最完整全面的参考书　　世界最畅销书系列

即学即用 DELPHI

[美] Jon Matcho & David Faulkner 著
陈一民 张微 等译

OWL
OLE/DDE
API DLL

Special Edition
USING DELPHI

- 开发环境
- 各种构件
- 软件产品交付
- OWI . OLE/DDE. 数据库开发
- 最新可视工具介绍
- 开发功能强大的Windows应用程序

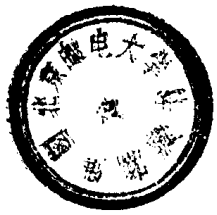
que®

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

即学即用 DELPHI

[美] Jon Matcho & David Faulkner

陈一民 张 微等译



電子工業出版社.

内 容 提 要

本书主要介绍 Delphi 开发环境、构件、应用程序开发和交付开发的应用程序。详细介绍 Delphi 的应用标准和全套开发技术，其作为一种可视开发工具赋予编程人员，使开发人员多快好省地进入开发环境，用 Delphi 的可视性和 Pascal 的编程技巧开发 Windows 应用程序软件。适于程序开发及软件工作者阅读。

USING DELPHI

[美]Jon Matcho & David Faulkner

即学即用 DELPHI

陈一民　张　微等译

责任编辑：和德林

*

电子工业出版社出版

北京市海淀区万寿路 173 信箱(100036)

电子工业出版社发行　各地新华书店经销

顺义县天竺颖华印刷厂印刷

*

开本：787×1092 毫米 1/16　印张：25.25 字数：640 千字

1995 年 12 月第一版　1995 年 12 月北京第一次印刷

印数：5000 册　　定价：48.00 元

ISBN7-5053-3357-7/TP·1290

著作权合同登记号：图字 01-1995-839

前　言

目前在计算机领域，美国微软公司推出的 MS Windows 各种版本为用户提供了图形用户界面和编程环境，操作起来界面友好，非常方便。使其成为当今世界上最流行的软件。今年 8 月，该公司又推出 MS Windows 95。这项软件功能更加强大，其应用又进入一个新的阶段。为了使 Windows 编程适合于广大开发者，许多编程语言供应商发布了自己的流行编译程序 Windows 版本。Borlard 公司推出了 Delphi，以提供基于 Pascal 等编程语言的可视编译程序。

Delphi 是 Borlard 公司的国际商标。它的正真意思是面向对象的可视编程工具。即用 Delphi 的可视性，结合 Object Pascal 语言的编程技巧，开发编写功能强大的 Windows 应用程序和数据库应用程序。现在用其它方法编写程序，枯燥无味，同时需要大量高水平的编程人员。而用 Delphi 只需一个程序员即可，并且享受到极大的编程乐趣。据初步试用者反映，其可节省工时 60%～80%。这的确是令用户兴奋的好工具软件。

Delphi 在美国市场推出后，便迅速流行起来。目前其刚刚进入中国市场，即已引起计算机界的敏感人士的极大注意。当然大多数人尚未听说过 Delphi，即使是一些资深的计算机人士也如此。为了向国内广大计算机同仁介绍 Delphi，我们引进了“Special Edition Using Delphi”这本书。中文名定为“即学即用 Delphi”。

该书是由美国 Jon Matcho 、David Faulkner 等七位作者共同完成的著作，又由美国有影响的计算机图书出版商 Macmillan 所属的 Que Corporation 于 1995 在美国出版。该公司已将中文版授权电子工业出版社。本书的介绍请详见引言部分。尽管叙述方法、用语用词不太习惯，但仍介绍得十分详尽，定会有所帮助。

消息灵通人士获知我们要出此书，要求我们一定要快。在各方面的呼声下，领导十分重视，译者和编者都以同样的心情尽快成书，与读者见面。时间紧，内容新，给翻译带来不少困难。有些术语提法欠准确就是最突出的问题。例如：Form、component、Project，国内的书有多种译法，本书依次译为窗体、构件、项目。这种译法觉得还不完全准确。但又找不出更好的译名。不过书中对各术语都有定义。请读者在阅读时了解其真正含义。参加本书翻译的有张微、陈一民、欧洋、康向东、张贤华、陈骁，陈永甫、谭秀华、张平同志进行了快速审读。参加本书编译审校及出版的同志均做出了巨大努力，非常辛苦。在此，向所有参与和关心此书出版的同志们致谢。印厂的同志几天就排完此书同样也向他们致谢。

