Delphi入门

目录

[一、Delphi 环境 3](#_Toc452990256)

[1.1 菜单栏 3](#_Toc452990257)

[1.2 工具栏 3](#_Toc452990258)

[1.3 控件面板（Tool Palette） 4](#_Toc452990259)

[1.4 属性查看器（Object Inspector） 5](#_Toc452990260)

[1.5窗体设计器 6](#_Toc452990261)

[1.6代码编辑器 7](#_Toc452990262)

[1.7 对象树视图 8](#_Toc452990263)

[二、Delphi 程序结构 8](#_Toc452990264)

[2.1 Project1.dpr 9](#_Toc452990265)

[2.2 Unit1.pas 9](#_Toc452990266)

[2.3 Unit1.dfm 9](#_Toc452990267)

[三、Object Pascal语言 10](#_Toc452990268)

[3.1 关健字及保留字 10](#_Toc452990269)

[3.2数据类型 11](#_Toc452990270)

[3.2运算符 15](#_Toc452990271)

[3.3 标点符号 17](#_Toc452990272)

[3.4 语法 17](#_Toc452990273)

[3.4.1变量定义 17](#_Toc452990274)

[3.4.2 常量定义 18](#_Toc452990275)

[3.4.3 赋值语句 18](#_Toc452990276)

[3.4.4 If..Then..Else语句 18](#_Toc452990277)

[3.4.5 Case 语句 19](#_Toc452990278)

[3.4.6 Repeat..Until语句 19](#_Toc452990279)

[3.4.7 While..Do语句 20](#_Toc452990280)

[3.4.8 For..Do语句 20](#_Toc452990281)

[3.4.9异常处理语句 21](#_Toc452990282)

[3.4.10过程与函数 22](#_Toc452990283)

[四、常用控件 24](#_Toc452990284)

[4.1 窗体 24](#_Toc452990285)

[4.2 标签控件(TLabel) 26](#_Toc452990286)

[4.3 文本框控件(TEdit) 26](#_Toc452990287)

[4.4 Memo控件(TMEMO) 27](#_Toc452990288)

[4.5 按钮控件(TButton,TBitBtn) 27](#_Toc452990289)

[4.5.1 TButton控件 27](#_Toc452990290)

[4.5.2 TBitBtn控件 27](#_Toc452990291)

[4.6 选项按钮(TCheckBox,TRadioButton) 28](#_Toc452990292)

[4.6.1 TCheckBox(复选框) 28](#_Toc452990293)

[4.6.2 TRadioButton(单选框) 28](#_Toc452990294)

[4.7 列表框(TListBox) 28](#_Toc452990295)

[4.8 组合框(TComboBox) 29](#_Toc452990296)

[4.9 Panel 控件(TPanel) 29](#_Toc452990297)

[4.10 对话框控件 30](#_Toc452990298)

[4.10.1OpenDialog控件 30](#_Toc452990299)

[4.10.2 SaveDialog控件 30](#_Toc452990300)

[4.10.3FontDialog控件 30](#_Toc452990301)

[4.10.4 ColorDialog控件 30](#_Toc452990302)

[4.10.5 PrintDialog控件 31](#_Toc452990303)

[4.10.6 ReplaceDialog控件 31](#_Toc452990304)

[4.11 数据集(DataSet) 31](#_Toc452990305)

[4.11.1 TADODataSet控件 31](#_Toc452990306)

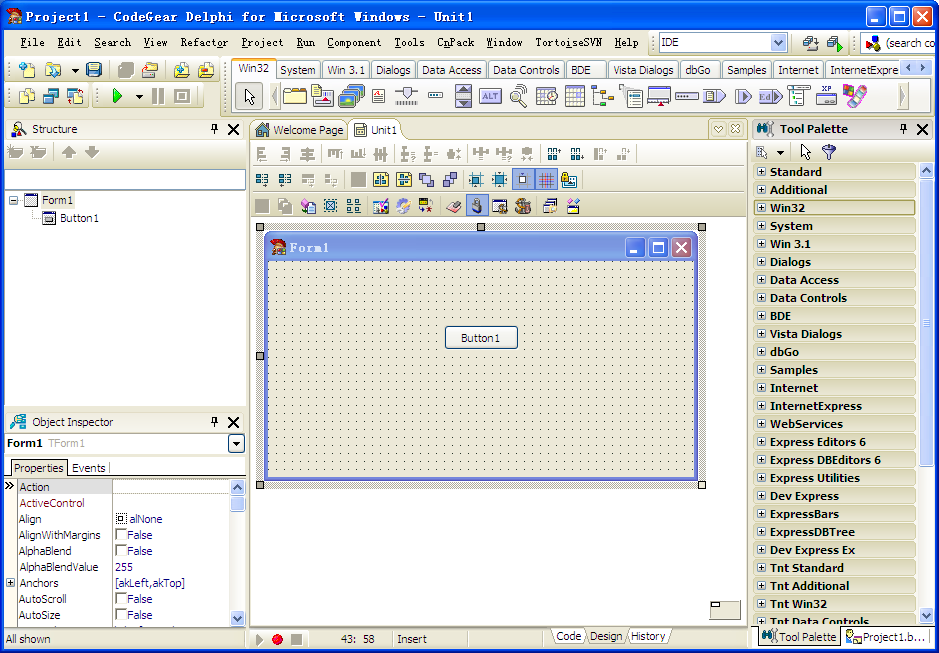
[五、例子 32](#_Toc452990307)

