Самоучитель

Никита Культин

Основы программирования в Delphi 7

2-е издание

Среда разработки

Назначение базовых компонентов

Программирование графики, мультимедиа и баз данных

Создание справочной системы и установочного CD

bhv + CD

Никита Культин

Основы программирования в Delphi 7

2-е издание

Санкт-Петербург
«БХВ-Петербург»
2009

УДК 681.3.068+800.92Delphi 7
ББК 32.973.26-018.1
К90

Культин Н. Б.

К90 Основы программирования в Delphi. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2009. — 640 с.: ил. + CD-ROM — (Самоучитель)

ISBN 978-5-9775-0310-5

Книга является руководством по программированию в среде Delphi 7. Описывается весь процесс разработки программы: от создания диалогового окна до организации справочной системы и создания установочного компакт-диска. Материал включает ряд тем, которые, как правило, остаются за рамками книг, адресованных начинающим — обработка символьной информации, использование динамических структур, работа с файлами. Рассматриваются вопросы программирования графики, мультимедиа и работа с базами данных. Приведено описание процесса создания справочной системы при помощи программы Microsoft HTML Help Workshop, установочного компакт-диска в InstallShield Express. Книга отличается доступностью изложения, большим количеством наглядных примеров. Во втором издании обновлены примеры и описана работа с базами данных Access на основе ADO. Прилагаемый компакт-диск содержит проекты, приведенные в книге.

Для начинающих программистов

УДК 681.3.068+800.92Delphi 7
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.08.08.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 51,6.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0310-5

Оглавление

Предисловие

Delphi — что это?

В последнее время резко возрос интерес к программированию. Это связано с развитием и внедрением в повседневную жизнь информационно-коммуникационных технологий. Если человек имеет дело с компьютером, то рано или поздно у него возникает желание, а иногда и необходимость, программировать.

Среди пользователей персональных компьютеров в настоящее время наиболее популярно семейство операционных систем Windows, и, естественно, тот, кто собирается программировать, стремится писать программы, которые будут работать в этих системах.

Несколько лет назад рядовому программисту оставалось только мечтать о создании собственных программ, работающих в среде Windows, т. к. единственным средством разработки был Borland C++ for Windows, явно ориентированный на профессионалов, обладающих серьезными знаниями и опытом.

Бурное развитие вычислительной техники, потребность в эффективных средствах разработки программного обеспечения привели к появлению систем программирования, ориентированных на так называемую "быструю разработку", среди которых можно выделить Borland Delphi и Microsoft Visual Basic. В основе систем быстрой разработки (RAD-систем, Rapid Application Development — среда быстрой разработки приложений) лежит технология визуального проектирования и событийного программирования, суть которой заключается в том, что среда разработки берет на себя большую часть рутинной работы, оставляя программисту работу по конструированию диалоговых окон и функций обработки событий. Производительность программиста при использовании RAD-систем — фантастическая!

Delphi — это среда быстрой разработки, в которой в качестве языка программирования используется Object Pascal. В основе идеологии Delphi лежит технология визуального проектирования и методология объектно-ориентированного событийного программирования. Она позволяет создавать программы различного назначения: от простейших однооконных приложений до программ работы с распределенными базами данных.

В настоящее время разработчики используют разные версии Delphi, в том числе и Delphi 7. Эта версия также широко используется и в учебных заведениях: в школах, техникумах, вузах. И это не удивительно. Как показывает опыт, научившись работать в Delphi 7, программист без особых усилий сможет перейти на другие, современные версии.

Borland Delphi 7 может работать в среде операционных систем от Windows 98 до Windows XP. Особых требований, по современным меркам, к ресурсам компьютера пакет не предъявляет (это выгодно отличает его от последующих версий, требующих наличия на компьютере Microsoft .NET Framework). Процессор должен быть типа Pentium или Celeron с тактовой частотой не ниже 166 МГц (рекомендуется Pentium II 400 МГц). Компьютер должен иметь оперативную память объемом 128 Мбайт (рекомендуется 256 Мбайт). На жестком диске должно быть достаточное количество свободного пространства (для полной установки необходимо приблизительно 475 Мбайт).

Об этой книге

В книге, посвященной программированию в конкретной среде разработки, необходим баланс между тремя линиями: языком программирования, технологией и средой разработки. Уже при первом представлении среды разработки, описании ее возможностей у автора возникает проблема: чтобы описать процесс разработки программы, объяснить, как работает программа, нужно оперировать такими терминами, как *объект*, *событие*, *свойство*, понимание которых на начальном этапе изучения программирования весьма проблематично. Как поступить? Сначала дать описание языка, а затем приступить к описанию среды разработки и процесса программирования в Delphi? Очевидно, что это не лучшее решение. Поэтому при изложении материала принят подход, в основу которого положен принцип соблюдения баланса между описанием языка, технологией программирования и средой разработки. В начале книги некоторые понятия, без которых просто невозможно изложение материала, даются на уровне определений.

Книга, которую вы держите в руках, — это не описание языка программирования Delphi и среды разработки Delphi. Это пособие (руководство) по программированию на языке Delphi в среде Delphi 7. В нем рассмотрена вся цепочка, весь процесс создания программы: от разработки диалогового окна и функций обработки событий до создания справочной системы и установочной дискеты.

Цель этой книги может быть сформулирована так: научить программировать в среде Delphi, создавать законченные программы различного назначения: от простых однооконных приложений до вполне профессиональных программ работы с базами данных.

Научиться программировать можно, только программируя, решая конкретные задачи. При этом достигнутые в программировании успехи в значительной степени зависят от опыта. Поэтому, чтобы получить максимальную пользу от книги, вы должны работать с ней активно. Не занимайтесь просто чтением примеров, реализуйте их с помощью вашего компьютера. Не бойтесь экспериментировать — вносите изменения в программы. Чем больше вы сделаете самостоятельно, тем большему научитесь!

Глава 1

Среда программирования Delphi

В данной главе описывается процесс установки Delphi. На примере программы **Скорость**, позволяющей вычислить скорость, с которой бегун пробежал дистанцию, демонстрируется процесс разработки программы в Delphi, технология визуального проектирования и событийного программирования, вводятся основные понятия и термины.

Установка

Установка Delphi 7 на компьютер выполняется с CD, на котором находятся все необходимые файлы, а также программа инициализации установки (Delphi Setup Launcher). Программа инициализации установки запускается автоматически, как только установочный диск будет помещен в дисковод.

В результате запуска программы инициализации установки на экране появляется окно **Delphi 7 Setup Launcher** (рис. 1.1), в котором перечислены программные продукты, которые можно установить на компьютер. Это, прежде всего, Delphi 7, сервер баз данных InterBase 6.5 и InstallShield Express — утилита, предназначенная для создания установочных CD.

Для того чтобы активизировать процесс установки Delphi, следует щелкнуть на строке **Delphi 7**. Процесс установки Delphi обычный. После ввода серийного номера (Serial Number) и ключа (Authorization Key) на экране сначала появляется окно с лицензионным соглашением, затем — окно **Setup Type** (рис. 1.2), в котором можно выбрать один из возможных вариантов установки: **Typical** (Обычный), **Compact** (Компактный) или **Custom** (Выборочный, определяемый пользователем).

Обычный вариант установки предполагает, что на жесткий диск компьютера будут скопированы все компоненты Delphi.

Рис. 1.1. В окне **Delphi 7 Setup Launcher** перечислены компоненты, которые можно установить на компьютер

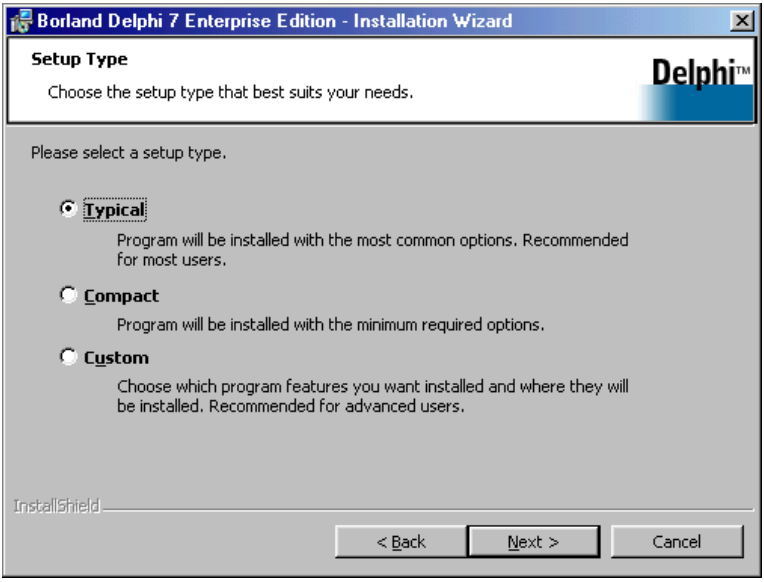

Рис. 1.2. В диалоговом окне **Setup Type** нужно выбрать вариант установки

Обычный вариант установки требует наибольшего свободного места на жестком диске компьютера, порядка 475 Мбайт (для комплекта Enterprise), однако

если на жестком диске компьютера достаточно свободного места, лучше выбрать именно этот вариант.

При *компактной установке* на жесткий диск компьютера копируются только самые необходимые компоненты Delphi. Компактный вариант требует наименьшего количества свободного дискового пространства. Однако в этом случае некоторые возможности среды разработки Delphi будут недоступны. В частности, при компактной установке на жесткий диск не копируются файлы справочной системы, некоторые компоненты и утилиты, примеры.

