Delphi 5

彻底研究

陈灿煌 陈周造 编著

内附光盘

博碩文化

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

Delphi 5.0

彻底研究

陈灿煌　陈周造　编著

中国铁道出版社

2001·北京

（京）新登字 063 号

北京市版权局著作权合同登记号：01-2000-2347 号

版 权 声 明

图书在版编目（CIP）数据

Delphi5.0 彻底研究/陈灿煌、陈周造编著. —北京：中国铁道出版社，2000.12

ISBN 7-113-03974-X

Ⅰ.D… Ⅱ.①陈… ②陈… Ⅲ.Delphi 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2000)第 58275 号

书　　名：Delphi5.0 彻底研究
作　　者：陈灿煌　陈周造
出版发行：中国铁道出版社（100054，北京市宣武区右安门西街 8 号）
策划编辑：苏　茜
特邀编辑：邓庆容
封面设计：冯龙彬
印　　刷：北京市彩桥印刷厂
开　　本：787×1092　1/16　印张：45.5　字数：1117 千
版　　本：2001 年 1 月第 1 版　　2001 年 10 月第 2 次印刷
印　　数：5001～7000 册
书　　号：ISBN 7-113-03974-X/TP・487
定　　价：80.00 元

出版说明

本书是阐述 Delphi 窗口开发环境的经典之作，内容丰富而充实，共分十五章进行讲解，主要包括可视化设计的集成开发环境、窗口程序的菜单设计、多人群组开发环境的建置、Two-Tier 数据库应用程序设计高级技巧、数据库维护辅助工具、Multi-Tier 应用程序设计高级技巧、Windows 程序设计高级技巧等内容。

书中配有一张范例光盘，您只要执行光盘上的 Setup.exe 安装程序，范例程序将会被安装在 C:\Program Files\Delphi5Example 目录下，依照各章建立不同的目录存放着每一章的范例源程序。由于时间匆忙，所附光盘为繁体版，若出现乱码，请用“东方快车”等汉化软件进行转换，敬请读者谅解。

本书由台湾博硕文化股份有限公司提供版权，中国铁道出版社计算机图书项目中心审选；林进、宁夕、高胜、王明、沈放、邓力、罗怡、葛兰、彭涛等同志完成了本书的整稿工作；廖康良、孟丽花、肖志军、陈贤淑等同志完成了本书的排版工作。

中国铁道出版社

2001．1

目　录

可视化设计的集成开发环境（IDE） 1

Delphi 提供了一个面向对象、可视化设计的快速应用程序开发环境，就是所谓的RAD(Rapid Application Development)，这种 RAD 环境主要是让你能以编写最少的程序代码来建立高运行效率的 Windows 应用程序(32 位)。Delphi 包含的所有 RAD 开发工具都是集成在IDE(Integrated Development Environment)内，图 1-1 就是 IDE 的操作环境，可以利用这个集成开发环境整个应用程序的设计，而不需要再依靠其它零散的工具程序，使程序开发环境能够简单一致，提高整体的项目开发效率。

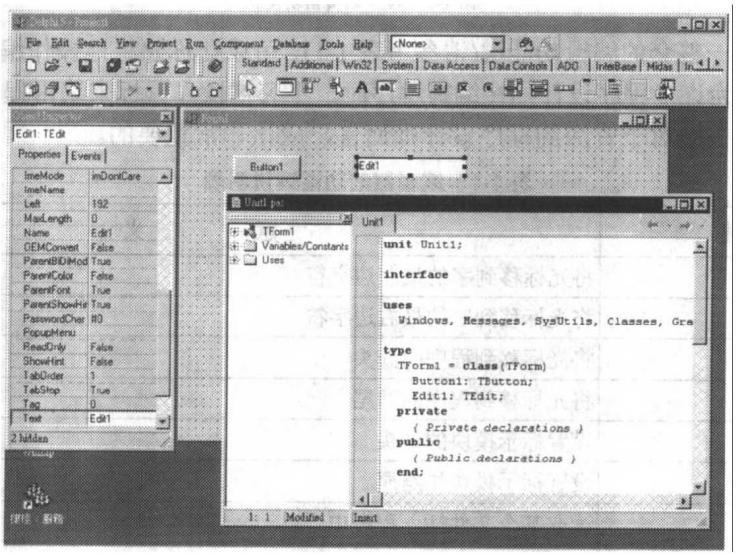

图 1-1 集成开发环境(IDE)