参与此书的人员较多，特别是编译审，统一名词译名就是一个大问题。加之时间太短，水平有限，错误和不当之处不可避免，敬请读者批评指正。

编者　1995.10

引　言

欢迎阅读专版“即学即用 Delphi”。本书是一本关于当前计算机产业中推出的激动人心的专版书。当今世界，Delphi 使程序员们再次感到编程是一件趣事。Windows 编程杂志《Windows 技术月刊》的编辑 J・D・Hildebrand 评论 Delphi 时说：“正如众人所知的，它将改变人们的生活。”读者如有机会即学即用 Borland 公司的这种强有力的开发环境，就会发觉他说得千真万确。只要是可重复使用的基于构件的体系结构，Delphi 就可使面向对象的编程和可视程序设计与友好的开发环境完美地结合起来，为快速开发应用程序及快速调用 Windows 程序开辟一条新路。

本书并非只局限于从 Delphi 中浏览菜单并描述每一步做什么。本书在章节安排上，使读可以快速地查询，获取自己所需要的知识。该书按逻辑结构分成四个部分及附录。这种结构便于读者根据自己的水平合理地安排学习进程。

(1)第一部分，“基础”。这部分包括在开始接触 Delphi 开发环境时所必须了解的概念。读完此部分后，就会了解如何全面掌握和使用 Delphi 工具。

(2)第二部分，“构件”。本书将各类构件编入相应的章节予以介绍。构件的编写是以用户觉得它们有用的方式组织的，而不是按它们出现在 Components 调色板上的方式组织的。所有的构件都涉及到了，并用代码实例对较复杂的构件加以说明，而并非简单标签式地说明构件。

(3)第三部分，“应用程序开发”。类似前一部分，这部分是全力以赴为应用程序开发者编写的，讨论的全部都是应用程序开发的问题，以及如何达到专业级水平的编程。

(4)第四部分“交付”。此部分帮助用户准备并交付自己的 Delphi 应用程序。同时也讨论了交付并装载 Delphi 应用程序所必须的文件。不必为术语“文件”烦恼，Delphi 的一组程序都可装载在一个简单的可执行文件中，而其它一些程序可能相当复杂。

(5)附录。这部分比其它部分更有参考价值。附录中有构件表及其相关的属性，方法和事件。这样，用户就可以快速确定各个构件应用哪些属性，方法和事件。

Delphi 有希望成为一种令人欣喜的产品。Delphi 中的应用标准和全套开发技术正进入一个新变革阶段。同时，它把最终用户的可视开发工具赋予广大基层骨干编程人员，而更多的非正式编程人员可进入编程环境。现在用户可同时拥有编程能力和速度，而在以前只能牺牲一个保全另一个。Delphi 将性能和速度融合在一个精心设计的开发环境中。当引入 Delphi 时，希望读者与作者一同享受这份惊喜，享受新的编程乐趣。

本书使用对象

Delphi 是一种针对程序员的开发工具，程序员最希望成为使用 Delphi 的能手，但并不一定是需要了解 Delphi 技术的唯一使用者。本书也适用于具有中、高级编程及开发经验的技术工作者。本书除满足程序员的需要外，还突破了一些技术范畴。书中某些章节可能对工作项目负责人比对程序员更有吸引力。最后，作者希望所提供的大量知识能使程序员快速掌握必要技能，发挥 Delphi 的威力，步入正确的轨道。

致程序员

本书主要面向那些熟悉第三代高级语言(3GL),Pascal 语言的中、高级程序员,但也并非绝对如此。本书不是为教授 Pascal 编程而设计的,而主要是为教授怎样利用 Delphi 可视性和 Pascal 编程技巧开发 Windows 应用软件而设计的。在此只需明白相对少量的代码段,但最好还是懂一些 Pascal 基本知识。Delphi 基本只需要简单控制用户界面,而不需要编写多少代码。

本书一些章节是为开发环境的程序员,而并非 Pascal 程序员而编写的。读者应仔细研究以下章节:

(1)第一章,"迈向 2000 年的高级 Pascal 编程"。本章讨论 Windows 编程是如何演变的,现在是怎样运行的。阐述某些贯穿全书的论题。读者如果缺乏事件驱动编程的经验,请仔细阅读本章内容。

(2)第二章,"理解环境"展示如何在 Delphi 的多窗口开发环境(单文档界面)中运行程序。同时讨论主要的 Delphi 外部对象。

(3)第三章,"学习语言"对于有一定 Pascal 基础的人,可作为 Pascal 的复习阅读本章,而那些想从其它编程语言转向 Pascal 的程序员可以以此为起点进行学习。如果读者不熟悉 Pascal 而有其它语言经验,本章帮助读者从其它语言转向 Delphi 开发。

