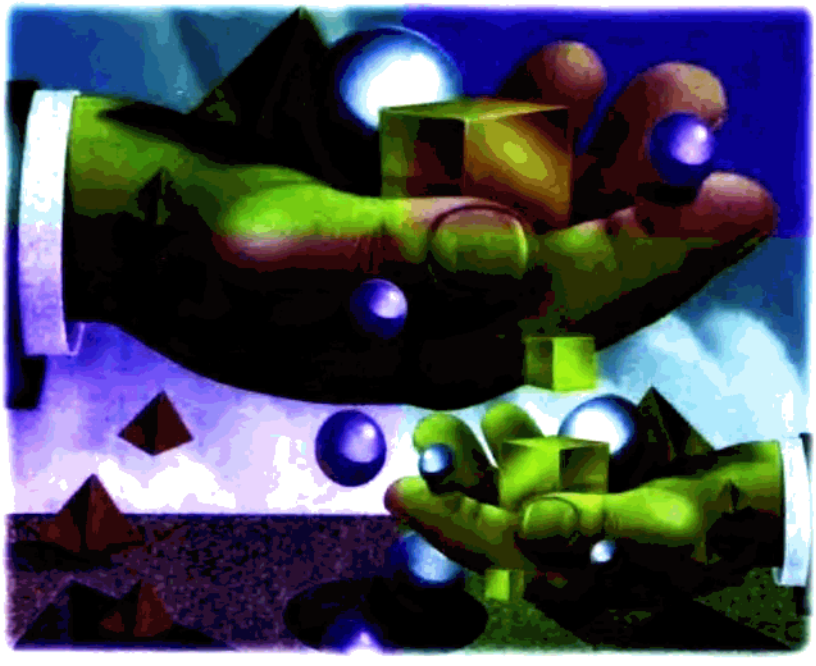
青松

Delphi 5

基础应用与在线数据库

余显强 编著



青岛出版社

出版者的话

有史以来，没有哪一门科学能像电脑这样飞速发展！新技术层出不穷，新产品不断涌现，电脑工作者必须不断学习、更新知识，才能跟上形势，不被淘汰。然而人们的精力是有限的，面对良莠不齐、铺天盖地而来的各种电脑著述和技术资料，你不可能有很多的时间一一鉴别和阅读。这时就需要专家们根据自己的实践经验给以精选和引导。

为此，青岛出版社聘请了具有丰富教学经验和实践经验的专家，组成《青岛松岗电脑图书》编委会，向广大读者介绍适合我国国情的、最新最实用的电脑及网络技术。

《青岛松岗电脑图书》编委会对这套丛书的质量负责，并郑重承诺：编、校、印刷质量符合国家新闻出版署的质量要求——差错率低于万分之一。

第一章　Delphi 简介

Delphi 是一个快速应用程序开发(Rapid Application Development，简称 RAD)工具，配合视觉的开发环境，让您迅速开发出具有效率的窗口应用程序。Delphi 采用“Object Pascal”为发展语言，并且加入视觉化的发展环境，通过其所提供的工具与编译器，使能快速产生与测试使用者所开发的雏形(Prototype)，或调整系统成为实际应用上的需要。

Delphi 可开发各种不同层面的应用程序，从一般性的公用程序到复杂的数据库应用系统程序。通过 Delphi 所提供的数据库存取元件(Dataset Component)，可迅速完成基本的数据库与主从架构的应用程序，并可利用数据显示元件，在设计时期(Designtime)即可看到实际应用程序界面的数据查询与改变的成果。

Delphi 3.0 版加强了因特网(World Wide Web，简称 WWW)开发功能。不仅简化开发的流程、降低系统的复杂性，更可涵盖各种不同的 WWW 应用程序标准，如：CGI、ISAPI、NSAPI 等。另外也提供了产生 ActiveX 元件与 ActiveForm 的功能。

当今，Multi-tier 的应用，对于日趋庞大复杂的应用程序开发，可以说是最佳的解决方案。而目前市场上也有部分能提供 Multi-tier 的工具，不过大都只是提供了程序分割的能力。这些工具只是借由 DCOM 或 CORBA 等分散运算的标准，将程序逻辑分割成数个部分，利用网络上的资源来分散运算。有些则是简单地利用现有 Internet Web 的方式来进行 3-Tier 的应用，离实际的应用面还有一段的距离。Delphi 不但能利用上述方式来开发应用程序，而且 Delphi3 的 Multi-tier 技术，会让开发此类型的应用程序变得更轻松方便与强大。

