

```
procedure TForm1.SplitColor(SCv: TCanvas;
var
```

编程实例教程系列

```
  ii,jj: Integer;
  rgb0,rgb1,Rv,Gv,Bv,Cv,Mv,Yv,Kv: Integer;
```

```
begin
  for ii:=0 to height-1 do
```

```
  begin
    for jj:=0 to width-1 do
```

```
    begin
      rgb0:=S_Cv.M[ii][jj];
      Rv:=GetR(rgb0);
      Gv:=GetG(rgb0);
      Bv:=GetB(Value(rgb0);
```

Delphi

```
      Cv:=Rv;
      Mv:=Gv;
      Yv:=Bv;
```

```
      if ((Rv=Gv) and (Cv=Mv)) then
        begin
```

```
          Kv:=Gv;Cv:=155;Mv:=55;Yv:=255;
        end
```

```
      else Kv:=255;
```

```
      if (nn=0) then rgb1:=Cv
```

```
      else if (nn=1) then rgb1:=Mv
```

```
      else if (nn=2) then rgb1:=Yv
```

```
      else rgb1:=Kv;
```

程序员经验点滴 桌面、网络编程实例集锦

王小华 编著

```
  D_Cv.Pixels[ii, jj] := TColor(RGB(Byte(rgb1), Byte(rgb1), Byte(rgb1)));
```

```
end;
```

```
end;
```

```
end;
```

```
procedure TForm1.CanvasCopy(SCv: TCanvas; x1, y1, w, h: Integer; DCv: TCanvas;
var
```

```
  cf1: TColor;
```

```
  ii,jj,ww,hh: Integer;
```

```
begin
```

```
  ww:=w-x1;hh:=h-y1;
```

```
  for ii:=0 to ww-1 do
```

```
  begin
```

```
    for jj:=0 to hh-1 do
```

```
    begin
```

```
      cf1:=DCv.Pixels[x2+ii,y2+jj];
```

```
      if (cf1<>cfWhite) then //
```

```
        SCv.Pixels[x1+ii,y1+jj]:=cf1;
```

```
    end;
```

```
  end;
```

```
end;
```

兵器工业出版社



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

Delphi

程序员经验点滴 桌面、网络编程实例集锦

王小华 编著

procedure TForm1.SplitColor(S_Cv: TCanvas;

var

ii,jj: Integer;

rgbo,rgb1,Rv,Gv,Bv,Cv,Mv,Nv,Kv: Integer;

begin

for ii:=0 to height-1 do

begin

for jj:=0 to width-1 do

begin

rgbo:=S_Cv.Pixels[ii,jj];

Rv:=GetR(rgbo);

Gv:=GetG(rgbo);

Bv:=GetB(rgbo);

Cv:=Rv;

Mv:=Gv;

Nv:=Bv;

if((Rv=Gv) and (Rv=Nv)) then

begin

Kv:=Gv;Cv:=Cv*(Mv+25);Nv:=25;

end

else (Kv:=255;

if (nn=0) then rgb1:=Cv

else if (nn=1) then rgb1:=Mv

else if (nn=2) then rgb1:=Nv

else rgb1:=Kv;

D_Cv.Pixels[ii,jj] := TColor(RGB(Byte(rgb1),Byte(rgb1),Byte(rgb1)));

end;

end;

end;

procedure TForm1.CanvasCopy(SCv: TCanvas; x1,y1,x2,y2: Integer; DCv: TCanvas;

var

cl1: TColor;

ii,jj,ww,hh: Integer;

begin

ww:=x2-x1;hh:=y2-y1;

for ii:=0 to ww-1 do

begin

for jj:=0 to hh-1 do

begin

cl1:=DCv.Pixels[x1+ii,y1+jj];

if (cl1<>clWhite) then //

SCv.Pixels[x1+ii,y1+jj]:=cl1;

end;

end;

end;

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方法的方法, 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; 一北京: 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

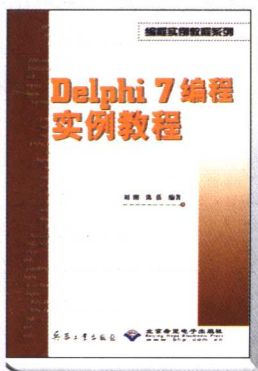
印 数: 1-4000

字 数: 532 千字

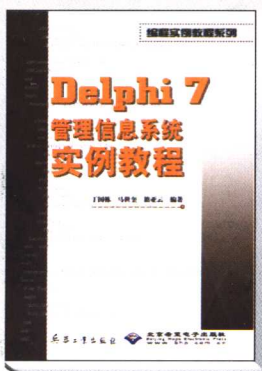
定 价: 35.00 元 (配 1 张光盘)