(4)第二部分。"构件",讨论 Delphi 中所有可得到的构件,以及如何把它们用于开发自己的应用程序。

(5)第三部分"应用程序开发",提供如何使用 Delphi 开发应用程序的信息资料。这些章节不仅包括应用基础,还包括 Delphi 强大的面向对象的工具,如 Browser,Database Desktop 和 ReportSmith。

(6)第四部分,"交付"。详细内容包括有关 Delphi 应用程序的交付。当程序员要把应用程序交付给用户时,这些章节有助于读者明白要交付完整的 Delphi 应用程序所必备的知识。

(7)附录。Delphi 的参考型信息一般都可在附录中查询。全部附录对程序员都很有使用意义。

书中的实例将使程序员尽可能快步地达到目标。本书并不打算取代 Delphi 文档资料或在线帮助——事实上是把参考做成在线帮助,使在线帮助成为 Delphi 产品不可分的一部分。本书的内容安排从逻辑结构和实用的角度帮助读者快速了解 Delphi 的关键技术和实质。书中许多例子着重讨论有效开发 Delphi 程序的关键性能和技术。

致项目负责人

Delphi 完全支持一个项目开发小组的工作。所有的编程组,无论大小,都会有项目或技术负责人负责讨论设计方案并作出最后的技术决策。作为一个能提出充分见解的称职的项目负责人,必须熟悉 Delphi 的几乎所有主要性能。本书的价值在于确定 Delphi 的哪些特点可以使用最适合的方式达到最理想的效果。

阅读本书后,所有的项目负责人都会了解,利用 Delphi 开发一个完整的应用程序需要在哪些方面努力。提供好的设想和建议会把人们引向令人满意的结果。领导者完全可以编写出一个较好的程序,但也不总是如此,最重要的是须首先获得第一手的方案。为此请阅读针对程序员和项目经理的有关章节。

致经理

程序员和工程项目负责人常常会接触到开发应用程序中的许多技术细节，例如怎样编写出最佳代码段。人们肯定会对一种熟悉的开发环境留下较深的印象，而同时常常会不准确地评价和比较其它此类产品的环境。经理必须做出决定，在将来的组织中使用什么样的环境，这些环境是否有效。根据探讨，可选定一个新的开发环境用于开发或评估。一旦确定开发环境，就要讨论和决定是否在新环境下工作以及是否继续利用其它环境的优点以利于开发。

Delphi 是具有复杂的构件重用特性和充分发挥小组开发权的一种环境。书中提供的信息可以协助经理了解 Delphi 是如何帮助开发小组的。欲获得更多知识，请阅读引言后部有关 Delphi 的性能小结："Delphi 性能部分"，并阅读以下章节：

(1)第二章，"理解环境"。本章包括了 Delphi 的主要开发工具，程序如何和什么时候使用它们。人们会感觉到 Delphi 不一般的特点。

(2)第三部分，"应用程序开发"。学习利用什么样的方法使用 Delphi 开发多窗口和多文档程序。对于今天的应用程序来说，存取数据是一个基本要求。第十章，"创建数据库应用程序"，专门叙述如何开发一个具有数据访问能力的应用程序。第十章及第六章，"使用数据约束构件"涉及到访问客户机/服务器数据。

(3)第四部分，"交付"。这部分的章节帮助用户准备并交付 Delphi 应用程序。这部分讨论交付所必需的文件和 Delphi 应用程序的装载。不必害怕，简单的应用程序都被装载在一个简单的可执行文件中，即使是一些复杂的数据驱动程序也只是用几个外部文件装载。

写在开始之际

如果用户有某种程度的编程语言的经验，熟悉新的第三、四代编程语言的速度会大大放慢。而对于具有 Object Pascal 内核和可视开发环境的 Delphi，学习新语言就不会有多大问题。一个新的 Delphi 程序员往往由于有其它语言，尤其是 Pascal 语言的编程经验而大大受益，但同时 Pascal 并非是绝对必要的。同样，具有许多其它开发环境的使用经验有时能提供大量的相关概念，这样常比有单一的 Pascal 经验更有利。

第三章，"学习语言"，每个人都应阅读。有一定 Pascal 编程经验的人则可快速浏览。即使读者了解 Pascal，无论如何也要读一读。Delphi 的 Object Pascal 语言是从 Borland Pascal 编译语言派生而来。随着 Delphi 自身 Pascal 语言的扩展，根据 Borland Pascal 5.5 版本对该语言增添了大量有价值的"基"和面向对象的扩充内容。