一般的开发工具如果要开发 Multi-tier 的数据库应用程序时，所有前端与数据库的连接以及每一个 Tier 间的信息传送，开发人员都需要自行撰写程序来完成这些繁杂的运作，不但费时而且还需要另外学习难度颇高的新开发技术，维护也相当困难。Delphi 3 在开发 Multi-tier 的数据库应用程序时，开发人员不需要了解与撰写其中的程序码，只要沿用既有的开发技术就可以完成所有工作。而应用程序可以由应用程序伺服器直接下载，前端的环境不需要任何的组态设定。配合上提供安全自动回复、负载平衡和减低网络负载的数据验证等技术，不管是开发或是维护所需要的成本都可以大为缩减。(节录自一通网站 http://www.atc.com.tw/intro/atc-tr/qa/delqa.htm 有关 Delphi3.0 问答集)。

一、Delphi 产品类型

① Delphi 3.0 Desktop 标准型版本，适合一般个人开发一般性的应用程序。

② Delphi 3.0 Developer，适合网络上或一般开发数据库应用系统。比 Desktop 版本增加数据字典(Data Dictionary)、ActiveForms、InstallShield Express 与 Open Tools API 等功能，并且包含单机的 InterBase 数据库系统。

③ Delphi 3.0 Client/Server Suite，适合开发主从架构的数据库应用系统。比 Developer 版本增加 Multi-Tier 架构的相关建置工具、WebServer 应用程序，以及一套包含四个使用权的 InterBase 数据库系统，并提供主从架构的运算环境等。

Delphi 3 的基本系统需求如下：

相容于不少于 Intel 486/66 MHz 的 CPU；

在 MicroSoft Windows 95，Windows NT 4.0 或 NT 3.51 上执行；

至少要有 12 Mb 的 RAM(建议 16 Mb 以上)；

配置 CD-ROM 驱动器；

VGA 或更高解析度的显示器；

鼠标或其他的指标设备。

有关 Delphi3 的新增功能或与其他开发工具的比较等等，我们不在此赘述。如果您有兴趣，可以浏览台湾宝兰(Borland)公司的首页 www.borland.com.tw，或产品代理商：兴德信息 www.sinter.com.tw 或一通科技 www.atc.com.tw 的网站内容，会有非常丰富的比较说明。

二、专有名词

(1) 原生码(Native Code)

Delphi 包含内建的 32 位原生码编译器，也就是编译成所谓的机械码。正由于它能产生独立且完备、原生可执行的应用程序文件(.EXE)，而不需要在执行期间(Runtime)载入动态连接程序库 (Dynamic Link Library，简称 DLL)，所以其执行速度比其他需载入 DLL 的直译式程序快好几倍。

(2) 视觉化元件(Visual Component Library，简称 VCL)

当 Borland 公司在 Turbo Pascal For Windows 推出了 Object Windows Library (OWL)时，提供了大幅简化 Windows 程序设计的工作。OWL 对象会自动完成许多必须由设计者自行撰写的程序码。而 Delphi 所采用的 VCL 可以说是 OWL 的后继者。虽然它采用与 OWL 相似的对象模型原理，但在实际使用上仍有极大的差异，因为最初是为了 Delphi 的视觉环境所制订的。它包含了许多不同应用目的的元件，使程序开发人员可以直接在 Delphi 中修改元件的外观与动作。这种视觉化的设计方式，取代了传统的尚在程序码中建立窗口和对话框与加入执行动作的作法。

元件是 Delphi 程序开发人员用来设计使用者界面(User Interface) 及提供一些非视觉化功能的根本。对元件撰写人员而言，元件就是 Object Pascal 程序中的对象，将预计提供的功能、外观、属性与事件等等处理程序封装起来，成为一个元件，提供程序开发人员运用。

(3) 面向对象(Object Orient)

面向对象的主要精神，是将数据和使用此数据的函数定义成一个新的类别(Class)，也就是一个新的数据型态(Data Type)。而将此类别的数据宣告成为变量即称为对象(Object)。

(4) 事件驱动(Event Driven)

事件驱动是一种基于当前问题状态的前向链接问题求解方法。Windows 就是一种“事件驱动”的操作系统，也就是当事件发生时，才会去执行程序。其执行过程如下：

事件：使用者按下表格中的按钮。(图 1.1)

