

Delphi Developer's Handbook

Delphi

高级开发指南

Marco Cantù

[美]

Tim Gooch 著

John F. Lam

王 辉 张晓晖 戴 英 等译



電子工業出版社

Publishing House of Electronics Industry

URL: <http://www.phei.co.cn>

Delphi Developer's Handbook

Delphi高级开发指南

Marco Cantu

[美] Tim Gooch 著

John F. Lam

王 辉 张晓晖 戴 英 等译

電子工業出版社

Publishing House of Electronics Industry

TP312

内 容 提 要

本书是《Delphi 3从入门到精通》的姊妹篇,也是著名Delphi专家Marco Cantu的高水平代表作。《Delphi 3从入门到精通》从基础知识到深入的编程技巧全面地介绍了Delphi的编程知识。本书在其基础上全面讨论了许多高深的专业话题(这些话题常常被其它Delphi书籍轻轻掠过,但编程者却很难了解或掌握)。例如,VCL专题、组件的所有权与访问的稳定性、RTTI、交互操作技巧、高级组件窗口、线程、Delphi的存储管理、扩展环境、向导(Wizard)与专家(Expert)的编制技巧、数据库编程等等。象这样深入介绍Delphi编程知识的书籍实属凤毛麟角。由于本书未重复介绍Delphi的基础知识,所以建议读者先阅读《Delphi 3从入门到精通》或类似的书籍,在有了较好的基础的前提下,本书会帮助读者插上起飞的翅膀,而成为高水准的Delphi专家,并由此推延出其它语言的高级编程技巧。



Copyright©1998 SYBEX Inc., 1151 Marina Village Parkway, Alameda, CA 94501. World rights reserved. No part of this publication may be stored in a retrieval system, transmitted, or reproduced in any way, including but not limited to photocopy, photograph, magnetic or other record, without the prior agreement and written permission of the publisher.

本书英文版由美国SYBEX公司出版,SYBEX公司已将中文版独家版权授予中国电子工业出版社及北京美迪亚电子信息有限公司。未经许可,不得以任何形式和手段复制或抄袭本书内容。

书 名: Delphi高级开发指南

著 者: [美] Marco Cantu Tim Gooch John F. Lam

译 者: 王 辉 张晓晖 戴 英

责任编辑: 晓 军 林 晶

印 刷 者: 北京天竺颖华印刷厂

装 订 者: 三河金马印装有限公司

出版发行: 电子工业出版社出版、发行

北京市海淀区万寿路173信箱 邮编: 100036 发行部电话: 68490077

北京市海淀区万寿路甲15号南小楼一层 邮编: 100036 发行部电话: 68215345

URL: <http://www.ehpher.com.cn>

经 销: 各地新华书店

开 本: 787×1092-1/16 印张: 49.25 字数: 1260千字

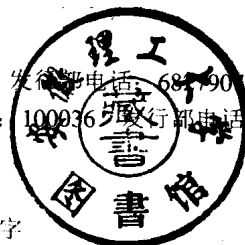
版 次: 1998年8月第1版 1998年8月第1次印刷

书 号: ISBN 7-5053-4753-5/TP·2289

定 价: 80.00元

著作权合同登记号 图字: 01-98-0471

凡购买电子工业出版社的图书,如有缺页、倒页、脱页者,本社发行部负责调换
版权所有·翻版必究



致 谢

首先，我们要感谢Borland与Delphi开发组的每个成员，多年来我们已经成为朋友。专业上的接触促成了专业外的深厚友谊；他们思想活跃，是我们研究工作的良师。我们尤其要感谢David Intersimone（Borland “最后的土著人”）和Nan Borreson（为我们提供了有关的参考书）。

我们还要感谢Sybex公司的员工，是他们的辛勤工作使此书得以面世。特别是Jim Compton和Peter Kuhns，在本书历时一年多的编写过程中，他们起到了关键的作用。同时对于本书多次的改动和修正，宽怀之处，崇尚尽美之心，提供最佳技术之意，昭然可见。Gary Masters也在这一年来提出过许多宝贵意见和帮助，发行人Tony Jonick和协调员Duncan Watson一拿到书稿后就迅速优质的付梓出版，衷心地感谢他们。

