

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

*** 如有印装问题可调换 ***

本书封面贴有西安电子科技大学出版社的激光防伪标志, 无标志者不得销售。

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1章 面向对象应用开发概述	1
1.1 面向对象的程序设计方法	1
1.2 面向对象程序设计中的基本概念	2
1.2.1 对象、类和实例	3
1.2.2 数据封装（信息隐蔽）	4
1.2.3 继承与派生	5
1.2.4 多态性	7
1.3 面向对象开发工具中的基本概念	8
1.3.1 消息与事件驱动	9
1.3.2 可视化	9
1.3.3 事件处理	10
1.3.4 组件	10
1.3.5 属性	11
1.3.6 方法	11
第2章 创建一个简单的 Delphi 应用程序	12
2.1 Delphi 5.0 的集成开发环境	12
2.1.1 主菜单及快捷按钮栏	13
2.1.2 组件板	14
2.1.3 对象监视器	14
2.1.4 窗体与代码编辑器	16
2.1.5 项目管理	17
2.1.6 环境参数设置	19
2.1.7 开发界面的调整	19
2.2 应用程序的开发过程	20
2.2.1 一个简单的应用程序	20
2.2.2 创建过程的基本步骤	21
2.3 应用程序的基本组成	24
2.3.1 项目文件	24
2.3.2 单元文件	25
2.3.3 窗体文件	27
2.3.4 变量的作用范围	27
第3章 输入、输出处理	29
3.1 程序实例：四则运算应用程序	29
3.2 窗体设计	30
3.2.1 窗体的主要属性	30
3.2.2 窗体的主要事件	32
3.2.3 窗体设计实例	32
3.3 基本输入、输出组件	33
3.3.1 标签（Label）	33
3.3.2 编辑框（Edit）	34
3.3.3 数字增减器（SpinEdit）	36

3.4 选择输入组件.....	37
3.4.1 列表选择组件.....	37
3.4.2 组合框.....	38
3.4.3 复选框.....	38
3.4.4 无线按钮.....	39
3.4.5 分组框.....	39
3.4.6 无线按钮组.....	40
3.4.7 选择输入组件的应用实例.....	40
3.5 按钮.....	42
3.5.1 基本按钮 (Button)	42
3.5.2 图形按钮 (BitBtn)	43
3.5.3 按钮组件的应用实例.....	43
3.6 输入、输出对话框.....	44
3.6.1 信息显示.....	44
3.6.2 信息对话.....	45
3.6.3 信息输入.....	46
3.6.4 应用信息.....	46
3.7 异常处理.....	47
3.8 四则运算应用程序的实现.....	48
3.8.1 功能要求.....	48
3.8.2 组件设置.....	48
3.8.3 实现要点.....	49
3.8.4 程序清单.....	50
第4章 文本编辑处理.....	52
4.1 程序实例: 文本编辑程序.....	52
4.2 文件管理过程调用.....	53
4.3 通用对话框组件.....	54
4.3.1 文件打开与保存对话框.....	54
4.3.2 字符串查找与替换对话框.....	56
4.3.3 字体与颜色设置对话框.....	59
4.4 多行编辑组件.....	62
4.4.1 Memo 组件.....	62
4.4.2 RichEdit 组件.....	63
4.5 文本编辑程序的实现.....	65
4.5.1 功能要求及组件设置.....	65
4.5.2 功能实现.....	65
4.5.3 程序清单.....	68
第5章 图形处理.....	72
5.1 程序实例: 循环队列演示程序.....	72
5.2 定时器组件.....	73
5.2.1 Timer 组件的基本使用方法.....	73
5.2.2 程序实例: 小球滚动程序.....	73
5.3 绘图.....	75
5.3.1 Canvas 对象的基本属性.....	75
5.3.2 使用 Canvas 的绘图方法.....	76
5.3.3 绘图板 (PaintBox) 组件.....	80
5.3.4 图形 (Shape) 组件.....	81

