可视化编程系列丛书

最新 Delphi 2.0/3.0 入门、应用实例与多媒体程序设计

曹国钧 主编

电子科技大学出版社

内 容 提 要

自从 1995 年 2 月推出 Delphi 1.0 以来，Borland 公司的 Delphi 和 Delphi 客户机/服务器开发工具在高性能的快速应用程序开发领域已经成为新的标准。由于 Borland 公司 32 位 Delphi 编译器（即 Delphi 2.0）采用了独特的本机代码（Native Code）编译器、可视工具、可扩展的数据库技术，已获得了众多的世界级大奖，并成为最畅销的十大软件之一。同时，Borland 公司也赢得了相当数量的第三方厂商的支持。现在，第三方厂商已经开发出几十种程序库和兼容工具，有关 Delphi 的技术参考书已有 30 多种，杂志六种，以及众多的 Delphi 技术培训机构也应运而生。

现在，Borland 开发的第二代 32 位 Delphi 开发环境（即 Delphi 3.0）也已经在 1997 年第一季度发行。新的 32 位版本引入了若干新的技术，以进一步提高开发人员的开发效率和应用软件的性能。新版的 Delphi 采用了新的 32 位本机代码优化技术，极大地提高了应用程序的性能。新版的 Delphi 还包括一个新的 32 位配有可扩展数据库引擎的高性能的 Borland Database Engine（即 BDE）。

本书分十一章介绍 Delphi 2.0/3.0。第一章到第五章介绍了可视化工具 Delphi 的入门知识及其程序设计基础，如 Delphi 的发展、安装与启动、菜单与界面的解释、实用工具详解、程序框架与第一个 Delphi 程序的设计等。第六章介绍了窗口设计的基础，如属性、事件、菜单语言控件等，它们是应用程序设计的精华所在。第七章介绍了在 Delphi 中如何使用图形对象与画布绘制图形与图像，并给出实例。第八章详细介绍了 Delphi 的媒体播放器控件及其应用实例，它们是多媒体程序设计的基础。第九章介绍了如何在 Delphi 程序中设计 Windows 式联机帮助系统及与程序的链接技巧。第十章介绍了十个应用实例及其编程技巧，帮助读者深入地理解 Delphi 的编程技术。在第十一章中，详细地介绍了 Delphi 中内建的子程序，并给出大量例子。在附录 A～D 中，给出许多翔实的资料。

本书不仅是一本 Delphi 入门书籍，也是程序员应用开发的极好参考手册，更是多媒体应用人员的指导书籍。因此，本书适用于各种层次的计算机爱好者。

最新 Delphi 2.0/3.0 入门、应用实例与多媒体程序设计

可视化编程系列丛书

曾国钦 主编

*

电子科技大学出版社出版

（成都建设北路二段四号）邮编 610054

四川省仁寿县印刷厂印刷

新华书店经销

*

开本 787×1092 1/16 印张 28.875 字数 700 千字

版次 1997 年 6 月第一版 印次 1997 年 6 月第一次印刷

印数 1—5000 册

ISBN 7—81043—739—9/TP·303

定价：31.00 元

序

Borland 公司在 1996 年、1997 年分别推出了全 32 位的 Delphi 2.0（包括其修订版 2.01）、3.0。在可视化编程工具家族中，Delphi 具有战胜 VB 的绝对优势。

Delphi 的设计者十分聪明：Delphi 结合当今最现代的电脑技术，将可视化编程、面向对象编程、C＋＋编程、数据库编程以及傻瓜式编程等融入其中。可以说，只要掌握了 Delphi 2.0/3.0，那么，就可以在 Windows 3.X/95/NT 操作系统中遨游了。

本书集多年使用 VB、Delphi 的编程经验，并且融入新版 32 位的 Delphi 程序设计的技巧，希望读者能通过对本书的学习快速地进入 Delphi 世界。