[5.1 简单计算器 32](#_Toc452990308)

[六、附录 33](#_Toc452990309)

[6.1 Delphi 快捷键 33](#_Toc452990310)

# 一、Delphi 环境



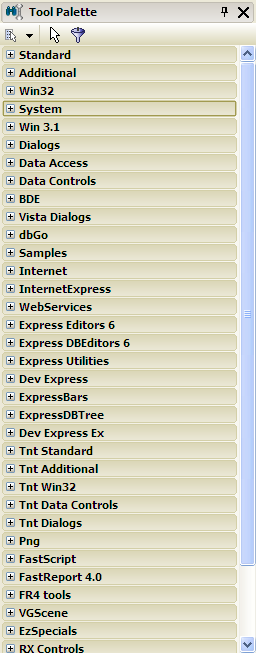
## 1.1 菜单栏



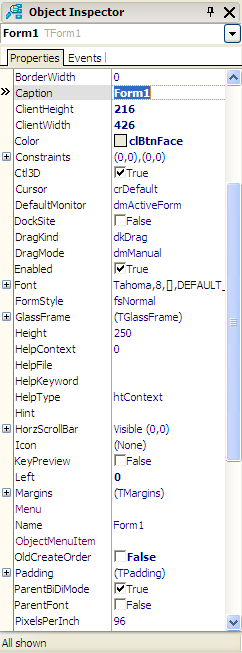
## 1.2 工具栏



## 1.3 控件面板（Tool Palette）



## 1.4 属性查看器（Object Inspector）



Object Inspector窗口含有两页：Properties页显示窗体中当前被选择部件的属性信息，并允许改变对象的属性；Events页列出了当前部件可以响应的事件。按动Object Inspector下端的“Events”页标签，使得Events页可见，这一定的事件后边的空白处，可以定义对象接受到相应事件时执行的动作。首次启动时，Object Inspector窗口显示的是当前窗体Form1的属性。Object Inspector根据对象属性的多少，决定是否有滚行显示。移动滚行条，可以查看当前对象的全部属性。

属性查看器的上部是一个下拉列表框，列出了当前活动窗体上所有控件以及它们的类名。这样，除了在窗体上直接用鼠标单击选中控件外，还可以通过对象查看器的下拉列表框选择。

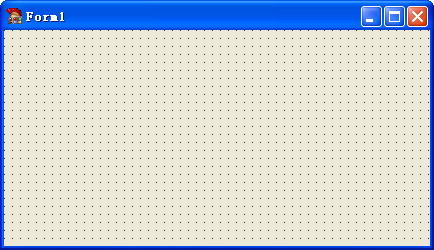
对象查看器的 【Properties】 页用于设置控件的属性，左边为控件属性名称，右边为该属性值。

★ 对象查看器 【属性】值若有 【…】符号，表明该按钮有进一步对话框可供设置。

★ 对象查看器 【属性】值若有下拉列表框，可以在双击空白处进行值的切换

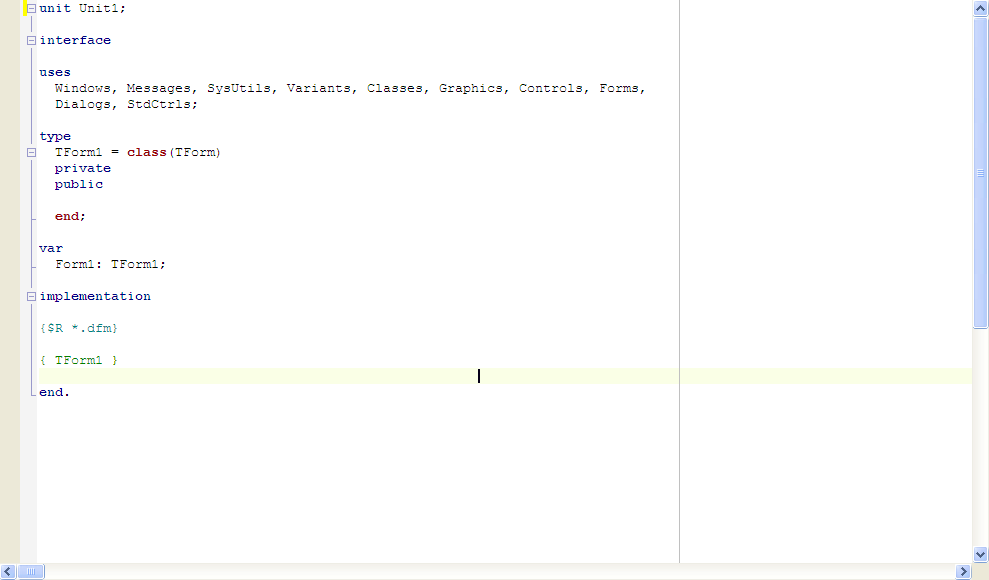
对象查看器的 【Events】页用于设置控件的响应事件。

## 1.5窗体设计器



窗体设计器是开展大部分设计的工作区域。首次启动Delphi时显示的是窗体Form1。可以把部件放在窗体中，通过移动位置、改变尺寸等操作随心所欲地安排它们，以此来开发应用程序的用户界面。您可以把窗体想象成一个可以放置其它部件的容器。窗体上有栅格(Grids)，供放置部件时对齐位置用，在程序运行时Grids是不可见的。

