bhv
www.bhv.ru
www.bhv.kiev.ua

Никита Культин

Delphi 6

Программирование на Object Pascal

САМОУЧИТЕЛЬ

Работа в среде Delphi 6

Графика, мультимедиа и базы данных

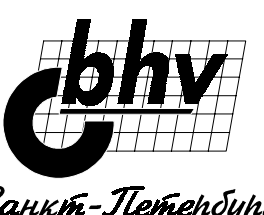
Использование MS HTML Help Workshop

Содержит примеры проектов

Никита Культин

Delphi 6.

Программирование на Object Pascal



Дюссельдорф ◆ Киев ◆ Москва ◆ Санкт-Петербург

УДК 681.3.06

Книга является руководством по программированию в среде Delphi 6. В ней рассматривается весь процесс создания программы: от разработки алгоритма и диалогового окна до отладки и создания справочной системы. Материал включает ряд тем, которые, как правило, остаются за рамками книг, адресованных начинающим программистам: обработка символьной информации, использование динамических структур, работа с файлами, создание справочной системы. Рассматриваются вопросы работы с графикой, мультимедиа и базами данных. Приведено описание процесса создания анимации в Macromedia Flash 5, создание справочной системы при помощи программы Microsoft HTML Help Workshop.

Книга отличается доступностью изложения, большим количеством наглядных примеров и адресована студентам, школьникам старших классов и всем изучающим программирование в учебном заведении или самостоятельно. Прилагаемая дискета содержит примеры приведенных в книге программ.

Для начинающих программистов

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зав. редакцией	*Наталья Таркова*
Редактор	*Михаил Кокорев*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Культин Н. Б.

Delphi 6. Программирование на Object Pascal. — СПб.: БХВ-Петербург, 2001. — 528 с.: ил.

ISBN 5-94157-112-7

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 24.07.01.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 42,57.
Тираж 5000 экз. Заказ
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар, № 77.99.1.953.П.950.3.99
от 01.03.1999 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН.
199034, Санкт-Петербург, 9-я линия, 12.

Содержание

Предисловие

Delphi — что это?

Интерес к программированию постоянно растет. Это связано с развитием и внедрением в повседневную жизнь информационных технологий. Если человек имеет дело с компьютером, то рано или поздно у него возникает желание, а иногда и необходимость, научиться программировать.

Среди пользователей персональных компьютеров в настоящее время наиболее популярно семейство операционных систем Windows и, естественно, что 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 генерации кода программы, 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.

Примечание

Delphi 6 может работать в среде операционных систем от Windows 98 до Windows 2000. Особых требований к ресурсам компьютера система не предъявляет: процессор должен быть типа Pentium или Celeron, оперативной памяти — не менее 32 Мбайт, достаточное количество свободного дискового пространства — порядка 200 Мбайт.

Об этой книге

В книге, которая посвящена программированию в конкретной среде разработки, необходим баланс между тремя линиями: язык программирования, программирование как таковое, среда разработки. Уже при первом знакомстве со средой разработки, представления ее возможностей у автора возникает проблема: чтобы объяснить процесс разработки программы, описать, 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 лучший вариант. Поэтому при изложении материала принят подход, в основу которого положен принцип соблюдения баланса между Object Pascal, методами 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 Object Pascal или среды разработки Delphi. Данная книга представляет собой учебное пособие по программированию на языке Object Pascal в среде Delphi 6. В ней рассмотрен весь процесс создания программы: от разработки диалогового окна и функций обработки событий до создания справочной системы и установочной дискеты.

Цель создания книги может быть сформулирована так: научить программировать в среде Delphi, т. е. создавать законченные программы различного назначения: от игровых до профессиональных утилит.

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 вы научитесь!

Введение

На примере программы вычисления скорости бега демонстрируется технология визуального проектирования и событийного программирования, вводятся основные понятия и термины.

Начало работы

Запускается Delphi обычным образом (рис. 1), т. е. выбором из меню **Borland Delphi 6** команды **Delphi 6**.