(版权所有 翻印必究 印装有误 负责调换)

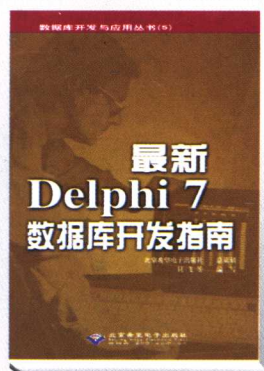
! Hotbook 好书推荐



CX-4615
定价: 35.00元



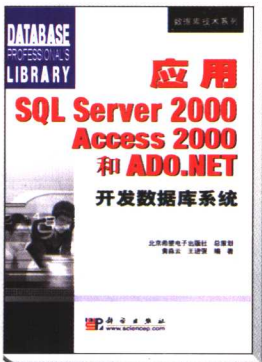
CX-4494
定价: 35.00元



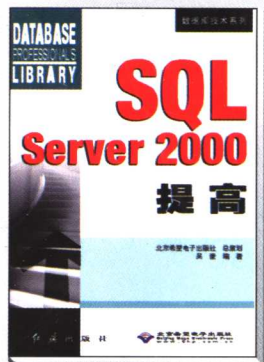
CX-4015
定价: 46.00元



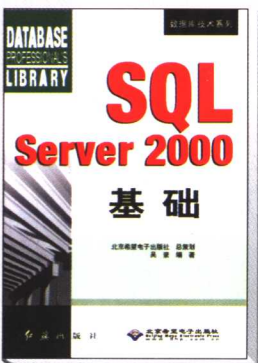
CX-4851
定价: 46.00元



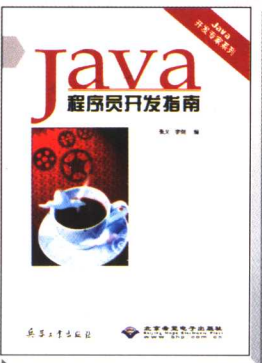
CX-4858
定价: 20.00元



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。

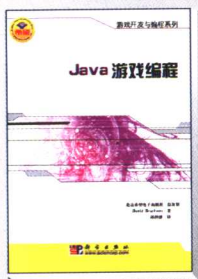
本书从应用的角度出发，以一个个小问题着手，讲述各个功能的实现方法，以及实现这些功能所应注意的问题，哪些是我们应该避免和克服的，哪些是我们应该发挥和继承的。

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，重点介绍它们的使用法，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