Выборочный вариант установки позволяет программисту выбрать только необходимые для работы инструменты и компоненты Delphi. Обычно этот вариант установки используют опытные программисты. Выборочный вариант можно предпочесть и в том случае, если на диске компьютера недостаточно свободного места для полной установки.

Выбрав вариант установки, нажмите кнопку **Next**. Если был выбран вариант установки **Custom**, то открывается диалоговое окно **Custom Setup** (рис. 1.3), в котором можно выбрать устанавливаемые компоненты, точнее — указать компоненты, которые устанавливать не надо. Чтобы запретить установку компонента, нужно щелкнуть на изображении диска слева от названия компонента и из появившегося меню выбрать команду **Do Not Install**.

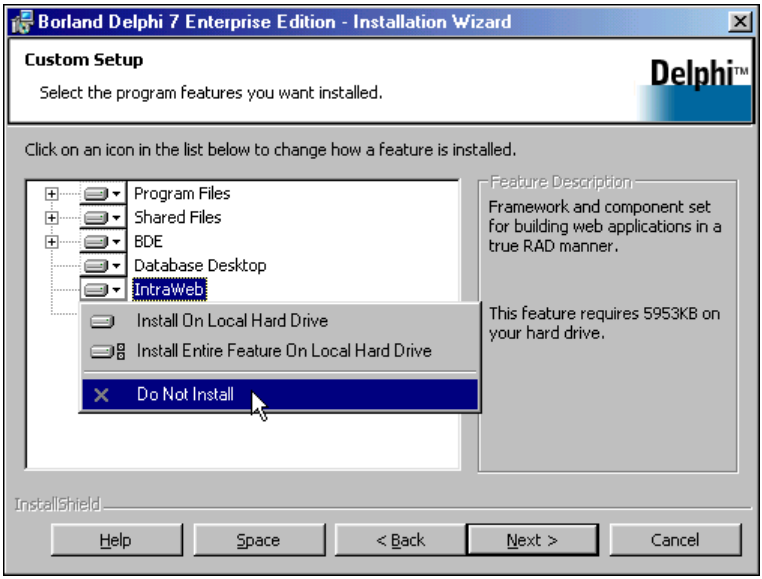

Рис. 1.3. Запрет установки компонента

Если выбран тип установки **Typical**, то в результате щелчка на кнопке **Next** открывается окно **Destination Folder**, в котором указаны каталоги, куда будет установлен пакет Delphi и его компоненты.

Очередной щелчок на кнопке **Next** открывает окно **Save Installation Database**, в котором пользователю предлагается сохранить информацию о процессе установки на жестком диске компьютера, что обеспечит возможность деинсталляции Delphi в дальнейшем без использования установочного CD. На этом процесс подготовки к установке заканчивается. На экране появляется окно **Ready To Install the Program**, щелчок на кнопке **Install** в котором активизирует процесс установки.

По окончании процесса установки на экране появляется окно с информационным сообщением о том, что установка выполнена (рис. 1.4). Щелчок на кнопке **Finish** закрывает это окно.

Теперь можно приступить к работе — запустить Delphi.

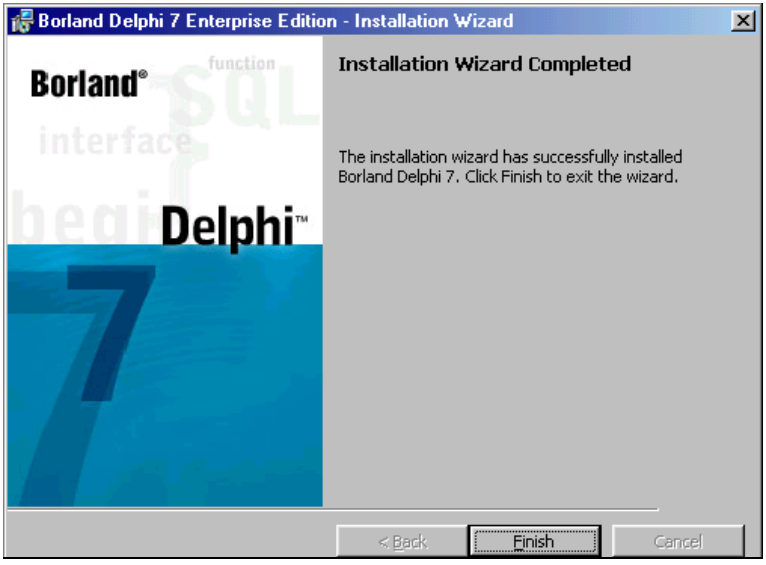

Рис. 1.4. Процесс установки завершен

Начало работы

Запускается Delphi обычным образом — выбором в меню **Пуск** команды **Все программы ▸ Borland Delphi 7 ▸ Delphi 7** (рис. 1.5).

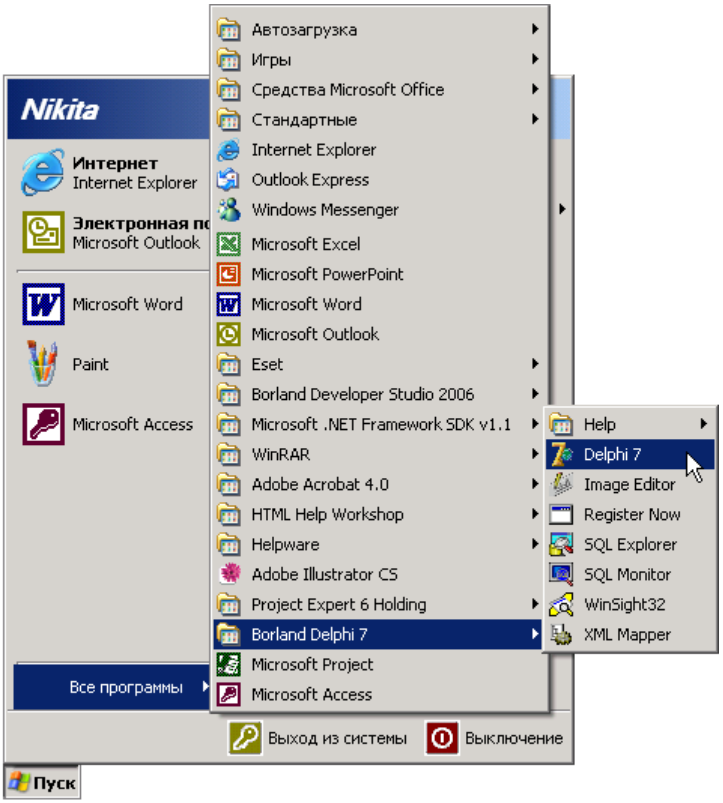

Рис. 1.5. Запуск Delphi

Вид экрана после запуска Delphi несколько необычен (рис. 1.6). Вместо одного окна на экране появляются пять:

- главное окно — **Delphi 7**;
- окно формы — **Form1**;
- окно редактора свойств объектов — **Object Inspector**;
- окно списка объектов — **Object TreeView**;
- окно редактора кода — **Unit1.pas**.

В главном окне (рис. 1.7) находится меню команд, панели инструментов и палитра компонентов.

Окно формы (**Form1**) представляет собой заготовку окна разрабатываемого *приложения*.

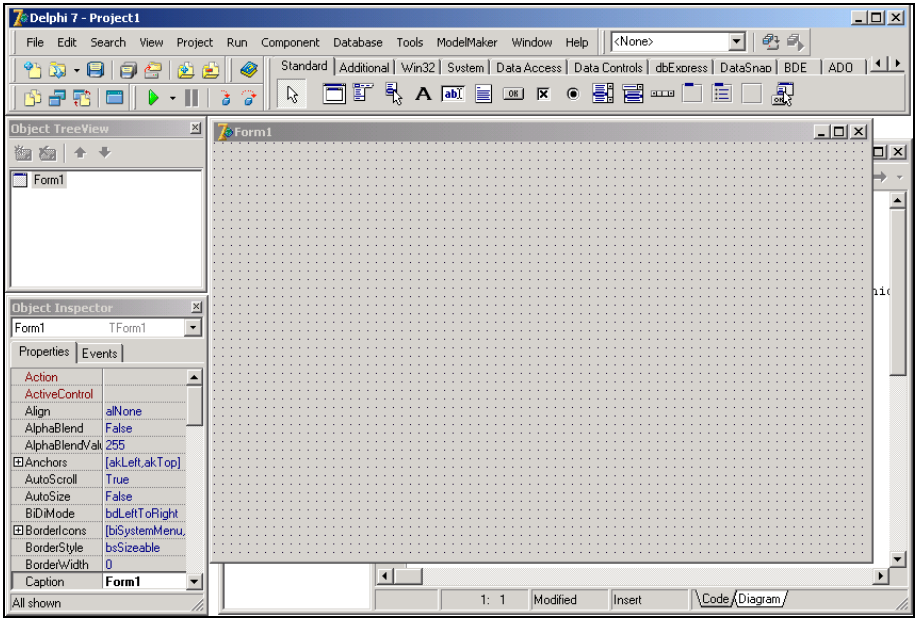

Рис. 1.6. Вид экрана после запуска Delphi

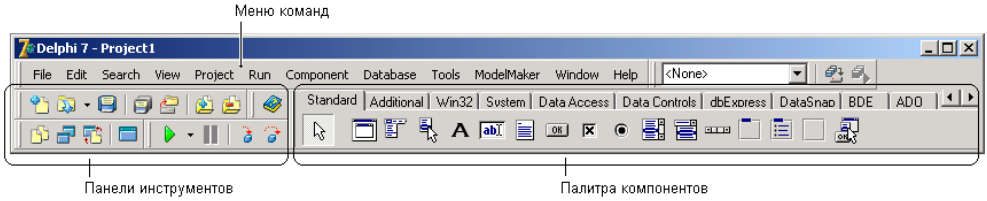

Рис. 1.7. Главное окно

Замечание

Программное обеспечение принято делить на *системное* и *прикладное*. Системное программное обеспечение — это все то, что составляет операционную систему. Остальные программы, предназначенные для решения *прикладных* задач, принято называть прикладными или приложениями.