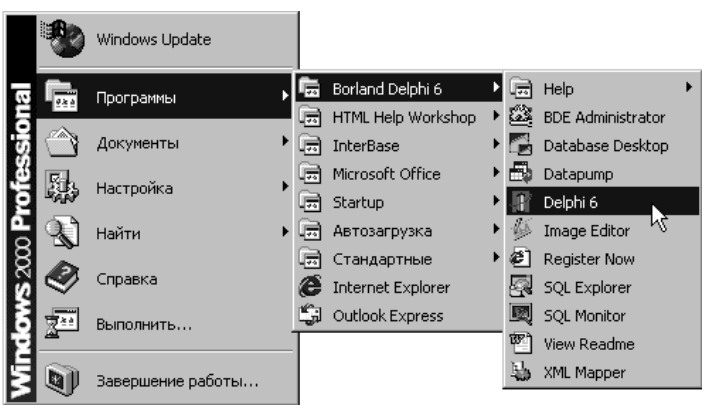


Рис. 1. Запуск Delphi

Вид экрана после запуска Delphi несколько необычен (рис. 2). Вместо одного окна на экране появляются пять:

- главное окно (**Delphi 6**);
- окно стартовой формы (**Form 1**);

- окно редактора свойств объектов (**Object Inspector** (Инспектор объектов));
- окно просмотра списка объектов (**Object TreeView** (Просмотр дерева объектов));
- окно редактора кода (**Unit1.pas**).

Окно редактора кода почти полностью закрыто окном стартовой формы.

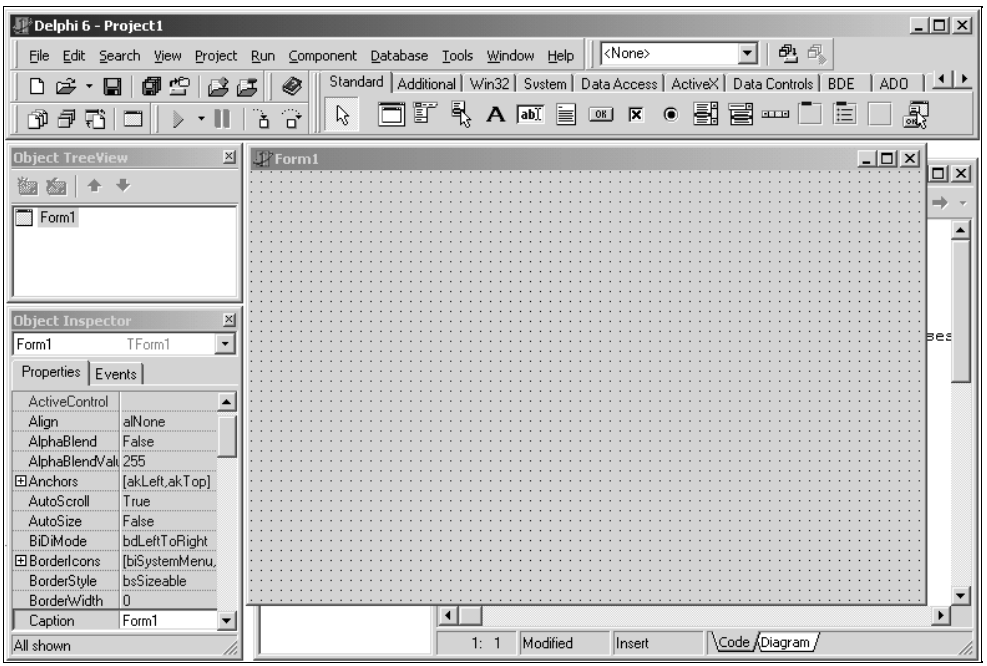


Рис. 2. Вид экрана после запуска Delphi

В главном окне (рис. 3) находится меню команд, панели инструментов и палитра компонентов.

Окно стартовой формы представляет собой заготовку главного окна разрабатываемого приложения.

