Delphi 程序设计基础

金林樵　主编
周智文　审

21世纪高职高专计算机科学与应用专业系列教材

机械工业出版社
CHINA MACHINE PRESS

21世纪高职高专计算机科学与应用专业系列教材

Delphi程序设计基础

金林樵　主编
周智文　审

机械工业出版社

Delphi 具有简明易学的特点，是开发 Windows 应用程序最为强大的工具之一，无论是初次接触 Windows 下程序设计的新手，还是有一定实践经验的 Windows 程序员，利用 Delphi 都能快速地开发出令人满意的应用程序。

本教程简明扼要地介绍了 Object Pascal 的语法、通过实例讲解了 Delphi 中各种常用组件的属性、方法和事件，以及它们的编程技巧。深入浅出地介绍了利用 Delphi 可视化开发环境进行程序开发的方法，通过实例详细地介绍了多媒体应用程序、单层数据库和 C/S 结构数据库应用程序的开发过程。

本书内容丰富，编程实例简捷实用，每章后均配有小结和习题，有利于学生对所学知识的巩固。本书适用于大专院校相关专业学生，及 Delphi 程序设计人员。

图书在版编目（CIP）数据

Delphi 程序设计基础 / 金林樵主编. —北京：机械工业出版社，2003.8
（21 世纪高职高专计算机科学与应用专业系列教材）
ISBN 7-111-12631-9

Ⅰ.D... Ⅱ.金... Ⅲ.软件工具-程序设计-高等学校：技术学校-教材 Ⅳ.TP311.56

中国版本图书馆 CIP 数据核字（2003）第 059936 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策　　划：胡毓坚
责任编辑：田　梅
责任印制：路　琳
高等教育出版社印刷厂印刷 · 新华书店北京发行所发行
2003 年 8 月第 1 版 · 第 1 次印刷
787mm × 1092mm1/16 · 18.5 印张 · 457 千字
0 001-5000 册
定价：26.00 元

21世纪高职高专计算机
科学与应用专业教材编委会名单

出 版 说 明

新世纪对高职高专教育提出了新的目标和要求，高职高专教育面临新一轮的改革和发展。为了进一步推进高职高专的教育，培养 21 世纪与我国现代化建设相适应的，具有较宽厚的文化基础底蕴，并在生产、管理、服务岗位第一线的技术型应用型人才。机械工业出版社与高职高专计算机科学与应用专业教材编委会联合组织了全国近百所院校的一线骨干教师，在交流、研讨的基础上，根据国家教育部的精神，以及高职高专教学改革的新思路、新突破、新经验、新举措，编写了此套“21 世纪高职高专计算机科学与应用专业系列教材”。目前已出版了 2 轮，近 30 种教材。随着教改的深入，新技术的出现，新一轮的高职高专教材将陆续出版。

第一轮教材更明确高职高专学生培养的定位，更强化学生实践能力和创新意识的培养，更反映现代高等职业技术教育的理念、方法和手段，更注重培养第一线的技术应用型人才。新的教材是将高职高专院校教学改革力度比较大，内容新颖，注重能力，体现创新的教材，或者各院校急需使用，适合社会经济发展新课题的教材列入选题规划，进行修编或新编。力求体现“定位准确、注重能力、内容创新、结构合理和叙述通俗”的编写特色。新教材是由个人申报，经各院校推荐，编委会会同专家评选，出版社立项出版的。

望各高职高专院校积极选用本套系列教材，及时提出修改意见，不断提高教材的编写质量。

高职高专计算机科学与应用专业教材编委会

机械工业出版社

前　言

面向对象的程序设计（OOP）理念已成为当前程序设计的主流，它改变了以往传统的程序设计模式，并日益显示出其强大的生命力。

Delphi 由于其强大的功能、快捷方便的开发模式，已成为一个极具代表性的面向对象的开发工具。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、代码自动生成等先进技术完美地融合在一起，给编程人员提供了一个超强高效的开发环境。

本书共分 9 章。第 1 章介绍了 Object Pascal 的基本语法要素、函数与过程的定义与调用、单元的概念、面向对象的概念及异常处理；第 2 章主要介绍了在 Delphi 中进行编程所需要的一些基本概念，详细说明了调试 Delphi 应用程序的方法；第 3 章介绍了 Delphi 的 VCL 可视化组件库的概念，组件的共有属性、共有方法和共有事件的功能及使用，以及在应用程序中进行窗体设计和菜单设计的方法；第 4 章介绍了文本编辑框的属性、方法、事件及使用，学习了输入组件的不同使用特点及编程方法，介绍了对话框的使用方法，并通过实例介绍了这些组件的使用；第 5 章介绍了按钮、工具栏及其他常用组件的使用；第 6 章介绍了 Delphi 中进行图形处理的方法包括绘图软件的制作、动画和多媒体设计；第 7～9 章介绍了数据库程序设计的基础知识以及单层数据库和 C/S 结构数据库应用程序的开发过程。