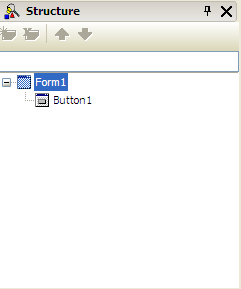
## 1.6代码编辑器



  代码窗口一开始处于窗体窗口之下。因为在Delphi中，设计用户界面直接在窗体中进行，运行结果和设计样板完全一致。当部件被放到窗体上时，Delphi会自动生成大部分的用户界面代码。您所应做的只是在它为您生成的框架中加入完成所需功能的程序段而已。点动Form1的状态行使代码窗口可见。

这个窗口中是代码编辑器。可以在其中书写Delphi应用程序的源代码。当程序中含有不止一个窗口时，会有几个库单元的源程序出现在代码编辑器中。代码编辑器的标题条中显示了当前正在编辑的库单元文件名。要查看某一特定程序的源代码，只需用鼠标点动写有该库单元文件名的页标签，就可以对该库单元进行编辑了。

## 1.7 对象树视图



# 二、Delphi 程序结构

.Bpg 工程项目包，多个工程文件的Dpr的集合。

.Dpr 应用系统的工程文件。

.Pas 单元文文件，存放Pascal源代码。

.Dfm 窗体界面文件，同Pas文件同名，而且一一对立，定义了窗体属性，用二进制格式存放

.Dof 选项文件，包括编译信息等，是一个文本文件，可以查看。

.Cfg 编译器可读的配置文件

.Dsk 保存工程文件的桌面排布情况，下次打开时可以恢复上次保存的桌面状态

.Dsm 集成开发环境IDE临时保存的文件，有时IDE总出错，可以将这个文件删除，系统会重新生成。

.Res Windows 的资源文件，可以定义光标，

.Dcu 编译后的单元文件，程序连接时需要。

.Dll Windows 动态链接库

.Dpk包文件的源代码

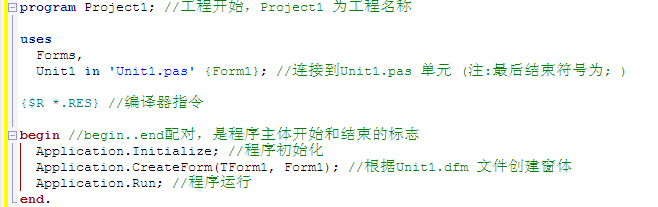
.Bpl 包，是一种特殊的Dll文件，用于代码重用和减少执行文件的长度。

.Exe 编译后的可执行文件.

典型的Delphi程序包括：Project1.cfg、Project1.dof、Project1.dpr、Project1.res、Unit1.dfm、Unit1.pas 等文件，这些文件相互独立又相互联系，共同组成一个完整的 Delphi 工程。其中尤其是Project1.dpr、Unit1.dfm、Unit1.pas 等三个文件，可以说是 Delphi 工程的主题。Project1.dpr 调用了 Unit1.dfm 和 Unit1.pas 两个文件，而这两个文件中前者负责生成程序的界面部分，后者负责程序的代码控制部分。

## 2.1 Project1.dpr

这个最简单的Delphi程序的工程头结构(Project1.dpr )



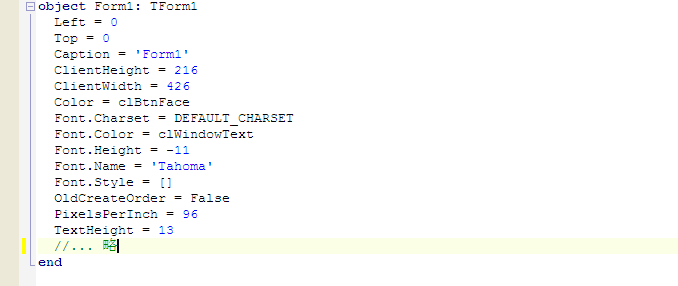
## 2.2 Unit1.pas

单元主体部分包括以下 Interface、Type、Var、Implementation 四大部分



## 2.3 Unit1.dfm

窗体文件是记录窗体上的控件属性信息(可按快捷键Alt+F12快速切换窗体与代码编辑器)

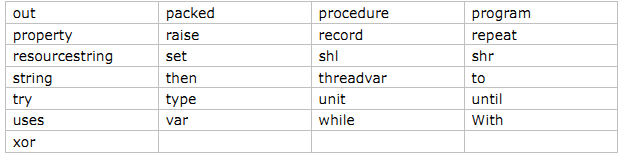
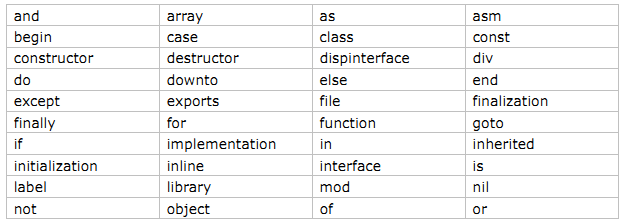