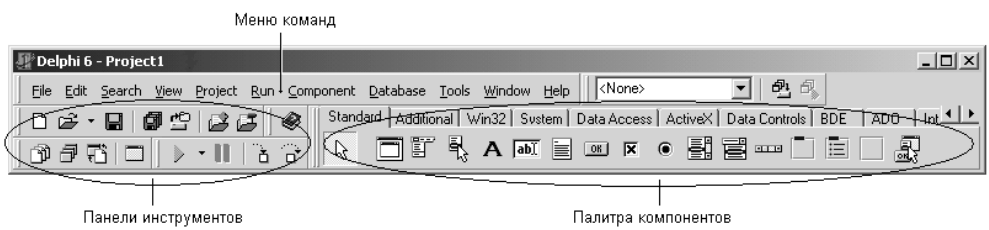
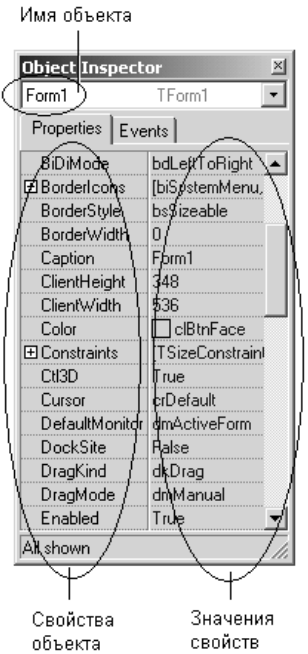


Рис. 3. Главное окно

Примечание

Программное обеспечение принято делить на системное и прикладное. Системное программное обеспечение — это все то, что составляет операционную систему. Остальные программы принято считать прикладными. Для краткости прикладные программы называют приложениями.



Окно **Object Inspector** (рис. 4) — окно редактора свойств объектов предназначено для редактирования значений свойств объектов. В терминологии визуального проектирования *объекты* — это диалоговые окна и элементы управления (поля ввода и вывода, командные кнопки, переключатели и др.). *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.

Рис. 4. Окно редактора свойств объекта

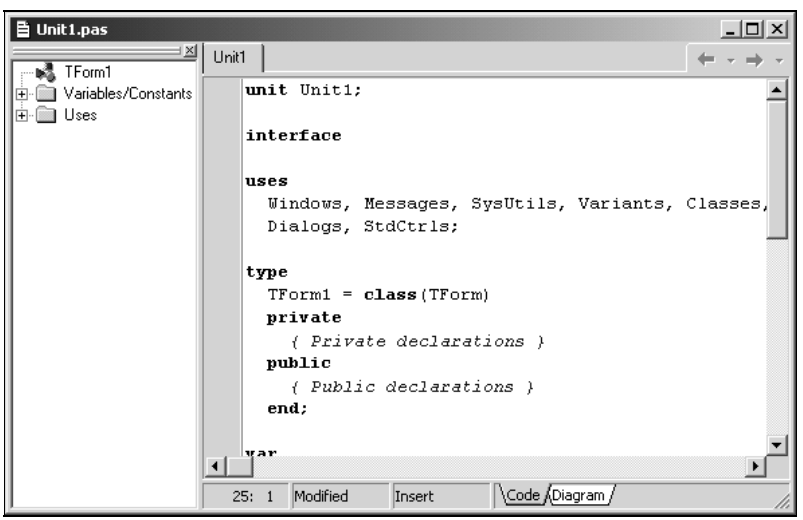


Рис. 5. Окно редактора кода

В окне редактора кода (рис. 5), которое можно увидеть, отодвинув в сторону окно формы, следует набирать текст программы. В начале работы над новым проектом это окно редактора кода содержит сформированный Delphi шаблон программы.

Первый проект

Для демонстрации возможностей Delphi и технологии визуального проектирования разработаем программу, вычисляющую скорость, с которой спортсмен пробежал дистанцию. Окно программы во время ее работы приведено на рис. 6.