Окно **Object Inspector** (рис. 1.8) предназначено для редактирования значений свойств объектов. В терминологии визуального проектирования *объекты* — это диалоговые окна и элементы управления (поля редактирования, поля отображения текста, командные кнопки, переключатели и др.).

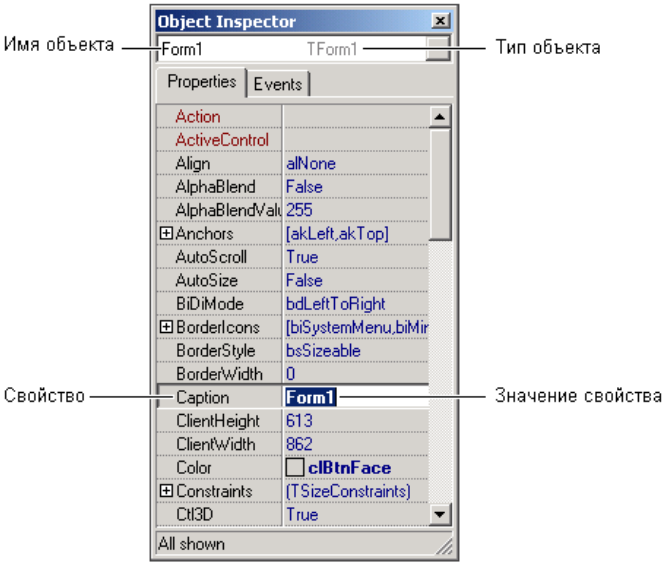

Рис. 1.8. На вкладке **Properties** перечислены свойства объекта и указаны их значения

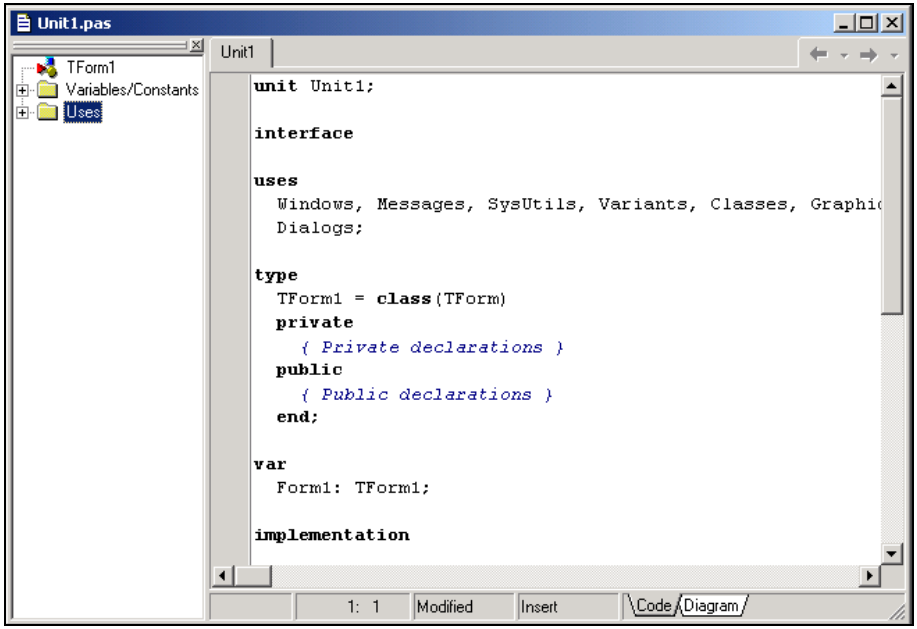

Рис. 1.9. Окно редактора кода

Свойства объекта — это характеристики, определяющие вид, положение и поведение объекта. Например, свойства `Width` и `Height` задают размер (ширину и высоту) формы, свойства `Top` и `Left` — положение формы на экране, свойство `Caption` — текст заголовка.

Окно редактора кода (рис. 1.9), которое можно увидеть, отодвинув в сторону окно формы или нажав функциональную клавишу <F12>, в начале работы над новым проектом содержит сформированный Delphi шаблон программы.

Первый проект

Процесс создания программы в Delphi рассмотрим на примере — создадим приложение, используя которое можно вычислить скорость, с которой спортсмен пробежал дистанцию. Вид окна программы во время ее работы приведен на рис. 1.10.

Для начала работы над новым приложением выберите в меню **File** команду **New ▸ Application**.

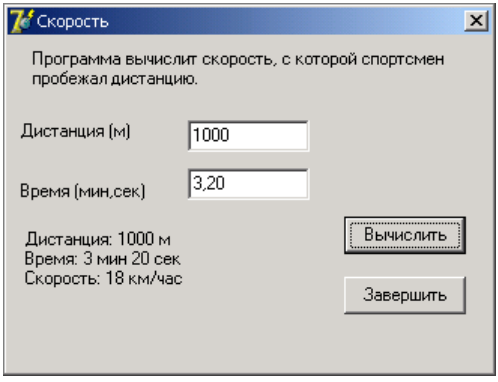

Рис. 1.10. Окно программы вычисления скорости

Форма

Работа над *новым проектом*, так принято называть разрабатываемую программу, начинается с создания стартовой формы.

Форма создается путем изменения значений ее свойств и добавления к форме необходимых компонентов (полей редактирования, полей отображения текста, командных кнопок и др.).

Свойства формы (табл. 1.1) определяют ее вид: размер, текст заголовка, вид рамки, а также положение на экране.

Для просмотра и изменения значений свойств формы и ее компонентов используется окно **Object Inspector**. В верхней части окна **Object Inspector** указано имя объекта, значения свойств которого отображается в данный момент. В левой колонке вкладки **Properties** (Свойства) перечислены свойства объекта, а в правой указаны их значения.

Таблица 1.1. *Свойства формы (объекта* `TForm`*)*

Свойство	Описание
`Name`	Имя формы. Используется для доступа к форме
`Caption`	Текст заголовка
`Width`	Ширина формы
`Height`	Высота формы
`Position`	Положение окна в момент первого его появления на экране (`poDesktopCenter` — в центре рабочего стола; `poCenterScreen` — в центре экрана; `poOwnerFormCenter` — в центре родительского окна; `poDesigned` — положение окна определяют значения свойств `Top` и `Left`)
`Top`	Расстояние от верхней границы формы до верхней границы экрана
`Left`	Расстояние от левой границы формы до левой границы экрана
`BorderStyle`	Вид границы. Граница может быть обычной (`bsSizeable`), тонкой (`bsSingle`) или отсутствовать (`bsNone`). Если у окна обычная граница, то во время работы программы пользователь может при помощи мыши изменить размер окна. Изменить размер окна с тонкой границей нельзя. Если граница отсутствует, то на экран во время работы программы будет выведено окно без заголовка. Положение и размер такого окна во время работы программы изменить нельзя
`BorderIcons`	Кнопки управления окном. Значение свойства определяет, какие кнопки управления окном будут доступны пользователю во время работы программы. Значение свойства задается путем присвоения значений уточняющим свойствам `biSystemMenu`, `biMinimaze`, `biMaximaze` и `biHelp`. Свойство `biSystemMenu` определяет доступность кнопки системного меню, `biMinimaze` — кнопки **Свернуть**, `biMaximaze` — кнопки **Развернуть**, `biHelp` — кнопки **Справка**

Таблица 1.1 *(окончание)*

Свойство	Описание
`Icon`	Значок в заголовке диалогового окна, обозначающий кнопку вывода системного меню
`Color`	Цвет фона. Цвет можно задать, указав название цвета или привязку к текущей цветовой схеме операционной системы. Во втором случае цвет определяется текущей цветовой схемой, выбранным компонентом привязки и меняется при изменении цветовой схемы операционной системы
`Font`	Шрифт, используемый "по умолчанию" компонентами, находящимися на поверхности формы. Изменение свойства `Font` формы приводит к автоматическому изменению свойства `Font` компонента, располагающегося на поверхности формы. То есть компоненты наследуют свойство `Font` от формы (имеется возможность запретить наследование)

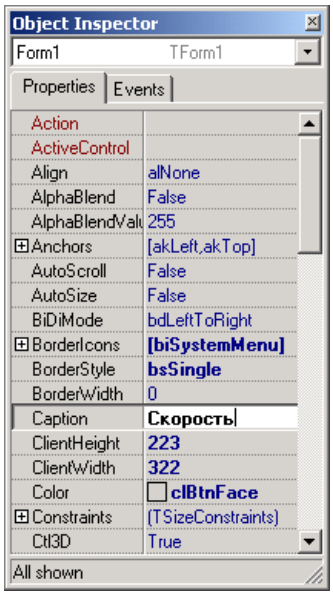

Рис. 1.11. Установка значения свойства путем ввода значения

При создании формы в первую очередь следует изменить значение свойства `Caption` (Заголовок). В нашем примере надо заменить текст `Form1` на `Скорость`. Чтобы это сделать, нужно в окне **Object Inspector** щелкнуть мышью в

строке `Caption` (в результате в поле значения свойства появится курсор) и ввести текст `Скорость` (рис. 1.11).