在进行操作 Delphi 以前，作者特别建议读者，无论技巧水平如何，阅读引言的剩余几节及下列各章：

(1)第一章，"迈向 2000 年前的高级 Pascal 编程"。本章讨论有关开发 Windows 应用程序的基础常识。Windows 的可视和事件驱动编程都包括在 Delphi 的技术内容中。

(2)第二章，"理解环境"。本章介绍 Delphi 的所有可视工具。读者阅读本章可以找到需要的内容。

(3)第三章，"学习语言"。那些从第三、四代或可视编程环境如 C 语言，Paradox(PAL 和 ObjectPAL)，Visual Basic 转道而来的程序员学习此章大有益处。编译类语言，尤其是 Pascal 语言，以其语法细致严密的特色使其它类语言自愧不如。读者只有在完全理解 Pascal 语言的基本

结构之后，才会在 Delphi 中编写出有效的代码。在调整自己的编程指导思想时请以此章为参考。

用 Delphi 编写 Windows 程序

Microsoft Windows 为标准用户提供图形用户界面(GUI)外壳和编程环境。用户应经常意识到，Windows 3.1x 作为一个操作系统，实际上只是一个 DOS 应用系统而并非真正意义上的操作系统。

Windows NT 和未来的 Windows 版本(Windows 95 和 Cairo)只要增加性能，就会具有直接包括在操作系统内的功能。这些操作系统运行 32 位程序，而 Windows 3.1x 上的程序都是 16 位的。Delphi 可以为这些新的操作系统编译 32 位的程序，快速执行程序。

图形用户界面(GUI)能提供比 DOS 类命令驱动式界面更复杂先进的以及与用户更友好的编程环境。由于 Windows 以直观的形式工作，用户能方便地切换任务并共享两个应用程序间的信息。Windows 还提供多任务切换，虚拟内存管理，拖和放，以及公共操作标准约定。尽管 Windows 受到世界各地计算机用户的极大欢迎，但开发者们还得照例面对极其繁杂的 Windows 环境，增加了编程负担。

为了使 Windows 编程适合于广大的开发者，许多主要的编程语言供应商都发布了流行的编译程序 Windows 版本。Borlard 公司推出了 Delphi，以提供基于 Pascal 编程语言的可视编译程序。本书假定用户熟悉在 Windows 中打开，关闭，移动和改变窗口尺寸等一套完整的面向用户的任务。在用 Windows 开发前，首先必须全面了解 Windows 程序的外观并体会 Windows 程序。有许多关于如何设计 Windows 界面的书，可能会更有价值。

Delphi 使开发功能强大的 Windows 应用程序变得快速而有趣。以往 Windows 应用程序开发常常需要大量高水平 C++编程人员，现在利用 Delphi 只需一个程序员就可编写了。

Delphi 的特点

Delphi 具有一系列广泛而富有创造性的特性，涉及从窗体设计到以透明方式支持所有流行的数据库格式。本书讨论 Delphi 的以下特点：

(1)构件的可重用性和可扩展性。开发人员使用 Delphi 后，不必再对诸如标签，按钮及对话框等 Windows 常见构件进行编程。读者也许常常发现在 Windows 中，不同的应用程序之间会多次出现同一对象。Choose File 和 Save File 对话框就是直接在 Delphi 中进行构件重复使用的例证。Delphi 包括这些及更多可重复使用的构件，允许用户控制 Windows 开发效果。Delphi 允许用户自定义这些构件，以所需的工作方式工作。见第七章“自定义和重用构件”详细内容。

可利用预先定义的可视及非可视对象，包括按钮，数据对象，菜单以及预先建立的对话框对象等。例如，利用数据对象即可无需编程，仅仅按鼠标按钮就能显示数据。这个有吸引力的对象列表使 Delphi 处于那些提供可重用构件结构的开发环境的领先地位。详见第二部分“构件”。

(2)VBX 支持。如果没有能力装进从第三方 VBX 商购来的最全面的“构件软件库”，那么构件重用有什么意义呢？Delphi 支持将封装的 VBX 对象直接装入 Component 调色板中，从而可以很容易地利用这些对象和工具。第五章“使用非可视构件”及第七章“自定义及重用构件”

都对 Delphi 的 VBX 构件有详细讨论。

(3)应用程序及窗体样板。Delphi 提供预先建立的窗体和应用程序样板,可以很快用于整个实际应用程序开发中。大量的公用对话框也包括其中。第八章"创建窗体"和第九章"创建应用程序"将进行更进一步讨论。

(4)可自定义开发环境。Component 调色板,Code Editor,应用程序样板和窗体样板都是一些 Delphi 完全可以自定义的领域。第二章"理解环境"将向读者叙述如何有效地利用 Delphi 开发环境。

