编程实例教程系列

Delphi 程序员经验点滴

桌面、网络编程实例集锦

王小华 编著

兵器工业出版社

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

编程实例教程系列

Delphi 程序员经验点滴

桌面、网络编程实例集锦

王小华 编著

兵器工业出版社
北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

内 容 简 介

本书从应用角度出发，积累了一个程序员多年工作的经验，从一个个我们经常遇到的小问题着手，讲述了Delphi程序设计中的经验与技巧，并给出了大量源程序代码，以及对代码的详细分析。不仅讲述了有些程序的功能，还从多个角度分析了实现该功能的方法，使读者能更透彻、全面地理解和运用Delphi。

本书包含的内容比较丰富，通过256个实用案例覆盖了Win32 API应用、组件的应用与改造、应用程序及系统、文件处理、图形图像、数据库、打印功能扩展、剪贴板应用、多媒体技术、网络等，适合于学习Delphi编程的广大程序爱好者，特别是对于一些想快速获得编程技巧与经验的读者，更具参考价值。

所配光盘为本书实例工程文件的源代码。

图书在版编目（CIP）数据

Delphi 程序员经验点滴：桌面、网络编程实例集锦 / 王小华编著；—北京：兵器工业出版社；北京希望电子出版社，2006.1

ISBN 7-80172-535-2

Ⅰ.D... Ⅱ.王... Ⅲ.软件工具—程序设计 Ⅳ.TP311.56

中国版本图书馆 CIP 数据核字（2005）第 099172 号

出　　版：兵器工业出版社 北京希望电子出版社

邮编社址：100089 北京市海淀区车道沟 10 号

100085 北京市海淀区上地信息产业基地 3 街 9 号 金隅嘉华大厦 C 座 610

发　　行：北京希望电子出版社

电　　话：（010）62520290（发行）（010）62532258（门市）

经　　销：各地新华书店 软件连锁店

印　　刷：北京媛明印刷厂

版　　次：2006 年 1 月第 1 版第 1 次印刷

封面设计：梁运丽

责任编辑：郭春临 宋丽华 但明天

责任校对：孙 红

开　　本：787×1092 1/16

印　　张：23.75

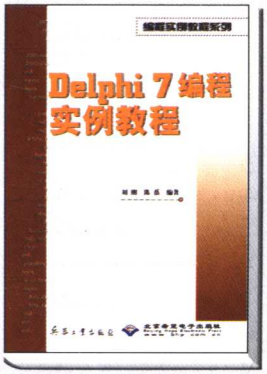
印　　数：1-4000

字　　数：532 千字

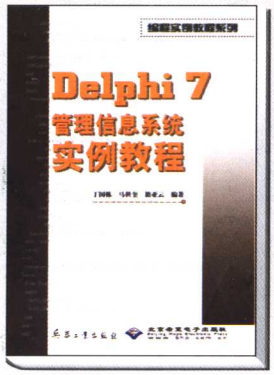
定　　价：35.00 元（配 1 张光盘）

希望 Beijing Hope Electronic Press www.bhp.com.cn

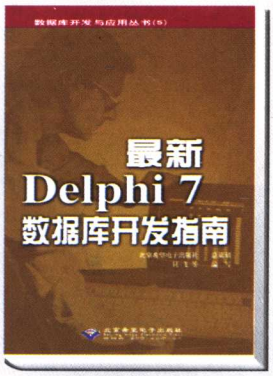
! Hotbook 好书推荐



CX-4615
定价：35.00元

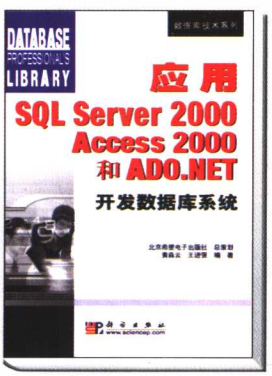


CX-4494
定价：35.00元

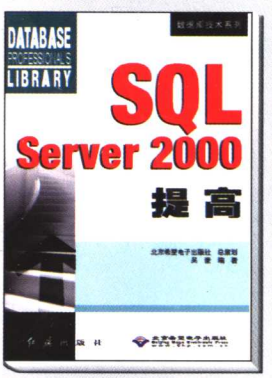


CX-4015
定价：46.00元

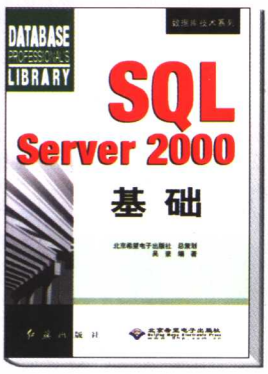
CX-4851
定价：46.00元



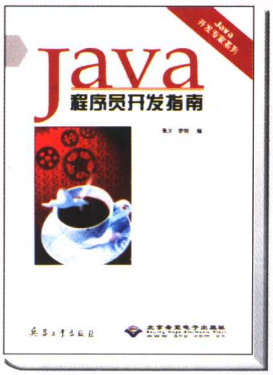
CX-4858
定价：20.00元



CX-4717