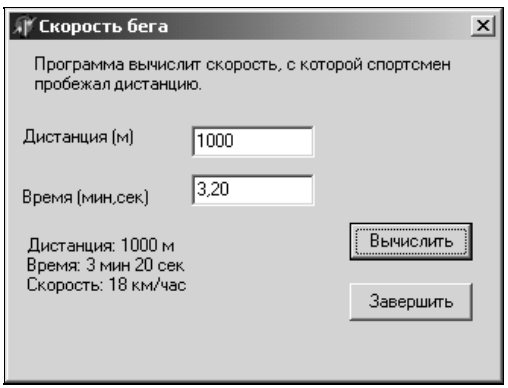


Рис. 6. Окно программы вычисления скорости бега

Для начала работы над новой программой запустите Delphi. Если вы уже работаете в среде разработки и у вас загружен другой проект, выберите из меню **File** (Файл) команду **New/Application** (Создать/Приложение).

Форма

Работа над *новым проектом*, так в Delphi называется разрабатываемое приложение, начинается с создания стартовой формы.

На этапе разработки программы диалоговые окна называют *формами*.

Стартовая форма создается путем изменения значений свойств формы **Form1** и добавления к форме необходимых компонентов (полей ввода и вывода текста, командных кнопок).

Свойства формы (табл. 1) определяют ее внешний вид: размер, положение на экране, текст заголовка, вид рамки.

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 мо-

мент. В левой колонке вкладки **Properties** (Свойства) перечислены свойства объекта, а в правой — указаны их значения.

Таблица 1. *Свойства формы*

Свойство	Описание
`Name`	Имя формы. В программе имя формы используется для управления формой и доступа к компонентам формы
`Caption`	Текст заголовка
`Width`	Ширина формы
`Height`	Высота формы
`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 (`biZizeable`), тонкой (`biSingle`) или отсутствовать (`biNone`). 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 **Свернуть** и кнопки системного меню, `biMinimaze` — кнопки **Свернуть**, `biMaximaze` — кнопки **Развернуть**, `biHelp` — кнопки вывода справочной информации
`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. 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)

При создании формы в первую очередь следует изменить значение свойства Caption (Заголовок). В нашем примере надо заменить текст Form1 на "Скорость бега". Чтобы это сделать, нужно в окне **Object Inspector** щелкнуть мышью в строке Caption, в результате чего будет выделено текущее значение свойства, в строке появится курсор, и можно будет ввести текст "Скорость бега" (рис. 7).

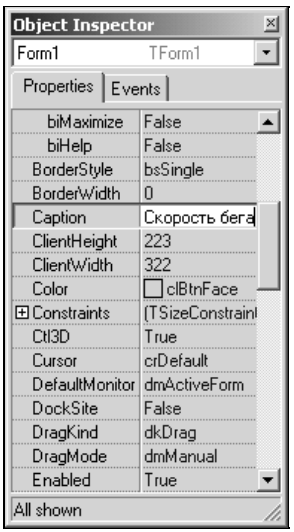


Рис. 7. Изменение значения свойства Caption

Аналогичным образом можно установить значения свойств Height и Width, которые определяют высоту и ширину формы. Этим свойствам надо присвоить значения 250 и 330, соответственно.

Примечание

Размер формы и ее положение на экране, а также размер других элементов управления и их положение на поверхности формы, задают в пикселах, т. е. точках экрана.

Форма — это обычное окно. Поэтому его 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.

Положение диалогового окна на экране после запуска программы соответствует положению формы во время ее разработки, 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.

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. 8).

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. 9). 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.