本书的最大特点是实用性，在内容选择上侧重主要功能和关键技术，并提供了相关的实例，书中所有程序都测试通过，在每章之后均配有小结和习题，有利于对所学知识的巩固。本教材参考学时为 72 学时。

本书由金林樵任主编。其中，第 3～9 章由金林樵编写，第 1、2 章和附录由何林编写。

由于时间仓促，书中难免有不足之处，敬请广大读者批评指正。

编　者

目　录

第 1 章　Delphi 的编程语言 Object Pascal

本章主要介绍与 Delphi 程序开发密切相关的编程语言 Object Pascal、编程时所使用的数据类型、Pascal 语法以及函数与过程的定义和调用规则。

1.1　Object Pascal 语言简介

对计算机语言发展史有基本了解的读者都听说过 Pascal 语言，Pascal 语言是由瑞士 Eidgenossische工业大学的 Niklaus Wirth 教授在 1971 年提出的。它是以 ALGOL 60 语言为基础，按照结构化程序设计原则设计出来的一种描述算法的语言，并以历史上著名的数学家 B.Pascal 的名字来命名。

1990 年以前，C 语言尚未普及，Pascal 语言是世界上使用最为广泛的语言，它以丰富的数据类型、严谨的语法规则、良好的结构化特性、高效的编译器、简洁明了的算法描述和自由优美的书写格式而广为流行，当时的《数据结构》教材都是采用类 Pascal 语言作为描述语言。

Pascal 的流行很大程度上依赖于 Borland 公司的拳头产品 Turbo Pascal，自从 1985 年 Turbo Pascal 推出以来，在最初的 6～7 年时间里，Turbo Pascal 以其操作简单、功能强大而占据了当时程序开发语言的半壁江山。此后 Borland 一直在不断地增强它的功能，在其 5.5 版本中引入了一个称之为里程碑的概念——对象(Object)，在 Pascal 语言史上产生了一次质的飞跃。

Delphi 是一个完全面向对象的可视开发平台，它采用面向对象的程序设计语言 Object Pascal 作为基础语言。Object Pascal 就是面向对象的 Pascal，它在传统的 Pascal 语言基础上增加了面向对象的内容、新增了若干数据类型、对编译器作了进一步的优化处理。

1.2　基本语法要素

本节主要描述构成 Pascal 语法最为基础的标识符、保留字和标准标识符的概念。

1.2.1　Pascal 中的标识符

标识符是用来定义程序中使用的常量、变量、自定义数据类型、函数、过程和单元文件等的名称。标识符的取名必须遵循以下规则：

- 必须以字母或下划线开头，其后可以跟零个或多个字母、数字或下划线。
- 标识符中的字符不区分大小写。如 User、USer 与 USER 是同一个标识符。
- 标识符中不允许包含空格。
- 禁止使用 Object Pascal 语言的保留字作为自定义标识符，如 procedure、do、while 等。同时尽量不使用 Object Pascal 的标准标识符，如 Single、Integer 等，否则可能会出现意想不到的后果。
- 标识符的长度一般不限，但只有前 255 个字符有效。

1.2.2 保留字和标准标识符

1. 保留字

保留字在 Object Pascal 语言中具有固定含义，系统已将它们做了特殊定义。因此，不允许读者在使用时再定义。

比如：absolute，Nil，and，not，begin，for，then，Inline，with，end，xor 等等都是系统定义的保留字。

2. 标准标识符

标准标识符也在 Object Pascal 语言中具有特殊含义，用来表示一些常用的命令。但与保留字不同，标准标识符可以被重新定义，一旦将标准标识符定义为自定义标识符，则系统原有的标准标识符将失效，不能使用。下面是 Object Pascal 语言中标准标识符的几个例子。

标准类型：integer，real，char

标准过程：read，write，reset，lineto

标准函数：abs，cos，round，sort，succ

1.2.3 常量与变量

1. 常量

程序运行过程中不能发生变化的值，称之为常量。如：3，－5.0，'pen'等分别表示整型、实型和字符串常量。

在 Pascal 语言中，定义一个常量要使用 const 说明语句。它的语法是：

```
const
    <常量标识符 1> = <常量 1>;
    <常量标识符 2> = <常量 2>;
```