# 三、Object Pascal语言

## 3.1 关健字及保留字

* 关键字是指构成Delphi语句语法的重要词汇。

如：



关健字一般在代码编辑器中默认会以粗体显示。关键字是由 Delphi 编译器决定意义的保留标志符，不能把关键字用作变量、方法、或类型的名字等。

* 保留字:所谓保留字可以理解为Delphi 系统自定义的，留给Delphi 系统自身使用的识别字。Windows、Messages、SysUtils、Classes、Graphics、Controls、Forms、Dialogs 等就是保留字。
* 自定义识别字: 所谓自定义识别字就是用户自己定义的识别字:

[Pascal 语言中的关键字及保留字](http://www.cnblogs.com/del/archive/2008/10/10/968856.html)

[Delphi 关键字详解](http://www.cnblogs.com/del/archive/2008/06/23/1228562.html)

## 3.2数据类型

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 分类 | | | | 范围 | 字节 | 备注 |
| 简单类型 | 序数 | 整数 | Integer | -2147483648 .. 2147483647 | 4 | 有符号32位 |
| Cardinal | 0 .. 4294967295 | 4 | 无符号32位 |
| Shortint | -128 .. 127 | 1 | 有符号8位 |
| Smallint | -32768 .. 32767 | 2 | 有符号16位 |
| Longint | -2147483648 .. 2147483647 | 4 | 有符号32位 |
| Int64 | -263 .. 263 | 8 | 有符号64位 |
| Byte | 0 .. 255 | 1 | 无符号8位 |
| Word | 0 .. 65535 | 2 | 无符号16位 |
| Longword | 0 .. 4294967295 | 4 | 无符号32位 |
| 字符 | AnsiChar(Char) | ANSI字符集 |  | 8位 |
| WideChar | Unicode字符集 |  | 16位 |
| 布尔 | Boolean | False < True Ord(False) = 0 Ord(True) = 1 Succ(False) = True Pred(True) = False | 1 |  |
| ByteBool | False <> True Ord(False) = 0 Ord(True) <> 0 Succ(False) = True Pred(False) = True | 1 |  |
| WordBool | 2 |  |
| LongBool | 4 |  |
| 枚举 |  |  |  |  |
| 子界 |  |  |  |  |
| 实数 |  | Real | 5.0×10-324 .. 1.7×10308 | 8 | [精度]15..16 |
| Real48 | 2.9×10-39 .. 1.7×1038 | 6 | [精度]11..12; 向后兼容 |
| Single | 1.5×10-45 .. 3.4×1038 | 4 | [精度]7..8 |
| Double | 5.0×10-324 .. 1.7×10308 | 8 | [精度]15..16 |
| Extended | 3.6×10-4951 .. 1.1×104932 | 10 | [精度]19..20 |
| Comp | -263 + 1 .. 263 - 1 | 8 | [精度]19..20 |
| Currency | -922337203685477.5808 ..  922337203685477.5807 | 8 | [精度]19..20 |
| 字符串 |  |  | ShortString | 255个字符 | 2..256B | 向后兼容 |
| AnsiString | 大约 231个字符 | 4B..2GB | 8位(ANSI)字符 |
| WideString | 大约 230个字符 | 4B..2GB | 多用户服务和 多语言应用程序;  和com定义的BSTR兼容 |
| 其他 | String String[0..255] PChar PAnsiString PWideString |  |  |
| 结构类型 | 集合 |  | Set | 最多256个元素[0..255] |  |  |
| 数组 | 静态数组 |  |  |  |  |
| 动态数组 |  |  |  |  |
| 记录 |  | Record |  |  |  |
| 文件 |  | File |  |  |  |
| 类 |  | Class |  |  |  |
| 类引用 |  | Class reference |  |  |  |
| 接口 |  | Interface |  |  |  |
| 指针类型 | 无类型指针 |  | Pointer |  |  |  |
| 有类型指针 | 预定义类型指针 | PAnsiString PString PByteArray PCurrency PDouble PExtended PSingle PInteger POleVariant PShortString PTextBuf PVarRec PVariant PWideString PWordArray |  |  |  |
| 过程类型 | 程序过程类型 |  | Procedural |  |  |  |
| 对象过程类型 |  | Procedural |  |  |  |
| 变体类型 |  |  | Variant |  |  |  |
|  | OleVariant |  |  |  |

#### 3.2.1大小写规则

  类型标识符是保留字，应当全部小写。Win32 API 类型常常全部大写，并且遵循诸如Windows.pas或其他API单元中关于特定类型名的规则。对于其他变量名，第一个字母应大写，其他字母则大小写交错。下面是一些例子：

**var**  
  MyString: **string**; *// 保留字*  WindowsHandle: HWND; *// Win32 API 类型*  I: Integer; *//在System单元中引入的类型标识*

#### 3.2.2浮点型

  不鼓励使用Real类型，因为它只是为了与老的Pascal代码兼容而保留的。通常情况下，对于浮点数应当使用Double。Double可被处理器优化，是IEEE定义的标准的数据格式。当需要比Double提供的范围更大时，可以使用Extend。Extend是intel专用的类型，Java不支持。当浮点变量的物理字节数很重要时(可能使用其他语言编写DLL)，则应当使用Single。