图 1.1

图 1.2

信息：按钮收到窗口送出的 Click 信息，触发此按钮的 OnClick 事件。

驱动：执行事件指定的程序。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Edit1.Text: ='你好！！';
end;
```

结果：数据栏显示出“你好！！”的内容。(图 1.2)

我们可以参考图 1.3 所示，逐一说明事件执行的过程：

① 当使用者按下表格中的按钮(假设该按钮名称为 Button1)

② 操作系统产生一个“OnClick”的信息给表格中的按钮 Button1。告诉 Button1 按钮“你被 Click 了”。

③ Button1 按钮接收到“OnClick”这一个信息，到 Button1 按钮本身的“事件处理区”查看要做何种处理程序?

④ 在 Button1 按钮的“事件处理区”中发现“当被 Click(即 OnClick)时，必须执行 TForm1.Button1Click 程序”。

⑤ 执行“TForm1.Button1Click 程序”。

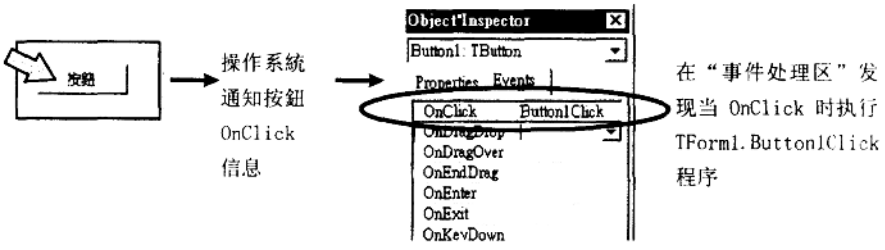


图 1.3

Delphi 所开发出来的系统是属于面向对象的程序，一个事件可以针对不同的对象。当然，不同的事件也可以针对同一个对象。不管对应哪一种方式，每一个事件都对应一个事件处理程序(Event procedure)。如果希望某一对象收到某一事件后能执行所需的程序，只要

将该程序放在对应的事件中即可。

(5) 属性(Property)

属性提供了程序开发人员对元件内部栏位(Field)的存取界面。或者说，属性就是对象的外观与行为。例如“车的颜色”，就是“车”这一个对象的属性。在 Delphi 中，每一对象均有其专属的属性，例如表格(Form)的属性：BorderStyle 可决定表格的边框类型、Width 可决定此一表格的宽度、Height 决定此一表格的高度、Caption 设定此表格的标题等等。

(6) 方法(Method)

在以往传统的程序设计中，程序(Procedure)和函数(Function)是获得结果的一种方式。而在面向对象的程序设计中，将程序或函数包装在对象里，这些被包装在对象中的程序和函数就叫做“方法”。

> 属性与方法常会困扰初学者，因为他们都是被包装在对象之中。只要记着属性是设定某一对象的外观与行为的参数；而方法则是该对象中所拥有可执行的程序(程序或函数)。

三、OOP 语言的特性

传统上，面向对象程序设计(OOP)包括数据与程序对象，以作为应用程序的基础。传统上 OOP 语言包括下列三种特性：

(1) 继承(Inheritance)

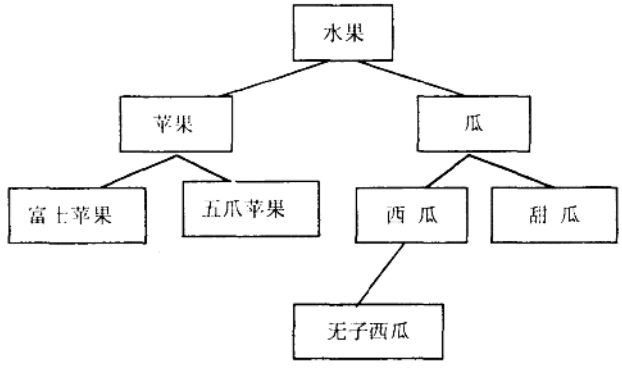


图 1.4 继承的范例

建立具备父对象属性及行为对象的能力，使程序开发人员能够利用继承的功能，建立更专门或更符合所需的后续对象。