两位技术审稿人，Ralph Friedman和Juancarlo Anez，帮助我们确定了本书的技术风格，并对一些复杂的范例提出了许多修改意见。

本书选配光盘也凝聚了许多人的心血。Sybex公司的Dale Wright和Molly Sharp处理了软件编辑的技术事宜，Mary Reeg处理了第三方软件使用许可的事宜。对于所有的软件提供者，我们感激在怀，尤其是那些为本书提供最新软件的人们。

我们感谢Delphi开发组为了该产品的未来所采取的合作和负责的态度；还要感谢本书及论文的读者以及他们不断的意见反馈和支持。

Marco希望感谢意大利的Delphi小组以及在意大利为Borland工作的人们。同时Marco要表达对妻子Lella的衷心谢意，感谢她对本书紧张的写作安排，以及为此度过的许多忙碌周末的宽容和理解。

Tim希望感谢Cobb Group的Mike Stephens和Mark Crane以及Ziff-Davis出版公司的许多其他成员，正是他们的帮助，使他能够忙中抽空致力于本书的写作。他还要表示对妻子Leeann、儿子Tyler和Spencer的诚挚谢意，感谢他们对时而脾气粗暴、时而心不在焉的丈夫与父亲的谅解。

前 言

每一种Delphi新版本的出现，都会相应地增添一些新特性，这些新特性也涉及到了大量的各种专业问题，Marco在《Delphi 3从入门到精通》（Sybex出版，美迪亚公司翻译）一书中深入剖析了Delphi的中初级特性，但是很多高级问题在该书中却没有论述，因此我们又组织编写了一本大型的Delphi编程用书《Delphi高级开发指南》。该书继承了上一本书通过范例解释问题的方法，重点揭示了那些在《Delphi 3从入门到精通》一书中没有涉及到的特性。在大量的专题讨论中，读者将学习到建立DLL与DCU向导，进行流水线式应用程序开发的方法，编写数据——感应组件的方法，以及如何充分利用客户机/服务器版Delphi对三级数据库结构的支持。正如书名暗示的一样，本书是为了满足从事专业Delphi应用程序开发工作的读者需要而编写的。

我们在本书中重点讨论了Delphi3的技术与特性，尽管很多内容也适用于以前的版本，甚至几乎可以确定，也适用于将来的版本。

读者将会发现，在本书中没有介绍基础知识。从一开始，我们就假定本书的读者已经熟悉了在《Delphi 3从入门到精通》或其它中级Delphi书籍中讨论过的基础知识。然而，充实的内容与实用的范例将有助于读者把自己的Delphi知识提高到一个新的层次上。

可以看到，本书是由多位作者合作编写的，但是我们采取了一种非常规的合作方法。Marco为本书设计了基本结构与技术风格，并负责编写了大部分范例、组件与附加工具。Tim整理了Marco的笔记和手稿，并将它们整理为可读性更强，更容易学习的形式。至于复杂的COM编程部分，John Lam则对此进行了精彩的论述，读者将会在第10章与第11章中体会到这一点。

本书的组织方式

我们将本书划分为四个主要部分：

第一部分：Delphi基础。前六章讨论了Object Pascal与VCL的专题，例如受保护属性与组件所有权的访问与流、稳定性、运行时类型信息（RTTI）、组件的建立以及使用组件包进行组件分配等有关的专题。

第二部分：Delphi与Windows。第7~9章重点讨论Delphi与Windows环境的交互作用，包括对VCL与Windows、高级窗口组件、线程以及Windows与Delphi的内存管理等专题的深入讨论。第10章与第11章对COM理论以及实现它的Windows与Delphi工具进行了深入研究。