5.3.5 处理重画事件.....	81
5.3.6 程序实例：动态图形程序.....	82
5.4 图像文件的处理.....	84
5.4.1 图像类.....	84
5.4.2 图像显示 (Image) 组件.....	85
5.4.3 图像组 (ImageList).....	86
5.4.4 程序实例：时钟模拟程序.....	87
5.5 循环队列演示程序的实现.....	89
5.5.1 功能要求及组件设置.....	89
5.5.2 实现要点.....	89
5.5.3 类定义.....	90
5.5.4 类的实现.....	90
5.5.5 界面功能的实现.....	91
5.5.6 程序清单.....	92

第二单元 深入编程

第6章 功能组织	97
6.1 程序实例：多功能应用程序.....	97
6.2 多文档界面设计.....	98
6.2.1 MDI 父窗体和子窗体.....	98
6.2.2 在项目中增删窗体.....	99
6.2.3 指定主窗体及窗体的生成方式.....	99
6.2.4 窗体的动态生成及关闭.....	100
6.2.5 子窗体的状态控制.....	100
6.3 主菜单 (MainMenu).....	101
6.3.1 创建主菜单.....	101
6.3.2 菜单项设计.....	102
6.4 快捷按钮及组合板.....	103
6.4.1 快捷按钮 (SpeedButton).....	103
6.4.2 组合板 (Panel).....	104
6.4.3 建立快捷按钮板.....	105
6.5 弹出式菜单 (PopupMenu).....	106
6.6 设置帮助功能.....	107
6.6.1 作成 RTF 文件.....	107
6.6.2 作成 HLP 文件.....	108
6.6.3 将 HLP 文件设置到应用程序中.....	110
6.7 多功能应用程序的实现.....	111
6.7.1 功能要求.....	111
6.7.2 组件设置.....	111
6.7.3 设计步骤.....	112
6.7.4 实现要点.....	112
6.7.5 程序清单.....	114
第7章 界面布置	117
7.1 程序实例：试题输入程序.....	117
7.2 多页卡界面设计.....	119
7.2.1 Win 3.1 中的有关组件.....	119
7.2.2 PageControl 组件.....	120

7.3 组件的布置.....	120
7.3.1 使用 Align 属性.....	121
7.3.2 使用 Anchor 属性.....	123
7.3.3 使用布件工具.....	123
7.4 分隔调整器.....	126
7.4.1 Splitter 组件的使用方法.....	126
7.4.2 Splitter 组件的使用示例.....	126
7.5 试题输入程序的实现.....	127
7.5.1 界面设计.....	127
7.5.2 设计步骤.....	129
7.5.3 实现要点.....	129
7.5.4 程序清单.....	131
第 8 章 操作设计	135
8.1 程序实例：组件设置程序.....	135
8.2 键盘控制.....	136
8.2.1 键盘操作的有关事件.....	136
8.2.2 键盘控制程序示例.....	138
8.3 鼠标控制.....	140
8.3.1 单击、双击和移动.....	140
8.3.2 鼠标的拖动.....	143
8.3.3 鼠标形状的改变.....	144
8.4 对象焦点的转移.....	145
8.4.1 取得焦点的方法.....	145/
8.4.2 焦点转移程序示例.....	146
8.5 组件设置程序的实现.....	146
8.5.1 界面外观及功能要求.....	147
8.5.2 组件设置.....	147
8.5.3 设计步骤.....	148
8.5.4 实现要点.....	148
8.5.5 程序清单.....	151
第 9 章 多媒体及对象处理	156
9.1 程序实例：多媒体播放程序.....	156
9.2 剪贴板对象.....	157
9.2.1 剪贴板类.....	157
9.2.2 剪贴板对象的使用.....	158
9.3 对象的链接和嵌入.....	159
9.3.1 OLE 的基本概念.....	159
9.3.2 对象容器及其使用方法.....	159
9.4 多媒体程序设计.....	161
9.4.1 Animate 动画组件.....	161
9.4.2 音响提示.....	161
9.4.3 MediaPlayer 组件.....	162
9.5 多媒体播放程序的实现.....	163
9.5.1 界面设计.....	163
9.5.2 实现要点.....	164
9.5.3 程序清单.....	165