Аналогичным образом следует установить значения свойств `Height` и `Width`, которые определяют размер формы — соответственно, высоту и ширину.

Размер формы и ее положение на экране, а также размер компонентов и их положение на поверхности формы задают в пикселах, т. е. точках экрана. Свойствам `Height` и `Width` надо присвоить значения 250 и 330 соответственно.

Форма — это обычное окно. Поэтому ее размер можно изменить точно так же, как любого другого окна, т. е. захватом и перемещением (с помощью мыши) границы. По окончании перемещения границ автоматически изменятся значения свойств `Height` и `Width`. Они будут соответствовать установленному размеру формы.

По умолчанию положение окна на экране сразу после запуска программы соответствует положению формы во время разработки программы, которое определяется значением свойств `Top` (отступ от верхней границы экрана) и `Left` (отступ от левой границы экрана). Значения этих свойств также можно задать путем перемещения окна формы при помощи мыши.

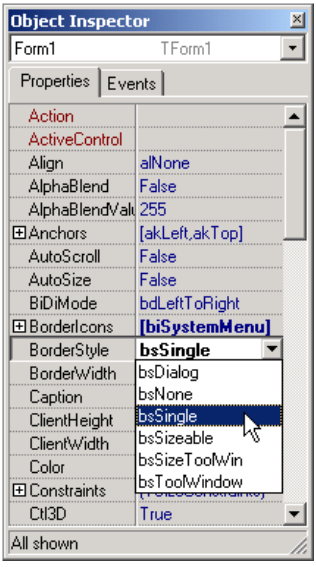

Рис. 1.12. Установка значения свойства путем выбора из списка

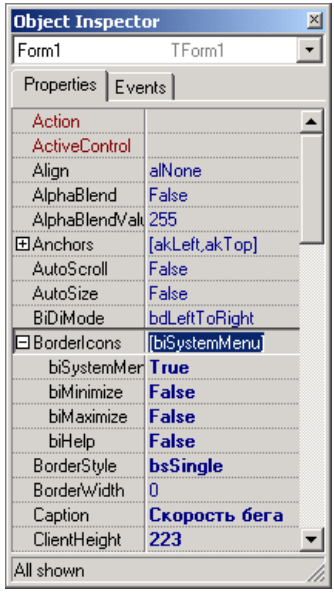

Рис. 1.13. Раскрытый список вложенных свойств сложного свойства `BorderIcons`

При выборе некоторых свойств, например BorderStyle, справа от текущего значения свойства появляется значок раскрывающегося списка. Очевидно, что значение таких свойств можно задать путем выбора из списка (рис. 1.12).

Некоторые свойства являются сложными, т. е. их значение определяется совокупностью значений других (уточняющих) свойств. Перед именами сложных свойств стоит значок "+", при щелчке на котором раскрывается список уточняющих свойств (рис. 1.13). Например, свойство BorderIcons определяет, какие кнопки управления окном будут доступны во время работы программы. Так, если свойству biMaximize присвоить значение False, то во время работы программы кнопки **Развернуть** в заголовке окна не будет.

Рядом со значениями некоторых свойств отображается командная кнопка с тремя точками. Это значит, что для задания значения свойства можно воспользоваться дополнительным диалоговым окном. Например, значение сложного свойства Font можно задать путем непосредственного ввода значений уточняющих свойств, а можно воспользоваться стандартным диалоговым окном выбора шрифта.

В табл. 1.2 перечислены свойства формы разрабатываемой программы, которые следует изменить. Остальные свойства оставлены без изменения и в таблице не приведены.

Таблица 1.2. *Значения свойств формы*

Свойство	Значение
Caption	Скорость
Height	250
Width	330
Position	poDesktopCenter
Font.Name	Tahoma
Font.Size	10
BorderStyle	bsSingle
BorderIcons.biMinimize	False
BorderIcons.biMaximize	False

В приведенной таблице в именах некоторых свойств есть точка. Это значит, что надо задать значение уточняющего свойства. После того как будут уста-

новлены значения свойств главной формы, она должна выглядеть так, как показано на рис. 1.14.

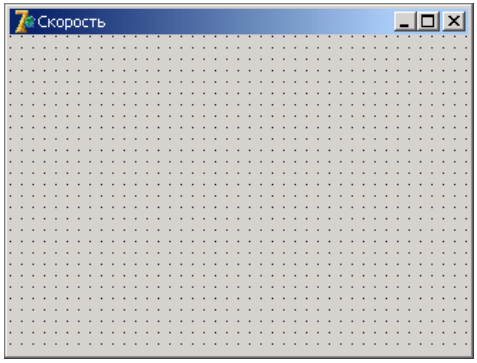

Рис. 1.14. Так выглядит форма после установки значений свойств

Компоненты

Программа вычисления скорости бега должна получить от пользователя исходные данные — длину дистанции и время, за которое спортсмен пробежал дистанцию. В подобных программах данные с клавиатуры, как правило, вводят в поля редактирования. Поэтому в форму надо добавить компонент `Edit` — поле редактирования.

Наиболее часто используемые компоненты находятся на вкладке **Standard** (рис. 1.15).

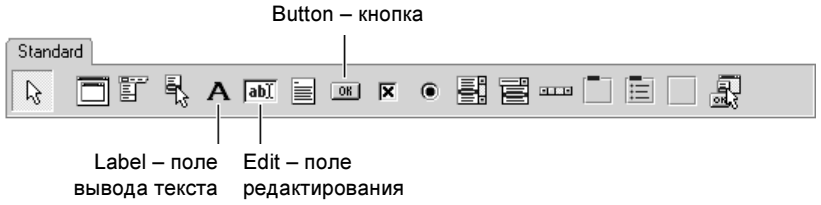

Рис. 1.15. На вкладке **Standard** находятся наиболее часто используемые компоненты

Для того чтобы добавить в форму компонент, необходимо в палитре компонентов выбрать этот компонент, щелкнув левой кнопкой мыши на его пикто-

грамме, далее установить курсор в ту точку формы, в которой должен быть левый верхний угол компонента, и еще раз щелкнуть левой кнопкой мыши. В результате в форме появляется компонент стандартного размера.

Размер компонента можно задать в процессе его добавления к форме. Для этого надо после выбора компонента из палитры поместить курсор мыши в ту точку формы, где должен находиться левый верхний угол компонента, нажать левую кнопку мыши и, удерживая ее нажатой, переместить курсор в точку, где должен находиться правый нижний угол компонента, затем отпустить кнопку мыши. На форме появится компонент нужного размера.

Каждому компоненту Delphi присваивает имя, которое состоит из названия компонента и его порядкового номера. Например, если к форме добавить два компонента `Edit`, то их имена будут `Edit1` и `Edit2`. Программист, путем изменения значения свойства `Name`, может изменить имя компонента. В простых программах имена компонентов, как правило, не изменяют.

На рис. 1.16 приведен вид формы после добавления двух компонентов `Edit` — полей редактирования, предназначенных для ввода исходных данных. Один из компонентов выделен. Свойства выделенного компонента отображаются в окне **Object Inspector**. Чтобы увидеть свойства другого компонента, надо щелкнуть левой кнопкой мыши на изображении нужного компонента. Можно также выбрать имя компонента в окне **Object TreeView** или из находящегося в верхней части окна **Object Inspector** раскрывающегося списка объектов.

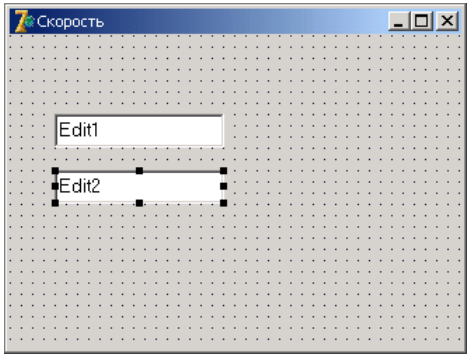

Рис. 1.16. Форма после добавления компонентов `Edit`

В табл. 1.3 перечислены основные свойства компонента `Edit` — поля редактирования.

Таблица 1.3. *Свойства компонента* `Edit`

Свойство	Описание
`Name`	Имя компонента. Используется для доступа к компоненту и его свойствам
`Text`	Текст, находящийся в поле ввода и редактирования
`Left`	Расстояние от левой границы компонента до левой границы формы
`Top`	Расстояние от верхней границы компонента до верхней границы формы
`Height`	Высота поля
`Width`	Ширина поля
`Font`	Шрифт, используемый для отображения вводимого текста
`ParentFont`	Признак наследования компонентом характеристик шрифта формы, на которой находится компонент. Если значение свойства равно `True`, то при изменении свойства `Font` формы автоматически меняется значение свойства `Font` компонента

Delphi позволяет изменить размер и положение компонента при помощи мыши.

Для того чтобы изменить положение компонента, необходимо установить курсор мыши на его изображение, нажать левую кнопку мыши и, удерживая ее нажатой, переместить контур компонента в нужную точку формы, затем отпустить кнопку мыши. Во время перемещения компонента (рис. 1.17) отображаются текущие значения координат левого верхнего угла компонента (значения свойств `Left` и `Top`).

