистр необходимо нажать

клавишу <Shift> или 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 строку "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 работу Delphi (что крайне маловероятно). Тогда придется всего лишь перезапустить программу, 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 подсказки, и файл справки. Только в самых простейших случаях, как например, первая программа, можно этого не делать.

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

1. Создайте проект типа VCL Forms Application.
2. Разместите на нем две кнопки и надпись. Используя способы выравнивания с панели Align или вручную, постарайтесь расположить их симметрично, выбрав подходящие размеры.
3. Назовите первую кнопку `Вывод надписи`, а вторую — `Очистка` (свойство Caption).
4. Для надписи выберите подходящий размер шрифта (выше говорилось, как это сделать).
5. Создайте обработчики событий для нажатия кнопки.
6. Для создания обработчика события нажатия кнопки необходимо в инспекторе объектов выбрать вкладку Events (События), где следует 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 мышью 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.

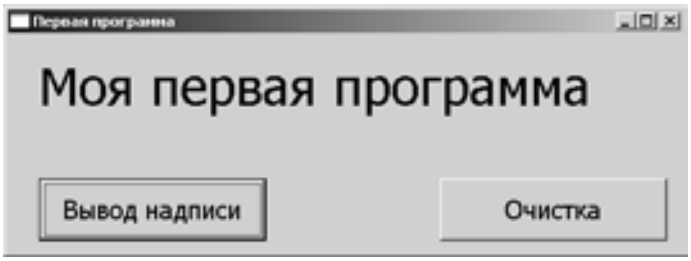

Рис. 1.12. Графический интерфейс первой программы

Полностью листинг вашей первой программы приведен ниже.

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

var
  Form1: TForm1;

implementation
{$R *.nfm}
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.Caption := 'Моя первая программа';
end;

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

Изучив листинг, вы найдете имена созданных вами объектов и выводимых надписей. Больше об этом листинге говорить не будем, пока не познакомимся с основами объектно-ориентированного языка Delphi. 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 Options, которое можно открыть, выбрав команду меню Tools⇨Options.... В окне выбирается вкладка Windows Form Designer, где находится настройка Grid Size, с помощью которой и устанавливаются размеры сетки для формы.

В этой же группе находятся такие настройки, как Show grid (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).

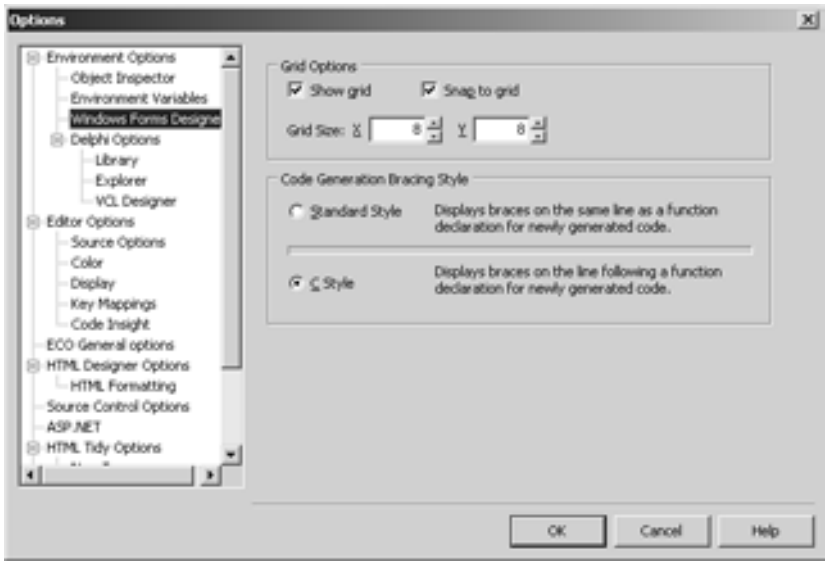

Рис. 1.13. Окно настроек Options

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 команду File⇨Save Project as... и сохраните проект и исходные файлы 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 расширениями `.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 Delphi. Для активизации справочной системы выберите команду Help⇨Delphi Help (Справка⇨Справка Delphi) или щелкните на пиктограмме Help contents (Содержание справочной системы) на обычной панели инструментов. При этом появится окно справочной системы Delphi (см. рис. 1.2).

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, для получения справки о типе выделите идентификатор типа, щелкните на нем правой кнопкой мыши и выберите команду Topic Search. Delphi предоставит справку о данном компоненте. Аналогичный результат можно получить, если нажать клавишу <F1>.

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

Сначала создадим календарь, разработка которого займет не более минуты.

1. Создайте проект типа VCL Forms Application.
2. Разместите на нем компонент типа `TMonthCalendar` (Календарь), который находится в разделе `Win32`.
3. Разместите календарь в форме и установите подходящие размеры формы (рис. 1.14).
4. Запустите программу и испытайте работу компонента `MonthCalendar`.
5. 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 компоненты `TFileListBox`, `TDirectoryListBox`, `TDriveComboBox` и `TFilterComboBox`, которые находятся в разделе `Win31`. Сделайте все то же самое, как и при создании календаря, за одним исключением. В инспекторе объектов нужно установить необходимые связи. Например, для свойства FileList компонента `DirectoryListBox1` необходимо установить значение `FileListBox`. Его просто нужно выбрать из раскрывающегося списка, который будет создан для свойства FileList. Это же нужно сделать и с другими компонентами. Для свойства формы Caption введите значение `Навигатор` и можете наслаждаться навигатором по файлам 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 щелчков 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 обеспечивает работу целого предприятия.

В дальнейшем мы познакомимся с 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 соответствующий код. Для того чтобы это делать, а самое главное, чтобы самому разрабатывать отдельные компоненты, необходимо знать язык программирования Delphi, на котором написана и сама Delphi. Поэтому все программы, разработанные в Delphi, хорошо интегрируются со средой Delphi. В следующих главах займемся изучением языка программирования Delphi.

Резюме

В данной главе вы ознакомились со средой программирования Delphi. Изучая рабочий стол Delphi, вы, вероятно, убедились, 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. Безусловно, пока еще не все понятно, но к этим вопросам мы будем возвращаться на протяжении всей книги.

Контрольные вопросы

1. Что такое среда программирования?
2. Назовите панели меню, которые могут располагаться на рабочем столе Delphi.
3. Какие окна располагаются на рабочем столе Delphi?
4. Что такое свойство?
5. Что находится на вкладках Code и Design?
6. Что такое компонент?
7. Какие стандартные компоненты вы уже знаете? Назовите их основные свойства.