第三单元 数据库编程

第 10 章 数据库操作	168
10.1 程序实例：人事库基本维护程序.....	168
10.2 Delphi 数据库工具.....	169
10.2.1 Delphi 数据库管理系统的特点.....	169
10.2.2 数据库桌面 (DBD).....	170
10.2.3 数据库驱动器 (BDE).....	172
10.2.4 数据库浏览器 (Database Explorer).....	173
10.3 数据库创建.....	174
10.3.1 定义数据库别名.....	174
10.3.2 建立数据表结构.....	174
10.3.3 建立索引.....	175
10.3.4 设置选项.....	175
10.3.5 输入部分数据.....	176
10.4 数据库窗体向导.....	176
10.5 数据库打印.....	178
10.5.1 打印页面的格式.....	178
10.5.2 常用的打印组件.....	179
10.5.3 打印功能的实现.....	179
第 11 章 数据库维护程序	181
11.1 程序实例：人事库维护程序.....	181
11.2 数据存取组件.....	182
11.2.1 Table 组件.....	183
11.2.2 Query 组件.....	185
11.2.3 TField 类.....	186
11.2.4 Datasource 组件.....	187
11.2.5 BatchMove 组件.....	187
11.3 数据库控制组件.....	188
11.3.1 DBGrid 组件.....	188
11.3.2 DBEdit、DBMemo 组件.....	189
11.3.3 DBImage 组件.....	189
11.3.4 DBNavigator 组件.....	190
11.3.5 DBLookupComboBox 组件.....	191
11.4 数据库操作的实现.....	192
11.4.1 移动记录指针.....	192
11.4.2 数据的存取及转换.....	193
11.4.3 编辑、确认与取消.....	194
11.4.4 增加、删除记录.....	194
11.5 人事库维护程序的实现.....	194
11.5.1 界面设计.....	194
11.5.2 设计步骤.....	196
11.5.3 实现要点.....	196
11.5.4 程序清单.....	199
第 12 章 数据库查询程序	203
12.1 程序实例：人事库查询程序.....	203
12.2 顺序查找.....	204

12.2.1 字符串比较、匹配函数	205
12.2.2 指定范围的顺序查找	206
12.2.3 指定组合条件的顺序查找	207
12.3 快速查询	209
12.3.1 用于快速查找的函数过程方法	209
12.3.2 快速查询示例	209
12.4 利用多表同步进行查询	211
12.4.1 建立多表同步的基本步骤	211
12.4.2 建立多表同步的程序示例	212
12.5 利用 TQuery 组件进行查询	213
12.5.1 直接设置 SQL 实现查询	213
12.5.2 通过组装 SQL 语句实现查询	214
12.5.3 通过设置 SQL 参数实现查询	215
12.6 树形组件及层次查询	215
12.6.1 TreeView 组件的基本使用方法	216
12.6.2 TreeView 组件的程序示例	217
12.7 人事库查询程序的实现	219
12.7.1 界面外观及功能要求	219
12.7.2 组件设置	219
12.7.3 设计步骤	220
12.7.4 实现要点	220
12.7.5 程序清单	223
第 13 章 数据统计程序	227
13.1 程序实例：人事信息统计程序	227
13.2 程序实现统计	228
13.2.1 统计图显示 (ChartFX) 组件	228
13.2.2 统计程序示例	230
13.3 使用 SQL 实现统计	231
13.3.1 决策组组件	231
13.3.2 涉及单表的统计示例	232
13.3.3 涉及多表的统计示例	234
13.4 人事信息统计程序的实现	237
13.4.1 界面设计	237
13.4.2 实现要点	237
13.4.3 程序清单	238
第 14 章 网络数据库编程	241
14.1 程序实例：网络人事库维护程序	241
14.2 系统结构	242
14.3 相关的组件	243
14.3.1 数据模块和远程数据模块	243
14.3.2 数据连接组件	243
14.4 几种传送方式	244
14.4.1 远程访问数据表	244
14.4.2 设置 SQL 语句进行访问	246
14.4.3 设置 SQL 参数进行访问	247
14.4.4 修改远程数据库中的记录	248
14.5 网络人事库维护程序的实现	248

