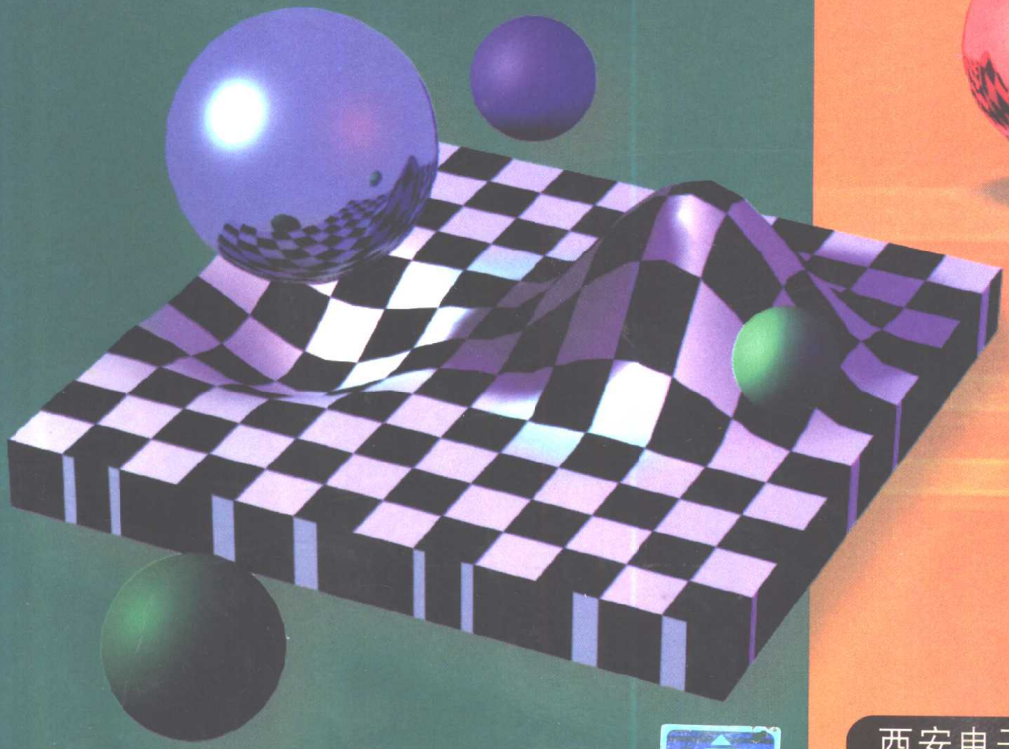
高等学校电子信息类教材

Delphi面向对象程序设计及其应用

朱振元 朱 承 编著



西安电子科技大学出版社
http://www.xduph.com

高等学校电子信息类教材

Delphi 面向对象程序设计及其应用

朱振元　朱　承 编著

西安电子科技大学出版社

2000

内 容 简 介

Delphi 是一个面向对象的软件开发工具，它可以直观地、快速地进行 Windows 应用程序的开发。本书在介绍 Delphi 基本操作及编程方法的同时，着重介绍了应用程序的开发过程和实现技巧。教材内容完全按照程序设计人员的视点进行组织，并遵循循序渐进的原则。全书由程序设计及应用开发两个部分组成，程序设计部分每一章中都有一个综合性的应用实例并围绕程序实例组织章节内容；应用开发部分则介绍一个完整的开发程序。

为适应专业素质教育，本书侧重于学生应用程序开发能力的训练与提高，可作为高等院校计算机专业的教科书，也可作为应用程序开发人员及电脑爱好者的技术参考书。

图书在版编目（CIP）数据

Delphi 面向对象程序设计及其应用 / 朱振元，朱承编著. —西安：西安电子科技大学出版社，2000.11
高等学校电子信息类教材
ISBN 7-5606-0936-8

Ⅰ. D… Ⅱ. ① 朱… ② 朱… Ⅲ Delphi 语言-程序设计-高等学校-教材
Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2000）第 49613 号

责任编辑 戚文艳 云立实
出版发行 西安电子科技大学出版社（西安市太白南路 2 号）
电 话 (029)8227828 邮 编: 710071
http://www.xduph.com E-mail: xdupfxb@pub.xaonline.com
经 销 新华书店
印 刷 西安辞源印刷厂
版 次 2000 年 9 月第 1 版 2000 年 9 月第 1 次印刷
开 本 787 毫米×1092 毫米 1/16 印张
字 数 470 千字
印 数 1~2 000 册
定 价 21.00 元
ISBN 7 - 5606 - 0936 -8 / TP • 0855