第三部分：Delphi环境的扩展。在第12~16章中，读者会学习到定制Delphi IDE，将开发过程流水线式以及安全化的有效方法。读者将学会建立属性编辑器、组件编辑器以及向导，研究版本控制系统，最后介绍了建立一个对象调试器的完整步骤。

第四部分：Delphi数据库编程。第17~20章深入研究了Delphi对数据库应用程序的支持。读者将学习建立数据——感应组件、定制DBGrid以及创建一个数据库窗体向导。读者还将学

习在三级数据库模型中进行操作，建立远程数据模块与客户端；读者还将研究Delphi用于在Web上公布一个数据库的工具。

在每一部分中，我们都尽可能多地创建范例与组件，来解释在原文中讨论的特性与技术。尽管范例的主要目的是教授有效的Delphi编程技术，但其中一些范例就其功能而言，还可以作为有用的工具。在本书中没有出现很长的源代码清单，而是重点介绍并解释核心代码。最后，本书中也没有复制在联机帮助与VCL源文件中可以查寻到的参考资料；我们将指导读者如何查寻这些信息。

本书选配光盘

在本书选配光盘上，读者可以发现所有在原文中描述过的范例、组件以及附加工具的源代码与可执行文件。这样，读者可以选择，简单地运行已完成的范例应用程序，研究其源代码，还可以按步就班地建立并编译各个范例。为了简化光盘上源代码的查阅，我们提供了源代码文件的HTML格式版本，并为了读者使用方便，将它们进行了链接。为了方便读者寻找某一特殊文件或代码范例，我们还相关参照了HTML文件中所有有效关键字与标志符。

在光盘上，读者还会发现各种工具、文档以及一些当前功能最强的Delphi配套产品的演示版本。光盘中提供了这些第三方工具的一个完整列表。

与作者的联系方式

要想获得最新信息以及对范例与原文的更新，读者可以访问Marco的Web站点：

<http://www.marcocantu.com>

或Sybex的站点：

<http://www.sybex.com>

读者还可以通过marco@marcocantu.com与Marco联系，或通过tim_gooch@cobb.com与Tim联系。

目 录

第一部分 Delphi基础	1
第1章 Object Pascal的秘密	1
1.1 Delphi的长字符串	1
1.2 Pascal中的整型按位操作	6
1.3 类引用	10
1.4 Delphi的保护异常	25
本章小结	29
第2章 VCL的秘密	30
2.1 组件的所有权	30
2.2 令人惊奇的Name属性	36
2.3 组件的查寻	40
2.4 类型安全的列表	42
本章小结	50
第3章 流与持久性	51
3.1 Delphi的流类	51
3.2 文件流	53
3.3 内存流	56
3.4 编写定制的流类	60
3.5 TReader与TWriter类	63
3.6 读写组件	69
3.7 读写窗体文件	83
3.8 TParser类的使用	91
本章小结	104
第4章 运行时类型信息	105
4.1 访问类型信息	105
4.2 序数类型的RTTI	109
4.3 对象方法指针的RTTI	115
4.4 类的RTTI	119
4.5 访问属性值	125
本章小结	132
第5章 建立组件	133
5.1 建立组件的原因与方法	133

5.2	建立简单的组件	138
5.3	复合组件：红绿灯	145
5.4	定制按钮	155
5.5	使用TCustom...类：数字编辑	165
5.6	使用Collection属性	167
5.7	建立组件中的高级话题	174
	本章小结	181
第6章	深入Delphi 3的组件包	182
6.1	组件的是与非	182
6.2	将组件打包	185
6.3	一个组件的组件包	188
6.4	组件包及其单元的列表	190
6.5	准备帮助文件	194
6.6	发布组件包集合	195
	本章小结	199
第二部分	Delphi与Windows	201
第7章	VCL与Windows	201
7.1	不使用VCL的Windows应用程序	201
7.2	Windows消息与Delphi事件	214
7.3	剖析内部Delphi消息	228
7.4	Delphi的窗体、Windows的保留内存及子类化	237
	本章小结	247
第8章	高级的窗口组件	248
8.1	建立Form-Extender组件	248
8.2	建立应用程序扩展器组件	256
8.3	非矩形控件	268
8.4	运行时的Dragging与Sizing组件	277
	本章小结	285
第9章	进程与存储器	287
9.1	存储管理	287
9.2	程序间的数据传递	307
	本章小结	318
第10章	Delphi与COM	319
10.1	组件软件、Delphi与COM	319
10.2	编程问题与COM的解决方法	321
10.3	COM编程问题与Delphi的解决方法	339
	本章小结	353