定价：52.00元



CX-4719
定价：46.00元



CX-4602
定价：45.00元

CX-4914
定价：48.00元

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

社址：北京市海淀区上地信息产业基地3街9号金隅嘉华大厦C座611
电话：（010）62978181（总机）通信：北京市清河6号信箱（100085）

前　言

Delphi 是一个优秀的编程工具，为什么有那么多程序员选择它呢？因为它有非常方便的集成开发环境，高效的数据处理能力和优化的程序代码。它集编程、调试、运行于一体，使初学者一看就懂，一用就明白，节省了入门时间与程序的生成时间。

本书从应用的角度出发，以一个个小问题着手，讲述各个功能的实现方法，以及实现这些功能所应注意的问题，哪些是我们应该避免和克服的，哪些是我们应该发挥和继承的。

全书共分为 16 章，共 256 个实用案例。第 1 章讲述 Pascal 语句的基本功，主要是帮助初学者加深对 Pascal 语言的理解。第 2 章讲述控件的应用与改造，选取了一些我们常用的控件，重点介绍它们的用法，以及改造控件的方法。第 3 章讲述了一些关于程序与窗口方面的应用知识。第 4 章介绍了与系统有关的常用功能实现方法。第 5 章讲述了设计安装程序与卸载程序的方法。第 6 章讲述了如何实现剪贴板的读写功能。第 7 章讲述了文件读写、目录增减等方法。第 8 章讲述了画布、图像显示、图像变幻、艺术显示、动画实现方法、位图的 CMYK 分色技术等。第 9 章讲述了如何编写自己的打印程序，以及如何进行各种打印设置。第 10 章讲述了动态链接库 DLL 的编写与调用方法。第 11 章讲述了如何在程序中读写注册表的方法。第 12 章讲述了内存与字符串常用的功能。第 13 章介绍了 OLE 与 DDE，以及程序中如何实现与 Word、Excel 的接口。第 14 章讲述了一些数据库的常用处理方法：数据源拷贝、记录的增删改、字段查询、表格控件的应用等。第 15 章讲述了与多媒体有关的知识。第 16 章介绍了网络应用方面的知识。

由于篇幅所限，不可能穷尽所有的应用方法，只是根据本人的编程实践，抽取出了一些比较有代表性的具体实例，结合自己多年的编程经验，试图尽量将它们解释清楚，希望给读者一些编程的灵感和启发。

Hotbook 好书推荐

CX-4463
定价：68.00元

CX-4502
定价：35.00元

CX-4578
定价：33.00元

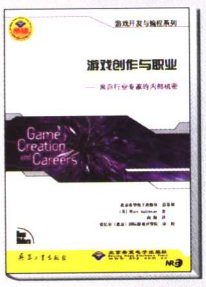
CX-4608
定价：42.00元

CX-4611
定价：48.00元

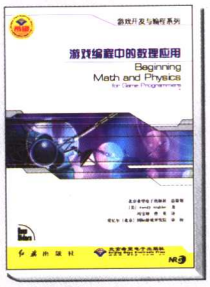
CX-4638
定价：58.00元

CX-4653
定价：48.00元

CX-4654
定价：58.00元

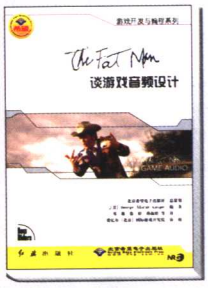


CX-4673
定价：58.00元



CX-4698
定价：45.00元

CX-4771
定价：52.00元



CX-4775
定价：25.00元

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

社址：北京市海淀区上地信息产业基地3街9号金隅嘉华大厦C座611
电话：(010) 62978181（总机）通信：北京市清河6号信箱(100085)