基本上，Delphi 的 IDE 环境包含有一些要素，分别是程序编辑器(Code Editor)、程序检索器(Code Explorer)、窗体(Form)、组件模板(Component Palette)、对象检查器(Object Inspector)、鼠标右键菜单(Popup Menus)、对象浏览器(Object Browser)、项目管理器(Project Manager)、快捷工具栏(Speed Bar)、调试器(Debugger)、联机帮助(On Line Help)。下面将针对 IDE 的每一个组成要素做一下说明：

1-1 程序编辑器（Code Editor）

这是一个具有完整功能的程序编辑器，你可以在上面同时编辑好几个原程序文件，它会将关键字(keyword)加粗表示，或是以不同的字体颜色来区别代表不同意义的程序模块(参考图1-2)。你可以通过主菜单的[File]选项或是快捷键，来维护编辑器内的原程序代码(打开、保存、另存为等)。

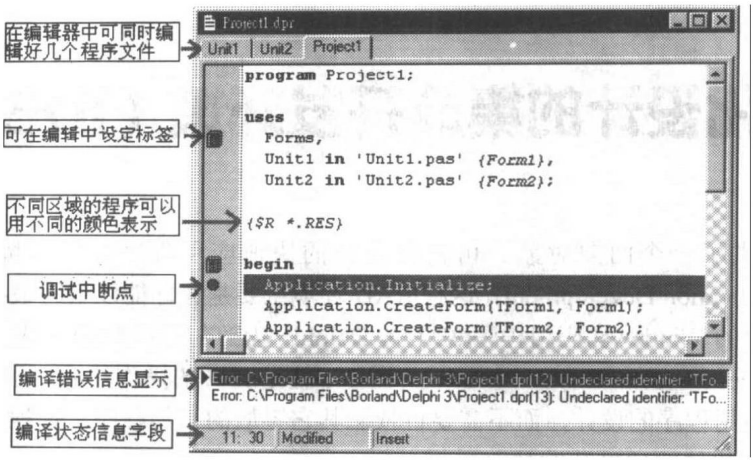

图 1-2 程序编辑器功能

它也提供有一整套的编辑功能键(请参考表 1-1)，可以让你熟练地操作编辑功能，增加程序编写效率，因为许多程序设计师在编写程序时，不习惯操作鼠标来编辑程序内容，所以这些编辑用的组合按键对于资深的程序设计师而言，应该非常重要的。

表 1-1 编辑器的功能键说明表

功 能 键	意 义
Ctrl+Left	将光标移到字的最左边字符
Ctrl+Right	将光标移到字的最右边字符
Ctrl+Home	将光标移到程序的开头
Ctrl+End	将光标移到程序的结尾
Ctrl+K+B	设置标示模块的开头
Ctrl+K+K	设置标示模块的结尾
Ctrl+K+R	读取某个文件插入程序中
Ctrl+K+W	将标示模块的程序代码写入文件中
Ctrl+K+P	将标示模块的程序代码输出到打印机
Ctrl+K+I	将标示模块的程序代码往后缩一格空白
Ctrl+K+U	将标示模块的程序代码往前推一格空白
Ctrl+K+C	复制标示模块的程序代码(没有用到系统的剪贴板)
Ctrl+K+V	粘贴标示模块的程序代码(没有用到系统的剪贴板)
Ctrl+C	复制标示模块的程序代码到系统的剪贴板
Ctrl+V	将剪贴板内的数据复制到编辑器
Ctrl+X	复制标示模块的程序代码到系统的剪贴板，并且删除原标示模块的程序代码
Del	删除光标所在的字符或是标示模块的程序代码
Ctrl+K+0,1,2,3…9	设置卷标(每页程序最多 10 个)
Ctrl+Q+1,2,3,4…9	将光标移到卷标处
Ctrl+F	查找字符串
Ctrl+R	查找并取代字符串