在本书的创作过程中，得到了国家医药管理局重庆医药设计院黄后软件工作室的同事们的大力协助，作者谨致以崇高的敬礼。

电子科技大学出版社的朱丹编辑一直给予本书以激励与支持，作者表示诚挚的感谢。

读者在阅读本书时有什么意见，请来信或来电联系。

地址：（630042）国家医药管理局重庆医药设计院

我的 E-Mail 地址为：

libozh@public.cq.sc.cn

1997.5.18　曹国钧　于黄后软件工作室

目　录

第一章　可视化编程工具及Delphi的概况

随着计算机技术的不断发展，编程工具也得到了迅速发展。在这些编程工具中，可视化编程工具已经受到用户的欢迎。“几分钟内就可编写一个 Windows 3.X/95 应用程序”已经不再是神话。

本章首先比较目前出现的可视化编程工具 Visual BASIC、Power Object、Delphi，然后介绍最新一代的 32 位编程工具 Delphi 2.0 的特点，最后介绍 Delphi 2.0 的编程思想。

1.1　可视化编程工具综述

Microsoft Visual BASIC 早已成为人们喜爱的开发工具，而随着 Borland Delphi 2.0 的推出，可视化编程工具逐渐进入热潮。它方便的操作、强大的数据库支持以及独有的预编译特性，使其成为一个后来居上者。因此，下面对当前三个典型的可视化开发工具进行了探讨，并介绍一下 Delphi 在代码优化上所作的努力。

Visual BASIC 推出至今已经有五六个年头了，但如今在可视化编程领域它遇上了一个可怕的挑战者，那就是 Borland 公司推出的 Delphi。Delphi 不但具有 Visual BASIC 的全部功能，还有颇多自己的创新，这使得它一出台就大有后来居上之势。另外，Oracle 公司也有一个同类的可视化产品，即 Oracle Power Objects 1.0。大致说来，Power Objects 1.0 是 Visual BASIC 的一个模仿，或许它主攻的客户是 Visual BASIC 的老版本用户。目前 Power Objects 1.0 还比较嫩，也才出到 1.0 版。

虽然 Borland 公司在其他软件方面似乎不甚如意，但 Delphi 却是一枝独秀。1996 年 2 月，Borland 公司推出了 Delphi 2.0 版，1997 年 3 月推出了 Delphi 3.0 版。它们已是一个支持 OLE 的 32 位软件了。目前，Microsoft 已开发出了 Visual BASIC 的下一代版本 5.0，Power Objects 的下一个版本估计也将很快出台，竞争的好戏还在后头。

很难说哪个产品就绝对优秀，正所谓“萝卜青菜，各有所爱”，编程人员都偏爱自己使得顺手的开发工具。下面的论述，只是想在开发能力、数据库支持、运行性能，还有一些其他方面对三方产品作些比较。

1.1.1　开发能力

粗略看来，这三家产品非常相似，它们都提供了可视化的开发环境，提供了对 OLE 构件（OCX）的支持，可以与多种数据库建立连接，并且，它们都有自己的底层编程语言等等（Visual BASIC 和 Power Object 用的是 BASIC，Delphi 用的是 Object Pascal）。但如果细细比较，它们的差别还是很明显的。

Visual BASIC 是第一个将即拖即放的开发方式引入快速开发环境的，从 Windows 3.X 到现在的 Windows 95 和 Windows NT，它一直有很好的易用性。当前的 Visual BASIC 版

本则在 OLE 支持上表现出优势。使用OLE，Visual BASIC 开发人员不但自己能访问 OLE 服务器，还能自己创建 OLE 服务器以让其他具有 OLE 支持的应用程序来访问。有了这个功能，开发人员就可以用所谓的分布式应用的思想来开发应用程序了。