目　录

第 1 章　Pascal 语言基本功

实例 1　容易理解但不一定会灵活运用的常量、变量、表达式

所谓常量，即在程序运行过程中，以一个固定的值表现的量。在 Pascal 语言中，用关键字 const 声明。对于常量，在使用时，注意下面两点。

（1）常量在声明时就要赋值。

```
const  abc=1234;
```

（2）常量的属性是只读的，在使用过程中，不能再被赋值。

上述在定义 abc 时已被赋值为 1234，如果在程序中再用语句 abc=22; 系统在编译时将不被通过。

定义常量时，可以带数据类型，也可以不带数据类型。下面两种定义方法都是正确的。

（1）带数据类型的常量定义法。

```
const
  PI : double = 3.1415;
  ii : Integer = 100;
  Note : string = '注意安全';
```

（2）不带数据类型的常量定义法。

```
const
  PI = 3.1415;
  ii  = 100;
  Note = '注意安全';
```

常量在什么时候能用到呢？举一个简单的例子：在几何运算中，常常遇到一个圆周率，它的值是 3.1415926。在运算时，可能需要多次调用这个值，而且这个值的小数位特别长，稍不留意就会写错。为了不致于每次都写这个值，常量的作用就发挥出来了。我们用一个常量定义 const　PI=3.1415926;以后，程序中凡有圆周率的地方，都可以用 PI 代替。

变量则与常量相反，它是一个变化的量。在程序设计中，可以根据实际情况，随时赋予它不同的值。变量由数据类型声明，定义一个整型变量，声明格式为：

```
aa : Integer;
```

定义一个双精度型的变量，声明格式为：

```
aa : double;
```

定义一个字符串型的变量，声明格式为：

```
aa : string;
```

全局变量可以在声明的同时赋值，例如：

```
aa : Integer=123;
```

局部变量不可以在声明时赋值。

变量只能在函数或过程头部声明，不可以在程序主体的 begin...end 之间声明。

根据声明的位置不同，变量可以分为全局变量与局部变量。

所谓全局变量，有一个简单的区分方法是，声明在函数之外的为全局变量，该变量可应用于每一个函数。声明于函数内部的为局部变量，该变量仅限于本函数使用。

我们最常犯的错误是将全局变量与局部变量取同一个名称。在程序编译时，不会发生任何错误，但是在程序调试时，却会出现莫名其妙的错误。如果你没有意识到同名的问题，无论怎么寻找，也找不出程序的错误。

谁都不能保证自己曾经声明过的变量会永远记得，那就想一个办法克服记忆的不足吧！在实际运用当中，我们常用的方法是，凡是全局变量，在变量名前加上标识符 G_，如声明某一串全局变量如下：

```
G_aa1,G_aa2,G_Width : Integer;
```

对于局部变量的声明，只能在函数或过程的头部，begin 之前，var 说明符之后，例如：

```
procedure Function1();
var
  aa : Integer;
  bb : Integer;
  cc : Integer;
begin
  aa:=123;
  bb:=234;
  cc:=aa*bb;
end;
```

对于相同的变量，可以放在一起声明，中间用逗号分开，例如：

```
procedure Function1();
var
  aa ,bb ,cc : Integer;
begin
  aa:=123;
  bb:=234;
  cc:=aa*bb;
end;
```

很多初学者往往不太喜欢使用变量，尤其不喜欢将变量写得太长，嫌麻烦。如有些人总是喜欢用 a,b,c,i,j,aa,bb,cc,ii,jj 等简单的字母表示。当然，在编写一些小程序时，本身用不了几个变量，很容易将变量的意思分清，用简单字母表示也未尝不可。要是大程序呢？上千个变量甚至上万个变量呢？你还能保证永远记住 i 与 j 这些变量是代表什么吗？

不要贪图简单而给程序检查时设置障碍。要知道，没有一次编写成功的程序，一个程序往往要经过无数次的修改、补充才能达到要求。如果你设置的变量不能清楚地表达意思，每一次检查与修改时，都要重新搞清楚各个变量的意思，想一想该有多麻烦？

在设置变量名，特别是全局变量名时，要尽可能地将变量的意思体现出来，不要怕名字太长。例如要定义表的宽度变量，如果你英语还可以，就用英文 Table_Width 表达，如果你汉语拼音不错，就用 Biao_Kuan 表达，非常一目了然。如果你用 a 表达，三天以后你还记得它表达的意思吗？

如果变量所要表达的意思很多，你尽量多用_符号将不同的意思连结起来，便于自己理解，也便于别人能看懂你的程序。

还要注意一点，Pascal 语言不区分字母大小写，这一点不同于 C 语言，与 Basic 语言相似。在 Pascal 中，变量 MYVAL 与 myval 是一样的。那么是否可以随便地改变变量的大小写呢？这样也不可取。在一段程序中，最好保持变量的写法一致，以便在检查程序时不至于眼晕而