(5)编译程序。Windows 的其它可视开发环境自称"编译"程序,而一般只是编译部分程序,再链接解释程序和预定义代码,形成可以执行文件。这种工作模式,使得许多程序员由于这种结构陷入开发困境。而 Delphi 生成完全编译的可执行程序,没有解释程序和预定义代码,这使 Delphi 程序比任何第三代语言都快,并且是世界上最快捷的数据库开发工具。Delphi 可以把简单的 Delphi 程序装载在单个可执行文件中,而不需要其它环境所要求的附加 DLLs。

(6)强大的数据存取功能。Delphi 中已建立的数据处理工具是 Borland Database Engine (BDE)。BDE 经历几代发展已经相当成熟。首先是众所周知的 ODAPI,然后是 IDAPI,现在 BDE 是一个标准的中介软件层,可以用来处理当前的所有流行的数据格式。BDE 是透明客户机/服务器处理的关键,可链接所有主要的客户机/服务器的开发商产品,如 Sybase SQL Server,Microsoft SQL Server,Oracle 和 Borland 的 InterBase。与 Microsoft 的 ODBC 相比,BDE 的主要优势在于通过与目标数据库的格式紧密链接,提高基本性能。第六章"使用数据约束构件"及第十章"创建数据库应用程序"叙述如何在应用程序中使用 BDE 显示数据。

书中常用语

本书将介绍一系列常用术语及习惯命名,尽可能使信息清晰完整。

术语

每一种环境,无论编程环境或其它,都有一套名词和通用的专业术语。这些术语有助于在特定环境中参与者之间的交流。试想在一场台球比赛中,如果没有这些术语,如何向别人解释用"球杆"击打"主球"了?在本书中,常用到以下术语。

(1)Object(对象)纵贯全书,作者交替地使用了"对象"和"构件"两词。通常,对象即指在运行时存在的构件——甚至一组构件。构件的含义则与对象稍有区别,偏重于指开发工具。构件在程序中的使用主要通过 Component 调色板(见第二章),是对象构造的基础。在程序运行时更多地是把构件看作对象,而不是一个部件。

(2)环境。在编程环境和开发环境中,本词在不同的上下文中略有差别。除了作者专指的那种用于编译面向命令或批命令程序的编程环境外,两者并无大的区别。开发环境更进一步的含义是指包含在单个应用程序中的种类繁多的一系列工具。

(3)异常。异常是指没有预料到错误。见第十四章"处理错误",详细叙述了异常及如何处理它们。

第一章"迈向 2000 年的高级 Pascal 编程",将帮助读者理解更多的与 Delphi 开发环境相关的术语的概念。

变量

Delphi 提供许多不同变量类型，以存储整数，实数(浮点)，Boolean(逻辑)，字符，字符串，以及指针等值。另外还有用户定义类型，集合，记录，以及对象变量。由于有这么多不同的类型，在读别人编写的程序代码时碰到含糊变量名将使人迷惑。例如，考虑以下赋值语句：

```
myVar:=newVar;
```

这里被赋值的是什么？如何知道自己是否没有试图执行非法类型转换呢？为了找到 myVar 和 newVar 所包含的数据类型，不得不查找它们在程序中什么地方定义的。这造成低效率的维护。

本书中的许多例子使用前缀命名约定。使用这种约定，一眼就能马上知道正在处理的变量和对象的类型。而不必寻找声明变量的文件。使用这种约定，前面一行代码可能如下面：

```
realMyVar:=intNewVar;
```

注意 real 和 int 用于前缀实际变量名。这样一看，现在就知道正把一个整数赋给一个实数。

除了处理包括单个对象类型的简单例子外，只要有可能就使用前缀命名约定。一些例子使用 Delphi 的缺省命名约定，以免分散对所讨论问题的注意。下面表格列出了对 Delphi 的许多不同数据类型的建议命名约定。

Category	Data Type	Prefix	Example
Integer	Integer	int	intDaysInDecade
	Shortint	sint	sintDaysInMonth
	Longint	long	longDaysInMillenium
	Byte	byte	byteDaysInQuarter
	Word	word	wordDaysInCentury
Real	Single	sngl	snglMaxValue
	Double	dbl	dblPrecise
	Extended	ext	extMorePrecise
	Comp	comp	compMinValue
	Real	real	realDollarVal
Boolean	Boolean	bool	boolDone
Char	Char	char	charKeyPressed
String	String	str	strLastName
Pointer	Pointer	ptr	ptrAnyType
PChar	PChar	pc	pcFirstName