当然，你也可以自己定义程序编辑器的一些设置值，像是字体显示的颜色、粗细、大小，或是编辑器附带的一些功能。你只要点选右键菜单的[Properities]选项，Delphi 将会弹出程序编辑器的参数设置窗口让你修改(请参考图 1-3)，整个参数选项共分成三类，分别放在 Editor、Display、Colors 三个页面内，由于选项太多，限于篇幅缘故，所以无法一一枚举，请自行测试。

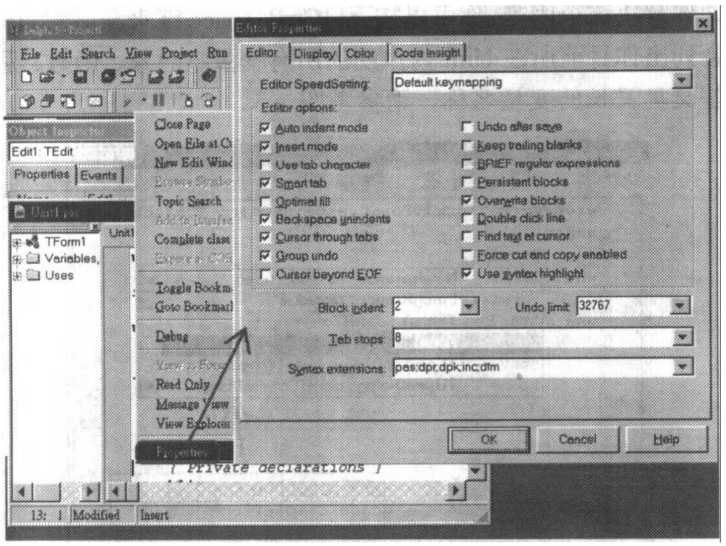

图 1-3 编辑器的参数设置

在 Delphi 中新增了一项 Code Insight 的功能，这个 Code Insight 功能又分有四个子功能，分别是 Code templates、Code completion、Code parameters、Tool-tip expression evaluation 等，下面是这些子功能的使用说明：

1. Code templates：这个功能会把 Object Pascal 语法中所有的命令语句列在一个下拉式的小窗口内(像 CASE、WHILE、FOR 等命令)，帮助程序设计师在忘记 Object Pascal 的命令语法时，可以不用再翻阅手册，直接在程序编辑器中得到帮助。你可以按下[CTRL]+J 键来激活 Code templates，当你激活这项功能之后，你可以直接用鼠标去点选想要取得的命令格式，或是用键盘输入命令，再按下 [Enter] 键来取得命令格式(请参考图 1-4)。