#### 3.2.3 Variant和OleVariant

  一般不建议使用Variant和OleVariant。但是，当数据类型只有在运行期才知道时(常常是在COM和数据库应用的程序中)，这两个类型对编程就有必要。当进行诸如自动化ActiveX控件的COM编程时，应当使用OleVariant；而对于非COM编程，则应当使用Variant。这是因为，Variant能够有效地保存Delphi的原生字符串，而OleVariant则将所有字符串转换为OLE字符串(即WideChar字符串)，且没有引用计数功能。

#### 3.2.4 枚举型

枚举类型名必须代表枚举的用途。名称前要加T字符作为前缀，表示这是个数据类型。枚举类型的标识符列表的前缀应包含2 - 3 个小写字符，来彼此关联。例如：

TSongType=(stRock, stClassical, stCountry, stAlternative, stHeavyMetal, stRB);

  枚举类型的变量实例的名称与类型相同，但没有前缀T ，也可以给变量一个更加特殊名称，诸如：FavoriteSongTypel、FavoriteSongType2等等。

#### 3.2.5 其它类型

暂未罗列其它类型(需了解可自行搜索)

## 3.2运算符

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **分类** | **运算符** | **操作** | **操作数** | **结果类型** | **范例** |
| 算术运算符 | + | 加 | 整数，实数 | 整数，实数 | X + Y |
| - | 减 | 整数，实数 | 整数，实数 | Result - 1 |
| \* | 乘 | 整数，实数 | 整数，实数 | P \* InterestRate |
| / | 实数除 | 整数，实数 | 实数 | X / 2 |
| div | 整数除 | 整数 | 整数 | Total div UnitSize |
| mod | 取模 | 整数 | 整数 | Y mod 6 |
| +(一元) | 符号等同 | 整数，实数 | 整数，实数 | +7 |
| -(一元) | 符号相反 | 整数，实数 | 整数，实数 | -X |
| 布尔运算符 | not | 否定 | 布尔型 | Boolean | not (C in MySet) |
| and | 与 | 布尔型 | Boolean | Done and (Total > 0) |
| or | 或 | 布尔型 | Boolean | A or B |
| xor | 异或(两个  条件同时  为True或  同时为fals  e时，为false  否则为true) | 布尔型 | Boolean | A xor B |
| 逻辑(按位)运算符 | not | 按位否定 | 整数 | 整数 | not X |
| and | 按位与 | 整数 | 整数 | X and Y |
| or | 按位或 | 整数 | 整数 | X or Y |
| xor | 按位异或 | 整数 | 整数 | X xor Y |
| shl | 按位左移 | 整数 | 整数 | X shl 2 |
| shr | 按位右移 | 整数 | 整数 | Y shr I |
| 字符串运算符 | + | 连接 | 字符串、压缩串、字符 | 字符串 | S + '.' |
| 指针运算符 | + | 指针加 | 字符指针，整数 | 字符指针 | P + I |
| - | 指针减 | 字符指针，整数 | 字符指针，整数 | P - Q |
| ^ | 指针解除参照 | 指针 | 指针的基类型 | P^ |
| = | 相等 | 指针 | Boolean | P = Q |
| <> | 不等 | 指针 | Boolean | P <> Q |
| 集合运算符 | + | 并集 | 集合 | 集合 | Set1 + Set2 |
| - | 差集 | 集合 | 集合 | S – T |
| \* | 交集 | 集合 | 集合 | S \* T |
| <= | 子集 | 集合 | Boolean | Q <= MySet |
| >= | 超集 | 集合 | Boolean | S1 >= S2 |
| = | 相等 | 集合 | Boolean | S2 = MySet |
| <> | 不等 | 集合 | Boolean | MySet<> S1 |
| in | 成员 | 序数，集合 | Boolean | A in Set1 |
| 关系运算符 | = | 相等 | 简单类型、类、类引用、接口、串、压缩串 | Boolean | I = Max |
| <> | 不等 | 简单类型、类、类引用、接口、串、压缩串 | Boolean | X <> Y |
| < | 小于 | 简单类型、串、压缩串、PChar | Boolean | X < Y |
| > | 大于 | 简单类型、串、压缩串、PChar | Boolean | Len > 0 |
| <= | 小于或等于 | 简单类型、串、压缩串、PChar | Boolean | Cnt<= 1 |
| >= | 大于或等于 | 简单类型、串、压缩串、PChar | Boolean | I >= 1 |
| 类运算符 | as | 转换 | 类和类的实例 |  |  |
| is | 判断 |  |  |
| = |  | 关系运算符 = 和<>也作用于类 |  |  |
| <> |  |  |  |
| 地址(@)运算符 | @X | 如果X是一个变量，那么@X返回X的地址。当编译指示 {$T-} 有效时，@X是Pointer类型;  而在编译指示 {$T+} 状态下时，@X是 ^T 类型，这里的T是X的类型。 | | | |
| @F | 如果 F 是一个例程(函数或过程)，那么@F返回 F 的入口点，@F的类型总是Pointer。 | | | |
| @类中方法 | 当 @ 适用于定义在类中的方法时，方法标识符必需被类的名称限定。例如: @TMyClass.DoSomething | | | |