继承的好处，是共用程序码及建立具备新特性的对象。如图 1.4 所示，“水果”是“苹果”与“瓜”的上一代(父)对象，所以“苹果”与“瓜”均继承了“水果”所有的特性，再加上个别的特性而成为一个新的对象。

(2) 封装(Encapsulation)

一个对象包含它可接受的信息、行为、属性与状态等。封装的意义即在于隐藏这些内部的处理程序，只允许通过一定的界面来存取。

(3) 多态(Polymorphism)

在调用对象的方法时，会依变量的实际状况调用适当的程序码。也就是说在同一程序中，同样的一件工作针对不同的处理对象，可以不同的方法来执行完成。

> Object Pascal 并不支持对象的多重继承。多重继承是指由两个不同的对象共同衍生新的对象，且所建立新的对象拥有两个父对象所有数据与程序的能力。

第二章　Delphi 的基本组成要素

一、启动 Delphi

当您安装好 Delphi3 之后，要启动 Delphi3 的方式如同您启动任何其他的 Windows 应用程序一样：

① 找到并以鼠标左键双击 Delphi3 的执行图示 

。

② 找到并以鼠标左键双击 DELPHI32.EXE 程序(预设的安装程序会将此文件放在\Program Files\Borland\Delphi 3\Bin 目录中)。

正确执行后将会出现如图 2.1 所示的 Delphi 程序整合开发环境（Integrated Development Environment，简称 IDE)。

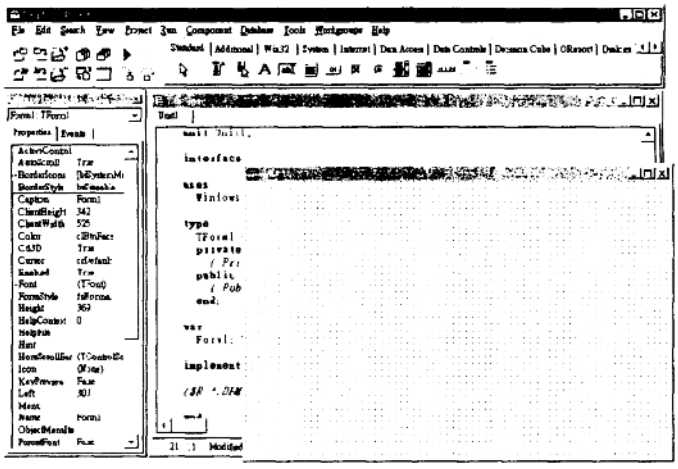
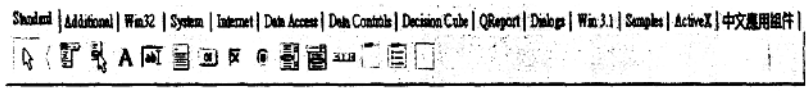


图 2.1　Delphi 的程序整合开发环境(IDE)

二、元件模板(Component Palette)



元件模板中，放置了许多可供您用来建立应用程序的元件。这些元件均建立在元件库(Component Library)中。如果您想改变这些元件所属的群组、增删所需的元件或群组的名称，可在主菜单中选择“Component|Configure palette”，如图 2.2 所列的元件库设定对话框中调整。

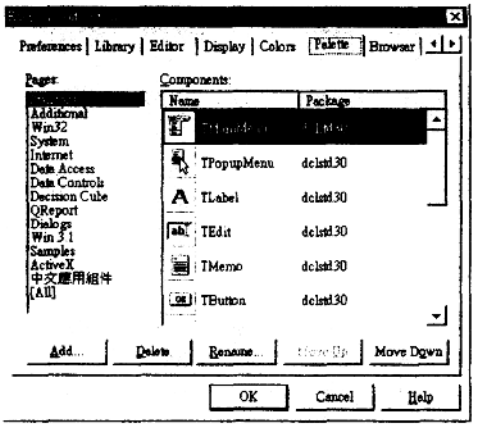


图 2.2　元件库设定对话框

如果您想寻找程序库中的某一元件，可在主菜单中选择“View|Component List”（如图 2.3 所示），在显示的对话框中输入您要查询的元件名称。查询到所需的元件后，按下“Add to form”按钮，即可将该元件加入现在正编辑的表格中。

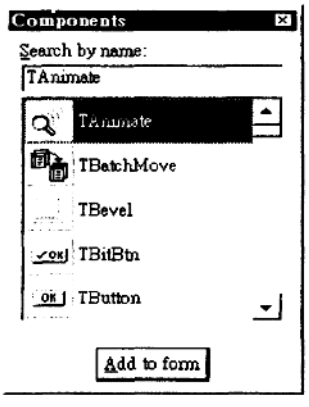


图 2.3　元件查询选取对话框

三、对象查看器(Object Inspector)