前面已经说过，Power Objects 是 Visual BASIC 的一个模仿，但它却同时有 Windows 和 Macintosh 的版本。可惜的是，Power Objects 却不支持已成标准的 ODBC，它宁愿提供自己的数据库驱动程序。Power Objects 也支持使用 OCX 构件，但不支持 VBX 构件。与其他产品不同的是，Delphi 使用 Object Pascal 作为它的底层编程语言。Delphi 提供了与 Visual BASIC 同样的 OLE 支持，它也能在应用中使用 OCX 构件。它最重要的特性还表现在它的预编译能力上，因为 Visual BASIC 和 Power Objects 都得让应用程序解释执行；这不免浪费了系统资源和 CPU 速度。Delphi 则预先将应用程序编译成经过优化的本地代码，其执行速度明显高于同类。

1.1.2 基于控件的开发

这三个产品都支持基于控件的开发（即用拼凑控件的方式开发应用程序），实际上，它们对基于控件的支持能力可能还要大过它们对面向对象的所谓封装和继承的支持能力，因为这样做对初学者有利。当然，它们也都有自己独特的面向对象支持方式。

控件有两类，一类是操作系统本身所带的本地控件，它们是不可移植的；另一类则是第三方厂商提供的控件库，如 VBX 和 OCX 等，它们存在着多个操作系统的版本。使用控件开发的一般思想是，尽量使用已有的控件来构造（或者说是拼合）应用程序，例如计算器、VCR 控件、数据库访问，这些都有现成做好的控件让用户引用，最好避免自己来创建本地控件。

一度统治控件市场的 Visual BASIC 控件（VBX）为什么让位给了 OLE 控件呢？其基本的原因就是标准问题。VBX 不是基于一个标准的控件开发平台。而 OCX 则不同，它使用的是基于 OLE 标准的控件开发接口。目前，Visual BASIC 和 Delphi 还同时支持 VBX 和 OCX，但 Power Object 只支持 OCX。

Delphi 的基于控件的开发环境是目前面向对象和控件开发性能最好的。Delphi 也是用它自己的控件库来书写的，在 Delphi 上开发应用可以简单到使用现成的控件库拼凑而以。Delphi 提供的控件库称为 VCL，它提供了在 Delphi 上开发应用的基石。VCL 几乎无所不包，它提供了用于界面设计、数据访问、列表框、菜单等的各种一般控件，它同样提供了诸如栅格、表、记事簿等各种实用控件。和 Visual BASIC 和 Power Objects 一样，Delphi 让程序员从控件板上直接拖放控件来构造应用程序，程序员也可以通过从 VCL 控件中继承其属性和功能来创建自己的控件，以恰好满足应用程序的需要。

如果程序员要使用第三方厂商的控件库（如 VBX 和 OCX），他只需将 VBX 或 OCX 登记入 Delphi 的控件板，然后就可以引用或继承 VBX 或 OCX 控件了。和 Delphi 比较相似，Visual BASIC 也可以让程序员以即拖即放的方式开发应用。它在集成环境中提供了一个Toolbox，所有的控件就可以从Toolbox 取得。Visual BASIC 的Toolbox 控件库也可以用第三方厂商的 OCX 控件扩展，不过 Visual BASIC 自带的控件库已很够用了。程序员也可以继承或修改控件的属性来创建自己的控件。

Power Objects 对控件的处理也比较类似，但遗憾的是，它只支持OCX，却不支持VBX

控件。看来，要让程序员喜欢它，还得有待更多的 VBX 厂商推出其 OCX 的版本。

但是，Power Object 在有一点上与 Visual BASIC 大不相同，那就是在对面向对象开发模型的支持上。Power Object 是将对象建立在实体（Forms）上而不是控制（Controls）上。

程序员可以将任何实体放入一个类，又可以将这些类放入一个实体，并继承原来实体的特性。对于含有相似实体的应用来说，这是比较方便的，因为只要程序员改动其中一个实体，则所有从这个实体继承而来的其他实体也相应改变。对于典型的 Client/Server 开发工具来说，这一点是很重要的。