运算符运算优先级

●一般情况，按照如下顺序运算：

(1)@,not

(2)\*,/,div,mod,and,shl,shr,as

(3)+,-,or,xor

(4)=,<>,<,>,<=,>=,in,is

●若有()，则从()由内外之运算优先；

●相同优先次序,则由左至右之运算优先

## 3.3 标点符号

【’’】(单引号)→字符串

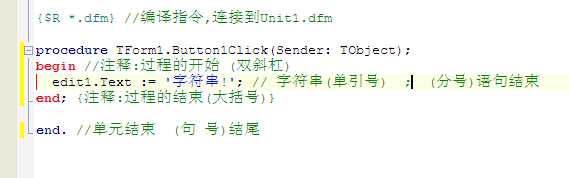
【//】(双斜杠)→注释

【{}】(大括号)→注释

【;】 (分号 )→语句结束

【.】 (句号 )→整个单元结束，一般加在整个单元的End符号之后

各种标点符号使用如图所示：



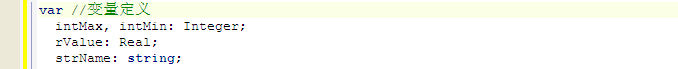
## 3.4 语法

### 3.4.1变量定义

变量是程序代码中代表一个内存地址的标识符，而此地址的内存内容在程序代码执行时可以被改变。

在使用变量前必须对它进行说明，即对它进行命名，并说明它的类型。在所有变量说明以前加上保留字【var】。

变量说明左边是变量的名称，右边则是该变量的类型，中间用【:】（冒号）隔开。



变量的名称应当能够表达出它的用途。循环控制变量常常为单个字母，诸如I 、J 或K 。也可以使用更有意义的名称，例如UserIndex。布尔变量名必须能清楚表示出True 和False 值的意义。

#### 3.4.1.1局部变量

  局部变量用于过程内部，如果需要的话，应当在过程的入口处立即初始化变量。局部的AnsiString 类型的变量自动被初始化为空字符串，局部的接口和dispinterface类型的变量自动被初始化为nil，局部的Variant和OleVariant类型的变量自动被初始化为Unassigned。

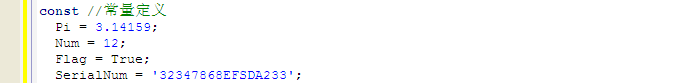
#### 3.4.1.2全局变量

  一般不鼓励使用全局变量。不过，有时候需要用到。即使如此，也应当把全局变量限制在需要的环境中。例如，一个全局变量可能只在单元的实现部分是全局的。

  全局数据如果将由许多单元使用，就应移动到一个公用单元里被所有对象使用。全局数据可在声明时直接初始化为一个值。注意，所有全局变量自动进行零初始化，因此，不要将全局变量初始化为诸如0 、nil、或Unassigned等空值。零初始化的全局变量在.EXE文件中不占空间。零初始化的数据保存在虚拟的数据段中，而虚拟数据段只在应用程序启动时才分配内存。非零初始化的全局数据则在.EXE文件中占空间。

### 3.4.2 常量定义

常量在说明时就被赋予了一个值，在程序执行过程中是不可改变的。

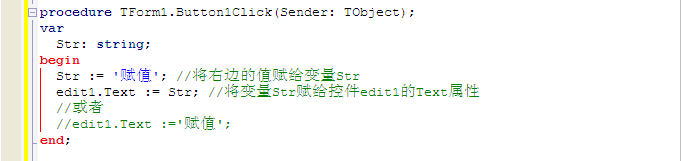


与变量一样，常量也有类型。不同的是，常量假设其类型就是常量说明中其所代表的值的类型。上面的几个常量的类型分别是实数型、整形、布尔型、字符串型。常量用【=】（等号）表示两边的值是相等的。

### 3.4.3 赋值语句

在事件处理过程中，最常做的工作就是把一个新值赋给一个属性或变量。

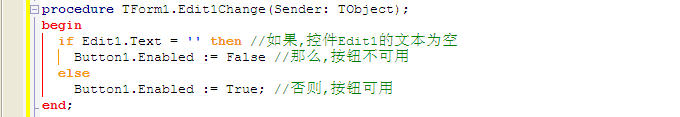
下文的赋值语句是一个OnClick事件。当单击按钮后，将文本控件edit1的Text属性置为等号右边的值:



注:赋值类型和接受此值的属性或变量的类型应相同或兼容

### 3.4.4 If..Then..Else语句

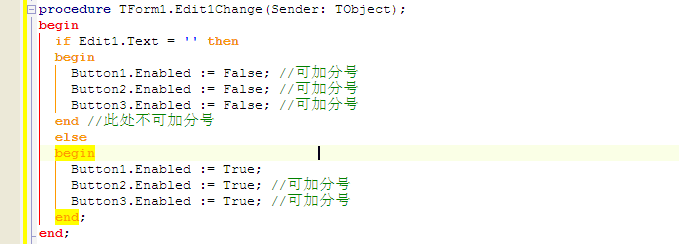
这个语句用于条件判断，意即“如果……那么……否则”，这个语句可以嵌套使用。



需要注意：if..then..else构成完整语句，之间不可有任何分号，否则必引起编译错误，如中间有

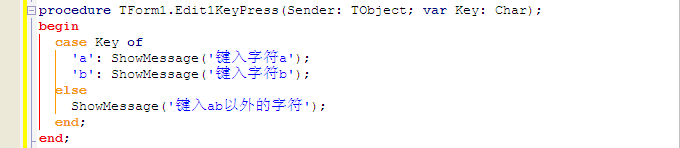
几个语句必须加上分号的话，那么应该在几个语句前后加上【begin..end】，而且【begin..end】的【end】

之后也不能加分号，也就是相当于用【begin..end】把几个语句并成一个语句，如下所示：



### 3.4.5 Case 语句

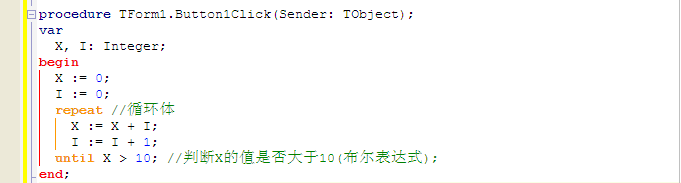
这个语句用于多分支的选择。如：



Case语句也可以用复杂的If..Then..Else语句替换。注：Case支持Integer,Boolean,Char,枚举等类型

### 3.4.6 Repeat..Until语句

repeat-until总是先执行循环体中的内容，然后判断，一旦判断成立循环即告停止(直到型循环)。如：



repeat与until之间的所有语句称为循环体。

说明：

1、repeat语句的特点是：先执行循环，后判断结束条件，因而至少要执行一次循环体。

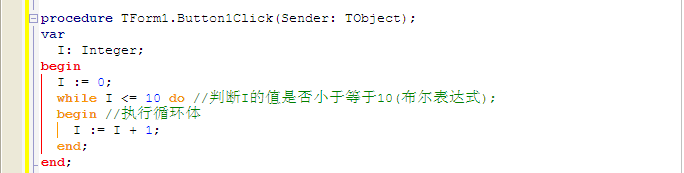
2、repeat-until是一个整体，它是一个（构造型）语句，不要误认为repeat是一个语句，　until是另一个语句。

3、repeat循环体可以是若干个语句，不需用begin和end。

注: repeat循环可以和下面的while循环相互转化。

### 3.4.7 While..Do语句

While语句则是先判断条件，如果判断成立则执行，否则停止(当型循环)。如：



注:while循环可以和上面的repeat循环相互转化。

### 3.4.8 For..Do语句

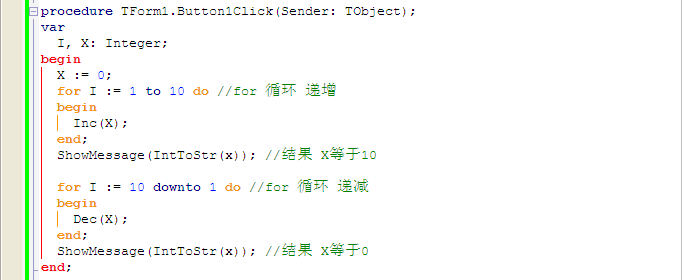
For语句用于已知循环次数的判断。

格式

　　 for 控制变量:=初值 to 终值 do 语句;

　　 for 控制变量:=初值downto终值 do 语句;

如:



For语句执行过程：

1、先将初值赋给左边的变量（称为循环控制变量）；

　　2、判断循环控制变量的值是否已“超过”终值，如已超过，则跳到步骤5；

　　3、结果未超过终值，则执行do后面的那个语句（称为循环体）；

　　4、控制变量返回步骤2；

　　5、循环结束，执行for循环后语句。

说明：

　　循环控制变量必须是顺序类型。例如，可以是整型、字符型等，但不能为实型。

　　循环控制变量的值选用to则为递增；选用downto则递减。

　　循环控制变量的值“超过”终值，对递增型循环，“超过”指大于，对递减型循环，“超过”指小于。

　　循环控制变量的初值和终值一经确定，循环次数就确定了。但是在循环体内对循环变量的值进行修改，常常会使得循环提前结束或进入死环。建议不要在循环体中随意修改控制变量的值。

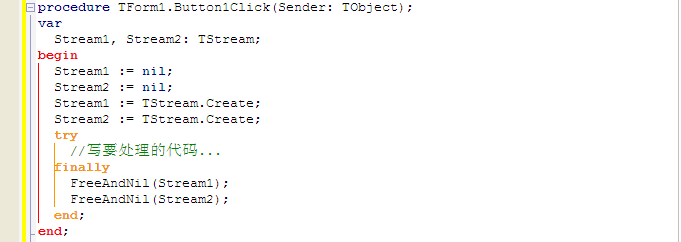
for语句中的初值、终值都可以是顺序类型的常量、变量、表达式。

### 3.4.9异常处理语句

  异常处理主要用于纠正错误和保护资源。这意味着，凡是分配资源的地方，都必须使用**try**...**finally**来保证资源得到释放。不过，如果是在单元的初始/结束部分或者对象的构造器/析构器中来分配/释放资源则例外。