对象

对象的性质很象变量。对象有范围和类型，并且需要相似的命名约定，以减轻 Delphi 应用程序的不便。对象也许是使用前缀类型命名约定的最大原因。设想一下阅读一份没有显示类型，按字母顺序排列的对象名清单，想象试图记住对象清单中哪个是按钮名该是多么困难。程序员不得不逐字地记住整个应用程序中的内容。

下面表格显示了在 Delphi 中一些不同构件类型的建议命名约定：

Class	Prefix	Example
TObject	obj	objUniversal
TForm	frm	frmMain, frmEntry
TTable	tbl	tblCustomer
TQuery	qry	qrySalesTotal
TStoredProc	sp	spGenSummaryData
TDataSource	dSrc	dSrcCustomer
TDBGrid	dbg	dbgCustomer
TButton	btn	btnOk, btnCancel
TRadioButton	rBtn	rBtnAppend, rBtnOverwrite
TMainMenu	mmob	mmobMainMenu, mmobMyMenu
TMenuItem	menu	menuFileOpen, menuFileSave

常量

Borland 已经包含了大量常量名，它们直接建立在 Delphi 中。有关所有 Delphi 常量的详细信息，参看附录 D，“常量”。

Borland 也对 Delphi 的常量名选择了前缀命名约定，如下面表格所示：

Constant Type	Prefix	Example
Border icons	bi	biSystemMenu
Border style	bs	biSizeable
Color	cl	clGreen
Cursor	cu	cuArrow
Font style	fs	fsBold
Form style	fs	fsMDIChild
Window state	ws	wsMaximized

记住前缀常量命名约定是 Delphi 自己所有的。用户定义的常量可由程序员创建，使用自己希望的约定名字。

在本书中，常量名以特定字体出现，例如，fsMDIChild。

属性

Delphi 提供缺省属性名，没有严格命名约定。缺省属性名在整本书中都使用。当命名用户

定义属性时，与缺省属性名使用相同的命名约定。

在本书中，属性名以特定字体出现—例如，Color。

过程

当在 if…then 语句中使用时，过程按可以用类似英语阅读的方式命名。没有额外的命名或语法约定用以命名过程名。例如，如果有个过程测试用户是否登入网络，可以把它命名为 userIsLoggedIn( )。如果对这个过程使用 if…then 语句，见如下所示：

```
if userIsLoggedIn( ) then
        doSomething( );
```

按这种方法命名，代码更容易阅读，提供了编程效率及全面可维护性。

本书所用范例

本书中所讨论的许多范例随着书中内容的发展逐渐变大，提供一种适合逐步进行应用程序开发的方法，而不是编写有限的小范例。这比对所讨论的 Delphi 每个功能都要求建立新的范例要节约时间。如果不理解某个范例，可从头查找该范例。

Delphi 为什么适合用户

如果读者怀疑 Delphi 或者只是想看看 Delphi 有些什么其它产品没有的功能，抽点时间读下本书。Delphi 建立了一个 Windows 开发环境标准，所有环境将通过它进行比较。今天，没有其它环境能提供 Delphi 具有的便于使用，功能强，速度快，及灵活性等性能。Delphi 填补了长期存在的3GL 和4GL 世界之间的断层，把两者的优点组合成一个功能强大并且有很高生产率的产品。

Delphi 为满足对构件重用的特别需求提供大量的功能。Delphi 的许多方面是可自定义的，允许开发环境随着开发队伍的技术发展而发展。当开发小组创建实用新对象时，这些对象——应用程序样板，窗体样板，以及构件—都可在将来开发项目中利用。

Borland 打赌 Delphi，这种按下一代 Windows 开发环境思想设计的开发工具，几年后将证明是开发 Windows 的首选工具。

第一部分　基　　础

第一章　迈向2000年的高级 Pascal 编程

Delphi 是一种开发环境，它使用了 Microsoft Windows 图形用户界面的许多先进特性和设计思想。Delphi 允许用户提供与 Windows 自身紧密联系的应用程序扩展控制。由于给用户提供这层控制，首先应该感到满意的是 Windows 是如何从用户角度出发进行工作。一旦用户对 Windows 操作熟悉后，就会更好地适应有效地开发 Windows 应用程序。

本章讨论开发 Windows 应用程序所涉及的基本概念。即使读者已经熟悉 Windows 编程，但本章所讨论的主题都是以 Delphi 的角度展开的，因而仍是有益的。本章讨论以下问题：