前　言

近年来计算机的软件开发技术有了变革性的飞速发展，一大批面向对象的开发工具相继出现，并日益显示出其强大的生命力。

Delphi 是一个极有代表性的面向对象开发工具，它将面向对象的程序设计方法与数据库技术、网络技术以及可视化、事件驱动、代码自动生成等先进技术完美地结合在一起，使用它可以直观地、快速地开发出高质量的 Windows 应用程序。

为了适应计算机软件开发技术的发展，并使计算机教材能较好地体现出专业素质教育的特点，我们编写了这本教材。

本教材以培养与提高学生的基本专业素质及综合应用能力为追求目标，在介绍 Delphi 基本操作及编程方法的同时，注重介绍了应用程序的开发过程和实现技巧，通过一些典型示例的介绍及综合开发实例的剖析，使读者能较快地具备使用 Delphi 进行应用程序开发的能力。

全书分为四个单元 18 章，由两部分内容组成，前一部分介绍 Delphi 程序设计，后一部分则介绍一个完整的开发实例。

教材内容完全按照程序设计人员的视点进行组织，并遵循循序渐进的原则。第一单元“基本编程”在介绍面向对象的基本概念及 Delphi 开发环境之后，围绕“输入输出”、“文本编辑”及“图形处理”等程序设计中的基本功能介绍了有关的应用程序的实现过程；第二单元“深入编程”则更深入地介绍建立高品质的应用程序所要求的各种功能，包括“功能组织”、“界面布置”、“操作设置”以及“多媒体及对象处理”等等。第三单元“数据库编程”介绍数据库及网络数据库的编程方法，分为“数据库操作”、“维护程序”、“查询程序”、“统计程序”及“网络数据库编程”等几个部分。每一章中都有一个综合性的应用实例并围绕该实例的实现过程组织编排的内容。第四单元“开发实例”介绍了一个完整的开发实例“Delphi 通用试题库管理系统”。为了便于对该实例进行剖析与教学，我们将整个系统按功能分为四个部分，每个功能部分又分成若干个实现步骤，先提出一个基本的实现，此后的各个步骤都在前一个步骤的基础上逐次扩展其功能，最后形成一个完整的功能部分。

本书所使用的 Delphi 是 Delphi5.0 的版本。

由于作者水平有限，时间仓促，书中难免存在缺点与疏漏，敬请读者及同行们予以批评指正。

朱振元

2000 年 8 月于长沙大学

目　录

第一单元　基 本 编 程

第二单元 深 入 编 程

第三单元　数据库编程

第四单元 开发实例

第一单元　基本编程

第 1 章　面向对象应用开发概述

Delphi 是一种面向对象的开发工具，它是完全按面向对象的程序设计方法构造应用程序的，因此在学习 Delphi 前，先了解面向对象程序设计的方法及特点是有必要的。本章介绍面向对象程序设计的方法、特点及其所涉及的基本概念。

1.1　面向对象的程序设计方法

在介绍面向对象程序设计方法之前，先来考察一下传统的面向过程程序设计方法的一些特点。

假如要编制一个程序演示栈操作的执行过程。在程序中用一个字符表示栈中的一个元素，由输入字符而引起栈中当前元素的变化。若输入字符“P”，则表示执行出栈操作；若为“E”，则表示退出执行，否则将该字符推入栈中。栈的初始状态为空，程序从开始起顺序地执行直至输入字符“E”。

使用面向过程的方法可编制以下一段程序：

```
Var top:link; ch:char;
Procedure push(el:char);
…
Function pop:char;
…
begin
  top:=NIL; ch:= 'a';
  while ch<> 'E' do
  begin
    输入一个字符存入 ch;
    对 ch 进行判别并进行相应的处理;
    输出栈中的当前元素
  end;
end;
```

在上述程序中，定义了一个变量 top 及过程 push 和函数 pop，它们都与栈操作有关，但在程序中并没有建立它们之间的必然联系，程序中的任何地方都可以对 top 进行访问。由