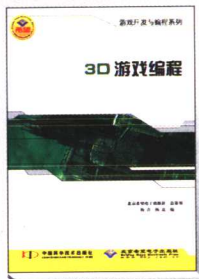
CX-4463
定价: 68.00 元



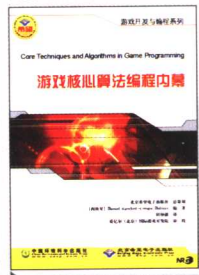
CX-4611
定价: 48.00 元



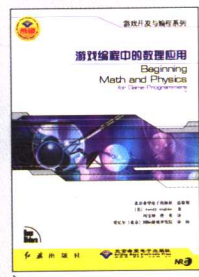
CX-4673
定价: 58.00 元



CX-4502
定价: 35.00 元



CX-4638
定价: 58.00 元



CX-4698
定价: 45.00 元



CX-4578
定价: 33.00 元



CX-4653
定价: 48.00 元



CX-4771
定价: 52.00 元



CX-4608
定价: 42.00 元



CX-4654
定价: 58.00 元



CX-4775
定价: 25.00 元

目 录

第 1 章 Pascal 语言基本功	1	实例 13 SpeedButton 按钮的 Flat 属性	25
实例 1 容易理解但不一定会灵活运用的 常量、变量、表达式	1	实例 14 StringGrid 单元格赋值	25
实例 2 整数、浮点、双精度, 使用 起来要心中有数	3	实例 15 StringGrid 单元格字体及背景 颜色设置	26
实例 3 逻辑操作, 很少用却很有用	3	实例 16 StringGrid 中文字的换行与 对齐方式	27
实例 4 过程与函数	4	实例 17 改造 Edit 只允许输入数字	29
实例 5 分支语句 case	5	实例 18 用 Edit 与 CSpinButton 组合制作 一个数量单位输入框	30
实例 6 条件分支判断语句 if	6	实例 19 在程序中动态创建 Edit	32
实例 7 for 循环	7	实例 20 去掉 Edit 进入时自动选择 文本功能	33
实例 8 条件循环用 while	7	实例 21 用代码设置选择文本	33
实例 9 repeat...until 循环	8	实例 22 在 Edit 中按回车键跳到 下一个控件	33
实例 10 continue, 让循环不做工作 自己走	8	实例 23 根据实际数量动态生成 Button	34
实例 11 break, 让循环刹车	9	实例 24 创建一个带图形的 CheckBox	37
实例 12 指针指向何方, 前途各不一样	10	实例 25 创建一个带图形的 RadioButton	38
实例 13 new 与 dispose	11	实例 26 获取与设定 Memo 中光标的 位置	39
实例 14 记录与集合	12	实例 27 控制 Memo 中文字的滚动	40
实例 15 class, 想说爱你不容易	14	实例 28 替换 Memo 控件中默认的菜单	40
实例 16 数据类型之间的转换	15	实例 29 将 Memo 中的内容保存为文本 文件	40
第 2 章 控件的应用与改造	17	实例 30 让 ListBox 显示图像列表	44
实例 1 无所不在的 MessageBox 对话框	17	实例 31 为 ListBox 添加水平滚动条	45
实例 2 随手拈来 ShowMessage	18	实例 32 为 ListBox 添加图案背景	47
实例 3 简便易用的 InputQuery 输入框	19	实例 33 在两个 ListBox 之间交换数据	47
实例 4 Label 控件显示多行文本	19	实例 34 改变 ListBox 列表内容的顺序	48
实例 5 Label 控件显示超长文本时 自动换行	19	实例 35 在 ListBox 中寻找字符串	49
实例 6 如何使 Label 控件上的文字 竖着显示	20	实例 36 让 BitBtn 显示多行文本	50
实例 7 鼠标指向 Label 控件时改变 说明文字与颜色	21	实例 37 动态地为 BitBtn 添加位图	50
实例 8 Edit 作为密码输入框	23	实例 38 获取 BitBtn 中位图的颜色值	52
实例 9 Edit 内可不可以显示图像	23	实例 39 为 GroupBox 加上图形背景	53
实例 10 去掉 Edit 控件回车后的提示音	23	实例 40 为 RadioGroup 加上图形背景	55
实例 11 静态/动态为 SpeedButton 添加 位图	24	实例 41 为 Combox 增加自动搜索功能	55
实例 12 让 SpeedButton 显示多行文本	25	实例 42 利用 ProgressBar 显示程序	