第11章	应用COM	354
11.1	脚本化应用程序	354
11.2	扩展应用程序	367
	本章小结	389
第三部分	扩展Delphi环境	391
第12章	属性编辑器	391
12.1	属性编辑器介绍	391
12.2	编写属性编辑器	394
12.3	属性编辑器的功能	398
12.4	现实中的属性编辑器	411
12.5	声音编辑器	412
12.6	编辑点集合	415
12.7	访问其它组件: Comparative Name编辑器	418
12.8	同时编辑多个组件	422
	本章小结	424
第13章	组件编辑器	426
13.1	编写组件编辑器	426
13.2	高级组件编辑器	431
13.3	用对象监视器编辑	448
	本章小结	454
第14章	向导	456
14.1	向导的基础	458
14.2	标准向导与项目向导	460
14.3	创建附件向导	471
14.4	列表模板向导	479
14.5	组件向导	484
14.6	编写窗体向导	499
14.7	基于代理的窗体向导	513
14.8	使用项目生成器	518
14.9	PasToWeb向导	520
	本章小结	523
第15章	其它Delphi扩展	524
15.1	外部工具和转换宏	524
15.2	版本控制系统界面	525
15.3	处理Delphi通知	530
15.4	ToolsAPI总结	546
15.5	使用定制设计模块	548
15.6	调整Delphi环境	557
	本章小结	571

第16章	对象调试器	572
16.1	RTTI帮助函数	572
16.2	浏览运行时属性	579
16.3	版本2: 组件树	584
16.4	版本3: 编辑属性	588
16.5	复制对象监视器的用户界面	594
16.6	在组件中包装对象调试器	607
	本章小结	610
第四部分	Delphi数据库编程	611
第17章	编写与数据相关的组件	611
17.1	数据链接	611
17.2	编写面向字段的与数据相关控件	615
17.3	生成定制的数据链接	622
17.4	定制DBGrid组件	639
17.5	可复制的与数据相关组件	643
	本章小结	652
第18章	扩展Delphi数据库支持	653
18.1	增强型数据库窗体向导	653
18.2	创建定制数据集合	669
18.3	在流中保存数据库数据	672
18.4	在定制数据集合中保存组件	692
	本章小结	706
第19章	远程数据模块与客户端	707
19.1	数据库三步曲	707
19.2	建立范例应用程序	711
19.3	向服务器添加约束	715
19.4	向客户端添加特性	717
	本章小结	726
第20章	在Web上公布数据库	727
20.1	静态与动态的Web页	727
20.2	CGI编程简介	729
20.3	CGI数据库编程	734
20.4	使用普通的ISAPI	744
20.5	Delphi 3的ISAPI支持	747
20.6	HTML生成器组件	759
20.7	作为数据库前端的ActiveForms	770
	本章小结	775
	关于本书选配光盘	777
	读者购盘说明	778

第一部分 Delphi基础

第1章 Object Pascal的秘密

- Delphi的长字符串
- Pascal中的整型按位操作
- 类引用
- Delphi的保护异常

学习Object Pascal是比学习别的语言更容易些，但即使对于非常有经验的Delphi程序员来说，一定仍存在着一些未知的区域未曾涉及。在本章中，我们将介绍这样一些鲜为人知的特性。这不能算是一种指导，而是对一些关键概念的深入理解，正是这些概念构筑了本书大多数编程工作的基础：