错看。

表达式是常量、变量、运算符的组合。在实际编程运用中，根据经验，介绍几点应该注意的方面。

（1）表达式不宜写得太长，对于运算关系复杂、运算符多、括号多的表达式，应该分开来写，不要图省事，尽量将一行拆开为多行来写。如某表达式：

```
aa:=bb*(1+b)*(2+c+3*(3+g));
```

可以写为：

```
aa:=bb*(1+b);
aa:=aa*(2+c+3*(3+g));
```

（2）对于有浮点运算的表达式，要注意采用双精度型，即用 double 声明的类型。类型不同的变量进行运算，要注意类型转换。

实例2 整数、浮点、双精度，使用起来要心中有数

在数学运算中，整数、浮点、双精度是我们常用到的，概念我就不说了，谈几点经验。

（1）要尽可能多运用整数进行运算，因为它运行的速度快。现在的计算机发展了，也许你感觉不出整数与浮点数之间运行速度的差别，但在处理大数据量的运算或大型数据库时，区别就很明显了。有时为了提高运算速度，有经验的程序员甚至将浮点数转化为整数运算。如某浮点数为 12.34，我们将它乘以 100，变为整数 1234，然后再参与运算。

（2）需要用到浮点运算时，少用 float 类型，多用 double 型。用 double 类型的变量参加运算后，有时候会出现一些小小的问题，如在小数位后几位多出一些数字，虽然四舍五入后不影响数据，但如果让这个多出来的数继续参与运算，累计的结果足以让你不知所措。

实例3 逻辑操作，很少用却很有用

所谓逻辑，即与、或关系，含有与、或关系运算符的表达式称为逻辑表达式。在 Pascal 语言里，与用 and 代替，或用 or 代替。

我们编写程序，大不了处理这样几类问题：表达式、判断、分支选择。判断有简单判断与复杂判断，在复杂判断中，少不了用到各种逻辑关系。比如有这样一个判断：筛选出年龄在 18~20 岁、长头发、身高 160 以上、体重小于 60 公斤的女孩。这么多关系，如果你用多重判断的方法，要多麻烦有多麻烦，例如：

```
if(Age>=18) then
begin
  if(Age<=20) then
  begin
    if(Hair) then
    begin
      if(Height>=160) then
      begin
        if(Weight<=60) then
        begin
          if(Type='Girl') then
          begin
           //表达式
          end;
```

```
        end;
      end;
    end;
  end;
end;
```

运用逻辑表达式，只有如下一条语句即可：

```
if((Age>=18) and (Age<=20) and (Hair=True) and (Height>=160) and (Weight<=60) and  (Type='Girl'))
begin
  //表达式
end;
```

在逻辑运算中，遇到与、或关系都存在的情况，该怎么办？

（1）头脑冷静，分清关系。

（2）用括号将每一个完整的关系括起来。不要吝啬括号，要大胆使用，多用一对括号没关系，少用一对括号可能就造成大错误。

同样是上面的例子，其中有一句“年龄在 18~20 岁”。这就是一个完整意义表达的句子，我们可以用括号将这两个条件括起来：

```
  if(  ((Age>=18) and (Age<=20))  and (Hair=True) and ( Height >= 160 ) and ( Weight <= 60 ) and
( Type='Girl'))
  begin
    //表达式
  end;
```

虽然增加了一对括号，丝毫不影响程序的结果。

只要关系清楚，层次分明，括号到位，逻辑运算关系就不会混乱。

举一个简单的例子，寻找 16~18 岁的男孩或女孩。逻辑表达式为：

```
if( ( (Age>=16) and (Age<=18) ) and ( (Type='Girl') or ( Type='Boy') ) )
```

我们可以将关系分解为两大部分：表示年龄关系的，表示类型关系的。这两大部分之间也是与关系。年龄关系中，大于等于 16 与小于等于 18 也是与关系，包含在年龄关系的括号里。类型关系中，男孩与女孩是或关系，要么表示男孩，要么表示女孩，包含在类型关系的括号里。如果你不注意分段，不注意括号之间的包含关系，写成如下的形式：

```
if((Age>=16) and (Age<=18)and (Type='Girl') or (Type='Boy'))
```

那是两种完全不同的结果。

实例 4　过程与函数

过程与函数都是自己写的一段程序代码，这段程序代码要完成指定的功能，也称之谓模块。在 C 语言中，只有函数概念，没有过程概念。C 语言中的函数，分为有返回值的与无返回值的；而在 Pascal 语言中，有返回值的称为函数，无返回值的称为过程。函数用 function 说明，过程由 procedure 说明。函数的返回值赋给 Result，它是无类型的，不管是整数、浮点数、字符串，都可以赋值于 Result。