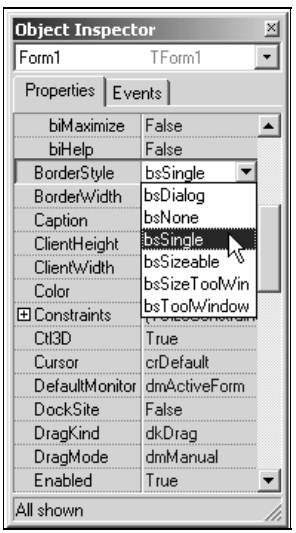


Рис. 8. Установка значения свойства путем выбора из списка

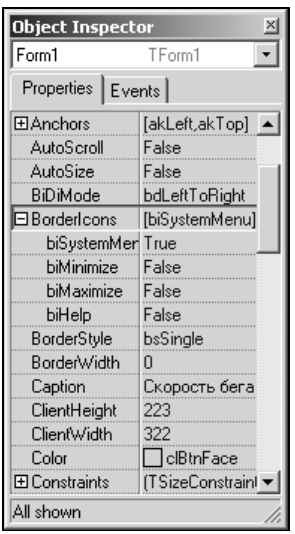


Рис. 9. Раскрытый список вложенных свойств сложного свойства `BorderIcons`

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

Таблица 2

Свойство	**Значение**
`Caption`	Скорость бега
`Height`	250

Таблица 2 *(окончание)*

Свойство	Значение
`Width`	330
`BorderStyle`	BsSingle
`BorderIcons.biMinimize`	False
`BorderIcons.biMaximize`	False
`Font.Size`	10

В приведенной таблице в именах некоторых свойств есть точка. Это значит, что надо задать значение уточняющего свойства. После того как будут установлены значения свойств главной формы, она должна иметь вид, приведенный на рис. 10.

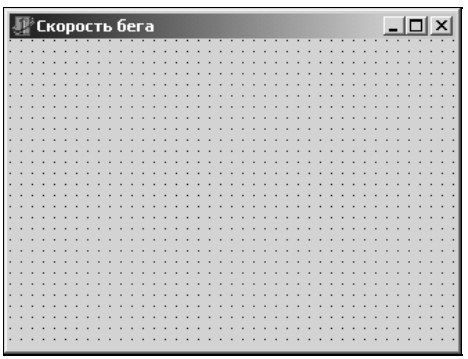


Рис. 10. Так выглядит форма после установки значений свойств

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. 11).

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

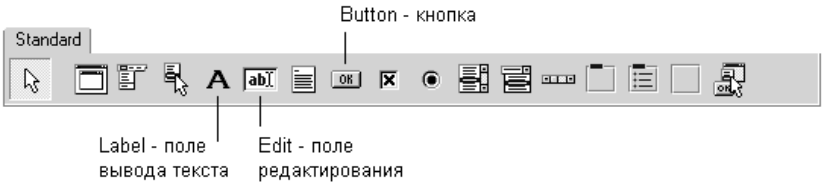


Рис. 11. Вкладка **Standard** содержит наиболее часто используемые компоненты

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, и 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. 12 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.

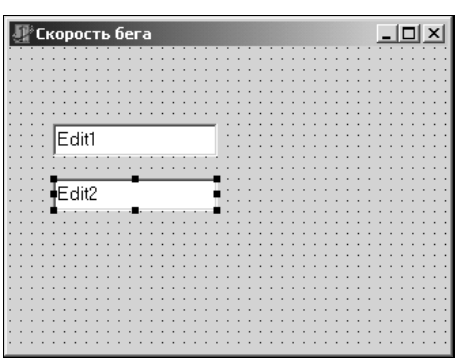


Рис. 12. Форма после добавления двух компонентов `Edit`

В табл. 3 перечислены основные свойства компонента `Edit` — поля редактирования.

Таблица 3

Свойство	Описание
`Name`	Имя компонента. Используется в программе для доступа к компоненту и его свойствам, в частности — для доступа к тексту, введенному в поле редактирования
`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. 13) 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. 14).

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. 15) или из списка в окне **Object TreeView** (рис. 16).