这三种产品都提供了一些表格处理能力，例如，Visual BASIC 捆绑了 Crystal Reports，并允许将它作为一个 OCX 加入到应用程序中；Delphi 则捆绑了 Borland 的 Report Smith，这两个表格处理应用基本上都具有了用户所需要的报表和图形显示能力。Power Objects 使用的是其独特的表格应用，对于成组的表格处理比较容易，但对于复杂的表格则并不理想。不过可以用其他表格应用来做替代方案（可以使用 CrystalReports，它可以被作为一个 OCX 集成到 Power Objects 中）。

1.1.3 数据库支持

三个产品都支持许多种数据库引擎，不管是本地的还是远地的，但它们访问数据库的方式则各有独特之处。

Delphi 的 Data Access 提供了 Delphi 用于访问数据库的对象，包括建立连接、访问数据、查询内库等等。Borland Database Engine（BDE）则是 Delphi 数据库方案的核心，Delphi 是通过 BDE 来与 Oracle、Sybase、Informix、In-terbase、DB2 以及 Microsoft SQL Server 等数据库通信的。使用 ODBC 标准接口，Delphi 也可以访问 DBASE 和 Paradox 的文件。Delphi 把许多 BDE 功能加入到控件中，从而使它成为三个产品中数据库开发能力最强的一个。Delphi 2.0 新加入的一个特点是，用户不需要第三方的开发工具，直接能够在 Delphi 的集成环境中进行数据库操作。

Visual BASIC 的数据库访问是建立在 Data Access Object（DAO）和 JET 数据库引擎上。此外，Visual BASIC 还提供了一些控制方面的新特性。JET 用于支持对表格的创建、维护或删除，还有如索引、查询等功能。Visual BASIC 和 JET 对数据库的访问则是通过 DAO。

在前几个版本中，Visual BASIC 的数据库能力似乎给人以马后炮的感觉，但现在 Visual BASIC 4.0 及 Visual BASIC 5.0 却已经在技术上赶上来了。

除提供传统的数据库能力以外，Visual BASIC 和 JET 还提供了诸如数据库复制和较好的安全机制等一些特别的功能。

Power Objects 对数据库的支持似乎只是面向服务器的。令人奇怪的是，它不支持标准的 ODBC，而只是对 Oracle、Sybase、Microsoft SQL Server 等有限的几个数据库提供本地驱动程序。对其他数据库，它不提供连接能力。可能的原因是，Power Objects 比较强调对数据库引擎的完全控制，而用 ODBC 很难做到这一点。在下一个版本中，它将会提供 ODBC 的支持。

Power Object 让所有的数据库请求都通过一个事务处理对象，统一由这个事务处理对象创建到数据库的连接，并将有关的表格、图、索引直观地显示出来，不同的图标代表不

同的数据库资源，这比上两家产品具有更好的直观性。

1.1.4 OLE 支持

三家产品都提供了对 OLE 的支持，再说明白点，这是一种能允许一个应用程序给另一个应用程序提供服务的公共接口。例如，通过 OLE 接口，Delphi 可以将数据库的查询结果发送给 Microsoft Excel。

Visual BASIC 是 OLE 的老手，实现 OLE 控制只需几个定义操作，十分方便。Visual BASIC 又是少有的几个能在远程网络上支持 OLE 的 软件之一，它允许程序员将应用程序分为几块，在各个不同的远程服务器上执行，这就是所谓“多层应用”的概念。不过，虽然 Visual BASIC 中提供了很先进的分布式应用功能，但这门技术还不太成熟，实用尚需一段时间。

Delphi 是通过一种“变种”的方式来实现 OLE 通信的。Delphi 提供了一个 TAuto Object 对象，通过对它继承来创建 OLE 服务者对象，再将服务者对象的属性和方法通知给其他应用。Delphi 另外还提供了一个辅助工具以使 OLE 操作变得简易。

Power Objects 目前还只能被看成是具有 OLE 能力，它还不能自己创建 OLE 服务者对象。

1.1.5 运行性能

运行性能包括开发工具性能、应用程序性能和数据库性能。由于 Visual BASIC 含有较多的 OCX 控件，在比较快的机器上它更能发挥优势。

Delphi 和 Visual BASIC 的 32 位版本表现出了较大的优越性。它们不但能生成 16 位的应用，还能利用 Windows 95 和 Windows NT 的多任务、多线程特性。比较起来，Power Objects 似乎更停留在 Windows 3.X 上，当然，它也能在 Windows 95 上运行。