(1)事件驱动编程原理

(2)对象：如何创建和使用对象，以及对象如何与事件相关联

(3)对象属性：它们是什么以及能为对象做些什么

可视编程环境使用一批术语和概念以说明构成应用程序的许多不同的“事物”。幸运的是，许多术语，包括对象，属性和事件，不管其基础语言是什么，在许多可视编程环境中有标准含意。本章将帮助读者对这些术语和概念进行分类，讨论事件驱动编程，而不单单是 Windows 编程，是如何演变发展至今的。

如果读者对事件驱动或对象驱动语言，或者对可视编程环境拥有经验，本章将帮助读者把在使用其它编程环境中所学的知识应用于 Delphi 中。本章以 Delphi 的观点，进行普遍意义上的可视化编程及概念的快速讨论。熟悉这些概念是理解如何运用 Delphi 的关键。

1.1　理解事件驱动编程

在图形用户界面(GUIs)出台以前，事件驱动编程已经很普遍，基本上任何程序员都可运用。事件驱动编程可以用许多方法实现。随着鼠标的出现，以及它给用户界面带来方便，事件驱动编程才成为用户和开发人员等的需求。

在事件驱动编程环境之前，自上而下的编程过程方式被认为是流行的技术。当构造的代码段管理大量处理时，自上而下的程序设计是非常有用的。实际上，使用自上而下的编程技术建立的应用程序往往产生精巧的、可维护的代码。然而，在事件驱动编程之前，使用自上而下的方法建立的应用程序常常出现与应用程序处理密切相关的过度复杂的菜单，和击键指令序列。这些应用程序的用户常常需要与他们所希望的代码有更紧的关系。

事件驱动编程不会取代过程编程，但通过一个结构框架可以实现，即使用户界面和具体事之间的更好地分离。Delphi 和其它事件驱动环境都提供了这样的结构框架，允许程序员更多地将精力集中于具体应用程序的逻辑上，而不必担心如何管理和控制用户的要求。

实际生活中，事件驱动的例子随处可见。设想一下家中的房间。下面发生的是由另外一些外部事件所激发(或驱动)的实际事件：

(1)按下遥控电源按钮后，电视打开。

(2)温度降至70度以下时，恒温器使供热系统产生更多热能。这里。恒温器触发一个事件以响应另一事件。当锅炉收到这个事件后，指示本系统燃烧更多的燃料产生热能。

(3)朋友打来电话。电话振铃以响应收到的信号。

(4)如果不接电话,响铃四次后,电话应答器接通电话并记录信息。

(5)朋友没有留言而是决定拜访。在门口,他按响门铃,作为听到这个事件的应答,主人从沙发上站起来去开门。

事件是实时发生的,并针对始发事件产生一个或多个动作。事件驱动编程的关键在于确定哪些事件需要专门处理。在使用 Windows 时,会触发许多事件,但只有那些与应用程序相关的事件才需要处理。也许程序员并不关心用户是否双击屏幕中的某块区域,但无论何时用户按下 Delete 键时,应用程序确实需要做出反应。事件驱动环境,包括 Delphi 在内,处理捕获按键事件,并允许程序员直接转入执行与按下 Delete 键有关的逻辑处理。

不少基于 DOS 的开发环境逐渐能提供事件驱动接口和编程支持,但没有 Windows 那样引人注目。所产生的框架结构能控制处理"Windows 型"的事件,如窗口移动,按钮按下,以及其它以前不具备的事件等。大多数情况下,从 DOS 事件驱动编程转至 Windows 编程的程序员,比那些直接进入 Windows 编程的程序员有较充分的准备。

Delphi 要求同时具备事件驱动编程概念及 Windows 本身的大量知识。开始在 Delphi 中编程时,具有事件驱动的编程经验是非常有帮助的,但并非完全必须。人们可以学习事件驱动编程,但仅仅通过 Delphi 是不能了解 Windows 是如何为用户运转的。Delphi 实际上是一种与 Windows 的内部函数紧密相连的低层语言。Delphi 把这些函数封装并加入外层部分,使之成为高层的构件、对象和方法。理解标准 Windows 特性将有助于以标准 Windows 方式利用 Delphi 的性能。

1.2 理解"基于"对象的编程

软件产业经历了几个发展阶段,促进了程序开发中新技术和新方法论的发展。这种提高往往伴随媒体的快速变化和一系列新词汇而出现。作为一个程序员,常常需要对这种局势加以归纳,并了解这种新技术是否能使复杂的工作变得较容易,尤其在程序员已经喜爱一种自己感觉最有效的环境时更应如此。若要转到另一种开发环境,其益处必须远远超过人们现用环境的性能。如果另一种不同的产品只提供一些新增的优点,转用它不会有太多收获。相反,人们一旦接触 Delphi 这种环境,就再也不会回头。