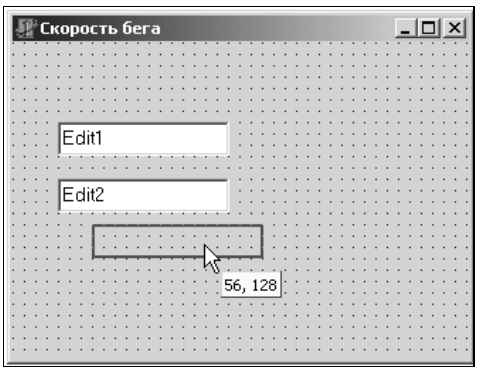


Рис. 13. Отображение текущих значений свойств `Left` и `Top` при изменении положения компонента

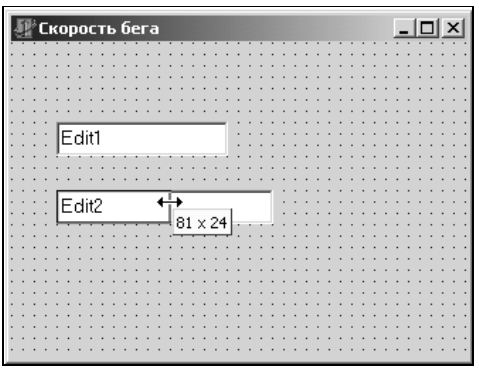


Рис. 14. Отображение текущих значений свойств `Height` и `Width` при изменении размера компонента

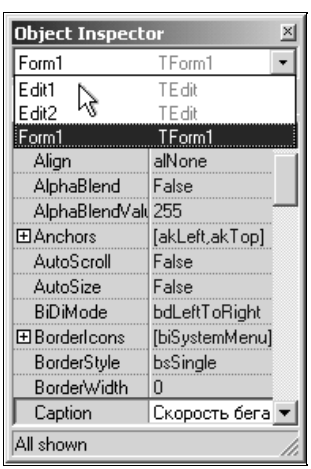


Рис. 15. Выбор компонента из списка в окне **Object Inspector**

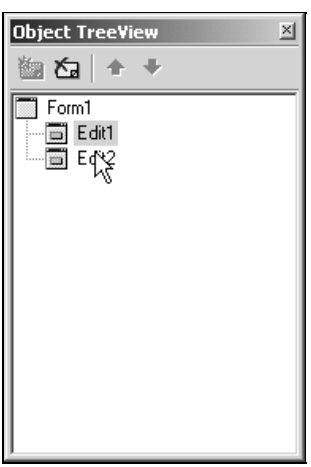


Рис. 16. Выбор компонента в окне **Object TreeView**

В табл. 4 приведены значения свойств полей редактирования `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 4

Name	Edit1	Edit2
Text		
Top	56	88
Left	128	128
Height	21	21
Width	121	121

Помимо полей редактирования в окне программы должна находиться краткая информация о программе и назначении полей ввода. Для вывода текста в форму используют поля вывода текста. Поле вывода текста — это компонент `Label`. Значок компонента `Label` находится на вкладке **Standard** (рис. 17). Добавляется компонент `Label` в форму точно так же, как и поле редактирования.

Рис. 17. Компонент `Label` — поле вывода текста

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 — для вывода результата расчета — скорости.

Свойства компонента `Label` перечислены в табл. 5.

Таблица 5

Свойство	Пояснение
Name	Имя компонента. Используется в программе 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

Таблица 5 *(окончание)*

Свойство	Пояснение
`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`. Эти свойства 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. 6, форма программы принимает вид, приведенный на рис. 18.

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.

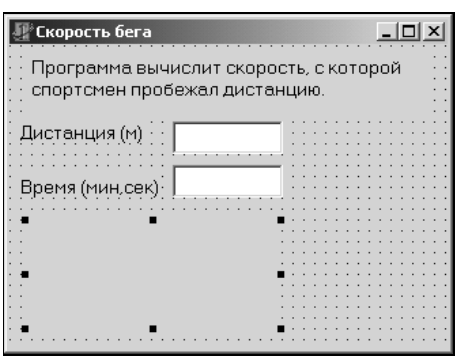


Рис. 18. Вид формы после добавления полей вывода текста

Таблица 6

Свойство	Компонент			
Name	Label1	Label2	Label3	Label4
AutoSize	False	True	True	False
WordWrap	True	False	False	True
Caption	Программа вычислит скорость, с которой спортсмен пробежал дистанцию	Дистанция (метров)	Время (мин, сек)	
Top	8	56	88	120
Left	8	8	8	8
Height	33	13	13	41
Width	209	102	832	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. 19). Свойства компонента приведены в табл. 7.