对象查看器、表格设计器(Form Designer)与程序编辑器(Code Editor)是 Delphi 程序开发环境中允许自由修改的三个重要的工具。通过对象查看器您可以改变元件的外表、行为与事件的发生。

在对象查看器的上方有一个元件下拉式列表(drop-down list)，列表中包含此一单元(Unit)中所有的元件名称，可以通过这个列表选择您要处理的元件，当然也可以直接利用鼠标选点表格上的元件。

在元件下拉式列表的下方，有两个页签(Page)，分别是您所选定元件的属性与事件。

① 属性页签(Properties)：显示此一元件所有的属性，提供您在此直接修改元件的属性内容。

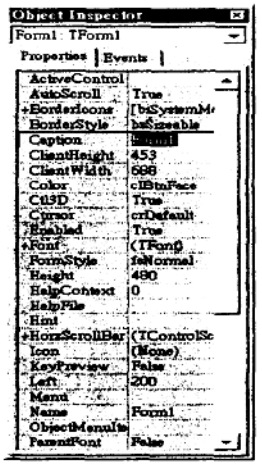


图 2.4 对象查看器

> 属性页签中所列出的属性，其实应该说是只有“公用”的属性，因为其他“私用”的属性是被封包在元件中，也有一些是“执行时期”才可修改的属性，是不会显示在属性页签中的。

> 您可以将鼠标移至对象查看器的上方，按下鼠标右键，设定对象查看器的一些功能，例如：隐藏(View|Object inspector 或 F11 恢复显示)、永远停留在窗口最上层、显示对象查看器的辅助说明文件等。

② 事件页签(Events)：列出了此一元件能够处理的所有事件。当指定的事件被触发时，即会执行此一事件页签中所属事件的程序码。当然，如果该事件中并未撰写任何程序码，您的程序也就不会做任何反应。

> 当元件被使用时，它会有预设的初值，您必须确定您所修改的属性是有效或有用的值，例如不要将 TButton(按钮)元件的背景颜色属性设为黑色，因为黑色的按钮如何显示它的名称?

> 当您改变了对象查看器中元件的任何属性，Delphi 都会将这些改变记录在此一表格的 DFM (Delphi form)文件中，提供编译时使用。不要试图去直接修改 DFM 文件的内容，除非您已确定明了 DFM 文件的结构。

四、快速菜单(SpeedMenu)

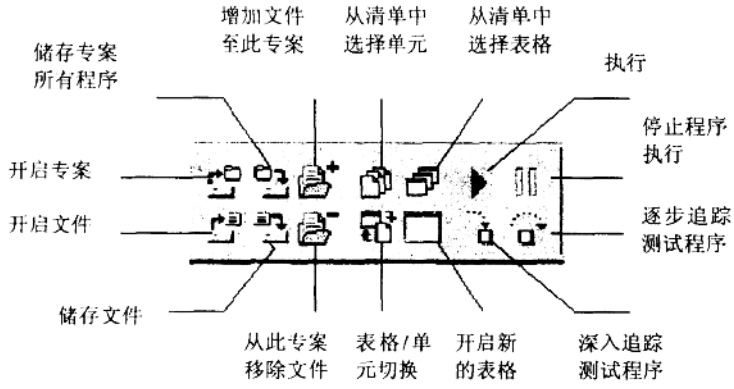


图 2.5　快速菜单

图 2.5 所示的快速菜单，是 Delphi 将常用的功能选项放置在工具栏上，方便您在设计时期使用。您也可以将鼠标移至快速菜单上，按下鼠标右键，在出现的浮动菜单中选择“Properties”选项，开启如图 2.6 所示的快速菜单编辑对话框。

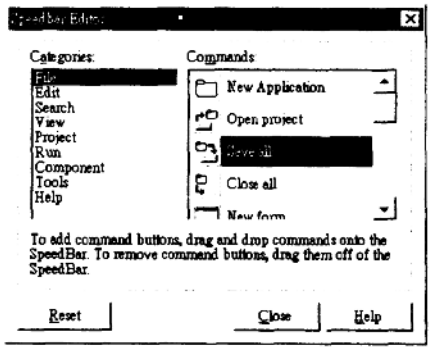


图 2.6　快速菜单编辑对话框

您可以通过拖曳的方式，将需要使用的工具图示拖曳至 Delphi IDE 的快速菜单中。如果要删除快速菜单中的工具图示，只需以鼠标左键按住该工具图示，拖曳至快速菜单之外的区域即可。

五、程序编辑器(Code Editor)