由于有预编译这一项，Delphi 应用程序的运行速度明显要快于其他两个。Visual BASIC 和 Power Objects 还要使用 P-Code 代码解释执行，这决定了它们在运行速度、内存占用上要弱于 Delphi。

三家的数据库性能不分伯仲，但 Visual BASIC 由于采用了标准的 ODBC 连接方式，而其他两家则采用本地驱动程序，理论上 Visual BASIC 要慢一些，但新发布的 32 位 ODBC 弥补了差距。

1.1.6 正确的选择

如何选择恰当的工具，这取决于用户使用的操作系统平台、开发工具易用性、性能和可扩展性。如果用户需要在 Windows 95 或 Windows NT 上开发，那就只能在 Delphi 和 Visual BASIC 上选择；如果用户需要让应用程序同时支持 Windows 和 Macintosh，那 Power Objects 就是当然的选择了。

总的来说，Visual BASIC 在开发的易用性上比其他两家要稍稍强些，而 Delphi 在开发能力上要占优势，但它略微复杂，不太适合初学者。Power Object 虽然也是一个完整的开发环境，但它的 OCX 支持较少，这会给开发者带来不便。如果用户对运行速度比较在意，选择 Delphi，它的预编译特性会让你满意。

在可扩展性上，只好选择 Visual BASIC 或 Delphi，它们都提供了很好的对 OLE 的支持。虽然 Visual BASIC 采用了所谓多层应用的结构，但这门技术毕竟还没有成熟，用者并不多。

1.2 新一代 32 位 Delphi 编译器 2.0

自从 1995 年 2 月推出以来，Borland 公司的 Delphi 1.0 和 Delphi 1.0 客户机/服务器(Client/Server Suite)开发工具在高性能的快速应用程序开发领域已经成为新的标准。由于 Borland 公司 32 位 Delphi 编译器采用了独特的本机代码 (native code) 编译器、可视工具、可扩展的数据库技术，已获得了众多的世界级大奖，并成为最畅销的十大软件之一。同时，Borland 公司也赢得了相当数量的第三方厂商的支持。现在，第三方厂商已经开发出几十种程序库和兼容工具，有关 Delphi 的技术参考书已有 30 多种，杂志六种，以及众多的 Delphi 技术培训机构也应运而生。

现在，Borland 公司新的 32 位版本 Delphi 2.0 已经引入了若干新的技术，以进一步提高开发人员的开发效率和应用软件的性能。新版的 Delphi 采用了新的 32 位本机代码优化技术，极大地提高了应用程序的性能。新版的 Delphi 还包括一个新的 32 位配有可扩展查询引擎的高性能的 Borland Database Engine。

下面介绍新版 Delphi 的特色和 32 位本机代码优化带来的好处。新版的 Delphi 为了充分利用 Windows 95 和 Windows NT 提供的功能，在许多方面做了改进工作。新版的 Delphi 完全支持 Windows 95 用户界面，包括许多新的界面控件，支持全部的 Windows 95 的 API，包括多线程、MAPI，支持 OLE 控件 (OCX)。这些功能都将极大地方便 Windows 95 下的开发人员和提高应用程序的性能。

1.2.1 新版的 32 位 Delphi 的目标

新版的 32 位 Delphi 的目标是：

- 通过新的 32 位优化编译器进一步提高性能
- 通过新的 32 位 Borland Database Engine 进一步提高客户机 /服务器应用程序的可扩展性和性能
- 支持 OLE 控件 (OCX) 和 OLE automation，提高对象的重用功能
- 全面支持 Windows 95 界面和 API
- 全面支持 32 位应用程序的开发和 Windows 95 及 Windows NT 的部署
- 与已有的 Delphi 应用程序完全兼容