Рис. 19. Командная кнопка — компонент `Button`

Таблица 7. *Свойства компонента* `Button`

Свойство	Описание
Name	Имя компонента. Используется в программе 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`
Left	Расстояние от левой границы кнопки до левой границы формы
Top	Расстояние от верхней границы кнопки до верхней границы формы

Таблица 7 *(окончание)*

Свойство	Описание
Height	Высота кнопки
Width	Ширина кнопки

После добавления к форме двух командных кнопок нужно установить значения их свойств в соответствии с табл. 8.

Таблица 8

Name	Button1	Button2
Caption	Вычислить	Завершить
Top	176	176
Left	16	112
Height	25	25
Width	75	75

Окончательный вид формы разрабатываемого приложения приведен на рис. 20.

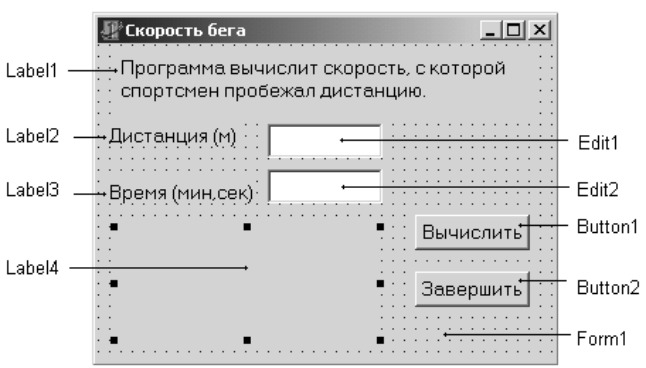


Рис. 20. Форма программы **Скорость бега**

Завершив работу по созданию формы приложения, можно приступить к написанию текста программы. Но перед этим обсудим очень важные при программировании под Windows понятия:

- ❒ событие;
- ❒ процедура обработки события.

Событие и процедура обработки события

Вид созданной формы подсказывает, как работает приложение. Очевидно, что пользователь должен ввести в поля редактирования исходные данные и щелкнуть мышью на кнопке **Вычислить**. Щелчок на изображении командной кнопки — это пример того, что в Windows называется *событием*.

Событие (`Event`) — это то, что происходит во время работы программы. В Delphi каждому событию присвоено имя. Например, щелчок кнопкой мыши — это событие `OnClick`, двойной щелчок мышью — событие `OnDblClick`.

В табл. 9 приведены некоторые события Windows.

Таблица 9

Событие	Происходит
`OnClick`	При щелчке кнопкой мыши
`OnDblClick`	При двойном щелчке кнопкой мыши
`OnMouseDown`	При нажатии кнопки мыши
`OnMouseUp`	При отпускании кнопки мыши
`OnMouseMove`	При перемещении мыши
`OnKeyPress`	При нажатии клавиши клавиатуры
`OnKeyDown`	При нажатии клавиши клавиатуры. События `OnKeyDown` и `OnKeyPress` — это чередующиеся, повторяющиеся события, которые происходят до тех пор, пока не будет отпущена удерживаемая клавиша (в этот момент происходит событие `OnKeyUp`)
`OnKeyUp`	При отпускании нажатой клавиши клавиатуры
`OnCreate`	При создании объекта (формы, элемента управления). Процедура обработки этого события обычно используется для инициализации переменных, выполнения подготовительных действий
`OnPaint`	При появлении окна на экране в начале работы программы, после появления части окна, которая, например, была закрыта другим окном и в других случаях
`OnEnter`	При получении элементом управления фокуса
`OnExit`	При потере элементом управления 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 соответ-

ствующего 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 на событие отличается от стандартной или не определена. Например, если по условию задачи ограничений на символы, вводимые в поле `Edit`, нет, то процедуру обработки события `onKeyPress` 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 `OnClick` для командной кнопки **Вычислить**.