运行中的进度	56	实例 25 制作透明窗口	84
实例 43 利用 Gauge 显示程序运行进度	58	实例 26 程序执行后自己删除自己	85
实例 44 利用 ScrollBox 实现应用程序 在屏幕上滚动	59	实例 27 枚举 Windows 系统中的字体	87
实例 45 动态设定 ScrollBox 的水平 与垂直滚动条	60	实例 28 巧用控件中的 Tag 属性	87
实例 46 为 ScrollBox 画上图案背景	61	实例 29 巧妙利用文本文件保存应用程序 的菜单内容	89
实例 47 为 TreeView 添加项目	61	实例 30 制作不规则形状的窗体	93
实例 48 为 TreeView 添加图形	63	实例 31 将自己的程序作为屏保	94
实例 49 获取 TreeView 中的项目值	65	实例 32 多线程技术	99
实例 50 对 TreeView 中的项目排序	65	实例 33 创建特色窗口标题条	100
第 3 章 应用程序与窗口	66	第 4 章 系统	102
实例 1 自定义应用程序的图标	66	实例 1 处理 Windows 应用程序的 命令行参数	102
实例 2 改变应用程序的光标	66	实例 2 把文件的 DOS 日期转换为 TDate	103
实例 3 让应用程序脱离支持文件而 独立运行	67	实例 3 格式化软盘	103
实例 4 窗口的初始化	68	实例 4 利用软盘序列号的加密方法	104
实例 5 闪烁窗体的标题栏	69	实例 5 不让应用程序显示在任务栏上	107
实例 6 让窗体的标题栏文字一个接一个 显示	69	实例 6 利用 WM_SYSCOMMAND 消息 启动屏保	108
实例 7 窗体定时关闭的小方法	70	实例 7 获取 Windows 的系统目录	109
实例 8 关闭多余窗口以减少内存开销	72	实例 8 制作托盘图标	112
实例 9 建立 MDI 子窗口	73	实例 9 执行控制面板上的程序	114
实例 10 关闭 MDI 子窗口	74	实例 10 修改系统日期	116
实例 11 获取 MDI 所有子窗口	74	实例 11 获取 Windows 系统信息	117
实例 12 打开一个新 MDI 子窗口时 关闭其他 MDI 子窗口	75	实例 12 调整显示分辨率	118
实例 13 为应用程序制作一个活动图标	75	实例 13 检测磁盘的容量	119
实例 14 隐藏桌面上的图标	76	实例 14 检测磁盘的剩余空间	120
实例 15 隐藏 Windows 开始菜单	77	实例 15 关闭 Windows 系统	120
实例 16 夺取程序的控件权	78	实例 16 删除文件至回收站	121
实例 17 用按钮关闭模式窗口	78	实例 17 更改 Windows 桌面	122
实例 18 利用程序代码关闭模式窗口	79	第 5 章 应用程序的安装与卸载	125
实例 19 让窗口永远显示在最前面	79	实例 1 设计自己的安装程序	125
实例 20 避免应用程序二次运行	79	实例 2 软件序列号设置的方法	126
实例 21 怎样关闭别的应用程序	80	实例 3 将应用程序放在桌面上	127
实例 22 模拟键盘输入	82	实例 4 卸载程序如何实现	129
实例 23 截获窗体的关闭信息以阻止 窗体关闭	82	实例 5 ocx、dll 文件的注册方法	131
实例 24 按下 Esc 键退出程序	83	实例 6 文件压缩与解压的方法	133
		实例 7 利用 TcompressionStream 压缩与 TdeCompressionStream	

解压文件.....	135
实例 8 将多个安装文件加入一个 安装程序中.....	139
实例 9 将应用程序加入启动组.....	140
实例 10 加入“发送到”菜单.....	141
实例 11 进行文件分割.....	142
实例 12 组合分割后的文件.....	145
第 6 章 剪贴板.....	148
实例 1 监视剪贴板的内容.....	148
实例 2 文本的复制与粘贴.....	149
实例 3 图像的复制与粘贴.....	150
实例 4 流与剪贴板.....	151
第 7 章 文件与目录.....	152
实例 1 在目录中搜索文件.....	152
实例 2 自制文件列表并按类型显示 位图.....	154
实例 3 INI 文件的读写.....	156
实例 4 获取驱动器类型.....	158
实例 5 删除隐含文件.....	159
实例 6 删除只读文件.....	160
实例 7 文件拷贝.....	161
实例 8 文件删除.....	162
实例 9 创建文件夹.....	162
实例 10 一次性创建多层目录.....	163
实例 11 删除文件夹.....	163
实例 12 获取文件的日期.....	163
实例 13 修改文件的日期.....	165
实例 14 将长文件名转换为短文件名.....	166
实例 15 只更改文件的扩展名.....	167
实例 16 将目录删除至回收站.....	167
实例 17 复制目录树.....	168
实例 18 检测文件是否被别的程序 打开.....	168
第 8 章 画布与图形图像.....	170
实例 1 RGB 与 TColor 的转换.....	170
实例 2 TCanvas 与 Font.....	171
实例 3 移动动画的实现.....	171
实例 4 利用 ScrollBox 滚动显示大 位图.....	175
实例 5 实现屏幕拷贝至剪贴板.....	176