过程示例：

```
procedure MyProc(ii:Integer);
var
  nn : Integer;
begin
  nn:=ii*12;
```

```
end;
```

函数示例：

```
function  MyFunc(aa,bb:Integer) : Integer;
var
  nn : Integer;
begin
  nn:=aa*bb;
  Result:=nn;
end;
```

函数、过程的调用与C语言一样，比如：

```
MyProc(22);              //过程调用
aa:=MyFunc(55,66);//函数调用
```

实例5 分支语句case

无论你编写大程序还是小程序，总少不了要用到分支语句，常用的分支处理语句case，经典的写法如下：

```
case 变量值 of
  值1 : 分支处理语句1;
  值2 : 分支处理语句2;
  值3 : begin
        多行分支处理语句;
        end
  ......
  else
    缺省分支处理语句N;
    break;
end;
```

其中的值一定是有序类型的数，比如整数。如果是字符串类型，就不要用这种语句，编译时肯定会出错。

这个语法中，常规情况下，每一个值对应一个功能，最后的else语句后面，表示缺省情况下的处理。如果你不加缺省处理，也不是不可以，但不规范。

多个值对应同一功能，语句写法如下：

```
case  VV  of
  22,33,44 : 分支处理语句;//多个值对应同一处理语句
  ......
  else
    缺省分支处理语句N;
    break;
end;
```

连续值对应同一功能，语句写法如下：

```
case  VV  of
  100..200 : 分支处理语句;//100~200连续值对应同一处理语句
  ...
  else
    缺省分支处理语句N;
    break;
```

```
end;
```

实例 6　条件分支判断语句 if

该语句就是一个条件判断语句，在写程序时，少不了要与它打交道。它的出现形式，无非就两种情况。

第一种情况：

```
if 条件表达式 then
begin
  处理语句;
end
else
begin
  处理语句;
end;
```

第二种情况：

```
if 条件表达式 1 then
begin
  处理语句;
end
else if 条件表达式 2 then
begin
  处理语句;
end
else if 条件表达式 3 then
begin
  处理语句;
end
......
else
begin
  处理语句;
end;
```

第一种情况就是处理非此及彼的判断，第二种情况是处理多种条件关系的判断。仔细看一看，它就是一个分支处理语句，这种分支比 case 语句更灵活，自由度更大，表达式中没有类型限制。但是用它作分支在处理速度上，恐怕没有 case 来得快。

在设计程序时，对字符串的分支处理往往是最多的，由于 case 分支处理语句对值有类型限制，因此在无法用 case 语句进行分支处理的情况下，都可以用 if 语句作为分支处理语句。比如最常用的字符串分支处理，往往就采用 if 语句。

```
procedure test(Name : string);
begin
  if  Name='张三'  then
  begin
    ShowMessage('老张');
  end
  else if Name='李四' then
  begin
```

```
    ShowMessage('老张');
  end
  else
  begin
    ShowMessage('其他');
  end;
end;
```

实例7 for循环

for循环是最常用的循环，它适用于重复次数比较确定的情况，主要有两种典型的用法。

```
for ... to do法:
var ii:Integer;
for  ii:=1 to  100  do
begin
  //处理程序
end;
for ... downto do法:
var ii:Integer;
for  ii:=100  downto  1  do
begin
  //处理程序
end;
```

实例8 条件循环用while

条件循环while的标准写法为：

```
while  条件关系 do
begin
  //处理程序
end;
```

当条件关系满足时就循环，不满足时就停止循环。但也有条件关系满足不了的情况，在这种情况下，该循环就成了死循环。死循环是程序设计中的一大忌，一定要在程序代码中多加判断条件，让程序跳出循环。

在实际应用程序设计中，死循环的程序设计方法有时却是故意采用的，比如：

```
var ii:Integer;
ii:=0;
while ii=0 do
begin
  处理语句;
end;
```

这可是个死循环程序，如果你不在循环的处理程序中加上中断语句，这个程序就将永无休止地循环下去了。

你也可能说，我不用这种循环，不就避免了吗？但这种循环往往是很有用的，有时候在想不出别的办法时，只有它才能帮你解决问题，为什么不用呢？

比如说，我们在不知道任何条件的情况下查找一件东西，找到后就退出循环。很自然，我们就联想到了“死循环”程序。但是，死循环也不是那么好用的，万一条件不考虑周到，