- Delphi实现长字符串的方法
- 按位操作
- 使用类引用识别类的层次，用类名动态地创建对象
- 受保护数据的访问

全面理解这些问题不但有助于后续章节内容的学习，而且会使读者大大增强Delphi编程的能力。事实上，这些概念正是深入理解Delphi的基础。

说明：我们将利用程序范例来研究这些问题，这些范例收录在本书选配光盘上。前言中曾提到，有多种查阅光盘内容的方式，但目录结构其实非常简单。主要的文件夹DDHCODE中包含了本书的所有范例，按照章节序号：01、02等等划分了次级的文件夹。在每一章的文件夹中，又为每个程序建立了子文件夹，其中包括该程序的EXE、PAS以及其它相关文件。因为范例是设计用来解释Delphi特性的，所以我们没有演示每个范例建立的完整过程，或列出完整的代码清单；我们只是根据当时的问题解释相关的代码。

1.1 Delphi的长字符串

首先，我们来看看Delphi（从版本2开始）最明显的一个特性：长字符串（在联机帮助中，读者可以看到这些字符串被描述为AnsiString对象，但因为它们最显著的特征是长度可以超过255个字节，所以大多数人称它们为“长字符串”）。32位版本的Delphi缺省地使用长字符串，尽管可以使用\$H编译指令来控制它们的使用。可以利用ShortString类型来继续使

用老字符串类型，但是只有在很少的情况下这样做才有意义，比如在记录或其它数据结构中拾取字符串时。

在Delphi 1与更早的Turbo以及Borland Pascal版本中，String变量通常存放在堆栈上。相比之下，长字符串是基于堆的对象（就象TObject派生类的对象一样），但可以自动地对它们分配与再分配。这一点是重要的，因为这样使得Delphi在多处（每个实例都是指向同一地址的指针）需要使用一个字符串时，可以利用引用计数来优化字符串的使用。例如，如果这样写：

```
var  
    S1: String ;
```

就可以在内存中自动建立一个指向长字符串的间接访问指针。该指针的初始值是nil。只要向该指针赋值：

```
S1 := 'Test' ;
```

Delphi就会为字符串分配足够的内存，使S1指向新字符串，并试着释放它原来使用的内存。

技巧：顺便提一下，可以使用SetLength函数强制Delphi为长字符串分配一段指定大小的内存。

长字符串使用引用模式（自动间接访问基于堆的字符串），最大的优点是使得引用计数成为了可能。对于每个字符串（在内存中），Delphi存储一个相关的整型数来指示字符串的大小，另一个整型数来指示字符串的引用号码。

假设要复制一个长字符串，可以这样写：

```
S2 := S1 ;
```

Delphi不用复制字符串S1（及其存储内容），只是增加引用计数并改变第二个字符串变量S2，分配给它与第一个字符串相同的内存地址。这样，在内存中只存放了一份字符串，但可以由两个变量来使用。如果后来改动了其中一个字符串，Delphi会将第一个字符串复制到新的内存地址，因为改动应该只影响一个字符串，而不是两个字符串。这项技术被称为copy-on-write（写复制）。当引用同一长字符串的所有变量都不在处于作用域内时，引用计数自动归零，而且Delphi会释放字符串所占内存。

1.1.1 字符串引用

为了帮助读者更好地理解Delphi是如何实现长字符串的，我们编写了一个小程序，名为LString，该程序演示了字符串内存地址分配与字符串引用的情况（事实上，我们在下一节会看到该程序还有其它功能）。读者可以在本书选配光盘的DDHCODE\01\LSTRING文件夹中运行LString程序并浏览其完整的源代码。

图1.1显示了LString项目启动时的主窗体。中部的列表框显示了Test字符串的有关信息：它的当前文本，它的长度（用不同的方法计算所得），它的内存地址（也就是说，应用字符串的变量值），以及它的当前引用计数。