程序编辑器（如图 2.7 所示），是您撰写与实际执行程序的“单元”(Unit)所在。单元内容包括常数、变量、程序(Procedure)和函数(Function)等宣告与程序实作部分。事实上，单元可供其他专案(Project)所共享。当产生一个新的表格时，一定会产生一个对应的单元，但是单元不一定有对应的表格。例如您在撰写一个背景处理数据的程序，不需要显示画面信息，因此只有单元(程序)而没有表格(显示画面)。

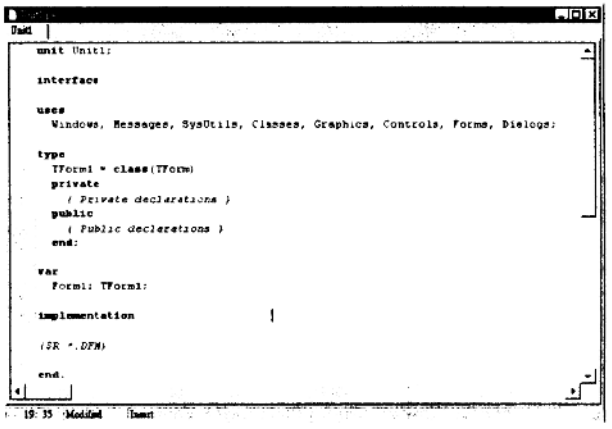


图 2.7　程序编辑器与预设的程式码

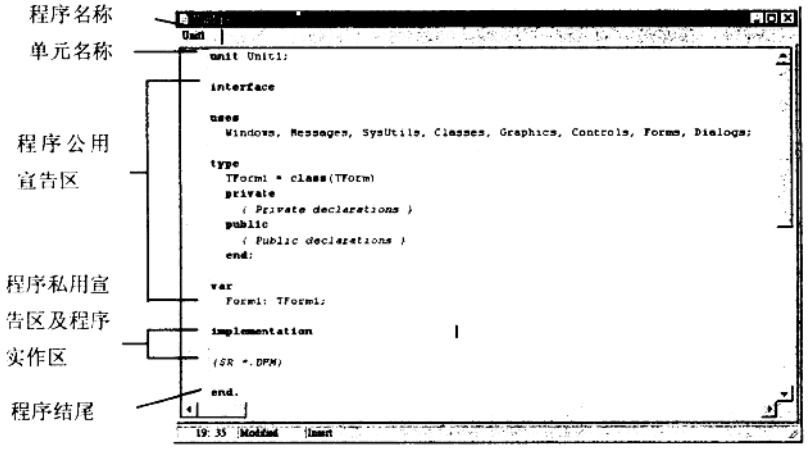


图 2.8　单元的基本结构

一个单元的基本结构如图 2.8 所示，今说明如下：

① 单元最先开始是一个 unit 保留字，后面所接的是此一单元的识别名称，也就是存文件时的程序名称。

② interface(界面部分)：在保留字以下，直到 implementation 识别字之前的区域。其所宣告的部分，可以让其他单元或应用程序调用，不过需注意的是，此处只是宣告部分，实际的程序码必须撰写于 implementation 之后。

uses 之后宣告的通常是 Delphi 内建的程序库名称，当表格上加入一个元件时，Delphi 会自动将该元件的程序库加入此处。当然您也可以加入您所开发的单元，使此一单元能通过 uses 的宣告，使用该单元的函数。

type 之后是类别的定义区，定义在此单元中所使用的各个类别。

var 之后所宣告的是将 type 中所定义的类别宣告成为变量，使其成为一个可执行的个体(Instance)，也就是一个对象。

> 当您选点在表格上的某一元件，或将光标停留在程序编辑器中该元件名称的位置时，按下 F1 键，系统会显示该元件的相关说明。如图2.9所示，说明中除了包含该元件的属性、方法与事件种类，还包括它的继承来源与程序库名称。

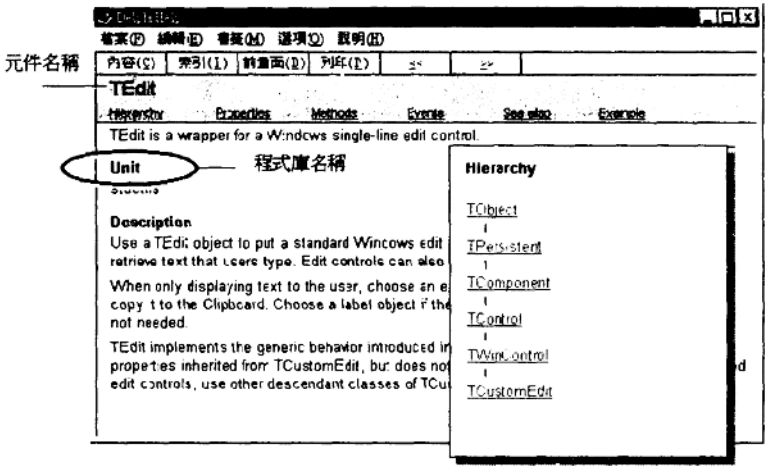


图 2.9　TEdit 元件的辅助信息

③ Implementation(实作部分)：保留字之后宣告的是此一单元的实际程序撰写区，包括事件处理的程序或您撰写的其他程序或函数。

> 程序(procedure)与函数(function)有什么差别呢？简单的说，程序与函数都是一组程序，只是程序没有传回值，而函数需要传回值。
>
> 宣告的方式：
>
> procedure 程序名称(参数 1：数据型态；参数 2：数据型态...)；
>
> function 函数名称(参数 1：数据型态；参数 2：数据型态...)：函数数据型态；

implementation 中也可以宣告 uses 与 var，这些宣告均表示此一单元私用的程序与变量后。例如在一个单元的程序中，有使用 UnitA 与 UnitB 两个单元的函数或型态，则 uses 应加入宣告：uses UnitA, UnitB。

此外，一个单元还包括两个选择性的部分：