Для того чтобы изменить размер компонента, необходимо его выделить, установить указатель мыши на один из маркеров, помечающих границу компонента, нажать левую кнопку мыши и, удерживая ее нажатой, изменить положение границы компонента. Затем отпустить кнопку мыши. Во время изменения размера компонента отображаются текущие значения свойств `Height` и `Width` (рис. 1.18).

Свойства компонента так же, как и свойства формы, можно изменить при помощи **Object Inspector**. Для того чтобы свойства требуемого компонента были выведены в окне **Object Inspector**, нужно выделить этот компонент (щелкнуть мышью на его изображении). Можно также выбрать компонент из

находящегося в верхней части окна **Object Inspector** раскрывающегося списка объектов (рис. 1.19) или из списка в окне **Object TreeView** (рис. 1.20).

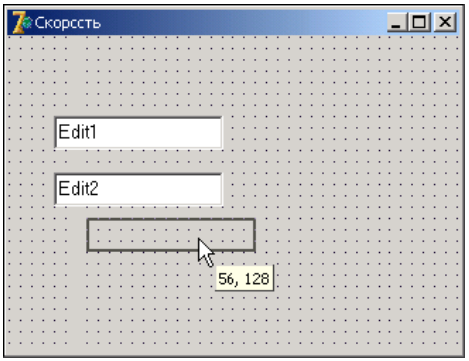

Рис. 1.17. Отображение текущих значений свойств `Left` и `Top` при изменении положения компонента

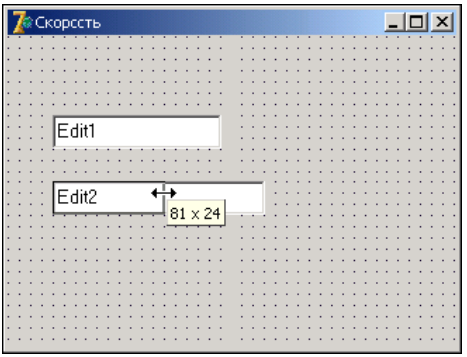

Рис. 1.18. Отображение текущих значений свойств `Height` и `Width` при изменении размера компонента

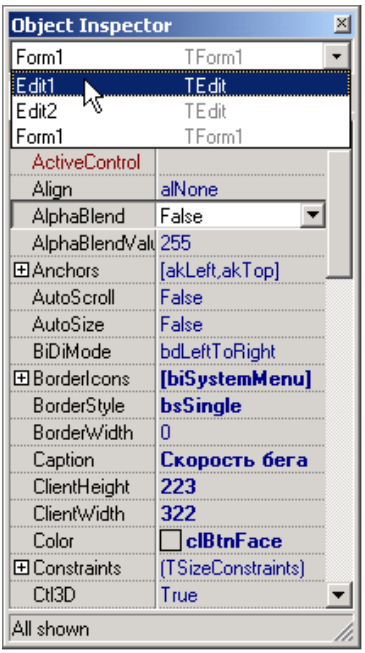

Рис. 1.19. Выбор компонента из списка в окне **Object Inspector**

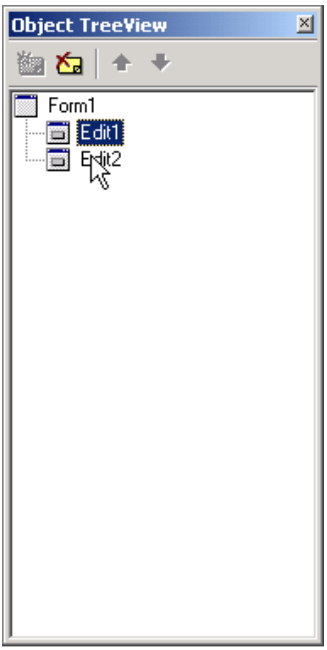

Рис. 1.20. Выбор компонента в окне **Object TreeView**

В табл. 1.4 приведены значения свойств компонентов `Edit1` и `Edit2`. Компонент `Edit1` предназначен для ввода длины дистанции, `Edit2` — для ввода времени. Обратите внимание на то, что значением свойства `Text` обоих компонентов является пустая строка.

Таблица 1.4. *Значения свойств компонентов* `Edit`

Компонент	Свойство	Значение
`Edit1`	`Top`	`56`
	`Left`	`128`
	`Width`	`121`
	`Height`	`21`
	`Text`	
`Edit2`	`Top`	`88`
	`Left`	`128`
	`Width`	`121`
	`Height`	`21`
	`Text`	

В окне программы должна отображаться информация как о назначении самой программы, так и назначении полей ввода. Отображение статического текста обеспечивает компонент `Label`. Значок компонента `Label` находится на вкладке **Standard** (рис. 1.21). Добавляется компонент `Label` на форму точно так же, как и поле редактирования.

В форму разрабатываемого приложения надо добавить четыре компонента `Label`. Первое поле предназначено для вывода информационного сообщения, второе и третье — для вывода информации о назначении полей ввода, четвертое поле служит для вывода результата расчета — скорости.

Рис. 1.21. Компонент `Label` — поле вывода текста

Свойства компонента Label перечислены в табл. 1.5.

Таблица 1.5. *Свойства компонента* Label *(поле вывода текста)*

Свойство	Описание
Name	Имя компонента. Используется для доступа к компоненту и его свойствам
Caption	Отображаемый текст
Font	Шрифт, используемый для отображения текста
ParentFont	Признак наследования компонентом характеристик шрифта формы, на которой находится компонент. Если значение свойства равно True, текст выводится шрифтом, установленным для формы
AutoSize	Признак того, что размер поля определяется его содержимым
Left	Расстояние от левой границы поля вывода до левой границы формы
Top	Расстояние от верхней границы поля вывода до верхней границы формы
Height	Высота поля вывода
Width	Ширина поля вывода
WordWrap	Признак того, что слова, которые не помещаются в текущей строке, автоматически переносятся на следующую строку

Следует обратить внимание на свойства AutoSize и WordWrap. Их нужно использовать, если поле вывода должно содержать несколько строк текста. После добавления к форме компонента Label значение свойства AutoSize равно True, т. е. размер поля определяется автоматически в процессе изменения значения свойства Caption. Если вы хотите, чтобы находящийся в поле вывода текст занимал несколько строк, то надо сразу после добавления к форме компонента Label присвоить свойству AutoSize значение False, свойству WordWrap — значение True. Затем изменением значений свойств Width и Height нужно задать требуемый размер поля. Только после этого можно ввести в свойство Caption текст, который должен быть выведен в поле.

После добавления полей вывода текста (четырех компонентов Label) и установки значений их свойств в соответствии с табл. 1.6 форма программы принимает вид, приведенный на рис. 1.22.

Обратите внимание, что значение свойства `Caption` вводится как одна строка. Расположение текста внутри поля вывода определяется размером поля, значением свойств `AutoSize` и `WordWrap`, а также зависит от характеристик используемого для вывода текста шрифта.

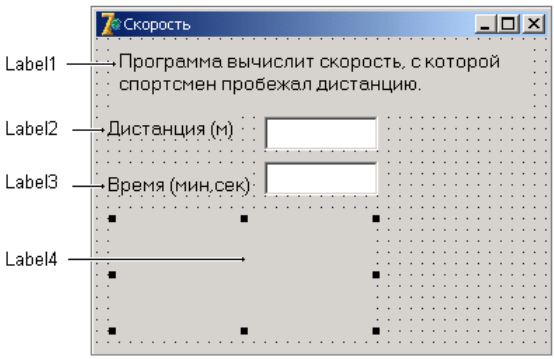

Рис. 1.22. Вид формы после добавления и настройки полей отображения текста (компонентов `Label`)

Таблица 1.6. *Значения свойств компонентов* `Label`

Компонент	Свойство	Значение
Label1	AutoSize	False
	WordWrap	True
	Caption	Программа вычислит скорость, с которой спортсмен пробежал дистанцию
	Top	8
	Left	8
	Height	33
	Width	209
Label2	Top	56
	Left	8
	Caption	Дистанция (м)

Таблица 1.6 *(окончание)*

Компонент	Свойство	Значение
Label3	Top	88
	Left	8
	Caption	Время (мин.,сек)
Label4	AutoSize	False
	WordWrap	True
	Top	120
	Left	8
	Height	41
	Width	273

Последнее, что надо сделать на этапе создания формы, — добавить в форму две командные кнопки: **Вычислить** и **Завершить**. Назначение этих кнопок очевидно.

Командная кнопка, компонент `Button`, добавляется в форму точно так же, как и другие компоненты. Значок компонента `Button` находится на вкладке **Standard** (рис. 1.23). Свойства компонента приведены в табл. 1.7.

Рис. 1.23. Командная кнопка — компонент `Button`

Таблица 1.7. *Свойства компонента* `Button`

Свойство	Описание
Name	Имя компонента. Используется для доступа к компоненту и его свойствам
Caption	Текст на кнопке
Enabled	Признак доступности кнопки. Кнопка доступна, если значение свойства равно `True`, и не доступна, если значение свойства равно `False`

Таблица 1.7 *(окончание)*

Свойство	Описание
`Left`	Расстояние от левой границы кнопки до левой границы формы
`Top`	Расстояние от верхней границы кнопки до верхней границы формы
`Height`	Высота кнопки
`Width`	Ширина кнопки

После добавления к форме двух командных кнопок нужно установить значения их свойств в соответствии с табл. 1.8.