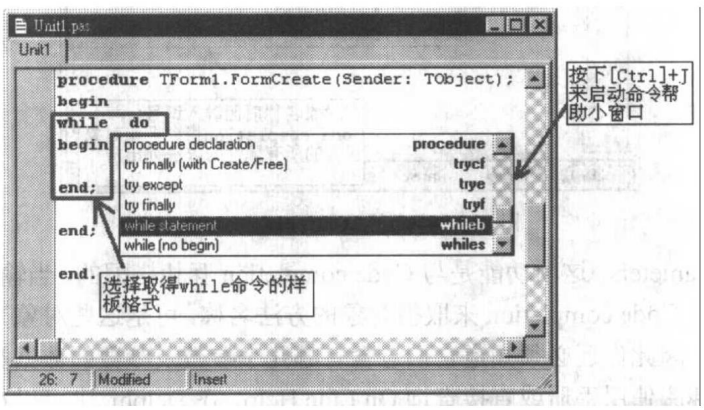

图 1-4 编辑器的 Code templates 功能

2. Code completion：在你编写程序的过程当中，有可能声明许多变量、对象、函数、过程等，而当你想键入这些声明的数据时，如果忘记了这些声明数据的名称，你还必须翻到最前面去把声明数据的名称复制到下面来，当这些声明数据多达上百个时，你就会感觉到头昏脑胀、两眼昏花，不知道想要的数据在哪里，而这项功能将可以帮助你解决这个困难。你可以按下 [Ctrl]+space 键来激活 Code completion 窗口，此窗口会列出你目前所在 Unit 内声明的所有变量、对象、函数、程序名称，帮助你取得这些又臭又长的变量名称，请参考图 1-5。

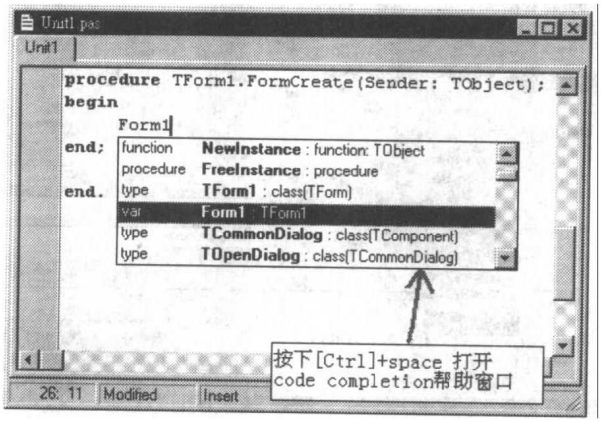

图 1-5 编辑器的 Code completion 功能

如果你输入的是对象名称，那么你一定接着会输入此对象的某个属性或方法，此时你只要在对象名称后面键入逗号，不到两秒钟的时间 Code completion 窗口将会把此对象包含的所有属性及方法找出来，列在窗口内，你可以按下[Enter]键或用鼠标点选来取得需要的属性或方法，参考图 1-6。

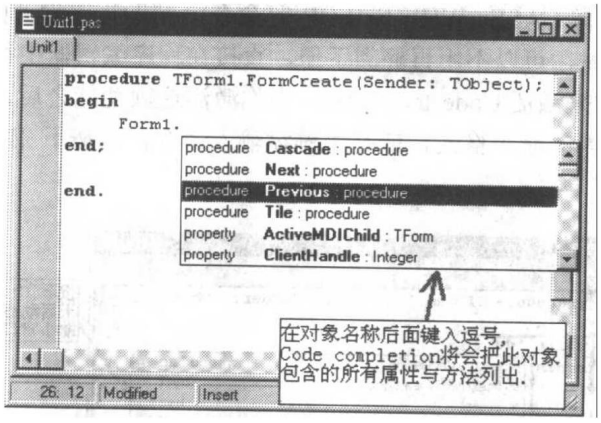

图 1-6 编辑器的 Code completion 功能

3. Code parameters：这项功能是与 Code completion 互相搭配的，当输入对象名称之后，可以激活 Code completion 来取得对象的方法名称，可是这些对象方法都是函数或过程格式，因此你还必须知道它们包含有哪几个参数及参数的数据类型。一般传统的作法是翻阅使用手册或直接查询 On Line Help，在 Delphi 中可以在对象方法后面键入“左括号”来激活 Code parameters，它将会把查询到的参数格式以 Hint 的方式