实例 6 改变画布的文字显示分辨率.....	177
实例 7 改变画布的分辨率.....	178
实例 8 不同分辨率的画布画同样 比例的图形.....	179
实例 9 位图的 CMYK 分色技术.....	181
实例 10 图像的放大与缩小.....	183
实例 11 艺术显示图片.....	183
实例 12 bmp 与 jpg 格式转换.....	187
实例 13 图像翻转.....	188
实例 14 获取图像中的颜色值.....	189
实例 15 防止图像闪烁.....	190
实例 16 图像的分层处理方法.....	192
实例 17 将位图的二进制数转化为 文本.....	193
实例 18 将彩色位图变为灰度位图.....	196
实例 19 在桌面上画图.....	197
实例 20 实现文本自适应显示区域.....	198
实例 21 获取应用程序的图标.....	200
第 9 章 打印.....	202
实例 1 认识 Tprinter.....	202
实例 2 打印画布.....	204
实例 3 简单文本打印.....	205
实例 4 位图打印.....	207
实例 5 打印纸张设置.....	207
实例 6 打印纸张横向与纵向调整.....	211
实例 7 打印纸张的定位方法.....	212
实例 8 打印比例设置.....	213
实例 9 设置打印颜色.....	214
实例 10 设置打印质量.....	214
第 10 章 动态链接库 dll.....	216
实例 1 创建 dll.....	216
实例 2 静态调用 dll 的步骤.....	218
实例 3 动态调用 dll 的步骤.....	219
实例 4 dll 中的 Form.....	222
实例 5 dll 入口与出口—DllEntryPoint.....	224
第 11 章 注册表.....	226
实例 1 设置注册表.....	226
实例 2 获取 Windows 的信息.....	228
实例 3 设置文件关联程序.....	229
实例 4 查找关联程序.....	229

实例 5 保存应用程序的运行状态	230	实例 12 用 Query 修改记录	267
实例 6 查找打印机的安装信息	231	实例 13 用 Table 删除记录	268
实例 7 查找显示器信息	231	实例 14 用 Query 删除记录	268
实例 8 保存自己的密码	232	实例 15 记录批量增加	268
第 12 章 内存与字符串	234	实例 16 记录批量修改	269
实例 1 去掉字符串的空格	234	实例 17 记录批量删除	269
实例 2 字符串大写与小写转换	235	实例 18 获取数据库中的表名	270
实例 3 字符串比较	235	实例 19 获取表中的字段名	270
实例 4 字符串的位加密方法	237	实例 20 求记录中字段的最大值	270
实例 5 查找与替换字符串	238	实例 21 对记录中的字段求和	271
实例 6 字符串截取	239	实例 22 对记录中的字段求平均	272
实例 7 判断汉字的内码	240	实例 23 恼人的 0 日期值	272
实例 8 动态内存分配方法	241	实例 24 定做个性化报表	272
实例 9 利用内存流读位图文件	241	第 15 章 多媒体	279
实例 10 利用内存流合并文件	242	实例 1 利用 TAnimate 制作动画	279
第 13 章 OLE 与 DDE	244	实例 2 利用 TTimer 制作动画	280
实例 1 OLE 容器控件	244	实例 3 检测声卡是否存在	281
实例 2 编辑 OLE 控件	245	实例 4 检测光驱中是否有 CD	282
实例 3 存取 OLE 对象	245	实例 5 弹开与关闭光驱	282
实例 4 与 Word 的接口	246	实例 6 禁止与启用光驱的自动播放 功能	283
实例 5 与 Excel 的接口	247	实例 7 电影全屏播放	284
实例 6 用数据库管理 Word 文档	250	实例 8 利用媒体控件录音	285
实例 7 DDE 客户端程序设计	253	实例 9 伴音的实现	287
实例 8 DDE 服务器端程序设计	255	实例 10 低级函数录音与放音	287
第 14 章 数据库	257	实例 11 对摄像头编程	294
实例 1 在程序中配置 ODBC 数据源	257	第 16 章 网络	302
实例 2 创建数据表	258	实例 1 获取电脑所在工作组名	302
实例 3 修改 Query 的查询结果集	260	实例 2 获取并修改计算机名	302
实例 4 利用 Query 实现 Pack 功能	260	实例 3 获取本机 IP	303
实例 5 利用 BatchMove 实现 Pack 功能	261	实例 4 设置网络驱动器映射方式	304
实例 6 查找的 Locate 方法	263	实例 5 链接自己的网页和邮件	305
实例 7 查找的 Query 方法	264	实例 6 测试是否联网	305
实例 8 Query 实现模糊查询	265	实例 7 在程序中启动拨号	306
实例 9 为 Table 添加记录	265	实例 8 在程序中挂断拨号	307
实例 10 用 Query 添加记录	266	实例 9 Socket 网络连接与数据传送	307
实例 11 用 Table 修改记录	267		

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，稍不注意就会写错。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;
```

```

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，也称之为模块。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，