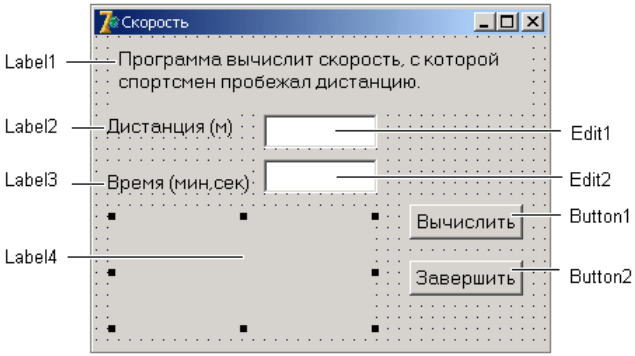

Рис. 1.24. Форма программы **Скорость**

Таблица 1.8. *Значения свойств компонента* `Button`

Компонент	Свойство	Значение
`Button1`	`Caption`	`Вычислить`
	`Top`	`120`
	`Left`	`224`
	`Height`	`25`
	`Width`	`81`

Таблица 1.8 *(окончание)*

Компонент	Свойство	Значение
Button2	Caption	Завершить
	Top	160
	Left	224
	Height	25
	Width	81

Окончательный вид формы разрабатываемого приложения приведен на рис. 1.24.

Событие и процедура обработки события

Вид формы программы **Скорость** подсказывает, как должна работать программа. Очевидно, что пользователь должен ввести в поля редактирования исходные данные и "нажать" кнопку **Вычислить** (сделать щелчок кнопкой мыши на изображении кнопки). Щелчок на изображении командной кнопки — это пример того, что называется *событием*.

Событие (Event) — это то, что происходит во время работы программы. У каждого события есть имя. Например, щелчок кнопкой мыши — это событие Click, двойной щелчок кнопкой — событие DblClick, нажатие клавиши клавиатуры — событие KeyPress.

В табл. 1.9 приведены некоторые события.

Таблица 1.9. *События*

Событие	Описание
Click	Щелчок кнопкой мыши
DblClick	Двойной щелчок кнопкой мыши
KeyPress	Нажатие клавиши клавиатуры
KeyDown	Нажатие (удерживание в нажатом состоянии) клавиши клавиатуры. События KeyDown и KeyPress — это чередующиеся, повторяющиеся события, которые происходят до тех пор, пока не будет отпущена удерживаемая клавиша (в этот момент происходит событие KeyUp)

Таблица 1.9 *(окончание)*

Событие	Описание
`KeyUp`	Отпускание нажатой клавиши клавиатуры
`Change`	Изменение содержимого поля редактирования
`MouseDown`	Нажатие кнопки мыши
`MouseUp`	Отпускание кнопки мыши
`MouseMove`	Перемещение указателя мыши
`Create`	Создание объекта (формы, элемента управления). Процедура обработки этого события обычно используется для инициализации переменных, выполнения подготовительных действий
`Paint`	Событие происходит при появлении окна на экране в начале работы программы, при появлении части окна, которая, например, была закрыта другим окном и в других случаях
`Enter`	Получение элементом управления (компонентом) фокуса
`Exit`	Потеря элементом управления (компонентом) фокуса

Реакцией на событие должно быть какое-либо действие. В Delphi реакция на событие реализуется как *процедура обработки события*. Таким образом, для того чтобы программа выполняла некоторую работу в ответ на действия пользователя, программист должен написать процедуру обработки соответствующего события. Следует обратить внимание на то, что значительную часть обработки событий берет на себя компонент. Поэтому программист должен разрабатывать процедуру обработки события только в том случае, если реакция компонента на событие отличается от стандартной. Например, если по условию задачи ограничений на символы, вводимые в поле `Edit`, нет, то процедуру обработки события `KeyPress` писать не надо, т. к. во время работы программы будет использована стандартная (скрытая от программиста) процедура обработки этого события.

Методику создания процедур обработки событий рассмотрим на примере процедуры обработки события `Click` для командной кнопки **Вычислить**.

Чтобы приступить к созданию процедуры обработки события, нужно сначала выбрать компонент (сделать щелчок левой кнопкой мыши на изображении компонента), событие которого надо обработать (обратите внимание, в верхней части окна **Object Inspector** отображается имя выбранного компонента). Затем в окне **Object Inspector** надо открыть вкладку **Events** (рис. 1.25).

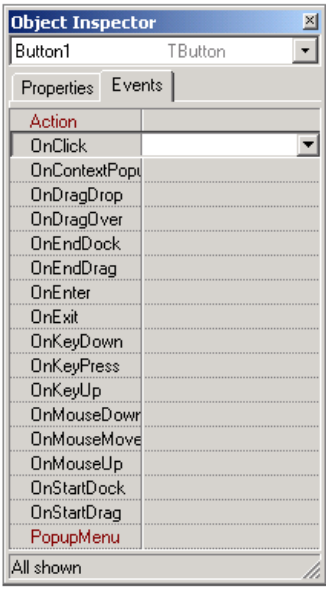

Рис. 1.25. На вкладке **Events** перечислены события, которые может воспринимать компонент

В левой колонке вкладки **Events** перечислены события, которые может воспринимать выбранный компонент. Если для события определена (написана) процедура обработки события, то рядом с именем события отображается имя этой процедуры.

Замечание

Строго говоря, на вкладке **Events** указаны не события, а свойства, значения которых определяют процедуры обработки соответствующих событий.

Для того чтобы создать процедуру обработки события, нужно сделать двойной щелчок мышью в поле редактирования, находящемся рядом с именем соответствующего события. В результате этого в программу будет добавлена процедура обработки события (ее имя появится на вкладке **Events**) и откроется окно редактора кода (рис. 1.26), в котором можно набирать инструкции, реализующие обработку события.

Сформированное Delphi имя процедуры обработки события состоит из двух частей. Первая часть имени идентифицирует форму, содержащую объект (компонент), для которого создана процедура обработки события. Вторая

часть имени — объект и событие. В нашем примере имя формы — `Form1`, имя объекта (командной кнопки) — `Button1`, а имя события — `Click`.

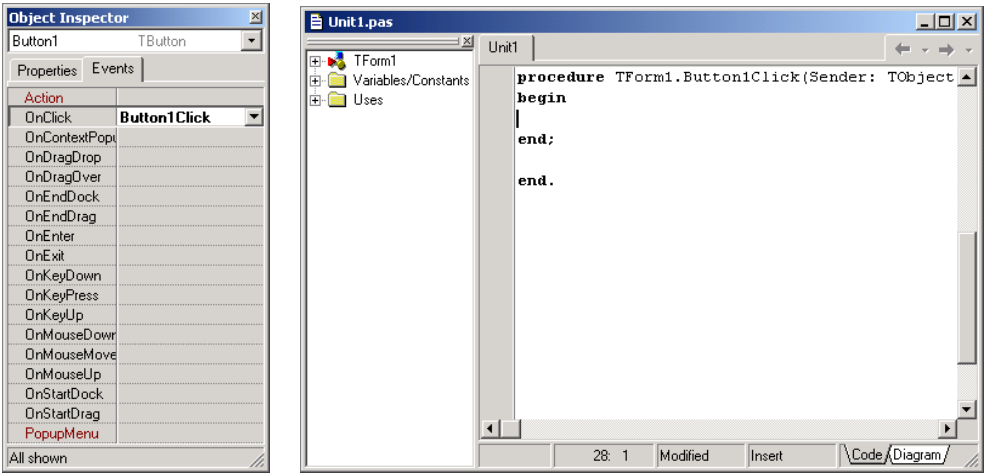

Рис. 1.26. Процедура обработки события, сгенерированная Delphi

Процедура обработки события `Click` для кнопки **Вычислить** приведена в листинге 1.1. Обратите внимание на то, как она представлена. Ее общий вид соответствует тому, как она выглядит в окне редактора кода: ключевые слова выделены полужирным, комментарии — курсивом (выделение выполняет редактор кода). Кроме того, инструкции программы набраны с отступами в соответствии с принятыми в среде программистов правилами хорошего стиля программирования.

Листинг 1.1. Процедура обработки события `Click` на кнопке `Button1`

```
// щелчок на кнопке Вычислить
procedure TForm1.Button1Click(Sender: TObject);
var
    dist : integer; // дистанция, метров
    t:     real;    // время, как дробное число

    min : real;    // время, минуты
    sek : real;    // время, секунды

    v: real;       // скорость
```

```
begin
    // получить исходные данные из полей ввода
    dist := StrToInt(Edit1.Text);
    t := StrToFloat(Edit2.Text);

    // предварительные преобразования
    min := Int(t);       // кол-во минут - это целая часть числа t
    sek := Frac(t) *100; // кол-во секунд - это дробная часть числа t

    // вычисление
    v := (dist/1000) / ((min*60 + sek)/3600);

    // вывод результата
    label4.Caption :=
              'Дистанция: '+ Edit1.Text + ' м' + #13 +
              'Время: ' + FloatToStr(min) + ' мин ' +
                          FloatToStr(sek) + ' сек ' + #13 +
              'Скорость: ' + FloatToStrF(v,ffFixed,4,2) + ' км/час';
end;
```

Приведенная процедура выполняет расчет скорости и выводит результат в поле `Label4`. Исходные данные вводятся из полей редактирования `Edit1` и `Edit2` путем обращения к свойству `Text` (свойство `Text` содержит строку, которая находится в поле редактирования). Преобразование строк в числа выполняют функции `StrToInt` и `StrToFloat`. Функция `StrToInt` преобразует строку в целое число, `StrToFloat` — в дробное. Обе функции проверяют символы строк, переданных им в качестве параметра, на допустимость. Если строка является изображением числа, то функция возвращает значение (число), изображением которого является строка-параметр. После того как исходные данные будут помещены в переменные `dist` и `t`, выполняются подготовительные действия и расчет. Сначала с использованием функции `Int`, которая "отбрасывает" дробную часть числа, выделяется целая часть значения переменной `t` — это количество минут. Затем дробная часть числа `t` (значение функции `Frac`) умножается на 100. Полученное таким образом значение представляет собой количество секунд. После этого выполняется расчет. Так как скорость должна быть выражена в км/час, то значения дистанции и времени, выраженные в метрах и минутах, преобразуются в километры и часы. Вычисленное значение скорости выводится в поле `Label4` путем присваивания значения свойству `Caption`. Для преобразования чисел в строки (свойство `Caption` строкового типа) используются функции `FloatToStr` и `FloatToStrF`.