此可见在面向过程的程序设计方法中，数据和过程之间并没有必然的联系，而且程序是从开始至结束顺序地执行的。

这种方法着眼于系统要实现的功能。从系统的输入和输出出发，分析系统要做哪些事情，进而考虑如何做这些事情，自顶向下的对系统的功能进行分解，来建立系统的功能结构和相应的程序模块结构。但是，当程序因某种原因需要修改时，常常要涉及到许多模块，有时因功能的改变要导致全部模块都要变更，这样的修改工作量极大并容易产生新的错误。

而面向对象的程序设计方法完全避免了面向过程方法中所存在的问题。

在面向对象的程序方法中，相关的数据及操作被统一在一个整体——对象之中。例如，在栈演示程序中，可以先将栈定义成类：

```
Tlz= class
  Private
    top:Tnode;
  public
    procedure init;
    procedure prnt;
    function   pop :elemtp;
    procedure push (el:elemtp);
end;
```

并建立一个相应的实体

```
lz1:=Tlz.Create;
```

然后通过对该实体执行相应的操作来实现演示程序的功能。例如，先将栈清空，然后将一个元素推入栈中，再显示这个栈，可执行下述代码：

```
lz1.init;
lz1.push(el);
lz1.prnt;
```

上述代码段不仅比较简洁，容易理解，而且也不会随类中实现细节的改变而改变。

面向对象的方法着眼于应用问题中所涉及的对象，识别为解决问题所需的各种对象、对象的属性及相应的操作，从而建立起对象的类结构。通过对类的实体施行相应的操作以及各实体间的消息传递来实现系统的功能。类的定义充分体现了抽象数据类型的思想，基于类的体系结构可以把程序的修改局部化。当类中数据的存储方式及操作的实现过程需要修改时，不会影响外界对该类实体的操作，从而使整个系统保持稳定。因此，用面向对象开发方法建立起来的软件易于修改，与传统的方法相比，程序具有更好的可靠性、适用性、可修改性、可维护性、可复用性和可理解性。

以上结合栈演示程序的实现过程，对面向过程与面向对象两种程序设计方法进行了粗略的比较，从中我们可以大致领略到面向对象程序设计的基本方法。

1.2 面向对象程序设计中的基本概念

在问题空间中，我们将客观世界的实体称为对象，不同对象之间的互相作用和互相通

信构成了完整的客观世界。如何将问题空间的这一思维模型直接映射到程序空间，也就是说，面向对象的程序设计方法提供什么概念来支持这一思维模型？其中最核心的概念是类、数据封装、继承和多态性。下面介绍这些基本概念。

1.2.1 对象、类和实例

在面向对象的程序设计中，对象是指应用问题中所出现的各种实体，它是由一组属性值和在这组值上的一组操作（方法）构成的，其中属性值确定了对象的状态。例如，在程序中常用的字符串、线性表、栈、队列等或在 Windows 应用程序中常见的窗体、组合框、编辑框、无线按钮等都可看作为对象。

对象在语言中是用类来定义的，类中定义了与某一种对象相关联的一组数据以及施与该数据的一组基本操作，对象是数据与操作的统一体。

在 Delphi 中，类的保留字是 class，下面是一个类定义的例子。

```
Tint= class
  Private
    v:integer;
  public
    constructor init(val:integer);
    procedure inc;
    procedure dec;
    function   get :integer;
end;
implementation
constructor Tint.init(val:integer);
begin
  v:=val;
end;
procedure Tint.inc;
begin
  inc(v);
end;
procedure Tint.dec;
begin
  dec(v);
end;
function   Tint.get :integer;
begin
  result:=v;
end;
```

在上述类定义中，所定义的对象相当于一个计数器，涉及的数据为一个整数 v，相关的操作为设置计数初值、计数增 1、计数减 1 以及读取计数值等。

在面向对象的程序设计中，如果某一个变量被定义成属于某一个类，那么该变量即可

成为这种对象中的一个实例。因此，对象与实例这两个概念在程序设计中可对应于类与变量。对象、类是抽象的概念，而实例、变量代表具体事物（但有时也将对象实例直接称为对象）。

例如，以下的语句定义了 Tint 类的两个对象实例：

```
var int1,int2: Tint;
```

在 Delphi 中，对象实例在定义后还要进行生成才能使用。在上例中，如果 int1、int2 的初值分别设定为 0 与 1，则可使用下述语句进行生成：

```
int1:=Tint.init(0);
int2:=Tint.init(1);
```