1.2.2 32 位编译器优化

由于采用了一系列新的代码优化技术，新版的 32 位本机代码编译器的性能得到大幅度提高。过去，优化编译器需要与复杂的原编译器进行交互操作才能提高性能，并且还要对代码实施额外的“Pass”，这显然降低了编译器的速度，妨碍快速的应用程序开发。新版的 32 位 Delphi 采用了许多新的技术，在优化时不需要猜测，直接生成高性能代码。此外，新版的 32 位 Delphi 编译器仍然是世界上最快的本机代码编译器，在奔腾机器上每秒编译

350 000 行程序。结果，开发人员不仅提高了开发效率，同时改进了应用程序的性能。

新版的 32 位编译器采用了一系列的自动优化技术，包括寄存器优化、消除过度栈调用、共用子表达式消除、循环引入变量等。所有这些优化不仅保证了执行的正确性，并且不改变源代码的语义。

新版的 32 位编译器的优化选项可以关闭，以用于基准测试的比较。新版的 32 位编译器可以生成“Pentium-safe FDIV”代码，保证了 Delphi 应用程序的浮点运算即使在有错误的奔腾芯片中也可以正确运行。

1. 寄存器优化

经常使用的变量和参数将会被自动放在 CPU 的寄存器中，这样便减少了访问该变量而必须执行的指令的数目。由于程序不再从内存中读取变量放入寄存器中，不仅是代码紧凑，而且提高了执行速度。这种优化是由编译器自动完成的，开发人员无需指定把哪一个变量或参数放在寄存器中。编译器会自动分析变量的生命周期，进行寄存器优化。例如：如果变量 I 和 J 分别仅用在独立的程序段中，编译器会把 I 和 J 放在同一个寄存器中。

2. 消除过度栈调用

在尽可能的情况下，传给函数和过程的参数最好放在 CPU 的寄存器中。这样一方面可以减少对内存的访问，另一方面减少了栈空间为零时存储变量和栈空间的管理，提高了执行速度，降低了内存开销。

3. 共用子表达式消除

在编译器翻译复杂的数据表达式时，会自动找出相同的子表达式，以防止运算多次。这样一来，开发人员可以自由、清晰地书写他们的数据表达式而无需考虑优化，把这一工作留给编译器。

4. 循环引入变量

编译器会自动使用循环引入变量来加速对数组或字符串的访问。如果一个变量仅用来索引一个数组，例如在一个 for 循环中，编译器会自动引入一个变量，以指针的方式来访问数组。如果变量的尺寸是 1、4 或 8，Intel 的基数索引（scale indexing）会进一步提高性能。

1.2.3 新的优化连接器

作为编译过程的一个部分，Delphi 使用一种新的 32 位连接技术。这种新的 32 位连接技术也将提供若干优化，提高代码运行效率。因为新的智能连接器将消除无用的函数和过程，以及无用的静态和虚方法，这样，Delphi 应用程序不仅代码紧凑，而且执行速度快。通过消除虚方法，应用程序的对象束定速度更快，因为不必要的代码已被自动删除。新的连接器由于采用了单元缓冲的方法，连接速度提高了 20%到 50%。这意味着第一次编译后，程序未发生改变的部分可以直接从内存读，而不是从硬盘。

连接器还采用了一种所谓的智能版本检查技术，尽量避免重新编译，从而加快了编译速度。例如，一个库单元的一个函数被四个不同表使用，一旦这个函数发生变化，这四个表都要重新编译，因为编译后的代码存储在硬盘上，符号表发生变化，整个代码必须重新编译。而现在采用了智能编译和内置版本检查，在 DCU 文件中的被编译的代码采用更安全和灵活的格式，只有那些被修改的函数才被重新编译。这给第三方的厂商也带来了好处，他

们的函数库在新版的 Delphi 发表后，也不需要重新编译。

此外，新版的 32 位 Delphi 支持 OBJ 文件格式，这便允许在 Delphi 和 C /C++之间方便地共享代码，以及创建和共享 DLL。由于市场上有大量的商业函数库，这一功能使 Delphi 可以充分利用它们。