下面是UpdateInfo对象方法的代码，我们首先在OnCreate事件处理程序中调用该对象方法，然后每当程序改变字符串时都将调用它：

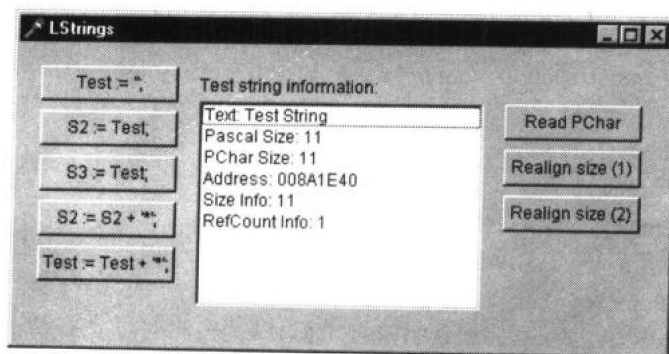


图1.1 Lstring程序启动时的输出

```

procedure TForm1.UpdateInfo;
begin
  with Listbox1.Items do
  begin
    Clear;
    Add ('Text: ' + Test);
    Add ('Pascal Size: ' + IntToStr (Length (Test)));
    Add ('PChar Size: ' + IntToStr (StrLen (PChar (Test))));
    Add (Format ('Address: %p', [Pointer (Test)]));
    Add ('Size Info: ' + IntToStr (GetSize (Test)));
    Add ('RefCount Info: ' + IntToStr (GetRefCount (Test)));
  end;
end;

```

注意，我们不但计算了字符串传统的“Pascal size”，而且计算了“PChar size”（查找空终止符）。下一节中会讨论到，在某些情况下，它们是有区别的。为了显示字符串的内存地址，我们只需将它的变量转换为指针，然后使用标准的Format库函数将它转换为文本。指针自己引用了字符串的第一个字节，这简化了字符串与PChar之间相互转换的过程。结果，Delphi将长度信息与引用计数存储在从字符串第一个字节开始的负偏移处，如图1.2所示。

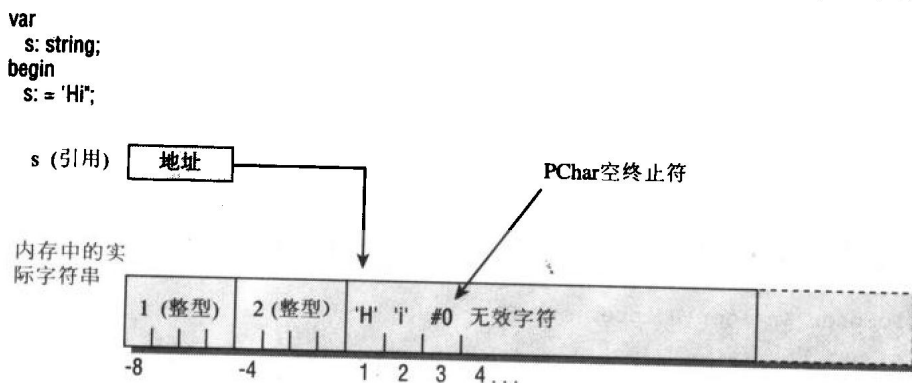


图1.2 一个长字符串的内存布局

不幸的是，Delphi没有提供访问引用计数与长度字节的标准方式。为了解决这个问题，我

们创建了两个定制函数，**GetRefCount**与**GetSize**，来检索该信息。下面是这两个函数的代码：

```
function GetRefCount (const s: string) : Integer;
var
  RefCountPointer: Pointer;
begin
  if Pointer(s) <> nil then
  begin
    RefCountPointer :=
      Pointer (Integer (Pointer (s)) - 8);
    Result := Integer (RefCountPointer^);
  end
  else
    Result := 0;
end;

function GetSize (const s: String) : Integer;
var
  SizePointer: Pointer;
begin
  if Pointer(s) <> nil then
  begin
    SizePointer := Pointer (
      Integer (Pointer (s)) - 4);
    Result := Integer (SizePointer^);
  end
  else
    Result := 0;
end;
```