④ initialization(初始化部分)：这一部分位于文件结尾的部分。内容通常包括此单元程序的所有初始化工作的程序码。这段程序会在主程序开始执行前被执行，而且只执行一次。

> 如果使用表格时，您也可以将初始化的程序码放在 OnCreate 事件中。当执行时此表格建立的时候，系统便会触发此表格的 OnCreate 事件。

⑤ finalization(结束处理部分)：这一部分位于初始化宣告与程序结束的 end. 之间。内容常包括所有程序结束时的善后工作的程序码，例如释放存储器、关闭文件等等。

> 您可以可在主菜单中选择 Tools|Environment Options 设定表格、编译与程序编辑器的相关系统参数。

> 千万不要直接去改变程序编辑器中，定义在 Type 或 Ver 中指定元件的名称，不然会造成 PAS 文件与 DFM 文件内容关连无法一致。切记一定要通过对象查看器来改变该元件的名称。

六、表格(Form)

当开启一个新的专案时，Delphi 会预设一个程序单元 Unit1 与显示一个预设的空白表格 Form1，您可以在此表格上逐一建立应用程序的使用者界面(User Interface)。

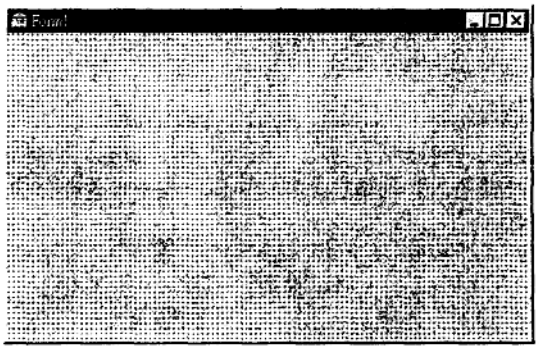


图 2.10 预设的空白表格

在表格上，您可以看到有许多排列整齐的格线(Grid)，这些格线是作为摆放元件时定位用的。Delphi 预设的间距为 8，您可以通过“Tools|Environment Options …”选项，在图 2.11 所示的环境参数设定功能中，调整 X、Y 轴的 Grid 间距数。

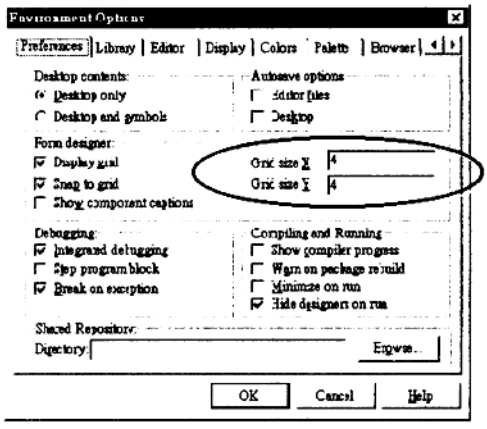


图 2.11　Delphi 的环境设定参数功能

七、练习撰写一个简单的程序

首先在主菜单上选择“File|New Application ”功能(在 Delphi 的 IDE 启动时，会自动先执行一次“New Application” 的功能)，系统会展开如前面图 2.1 的画面。

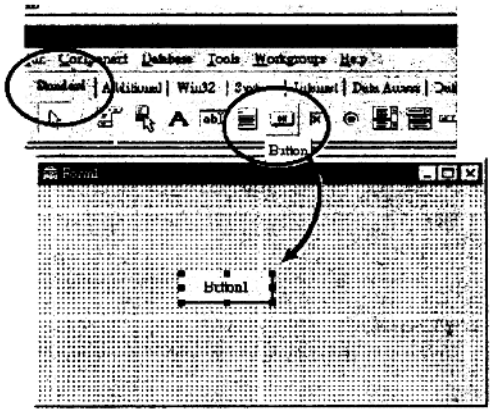


图 2.12　在表格设计器中加入元件

> 当您一启动 Delphi 时，Delphi 会自动执行 New Application 的选项显示整个 IDE。

将光标移到元件模板的标准元件群组“Standard”中（见图 2.12），先选点按钮元件，再将光标移至空白表格中按下鼠标，即完成加入一个元件的工作。接着再以相同的方法加入一个栏位元件。

这时您可以试着调整表格的大小，或移动表格中的元件到适当的位置(如图 2.13 所示)。

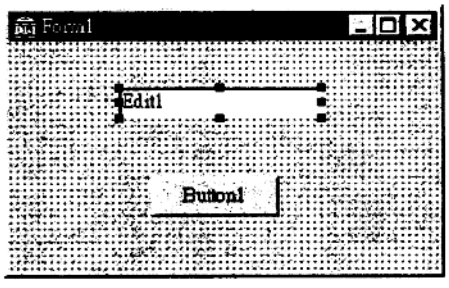


图 2.13　练习范例的表格

此时使用“View|Toggle Form/Unit”选项或按下 F12，显示程序编辑器。可以看到 Delphi 已帮您完成的程序如下：