在80年代中期,商业出版物鼓吹人工智能(AI),随之掀起一场新的热浪。由于种种原因,这种音调不久即减退了,也许部分原因是由于软件产业引进了面向对象的编程(OOP)概念。与 AI 的衰退不同,OOP 清楚地定义了整套对象,并有能力向软件开发者交付它的承诺:触手可及的促进可重复使用开发的工具。虽然有人声称 OOP 效率低,并且比传统环境开销大,但与其替代品相比,OOP 具有很大吸引力。

面向对象的编程由于其创建可重复使用代码及更好地模拟现实世界环境的能力,已经被宣布为对自上而下编程的优胜者。Microsoft Windows 使转向 OOP 语言成为更合理的决定;这是由于 OOP 与 GUI 开发相互促进,已经发展至互相不可分离的程度。使用一种通用的非 OOP 语言如 C 语言,进行 Windows 应用编程不是一件轻松的事。仅仅显示有"Hello World!"消息的一个简单窗口,就必须写出极其庞大的 C 代码。OOP 拥护者表示,通过给语言中增加扩展,把函数封装进 Windows 编程所必需的"对象"中,面向对象的语言使复杂的工作变得容易了。

软件开发者不久就开始更加注意目前编译器的面向对象后继者,如从 C 到 C++。80年代

后期，出现了基于 DOS 的 Pascal 和 Modula-2编译器的扩展 OOP 语言。具备 OOP 性能的各类语言（如 C++，Pascal，SmallTalk，Actor，和 Modula-2），完全溶入 Windows 应用编程也只是个时间问题。

虽然这些早期的 OOP 环境确实具有真正的面向对象的特征，但它们缺乏轻松地画出可视对象及管理与外部事件交互的能力。尽管它们组织得很好，但仍不得不编写大量的编码，使对象的行为如所希望的那样。一般来说，OOP 对于普通程序员是很难学会和使用的，因而妨碍了对它的认可。

为了填补这项空白，Microsoft 推出了 Visual Basic，一种基于初学者编程语言 BASIC（Beginners All-purpose Symbolic Instruction Code）的可视编程环境。尽管 Visual Basic 缺少指针和正规的 OOP 语言扩展，但它提供了大量的基本 OOP 元素。Visual Basic 毫无疑问已经是 Windows 编程最流行的开发工具，这清楚地阐明了应用程序开发可视部分的重要性。

随着 Borland 推出 Delphi 后，不必再从没有真正 OPP 性能的可视编程及没有可视编程性能的 OOP 之间作出选择。Delphi 允许在一个具有真正 OOP 扩展的可视编程环境中，使用它的 Object Pascal 语言。这种组合是革命性的，使可视编程与一个功能极其强大的，面向对象的开发框架紧密结合起来。

1.3 理解对象

由于软件产业的市场机制，专有名词“对象”自从由面向对象编程引进后，已经失去其部分含义。现在对象一词比过去使用得更加随便了，许多产品把这个新出现的新词用在其包装盒上，声称它们是面向对象的，其实它们也许只在技术上是事件驱动的或基于对象的。

实际上，基于对象也不错，如果读者曾经使用过基于对象的编程环境，如 Paradox for windows'ObjectPAL 或 Visual Basic，就应该已经熟悉了对象、事件和属性。

1.3.1 什么是对象？

定义一个对象不必使用复杂的编程术语。简单地说，对象是一个可以完成许多事的事务。在 Delphi 中，同其它可视编程环境一样，对象是指以下各项，如按钮、标签、列表框，字段等，从本质来说，任何一个被创建的项目都是在应用程序中加以利用。对象是一种定义不确切的术语，用于定义不太具体的事物。对象是那些构成应用程序的项目，可以涉及人们想要讨论的任何范围。本书可能把窗体上的一些如框体的简单事物称为对象，随后可能将整个窗体称为对象。一个单独的记帐应收票据模块也可被当作对象。当使用对象时，根据上下文环境决定各项目的含义。

思考一下给对象下的不确切定义，可能要感谢开发商为程序员提供了真正需要的：描述“一件事件”的专有名词。

为了使全书清晰明了，对象是指，当它存在于运行时或放置在窗体上后，所包含的构件。

1.3.2 创建对象

可视编程环境为创建应用程序增加了一个新内容，在程序执行以前，可以把对象画在屏幕上。没有可视编程时，“画”的过程需要写出实际的源代码，创建并自定义这些对象。只有在程序执行阶段才有可能看到所编写的对象。在这种环境下，要使对象的外表及行为如所需要的那