常量的数据类型通过等号“=”右边的数据的类型隐含确定。

例如下列程序段中定义了 3 个常量 a，b 和 c：

```
const
  a = 10.5;                    // 常量 a 为实型
  b = -2;                      // 常量 b 为整型
  c = 'a * b=';                // 常量 c 为字符串
begin
  ShowMessage('a * b=' + FloatToStr(a * b));
end;
```

当 Delphi 编译器编译到 ShowMessage 函数时，就将 a 替换成 10.5，而将 b 替换成－2。程序运行后显示：a * b= －21。但要注意，既然已将 a，b 和 c 定义成了常量，就不能再在该常量的作用域内使用诸如 a:=30；这样的赋值语句，这是因为常量的值是不能动态修改的。

同时 Delphi 为方便读者的使用，预定义了一些常量，如圆周率 PI、布尔型值 True 和 False 等，这些已预定义的常量读者不必定义就可直接使用。

但要在程序中使用其他常量，则必须先定义，后使用。否则将出现语法错误。

2. **变量**

在程序执行中,其值可由程序修改而发生变化的,称为变量。变量实际上是内存中一块命名的存储单元,用于存储在程序运行过程中不断发生变化的值。

Pascal 语言是一个强类型的语言,所有的变量必须先定义后使用,即在使用该变量前必须先定义该变量所使用的数据类型。数据类型可以是标准数据类型,也可以是自定义的数据类型。否则系统就将其认为是没有定义的标识符而在编译时产生语法错误。

Pascal 中变量说明的格式为:

```
var
  <变量 1[,变量 2]…> : <数据类型 1>;
  <变量 n> : <数据类型 n>;
  ⋮
```

一个 Var 可以含有多个不同的变量说明。当有多个变量属于同一数据类型时,可以在各变量名间使用逗号“,”分隔。每说明一个变量,系统就在内存中为其分配相应的存储单元。

下面的程序段是使用变量的例子:

```
var
  R :       Integer;
  Area:     Real;
begin
  R := 56;
  Area := PI * R * R;
end;
```

在该例中,说明了两个数据类型分别为 Integer 和 Real 的变量 R 和 Area。

1.2.4 注释

为了增加程序的可读性,减轻程序的维护工作量,在程序中使用注释是一个良好的习惯。在 Object Pascal 中,注释可以使用“{”和“}”或“(*”和“*)”将注释信息括起来,但必须成对出现。若仅注释一行,则在注释信息的开始处加上“//”即可。对于注释,Delphi 的编译器在编译时将自动忽略掉。

1.3 数据类型

在程序中使用的常量、变量、表达式和函数都必须指定一个数据类型。不同的数据类型,其值在内存中的存储格式是不同的,可以参于的运算也有所不同。

Object Pascal 语言所支持的数据类型见表 1-1 所示。

序数类型定义了有序值的集合,在该集合(即其取值范围)中,每个值都有一个固定的序号、一个前趋值(第一个除外)和一个后继值(最后一个除外)。简单类型中除实型外都是序数类型。Pascal 中预定义了如表 1-2 所示的序数函数或过程。

表 1-1 Object Pascal 支持的数据类型

分　类	数 据 类 型	是否为序数类型
简单类型	整数类型	是
	字符型	是
	布尔型	是
	枚举型	是
	子界型	是
	实型	否
字符串类型	字符串类型	否
结构类型	集合类型(Set)	否
	数组类型(Array)	否
	记录类型(Record)	否
	文件类型(File)	否
	类类型(Class)	否
指针类型	指针类型(Pointer)	否

表 1-2 序数函数或过程

函数或过程名	参　数	说　明
Ord()	序数类型表达式	返回表达式值所对应的序号
Pred()	序数类型表达式	返回表达式值所对应的前趋值
Succ()	序数类型表达式	返回表达式值所对应的后继值
High()	序数类型标识符或变量	返回该序数类型所能表示的最大值
Low()	序数类型标识符或变量	返回该序数类型所能表示的最小值
Inc()	序数类型变量	变量值加 1
Dec()	序数类型变量	变量值减 1

对于整型值来说,序号就是其值本身,前趋值就是比其小 1 的数,后继值就是比其大 1 的数;对于布尔型值来说,False 的序号为 0,而 True 的序号为 1,False 没有前趋值,True 也没有后继值;对于其他类型的序数类型,其第一个值的序号为 0,第二个值的序号为 1,…,依次类推。