В результате нажатия кнопки **Завершить** программа должна завершить работу. Чтобы это произошло, надо закрыть окно программы. Делается это при помощи метода `Close`. Процедура обработки события `Click` для кнопки **Завершить** приведена в листинге 1.2.

Листинг 1.2. Процедура обработки события `Click` на кнопке `Button2`

```
// щелчок на кнопке Завершить
procedure TForm1.Button2Click(Sender: TObject);
begin
   Form1.Close; // закрыть главное окно программы
end;
```

Редактор кода

Редактор кода выделяет ключевые слова языка программирования (`procedure`, `var`, `begin`, `end`, `if` и др.) полужирным шрифтом, что делает текст программы более выразительным и облегчает восприятие структуры программы.

Помимо ключевых слов редактор кода выделяет курсивом комментарии.

В процессе разработки программы часто возникает необходимость переключения между окном редактора кода и окном формы. Сделать это можно при помощи командной кнопки **Toggle Unit/Form**, находящейся на панели инструментов **View** (рис. 1.27), или нажатием клавиши <F12>. На этой же панели инструментов находятся командные кнопки **View Unit** и **View Form**, используя которые можно выбрать нужный модуль или форму в случае, если проект состоит из нескольких модулей или форм.

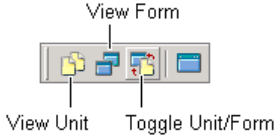

Рис. 1.27. Панель инструментов **View**

Система подсказок

В процессе набора текста программы редактор кода выводит справочную информацию о параметрах процедур и функций, о свойствах и методах объектов.

Например, если в окне редактора кода набрать `MessageDlg` (имя функции, которая выводит на экран окно сообщения) и открывающую скобку, то на экране появится окно подсказки, в котором будут перечислены параметры функции `MessageDlg` с указанием их типа (рис. 1.28). Один из параметров выделен полужирным. Так редактор подсказывает программисту, какой параметр он должен вводить. После набора параметра и запятой в окне подсказки будет выделен следующий параметр. И так до тех пор, пока не будут указаны все параметры.

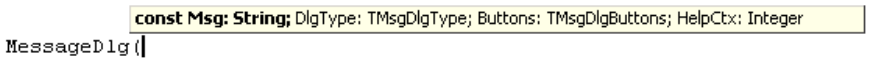

Рис. 1.28. Пример подсказки

Для объектов редактор кода выводит список свойств и методов. Как только программист наберет имя объекта (компонента) и точку, так сразу на экране появляется окно подсказки — список свойств и методов этого объекта (рис. 1.29).

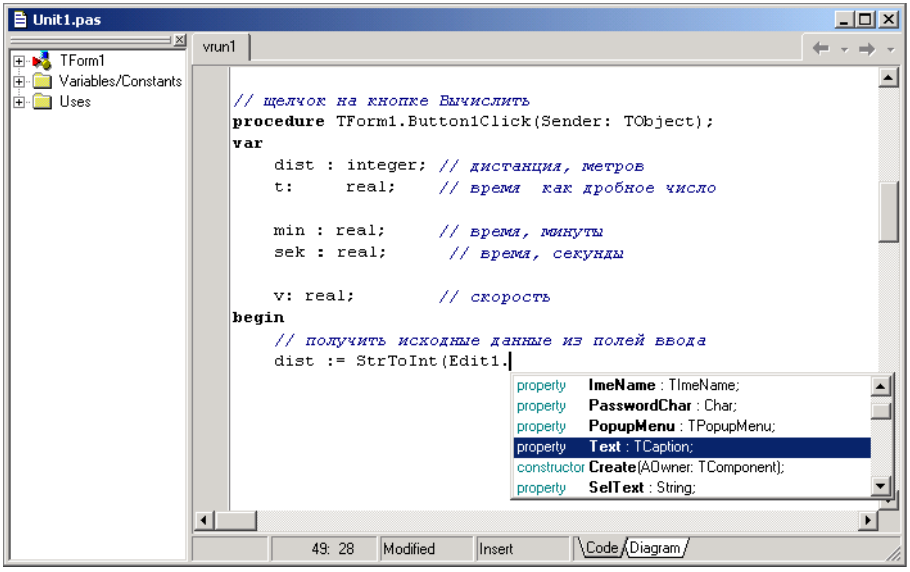

Рис. 1.29. Редактор кода автоматически выводит список свойств и методов объекта (компонента)

Перейти к нужному элементу списка можно при помощи клавиш перемещения курсора или набором на клавиатуре нескольких первых букв имени нужного свойства или метода. После того как будет выбран нужный элемент списка и нажата клавиша <Enter>, выбранное свойство или метод будут вставлены в текст программы.

Система подсказок существенно облегчает процесс подготовки текста программы, избавляет от рутины. Следует обратить внимание, если в процессе набора программы подсказка не появилась, то это значит, что программист допустил ошибку, скорее всего, неверно набрал имя объекта, процедуры или функции.

Навигатор кода

Окно редактора кода разделено на две части (рис. 1.30). В правой части находится текст программы.

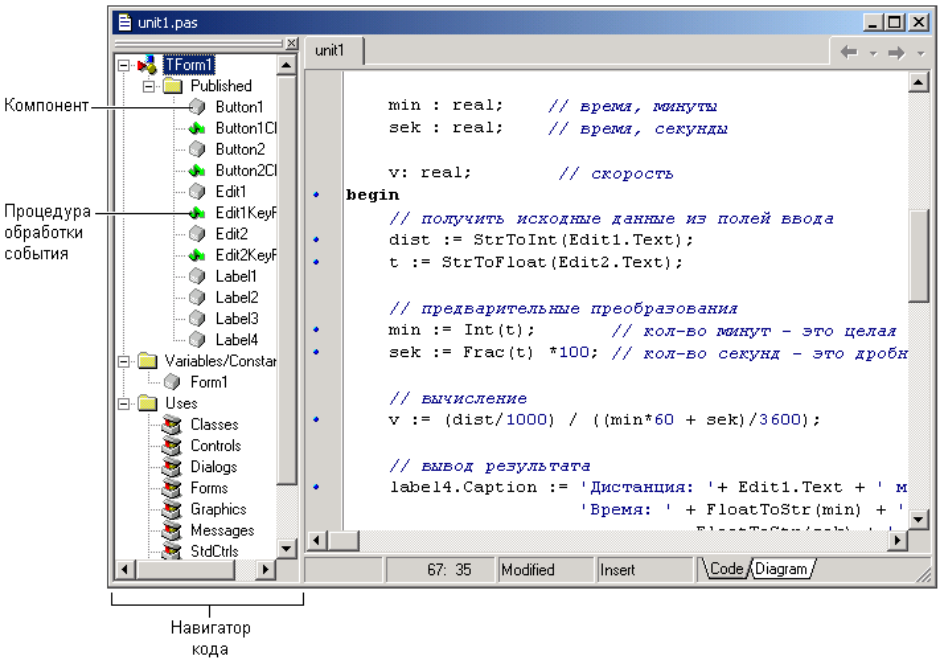

Рис. 1.30. Окно **Code Explorer** облегчает навигацию по тексту программы

Левая часть, которая называется *навигатор кода* (**Code Explorer**), облегчает навигацию по тексту (коду) программы. В иерархическом списке, структура которого зависит от проекта, над которым идет работа, перечислены формы проекта, их компоненты, процедуры обработки событий, функции, процедуры, глобальные переменные и константы. Выбрав соответствующий элемент списка, можно быстро перейти к нужному фрагменту кода.

Окно навигатора кода можно закрыть обычным образом. Если окно навигатора кода не доступно, то для того чтобы оно появилось на экране, нужно из меню **View** выбрать команду **Code Explorer**.

Шаблоны кода

В процессе набора текста удобно использовать *шаблоны кода* (**Code Templates**).