显示出来，参考图 1-7。

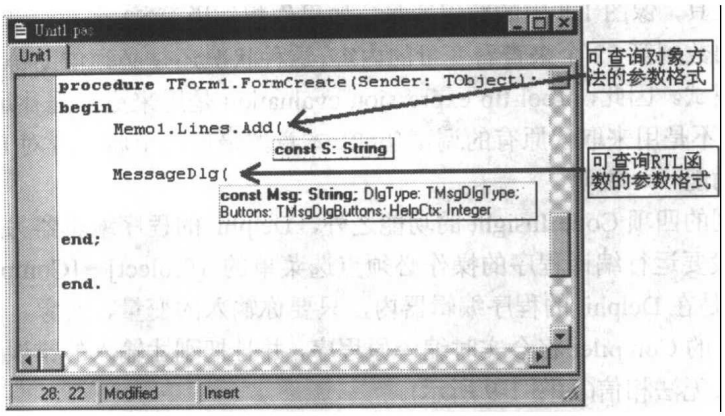

图 1-7 编辑器的 Code parameters 功能

Code parameters 不但可以查询到对象方法的参数格式，还有运行时期函数库(RTL)的函数及过程、自行定义的函数及过程、对象产生的事件过程等等，Code parameters 都可以查到这些函数或过程的参数格式。

4. Tool-tip expression evaluation：这项功能应该是 Code Insight 四项功能中最重要的一项。过去在程序设计调试时，你必须把要被调试的变量名称加入到 Watch 窗口，然后一步一步运行来观察变量内容的变化。可是这种调试方式具有两项缺点：第一，Watch 窗口可以一次显示在屏幕上的变量个数是有限的，当你加入好几十个变量到 Watch 窗口内时，你必须用鼠标慢慢地翻页寻找想要观察的变量；第二，有些 Local 的变量在某个程序下是有效的，当跳出此程序后，这些 Local 变量就变成无效，你必须删除它们，另外再加上目前所在程序的 Local 变量，不断地在重复这些工作，增加了不少调试时的工作负担。

   Tool-tip expression evaluation 就是用来解决你上述的调试问题。你只要在程序调试状态下，将鼠标指针指向想要观察的变量，即可马上激活了 Tool-tip expression evaluation 的功能，它将会把此变量目前的值以 Hint 的方式显示出来，参考图 1-8。

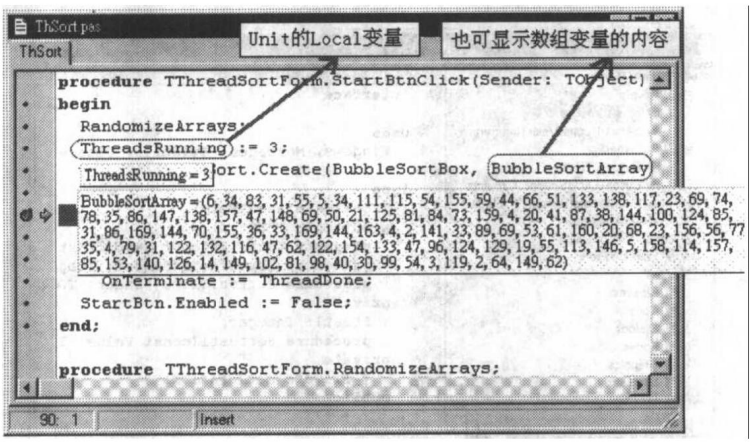

图 1-8 编辑器的 Tool-tip expression evaluation 功能

当然有了 Tool-tip expression evaluation 功能，并不代表你就可以完全放弃原先使用的调试工具，像图 1-8 中的数组变量，如果你想知道第 52 个数组节点的值，你不可能在上面数到第 52 个来查看变量的内容，这种状况还是必须借助传统的调试工具来帮助你调试。因此，Tool-tip expression evaluation 是用来更加强补足 Delphi 的调试功能，而不是用来取代原有的调试功能。本章的最后一节将会针对 Delphi 提供的所有调试功能一一说明。

除了前面提到的四项 Code Insight 的功能之外，Delphi 的程序编辑器另外还包含一项实时编译功能，一般要运行编译程序的操作必须点选菜单的 [Project]→[Compile] 或 [Project]→[Build All]，可是在 Delphi 的程序编辑器内，只要你输入的变量、对象、函数、过程名称不存在时，Delphi 的 Compiler 将会实时编译原程序，并且把刚才输入的错误显示在信息窗口内，速度快地让人无法相信(如图 1-9 所示)。