1.2.4 增强的 32 位功能

新的 32 位本机代码编译器运行在 32 位“平板”地址空间，完全消除了 16 位分割的段地址空间带来的限制。对程序员来说，这意味着无需求助于 Windows API，便可以充分利用机器的内存空间。例如，程序员可以自由地声明大尺寸的数据结构，在 Windows 95 下甚至可以创建一个 1G 的数组。而以前用 16 位时，数据结构不可能超过 64K 的限制。

1.2.5 新的 32 位数据类型

新的 32 位本机代码编译器引入了几个新的数据类型，以充分利用 32 位的大地址空间，但又不丧失对 16 位代码的控制。新的数据类型包括：

- 长字符串，仅受制于操作系统内存
- 宽字符，双字节字符
- 变体，为数据库访问和 OLE automation 提供运行时刻的变量类型改变的机制。

1.2.6 改进的编译器错误消息和诊断

对于一个本机代码编译器，人们常常忽视的一个优点是，它可以在程序执行前，向程序员提供完全的程序检查。编译器通常可以不通过解释器发现由于错误代码导致的逻辑错误。因为 Object Pascal 语言是一种强类型的语言，它可以防止程序员在类型转换方面的错误。新的 32 位编译器还有一个“mutil-error”体系，在发现错误之后，可以继续编译以发现更多的错误。这对大程序的开发非常有利。

新的 32 位编译器改进了错误消息和诊断，方便了程序员查错，防止了许多错误的发生，包括：

- 使用没有初始化的变量和指针
- 无用的变量
- 无用的函数返回值
- 空循环
- 类型不匹配

在语法错误方面，编译器也提供了更好的诊断信息，这对于刚学习 Object Pascal 的程序员非常有价值。编译器提供的诊断信息不仅是在哪一行出现错误，而且有对错误的解释。诊断信息包括一般性的语法错误，例如：忘记了分号；在 ELSE 语句前加了分号等。

1.2.7 本机代码编译器的优点

一般的 4GL 系统生成 C 代码，然后用 C 编译器进行编译。而 Delphi 直接生成优化后的本机代码，这使得 Delphi 开发速度快，生成代码执行效率高。它比两阶段代码生成方法的优越之处在于：快速反复，Delphi 采用了世界上最快的本机代码编译器，提高了开发效

率，鼓励快速应用程序开发 RAD (Rapid Application Development)；易于测试，在开发环境中开发的代码和实际运行的代码完全一样，开发人员无需担心解释器和编译后执行的代码效果不一样；高层调试，由于程序员写的代码直接编译成机器代码，调试器允许在源代码级上调试，而不是在系统生成的 C 代码上进行调试；维护简单，开发人员仅在 Object Pascal 层维护代码，而不是在 4GL 和 C 两个层次上进行维护。

P-code 系统和代码生成器在历史上曾发挥过作用，例如：早期在 Apple II 上运行的语言采用了 UCSD P-code 系统。但是由于的它的执行效率低，现在许多开发人员不再愿意采用这样的代码。同样，以前许多 C++版本都是先生成 C 代码，然后借助 C 编译器编译成可执行代码，但是，这样做的缺点是编译速度慢，难以调试。所以，现在大多数 C++编译器均直接生成本机代码。

32 位的 Delphi 和 16 位代码完全兼容。16 位代码程序转变成 32 位代码程序必须重新编译，但是，要作改写的程序非常少。在大多数情况下，开发人员仅需把 16 位代码程序装入开发环境，重新编译就行了。Delphi 会自动处理 Windows 消息类型的改变，而控制代码却不需要修改。需要改变的是那些依赖于物理表示的代码，包括：16 位嵌入汇编；在 Win32 中改变的 Windows API；依赖于整数物理大小的记录和子程序。

在 32 位的 Delphi 中，整数类型的大小为 4 个字节，而不是 16 位中的 2 个字节。为了保持兼容，Delphi 提供了一个新的类型 SmallInt。