```
unit Unit1;

interface

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

type
TForm1 = class(TForm)
Button1: TButton;
Edit1: TEdit;
private
{ Private declarations }

public
{ Public declarations }
end;

var
Form1: TForm1;
```

```
implementation

{$R *.DFM}
end.
```

注意其中 TForm1 = class(TForm) 表示一个对象继承的行为，TForm1 继承了所有 TForm(Delphi 所制作的表格元件)的能力。而在本程序中，TForm1 还增加了两个 TButton 与 TEdit 元件。在 var 区宣告变量 Form1: TForm1 表示产生一个可以执行的实体，也就是您在图 2.13 上所看到的那一个表格对象。

> 为什么元件的名称之前都有一个 T，因为这表示它还是一个型态，并不是一个对象。这是一种良好的命名习惯，方便程序人员以后区辨名称的类型，在“Object Pascal 语言”介绍使用者自定类型时，建议您所自定的数据型态之前也可以加一个 T。

> 当您将一个元件加入到表格时，每一个元件均有一个属性“元件名称”(name)，代表这一个元件在这一个表格的识别名称，也就是对象的变量名称。每一表格中各元件的识别名称必须是惟一的。当产生时，虽然 Delphi 会给予预设值，例如 Button1、Button2等。但是当一个表格拥有多个按钮、数据栏位等元件在其上时，使用预设的名称只会增加程序编辑的困扰。建议将表格中的元件给予适当的命名，例如将“离开的按钮”其元件名称改为 pbExit (pb 表示此名称是一个 Push Button 元件)；“确认的按钮”其元件名称改为 pbOk 等等。当以后您在审视程序时，可以很明确的辨识出此一名称是属于哪一元件，以及它的作用。目前撰写窗口程序的命名方式，有一套比较常被使用的命名原则，称为 Hungarian Notation。您可以参考该命名原则，对您的程序中所使用的识别字给予适当的命名。

您可以试着去修改下列属性：练习将表格 Form1 的 Caption 改成“练习”、光标在表格上的形状(Cursor 属性)、表格的颜色(Color 属性)等等。您可能已发现，改变表格的大小，除了可用鼠标直接拖曳表格来改变外，也可通过修改表格的 Height 与 Width 属性来改变；通过 Top 与 Left 属性来改变表格的位置。

接着设定按钮元件 Button1 的属性，希望能在执行时，光标停留在此按钮上的时候，会有提示讯息出现。首先将 Button1 的 ShowHint 属性设为 True，在 Hint 属性中输入您要显示的内容。