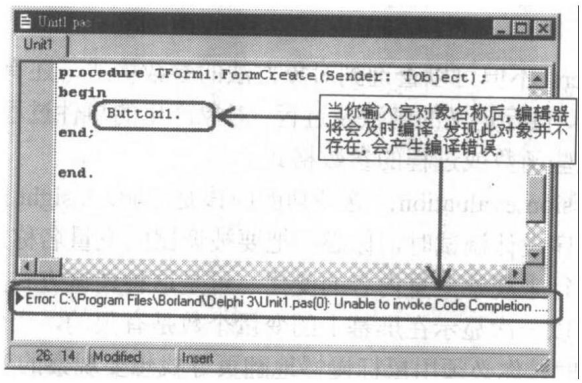

图 1-9 编辑器的自动实时编译功能

1-2 程序检索器(Code Explorer)

当你打开 Delphi 时，你是否发现程序编辑器(Code Editor)和旧版有点不同，左半边似乎多了一个树状结构的窗口，没错！这个窗口就是 Delphi 新增的 Code Explorer 工具窗口(如图 1-10 所示)。

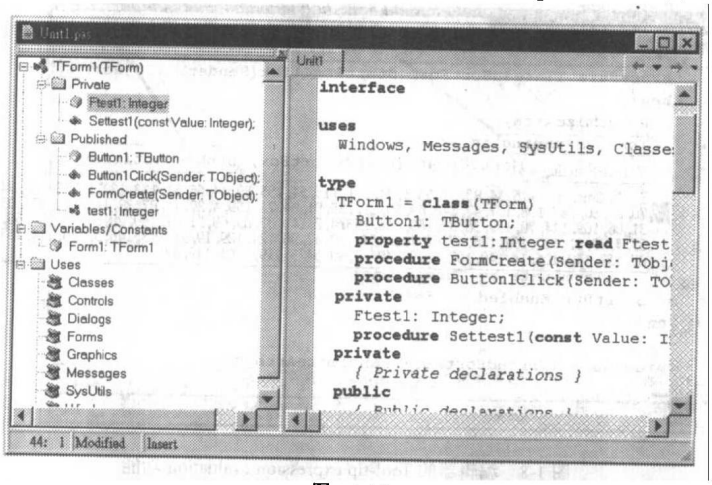

图 1-10

Code Explorer 对于 Delphi 的程序设计师而言，是一项早就该提供的功能，由于 Windows 的程序写法是属于事件驱动的方式，也就是说大部分的程序代码都会写在对象的事件内，若是遇到程序要修改时，你就必须先 Focus 到组件上，然后打开 Object Inspector，找到想要修改的事件程序，再点两下就会找到事件程序的程序段落，若是包含的对象不多那还好，如果遇到程序很大或是对象又很多时，光是找程序就晕头转向了。有了 Code Explorer，这些问题都将迎刃而解。

Code Explorer 会把你在 Unit 内声明的所有组件、属性、方法、程序、函数、各种类型的变量通通建成一个树状结构，你可以依据不同段落的声明，展开不同的节点，往下展开一直找到你所需要的节点为止，然后移动鼠标到此节点上点两下左键，右边的 Code Editor 将会自动找到对应的声明，并且把键盘光标移到此处，如此你将不用借助其它工具窗口，只要通过 Code Explorer 就可以很快地在 Code Editor 内找到你所需要的程序段落。

另外，你可以在 Code Explorer 的树状节点上点选右键菜单，其中有项功能是 New，这项功能是让你直接在 Code Explorer 的节点上往下新增一个节点，然后会在 Code Editor 上产生对应的程序代码声明，可是实际测试的结果，不论是在哪一层点选 New 功能，Code Explorer 都会把新节点建立在 PUBLIC 的声明段落内，不知这是否算是 Delphi 的臭虫。

另外，你可以点选菜单 Tools|Environment options 的 Explorer 页面来设置 Code Explorer 的一些参数(如图 1-11 所示)，这些参数设置将会影响 Code Explorer 的功能运作，请你自行测试每项参数的功能，限于篇幅的关系，在此将不做进一步的说明。