新的 32 位的 Delphi 还提供了单元别名机制，可以为同一个单元起多个名字。这将有助于动态地修改一个单元，而充分利用 32 位的优势。例如，在 16 位的 Delphi 中有两个分开的单元：WinTypes 和 WinProcs。在 32 位版本中，这两个单元可以联合成一个单元。为了保证兼容性，单元别名被引入，使得老的代码无需修改。

用新的 32 位的 Delphi 开发的应用程序，如果没有利用 32 位的功能，可以在 16 位下编译，运行在 Windows 3.X 上。如果采用了 32 位的功能，例如：Windows 95 界面、Win32 API 函数等，则必须修改后才能编译成 16 位代码。

总之，新的 32 位的 Delphi 是一个崭新的产品，它建立在高性能的优化编译器之上，充分挖掘了 Windows 95/NT 的 32 位功能。由于采用了编译优化、连接优化、新的 32 位数据类型，它明显胜过 P-code 解释系统。此外，32 位的 Delphi 拥有众多的封装了 Windows 95 界面的控件，完全支持 32 位 OLE 控件(OCX)和 OLE automation，以及强大的 Borland Database Engine。

1.3 Delphi 编程思想

Delphi 可以说是可视化面向对象编程的一场革命。可视化编程继承了 Visual BASIC 已有的成果，使用户在程序设计时要写的代码为最少。另一方面，Delphi 继承了其前辈 Borland Pascal 7.0 for Windows 已有的成果，完全采用了 OOP 的面向对象程序设计的思想，这样在使用 Delphi 时，就更易于程序的开发与维护。

1.3.1 面向对象编程的基础

面向过程的程序设计方法从解决问题的每一个步骤入手，适合于解决比较小的简单问

题。C 语言采用面向过程的程序设计模型，但是由于 C 本身几乎没有支持代码重用的语言结构，并且缺乏统一的接口，使得当程序的规模达到一定程度时，程序员很难控制其复杂性。

假如要写一个程序来计算一个学期中进行三次测试后一个班的平均成绩。面向过程的程序设计方法将采用如下步骤：

(1) 输入学生姓名。

(2) 输入第一次测验的分数。

(3) 输入第二次测验的分数。

(4) 输入第三次测验的分数。

(5) 计算并显示平均分。

图 1-1 为上述步骤的逻辑流程图。

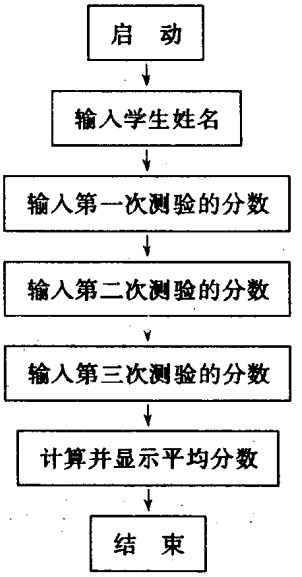

图 1-1　面向过程程序设计方法的逻辑流程图

程序首先在屏幕上显示一个提示符，请用户输入学生姓名，然后，下一个屏幕又提示用户输入第一次测验的成绩，再下一个屏幕提示输入第二次、第三次测验的成绩。输入了三次测试成绩后，则会出现一个屏幕以显示计算出的平均成绩。这个步骤是有逻辑关系的，而且遵循事件的顺序结构。然而，使用这样的程序，您必须遵循能设计的过程，而不能让用户有额外的空间在进入了第五步以后再去改变第三步的成绩。

虽然在这样的程序中也有许多处理例外的方法，但是，即使这样的例外处理也仍然是有顺序的、过程驱动的结构。

面向对象的程序设计方法则按照现实世界的特点来管理复杂的事物，把它们抽象为对象，具有自己的状态和行为，通过对消息的反应来完成一定的任务。我们可以将图 1-1 改为面向对象的程序设计的流程框图，如图 1-2 所示。