Чтобы приступить к созданию процедуры обработки события, надо сначала в окне **Object Inspector** выбрать компонент, для которого создается процедура обработки события. Затем, в этом же окне нужно выбрать вкладку **Events** (События).

В левой колонке вкладки **Events** (рис. 21) перечислены имена событий, которые может воспринимать выбранный компонент (объект). Если для события определена (написана) процедура обработки события, то в правой колонке, рядом с именем события, выводится имя этой процедуры.

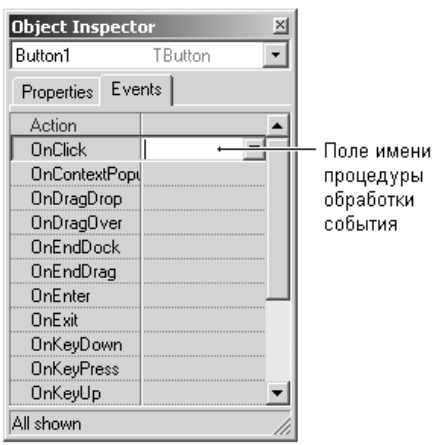


Рис. 21. На вкладке **Events** перечислены события, которые может воспринимать компонент

Для того чтобы создать функцию обработки события, нужно сделать двойной щелчок мышью в поле имени процедуры обработки соответствующего события. В результате этого откроется окно редактора кода, в которое будет добавлен шаблон функции обработки события, а в окне **Object Inspector**, рядом с именем события, появится имя функции его обработки (рис. 22).

Delphi присваивает функции обработки события имя, которое состоит из двух частей. Первая часть имени идентифицирует форму, содержащую объект (компонент), для которого создана процедура обработки события. Вторая часть имени идентифицирует сам объект и событие. В нашем примере имя формы — `Form1`, имя командной кнопки— `Button1`, а имя события — `Click`.

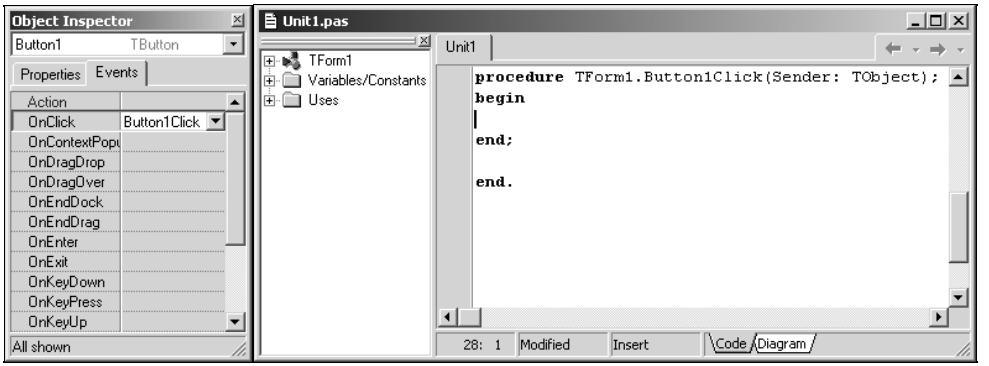


Рис. 22. Пример процедуры обработки события, сгенерированной Delphi

В окне редактора кода между словами `begin` и `end` можно печатать инструкции, реализующие функцию обработки события.

В листинге 1 приведен текст функции обработки события `OnClick` для командной кнопки **Вычислить**. Обратите внимание на то, как представлена программа. 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.

Листинг 1. Процедура обработки события `OnClick` на кнопке Вычислить

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

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

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

    // предварительные преобразования
    min := Trunc(t);  // кол-во минут — это целая часть числа t
    sek := Trunc(t*100) mod 100; // кол-во секунд — это дробная часть
                                 // числа t
```