又如，在操作界面的设计中，可以在一个窗体中设置多个编辑框，尽管其位置及外观都不相同，但它们都是属于编辑框组件（类）所派生的对象实例。

1.2.2 数据封装（信息隐蔽）

数据封装是指在面向对象的程序设计中，对象的实现过程（包括数据的存储方式、操作的执行过程）作为私有部分被封装在类结构中，使用者不能看到，也不能直接操作该类中所存储的数据，而只能根据对象提供的外部接口来访问或操作这些数据。

例如，在 1.2.1 小节中所给出的 Tint 类中，变量 v 被封装在类中，类的外界不能直接访问它，因此我们不能使用以下的语句来使实例 int1 中的变量 v 增加 1：

```
int1.v:=int1.v + 1;
```

而只能通过 Tint 类中所提供的接口来访问它，以下的语句才是合法的：

```
int1.inc;
```

在类定义中实现数据封装，就像在学校的四周砌了一道高高的围墙，所提供的接口就像学校所设置的传达室，这样外来人员只能通过传达室来访问学校中的人，这对学校中的人来说是比较安全的。

在传统的面向过程的方式中，虽然也提供过程与函数的调用接口，但并没有将对象作为程序的基本构件，将一组相关的数据及操作集中在一起，在数据与操作之间缺乏明确的关系，这就不能达到数据封装的目的。

数据封装无论是对使用者或实现者都是相当有利的。从使用者的角度来看，只要了解类定义的接口部分，即可操作对象实例，而不必去关心其实现细节。这就好比我们使用手表，只要按操作说明使用它，而不必了解手表的内部结构。这样使用者在开发过程中就可以集中精力去解决应用中所出现的问题，使问题得到简化，而且程序设计的表达方式也更加简练、直观。从实现者的角度来看，数据封装也有利于编码、测试及修改。因为只有类中的成员函数才能访问它的私有成员，这样做可以使错误局部化，一旦出现错误或者有必要改变数据的存储方式或改变内部的处理过程，也不至于影响其它模块。只要向外界提供的接口方式不变，其它所有使用该对象的程序都可以不变，从而大大提高了程序的可靠性和稳定性。

数据封装是通过将相应的数据定义成私有变量来实现的。“私有”是一种访问权限，在 Delphi 中允许使用以下 3 种访问权限，其名称与含义如下：

名　称	保　留　字	访 问 权 限
公有	public	任何可以引用该类的地方都可以访问
保护	protected	只有在本类及派生类中可以访问
私有	private	只有在本类中的方法可以访问

一般可将要封装的数据定义成私有变量，而将向外界提供的对该变量进行访问的过程（或函数）定义为公有过程。

在 Delphi 中，上述所有保护等级都是以单元为界限来设定的，在同一单元内的各类之间，不存在保护等级上的区别；同一单元内不同类之间的关系如同 C＋＋语言中的“友元”之间的关系，不受保护等级的约束。

1.2.3　继承与派生

在面向对象的方法中，类与类之间存在着继承关系，继承机制是在面向对象方法中最有特色的方面。所谓继承，是指从已定义的类导出新类时，新类将自动包括原有类的全部数据和方法，这种导出新类的过程称为派生。

在 Delphi 中，类的继承的表示方法如下：

```
Tnewobject= class(Toldobject)
              …
            end;
```

上述类定义中，Tnewobject 是从 Toldobject 派生而来的，Tnewobject 称为子类（或派生类），而 Toldobject 称为父类（或基类），子类继承了父类的所有数据和方法。

这种继承关系与客观世界中存在的一般与特殊的关系相类似。例如，Windows 的界面设计中，可将窗体分为一般窗体和对话框，而对话框又可分为打开对话框、确认对话框等，这些窗体都有窗体的共同特征，但不同的窗体又有自身的不同特征。我们可以将窗体定义成基类，建立它的派生类，如对话框、打开对话框等。将各派生类中的共同部分集中到基类中去，派生类中只保留自己特有的属性和方法，派生类的各对象独享该派生类的属性和方法，同时还能共享基类中共有的属性和方法，这样做的好处是可以合理地将各个对象的属性和方法分配到所有的类中，减少数据存储程序代码的重复。

下面是一个继承的例子：

```
Tperson= class
  Private
    Name:string;
    age:integer;
  public
    constructor init(aname:string;aage:integer);
    function   getname :string;
    function   getage   :integer;
end;
```