14.5.1 界面设计	248
14.5.2 实现要点	249
14.5.3 程序清单	250

第四单元 开发实例

第 15 章 STGL 系统封面及主菜单	254
15.1 STGL 系统设计概要	254
15.1.1 总体功能设计	255
15.1.2 数据库结构设计	255
15.1.3 屏幕外观及操作设计	256
15.1.4 可靠性及适应性设计	257
15.2 主子窗体	257
15.2.1 界面外观及功能要求	257
15.2.2 实现步骤	258
15.2.3 实现技巧与要点	258
15.2.4 组件设置	258
15.2.5 功能实现	259
15.3 系统封面与密码输入	260
15.3.1 界面外观及功能要求	260
15.3.2 组件设置及实现要点	261
15.3.3 功能实现	261
15.4 密码设置	262
15.4.1 界面外观及功能要求	262
15.4.2 实现要点及组件设置	262
15.4.3 功能实现	263
第 16 章 STGL 系统题库维护	265
16.1 基本的维护功能	265
16.1.1 界面外观及功能要求	265
16.1.2 实现技巧与要点	267
16.1.3 组件设置	267
16.1.4 处理流程	268
16.1.5 程序编制	269
16.2 题库范围的设置	270
16.2.1 界面外观及功能要求	270
16.2.2 组件设置及实现要点	270
16.2.3 处理流程	271
16.2.4 程序编制	271
16.3 图形的装入与维护	273
16.3.1 界面外观及功能要求	273
16.3.2 组件设置及实现要点	273
16.3.3 处理流程	274
16.3.4 程序编制	275
16.4 公式的装入与维护	277
16.4.1 界面外观及功能要求	277
16.4.2 组件设置及实现要点	277
16.4.3 处理流程	277
16.4.4 程序编制	278

第 17 章 STGL 系统选题及成卷	280
17.1 指定选题	280
17.1.1 界面外观及功能要求	280
17.1.2 组件设置及实现要点	281
17.1.3 功能实现	281
17.2 分值处理	283
17.2.1 界面外观及功能要求	283
17.2.2 组件设置及实现要点	284
17.2.3 处理流程	284
17.2.4 程序编制	284
17.3 自动选题	285
17.3.1 界面外观及功能要求	285
17.3.2 实现技巧与要点	286
17.3.3 组件设置	286
17.3.4 处理流程	286
17.3.5 程序编制	287
17.4 试卷组装	290
17.4.1 界面外观及功能要求	290
17.4.2 组件设置及实现要点	291
17.4.3 处理流程	291
17.4.4 程序编制	292
第 18 章 STGL 系统辅助功能	294
18.1 试卷分析	294
18.1.1 界面外观及功能要求	294
18.1.2 组件及变量	295
18.1.3 处理流程	295
18.1.4 程序编制	296
18.2 题库打印	297
18.2.1 界面外观及功能要求	297
18.2.2 组件设置及实现要点	297
18.2.3 处理流程	298
18.2.4 程序编制	298
18.3 题库备份	300
18.3.1 界面外观及功能要求	300
18.3.2 组件设置及实现要点	301
18.3.3 功能实现	301
18.4 课程维护	302
18.4.1 界面外观及功能要求	302
18.4.2 组件设置及实现要点	303
18.4.3 处理流程	303
18.4.4 程序编制	304
18.5 章节、题型库维护	306
18.5.1 功能要求及实现要点	306
18.5.2 功能实现	307
参考文献	309

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;age:integer);
    function  getname :string;
    function  getage  :integer;
end;
```