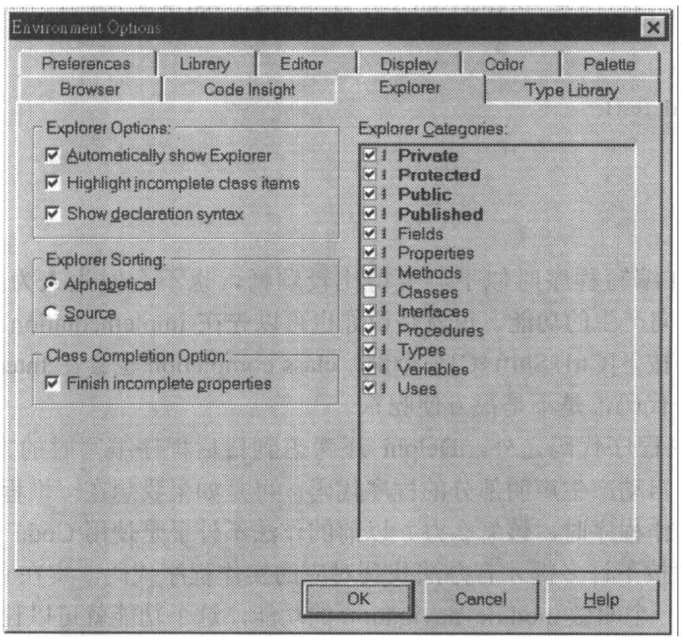

图 1-11

除了前面提到的功能外，Code Explorer 还有另外两种功能，分别是 class completion 及 module navigation。class completion 也是一项可以大幅提高程序开发效率的功能，例如，当你在 TForm1 下想要声明一个 property 时，传统的做法是必须先到 Interface 段落内编写 property 的声明程序，然后再到 implementation 段落内编写实际的程序代码，实在有点麻烦。class

completion 功能则会自动帮你产生程序框架，例如，假设你在 TForm1 下声明一个名为 Test1 的过程(整数类型)，程序声明内容如下所示：

```
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;
type
  TForm1 = class(TForm)
  Private
    Procedure Test1;
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
```

以前的作法是，除了 Interface 段落内的声明之外，你还必须到 implementation 段落内把此过程实作部分的程序代码写上去，过程上似乎有点麻烦，而 class completion 功能就是用来解决这个麻烦。当你在 Interface 段落内键入程序 Test1 完整的声明之后，请同时按下[Ctrl+Shift+C]这三个键来激活 class completion 功能，class completion 将会自动在 implementation 段落内产生程序 Test1 的实作程序框架，如下所示：

```
implementation
procedure TForm1.Test1;
begin

end;
```

如此一来，在编写程序时似乎会变得比较顺畅，也不会发生人为的输入错误。class completion 具有反向产生的功能，也就是说你也可以先在 implementation 段落内输入程序实作部分，同样地再按下[Ctrl+Shift+C]三个键，class completion 也会在 Interface 段落内自动产生声明部分的程序代码，是不是很方便呢？

除了自动产生程序代码之外，Delphi 还考虑到日后程序编写时的方便性，虽然 class completion 会帮你自动产生声明部分的程序代码，可是如果我想在一堆程序、函数声明中，马上找到对应的实作程序时，该怎么办？以前的作法不外乎是使用 Code Editor 的 Search 功能，可是你却必须按下许多键才有办法找到对应的实作程序代码，实在有够麻烦。而 Code Explorer 另外提供一个叫做 module navigation 的功能，这个功能就可以让你迅速地找到对应的实作程序代码。请先把光标移到程序声明的部分，然后同时按下[Ctrl+Shift+Down]三个键，module navigation 将会马上把光标移到对应的实作程序代码部分。同样地，你也可以在实作程序代码的位置按下[Ctrl+Shift+Up]三个键，module navigation 也会马上把光标移到程序对应的声明部分。

有了 class completion 和 module navigation 这两个功能，让你在实际 coding 的过程中更加