1.3.1 简单数据类型

1. 整数类型

Delphi 中的 Object Pascal 语言支持如表 1-3 所示的几种整数类型。

表 1-3 Object Pascal 语言所支持的整数类型

类　型	取值范围	说　明
Integer	-2147483648～2147483647	32Bit 有符号数
Shortint	-128～127	8Bit 有符号数
Smallint	-32768～32767	16Bit 有符号数
Longint	-2147483648～2147483647	32Bit 有符号数
Int64	-2^{63}～$2^{63}-1$	64Bit 有符号数
Byte	0～255	8Bit 无符号数
Word	0～65535	16Bit 无符号数
Longword	0～4294967295	32Bit 无符号数
Cardinal	0～4294967295	32Bit 无符号数

下面的程序段说明了整数类型变量的说明及序数函数的使用：

```
var
  K, Kfun :        Smallint;
  UK, UKfun:       Longword;
begin
  K := -18;
  UK: = 19;
  Kfun := Ord(K);                    // Kfun 的值为 -18
  UKfun: = High(UK);                 // UKfun 的值为 4294967295
  Inc(K);                            // K 的值为 -17
  ShowMessage('UKfun =' + IntToStr(UKfun) + ',K=' + IntToStr(K));
end;
```

此外，在 Pascal 语言中除可正常使用十进制整型常数外，还允许使用十六进制整型常数，其表示方法是在十六进制整型常数前加上“$”。比如，$10 代表十进制数 16，$AB 代表十进制数 171。

2. 字符型

Object Pascal 中的字符型数据共有三种：Char、AnsiChar 和 WideChar。其中 Char 和 AnsiChar基本等价，用来存放一个 ASCⅡ字符；而 WideChar 是广域字符，用来存放 ASCⅡ字符和 Unicode 字符。

可以在一对单引号内包含一个字符来表示一个字符常量，如'B'、'f'、'%'等。因在计算机中是用字符的 ASCⅡ码来表示该字符的，所以在 Object Pascal 中也可以使用“#”后缀一个 ASCⅡ码来表示一个字符，如用#49 表示字符'1'，用#65 表示字符'A'等。由于有这种特点，因此，用这种方法可以很方便地表示一个控制字符，如用#8 表示“退格”，用#13 表示“回车”，在后面要介绍的输入组件中，用它可以很方便地控制用户的输入用键。

函数 Chr()，是 Ord()函数的反函数，用来返回一个 ASCⅡ码值所对应的字符。如 Chr(65)的返回值是字符'A'。

3. 布尔型

布尔型(Boolean)用来表示逻辑量，该类型的取值只有两个：False 和 True，分别表示逻辑“假”和逻辑“真”。布尔型主要用于关系和逻辑(And、or 和 Not)运算，用来控制程序的执行流

程,因此布尔型数据很重要。

4. 枚举型

在 Pascal 语言中,可以使用整型和实型来解决数值问题;可以用布尔型来代表事物的真假。但是,当要表示一个较为抽象的数据时,Pascal 提供的基本数据类型就无能为力了。

为此,Pascal 语言提供了用户自定义数据类型的功能。在 Pascal 语言中,自定义数据类型均用保留字 type 来定义。

枚举类型是一种简单的用户自定义类型,和整型等数据类型一样,也是一种序数类型。

(1) 枚举类型的定义

Pascal 中定义枚举类型的语法格式为:

```
type
  <枚举类型标识符> = (e1, e2, e3, … , en);
```

上述格式中的 e1～en 代表标识符。

下面的例子说明了怎样定义一个星期中的每一天:

```
type
  Week = (SUN, MON, TUE, WED, THU, FRI, SAT);
```

有了上面的类型定义后,就可以在变量定义中使用它作为新的数据类型来定义变量了。如:

```
var
  Day: Week;
```

枚举类型的功能是基本数据类型所不能代替的。而且使用枚举类型还可以节省大量的内存;当定义的枚举类型中元素的个数在 256 个以内时,该枚举类型变量在内存中仅占 1B。但如果定义 1 个 Shortint 变量,则要占用 1B,用 Shortint 变量来描述一周 7 天的话,要定义 7 个 Shortint 变量,就要占用内存 7B。

(2) 枚举类型的运算

枚举类型可以进行赋值和关系两种运算。

1) 赋值运算

假设有如下的类型定义和变量说明:

```
type
  MyObject = (book, computer, TV, printer, table);
var
  M1, M2: MyObject;
```