在这两个函数中，读者可以注意到一个特殊的转换组合。首先，将字符串转换为一个**Pointer**：

`Pointer(s)`

输出变量地址。其次，将指针转换为一个**Integer**：

`Integer(Pointer(s))`

这是必须的，因为只有这样，我们才可以在地址上执行指针运算（减八位用于引用计数，减四位来检索长度字节）。最后，将字符串转换回**Pointer**：

`Pointer(Integer(Pointer(s))-8)`

初始化一个临时**Pointer**变量是必须的，然后我们可以对其进行间接访问，并将其转换为整型数。

可以看到，我们将这些函数的字符串参数声明为了**const**参数。这一点是重要的，因为将字符串作为一个非**const**参数进行传递，会复制字符串，并且会临时增加引用计数。读者还

可以注意到,我们测试了字符串是否为`nil`,这代表了一个空字符串或一个新的,未初始化的字符串。

现在,单击窗体左边五个按钮中时,就可以改变字符串的状态,并可以在列表框中看到结果。每个按钮的功能都很容易理解,因为我们将代码复制到了它们的标题中。下面是一个范例:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
    S2 := S2 + '*';
    UpdateInfo;
end;
```

通过单击不同的按钮,可以实现:

- 重新设置字符串的值为空字符串 (`Test := ''`; 按钮)。该操作会释放已分配的内存并设置字符串的地址为`nil`。
- 建立更多对字符串的引用 (`S2 := Test`; 按钮与 `S2 := Test`; 按钮)。
- 改变其它某个字符串的文本 (`S2 := S2 + '*'`; 按钮); 如果`S2`先前赋给了`Test`,该操作会减少`Test`字符串的引用计数。
- 改变原`Test`字符串的文本 (`Test := Test + '*'`; 按钮)。如果已经建立了其它对字符串的引用,Delphi会将`Test`复制到一个新的内存地址(当用户运行程序时,可以马上在屏幕上看到结果)。

1.1.2 长字符串与PChar问题

窗体右边的按钮解释了由长字符串与PChar指针引起的一个复杂问题。第一个按钮的OnClick事件的处理程序执行了一步很简单的操作;通过调用GetWindowText Windows API函数,它向Test字符串复制了窗体的标题。为了做到这一点,我们需要将字符串变量转换为一个PChar:

```
procedure TForm1.Button6Click(Sender: TObject);
begin
    SetLength (Test, 100);
    GetWindowText (Handle, PChar (Test), 100);
    UpdateInfo;
end;
```

该操作的结果是,如果存在其它对字符串的引用,Delphi将向新的内存地址复制字符串并更新该复制。然而,问题出现了,如图1.3所示。字符串的Pascal size(检索内部字符串信息并使用Length函数返回)与StrLen函数(它寻找空终止符)实际返回的精确长度不相一致。因为Delphi无法知道,用户通过向GetWindowText函数传递字符串的地址改动了它,所以它不能自动地为字符串重新设置长度信息。

例如,如果单击Read PChar按钮,然后单击左边的最后一个按钮,可向字符串添加新文本,Delphi将向字符串的末尾追加新字符。不幸的是,在该位置留有空终止符,因此该操作不会影响字符串的实际输出。

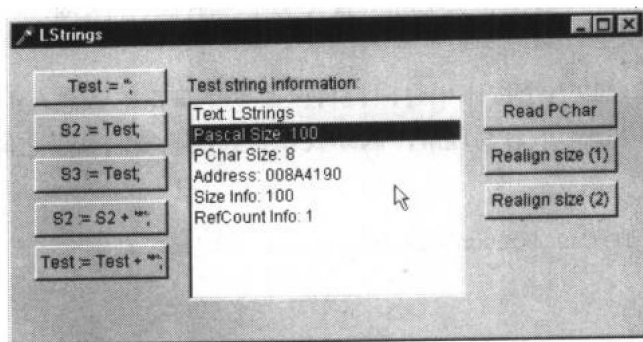


图1.3 单击Read PChar按钮时，LString程序的输出

为了解决问题，可以使用一个小技巧。**LString**程序实现了两个转换。第一个**Realign Size**按钮通过一个冗余转换修正了字符串长度问题。下面的代码将数据类型转换回字符串（在又一次将它转换为**PChar**之后）：