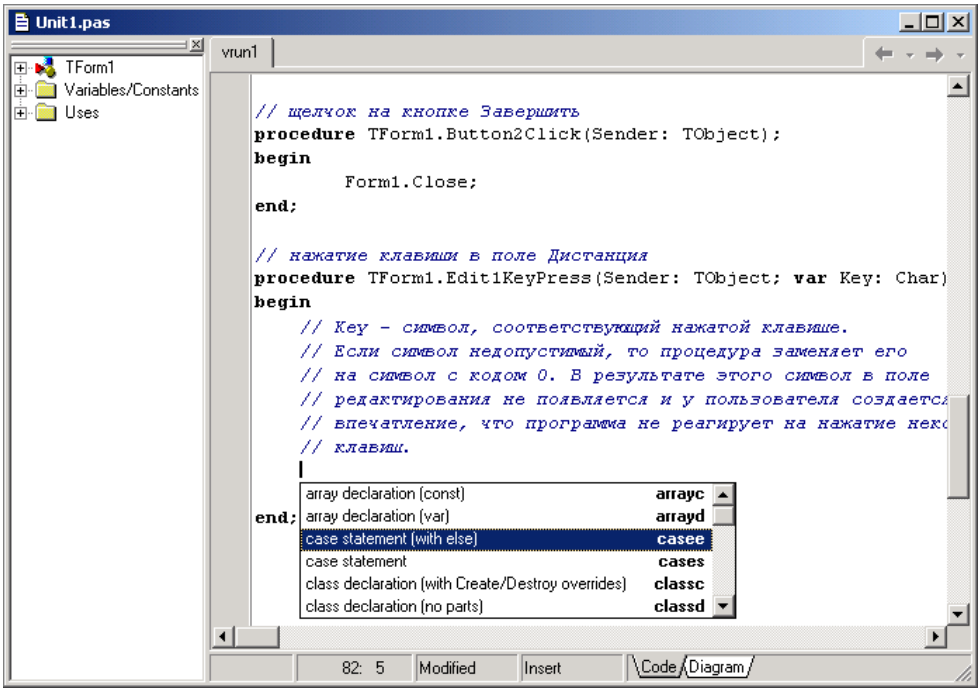

Рис. 1.31. Список шаблонов кода отображается в результате нажатия <Ctrl>+<J>

Шаблон кода — это инструкция программы, записанная в общем виде. Например, шаблон для инструкции `case` выглядит так:

```
case  of
    : ;
    : ;
  else ;
end;
```

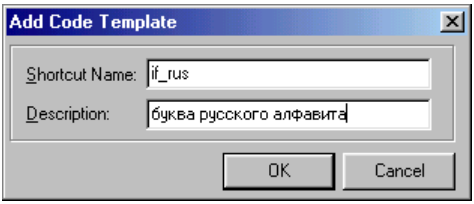

Рис. 1.32. В поля диалогового окна надо ввести имя шаблона и его краткое описание

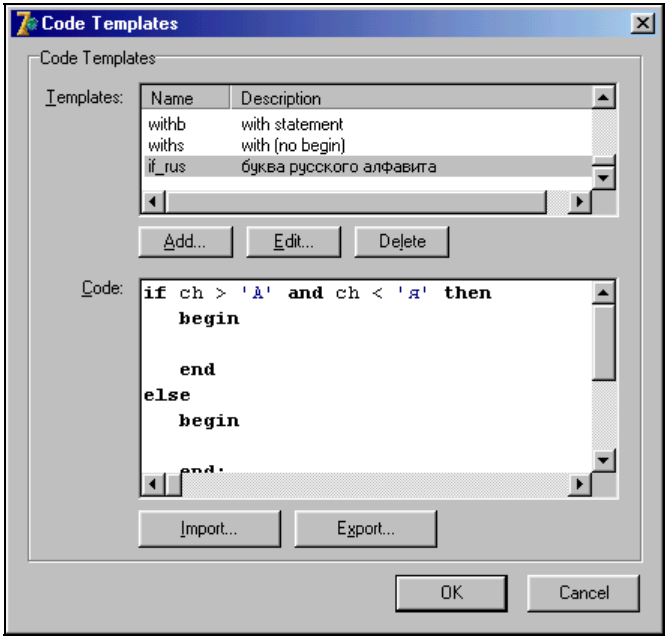

Рис. 1.33. Пример шаблона кода программиста

Редактор кода предоставляет программисту большой набор шаблонов: объявления массивов, классов, функций, процедур; инструкций выбора (`if`,

`case`), циклов (`for`, `while`). Для некоторых инструкций, например `if` и `while`, есть несколько вариантов шаблонов.

Для того чтобы в процессе набора текста программы воспользоваться шаблоном кода и вставить его в текст программы, нужно нажать комбинацию клавиш <Ctrl>+<J> и из появившегося списка выбрать нужный шаблон (рис. 1.31). Выбрать шаблон можно обычным образом, прокручивая список, или вводом первых букв имени шаблона (имена шаблонов в списке выделены полужирным). Выбрав в списке шаблон, нужно нажать клавишу <Enter>, шаблон будет вставлен в текст программы.

Программист может создать свой собственный шаблон кода и использовать его точно так же, как и стандартный. Для того чтобы создать шаблон кода, нужно из меню **Tools** выбрать команду **Editor Options**, на вкладке **Source Options** щелкнуть по кнопке **Edit Code Templates**, в появившемся диалоговом окне **Code Templates** щелкнуть на кнопке **Add** и в появившемся окне **Add Code Template** (рис. 1.32) задать имя шаблона (**Shortcut Name**) и его краткое описание (**Description**). Затем, после щелчка на кнопке **OK**, в поле **Code** диалогового окна **Code Templates** ввести шаблон (рис. 1.33).

Справочная система

В процессе набора программы можно получить справку, например, о конструкции языка или функции. Для этого нужно в окне редактора кода набрать слово (инструкцию языка программирования, имя процедуры или функции и т. д.), о котором надо получить справку, и нажать клавишу <F1>.

Справочную информацию можно получить, выбрав из меню **Help** команду **Delphi Help**. В этом случае на экране появится стандартное окно справочной системы (рис. 1.34). В этом окне на вкладке **Предметный указатель** нужно ввести ключевое слово, определяющее тему, по которой нужна справка. Как правило, в качестве ключевого слова используют первые несколько букв имени функции, процедуры, свойства или метода.

Структура проекта

Проект Delphi представляет собой набор программных единиц — модулей. Один из модулей — главный, содержит инструкции, с которых начинается выполнение программы. Главный модуль приложения полностью формируется Delphi.

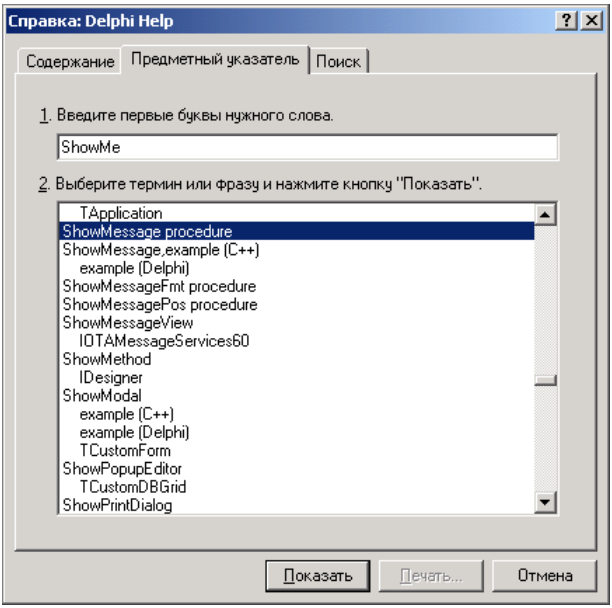

Рис. 1.34. Поиск справочной информации по ключевому слову

Главный модуль представляет собой файл с расширением dpr. Для того чтобы увидеть текст главного модуля приложения, нужно в меню **Project** выбрать команду **View Source**.

В листинге 1.3 приведен текст главного модуля программы вычисления скорости бега.

Листинг 1.3. Главный модуль приложения *Скорость*

```
program speed;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

Начинается главный модуль словом `program`, за которым следует имя программы, которое совпадает с именем проекта. Имя проекта задается в момент сохранения проекта, и оно определяет имя создаваемого компилятором исполняемого (exe) файла программы. Далее за словом `uses` следуют имена используемых модулей: библиотечного модуля `Forms` и модуля формы `Unit1.pas`.

Строка `{$R*.RES}`, которая похожа на комментарий, — это директива компилятору "подключить файл ресурсов". Файл ресурсов содержит *ресурсы* приложения: пиктограммы, курсоры, битовые образы и др. Звездочка показывает, что имя файла ресурсов такое же, как и у файла проекта, но с расширением res.

Файл ресурсов не является текстовым, поэтому увидеть его содержимое с помощью редактора текста нельзя. Для работы с файлами ресурсов используют специальные программы. Например, с Delphi поставляется утилита Image Editor, позволяющая работать res-файлами.

Исполняемая часть главного модуля находится между инструкциями `begin` и `end`. Инструкции исполняемой части обеспечивают инициализацию приложения, создание стартовой формы и запуск программы.

Помимо главного модуля каждая программа включает в себя еще, как минимум, один модуль формы, который содержит описание стартовой (главной) формы приложения и поддерживающих ее работу процедур.

В листинге 1.4 приведен модуль формы программы **Скорость**.

Листинг 1.4. Модуль главной формы программы *Скорость*

```
unit unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
```

```
    Label3: TLabel;
    Label4: 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}

// щелчок на кнопке Вычислить
procedure TForm1.Button1Click(Sender: TObject);
var
   dist : integer; // дистанция, метров
   t    : real;    // время, как дробное число
   min  : real;    // время, минуты
   sek  : real;    // время, секунды
   v    : real;    // скорость
begin
   // получить исходные данные из полей ввода
   dist := StrToInt(Edit1.Text);
   t := StrToFloat(Edit2.Text);

   // предварительные преобразования
   min := Int(t);       // кол-во минут - это целая часть числа t
   sek := Frac(t) *100; // кол-во секунд - это дробная часть числа t

   // вычисление
   v := (dist/1000) / ((min*60 + sek)/3600);

   // вывод результата
   label4.Caption :=
                'Дистанция: '+ Edit1.Text + ' м' + #13 +
                'Время: ' + FloatToStr(min) + ' мин ' +
```