则以下赋值语句是合法的:

```
M1 := TV;
M2 := book;
```

而以下赋值语句则是非法的:

```
M1 := black;
```

```
M2 := desk;
```

由此可见,枚举类型的变量在进行赋值运算时,不能赋以类型定义以外的值。否则,将出现错误。

2) 关系运算

Pascal 语言规定,枚举类型元素从左至右各有一个从 0 开始的整数序号。如在上面举过的 MyObject 的类型中,book 的序号为 0,Computer 的序号为 1,而 table 的序号为 4。

因此,枚举类型可以进行六种关系运算。在上面的例子中 M1 和 M2 进行比较运算完全合法,且 Ml > M2。

5. 子界型

有时,程序中不希望用到某种有序类型的全集,只是使用其中的若干数据值,如 Char 的数据范围从 #0～#255,而用户可能只想使用其中的'A'到'Z',在这种情况下就需要使用子界型。子界型,顾名思义就是某种有序类型的子集,即限制数据集的边界。

(1) 子界类型的定义

Pascal 中定义子界类型的语法格式为:

```
type
  子界类型标识符 = e1..e2;
```

上述格式中,e1,e2 代表子界类型的下界和上界,可以是一个常量或常量表达式。

使用子界类型时,要注意以下几点:

- 子界类型的上、下界必须是同一序数类型,该序数类型称为子界型的宿主类型;
- 定义子界类型前,其宿主类型必须事先已定义;
- 子界类型的下界不能大于上界。

例如可用子界类型来表示人的年龄,其定义方法为:

```
type
  Mage = 0..120;
```

但下面的定义不符合语法规定:

```
type
  Pe = 1340..90;          // 下界大于上界
  Ak = 10..'A';           // 下界与上界不是同一数据类型
  Rt = 9.375..177.6;      // 下界 9.375 与上界 177.6 是实数,不是序数类型
```

(2) 子界类型的运算

子界类型的运算与其宿主类型的运算完全相同,只不过其运算范围受到了上、下界的限制而已。如有以下的定义和说明:

```
type
  ID = 1..16;
  StuNo = 1..50;
  NameChar = 'A'..'Z';
var
```

```
i: ID;
st: StuNo;
Fn: NameChar;
```

则下列语句是合法的：

```
i := 9;
st := i + 20;
Fn := 'B';
```

而下列语句却是非法的：

```
i := 19;          // 超出了 1..16 的范围
Fn := 'a';        // 超出了'A'..'Z'的范围,因为'Z'的 ASCⅡ码值<'a' 的 ASCⅡ码值
```

6. 实数类型

用前面讲述过的整型、字符型、布尔型、枚举型和子界型等数据类型可以表示整数、字符、逻辑量和枚举量，但却无法用来存储一个实型数。如将一个 1088.85 的值，用整数类型来表示时，就变成 1088 了，这在需要使用实型数的场合（如财务软件），是不行的，所以 Object Pascal 提供了实数类型。

Delphi 的 Object Pascal 支持如表 1-4 所示的几种实数类型。

表 1-4 Object Pascal 语言所支持的实数类型

类 型	取值范围	有效位	占用字节数
Real	$5.0\times10^{-324}\sim1.7\times10^{308}$	15～16	8
Real48	$2.9\times10^{-39}\sim1.7\times10^{38}$	11～12	6
Single	$1.5\times10^{-45}\sim3.4\times10^{38}$	7～8	4
Double	$5.0\times10^{-324}\sim1.7\times10^{308}$	15～16	8
Extended	$3.6\times10^{-4951}\sim1.1\times10^{4932}$	19～20	10
Comp	$-2^{63}+1\sim2^{63}-1$	19～20	8
Currency	-922337203685477.5808～922337203685477.5807	19～20	8

1.3.2 字符串类型

前面描述的字符型数据只能表示单个字符。但在程序设计中，需要表示由一串字符组成的数据，如表示姓名、家庭地址等，要表示这种类型的数据，就需要使用字符串类型。

Delphi 的 Object Pascal 支持如表 1-5 所示的四种字符串类型。

表 1-5 Object Pascal 语言所支持的字符串类型

类 型	容 纳	是否以空字符结束	编译开关	最大长度
ShortString	Char	否		255
String	Char	否/是	{$H-}/{$H+}	$255/2^{31}$
AnsiString	Char	是		2^{31}（约 2GB）
WideString	WideChar	是		2^{30}（约 1GB）