```
Test := PChar (Test);
```

这与下面写法效果相同：

```
Test := String (Pchar (Test));
```

因为**Delphi**会自动地将**PChar**指针转换为长字符串。

第二个**Realign size**按钮将长字符串的长度重新设置为**PChar**字符串的当前长度：

```
SetLength (Test, StrLen (Pchar (Test)));
```

读者可以猜到，该操作实际减小字符串在内存中的长度；而且因为它不会移动字符串，所以它应该比第一种方法的速度快（第一种方法需要**Delphi**复制字符串，可以从字符串的地址上看出这一点）。在冗余转换或叫长度重新调整之后，字符串就可以象以前那样正确显示了。

1.2 Pascal中的整型按位操作

整型值的按位操作在**Pascal**中并不象在**C/C++**中那样常见。典型的**Delphi**应用程序更多的是使用集合，以一种更清晰、可读性更强、类型更为安全的方式，来处理成组的二进制值，而且我们不必自己关注在给定的字节内设置或清除位。**Delphi**使用集合的一个例子是文本的**Style**属性。可以这样改变**Style**属性

```
Style := Style + [fsBold, fsItalic] - [fsUnderline] ;
```

该语句清除了**Underline**属性，并向**Style**集合添加了**Bold**与**Italic**属性。

然而，当使用**Windows API**时，我们需要使用来自**C/C++**的技术来改变一个16位或32位值的单个位。每当用户向一个**Windows API**函数传递一个标志时，它基本上就可以算是一个位的集合。

在**Delphi**应用程序中有两种方法来进行位操作：使用**Pascal**按位操作符，或使用**Tbits**类。

1.2.1 使用按位操作符

Pascal提供了多个布尔操作符与标准操作。例如，shl与shr操作符允许用户分别左移或右移一个整型值。同样，and、or、xor与not操作符可以用来对布尔值或整型值的位执行标准的布尔操作（在第二种情况下，这些操作符被称为按位操作符）。下面，我们提供了一些常用的函数来改动与测试整型值内的指定位：

```
function IsBitOn (Value: Integer; Bit: Byte): Boolean;
begin
    Result := (Value and (1 shl Bit)) <> 0;
end;

function TurnBitOn (Value: Integer; Bit: Byte): Integer;
begin
    Result := Value or (1 shl Bit);
end;

function TurnBitOff (Value: Integer; Bit: Byte): Integer;
begin
    Result := Value and not (1 shl Bit);
end;
```

为了了解这些函数的执行情况，我们需要研究一下Bits1范例。在主窗体上，可以发现一个UpDown组件，一个只读编辑框（值域限定为0-31），以及一个方框内的五个复选框（每个框对应着我们将要调整的整型值的一位），如图1.4所示。为了简化测试与设置每个复选框状态的代码，我们将每个复选框的Tag属性设置为该位的位置值，这将有助于我们在运行时识别它们。

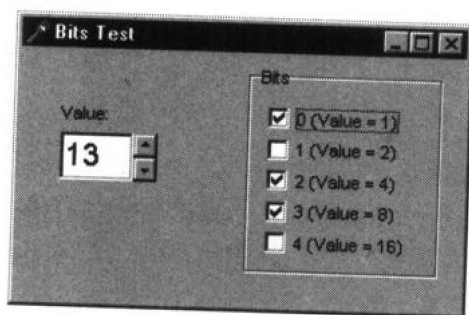


图1.4 Bits程序显示的窗体

当用户通过单击上下箭头来调整编辑框的值时，我们使用下列事件处理程序来调整复选框：

```
procedure TForm1.Edit1Change(Sender: TObject);
var
    I: Integer;
    CheckBox: TCheckBox;
begin
```