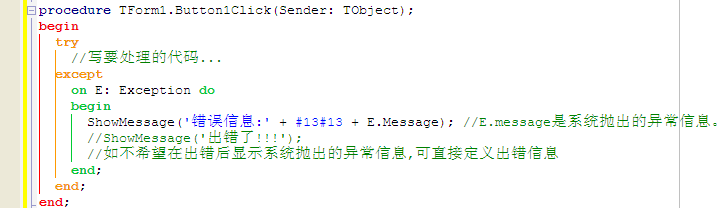
#### 3.4.9.1try...finally的用法

如:



#### 3.4.9.2**try...except的用法**

如:



  如果你希望在发生异常时执行一些任务，可以使用**try**...**except**。通常，没有必要为了简单地显示一个错误信息而使用**try**...**except**，因为Application对象能够自动根据上下文做到这一点。如果要在子句中激活默认的异常处理，可以再次触发异常。

### 3.4.10过程与函数

定义过程或函数最好在名称前加上表示此功能动作的动词为前缀,同时加入此过程或函数的功能描述(方便阅读)

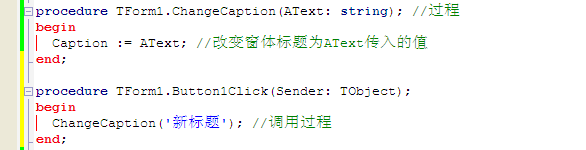
#### **3.4.** 10**.1过程**

过程名应当以大写字母开始，且大小写交错以增加可读性

结构如下所示：

procedure 过程名( 参数;参数..);

如:

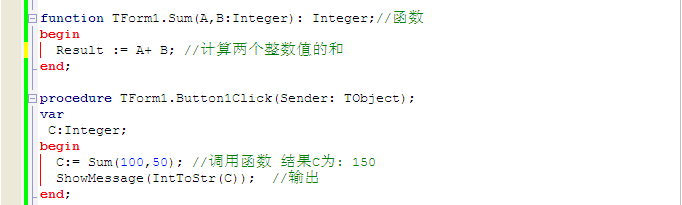


#### **3.4.** 10**.2函数**

结构如下所示：

function 函数名( 参数;参数..):返回值类型;

如:



过程和函数就是用来完成特定功能的一些代码组成的程序块。过程和函数的惟一区别就是：函数有返回值而过程没有。

#### **3.4.9.3作用域**

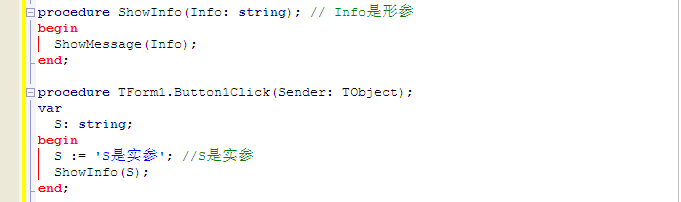
一个Unit（单元）中，声明于interface部分（即interface和implementation关键字之间的过程(函数)称为全局过程(函数)。另一个单元uses（引用）这个单元后，可以调用这些全局过程(函数)。

变量的作用域也是类似的，如果定义在interface部分，那么是全局的（典型的如：var Form1: TForm1），否则是局部的，只能在本单元使用。

#### **3.4.9.4参数传递**

声明/实现一个过程使用的参数称为形式参数（简称形参），调用过程时传入的参数称为实际参数（简称实参）。

如:



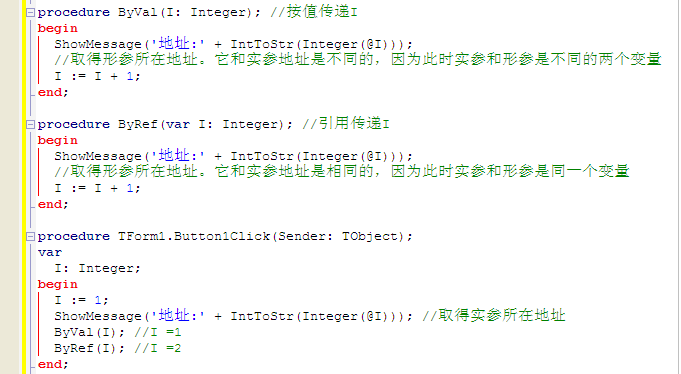
参数传递分两种：按值传递和引用传递。这两种方式的本质区别是：

按值传递时，形参和实参是两个变量，它们开始时的值是相同的，即实参的数据被拷贝一份传递给了形参。所以此时，形参的改变不会影响到实参。

引用传递时，形参和实参是同一个变量，可以将它们之一看做是另一个的别名。所以此时，形参改变时，实参跟着改变。

默认情况下，参数是按值传递的，传递的是数据拷贝；如果加了var前缀，则成了引用传递。

如：



按值传递的参数可以指定默认值，比如上面的ByVal可以是这样：

procedureByVal(I: Integer = 0);

调用它时可以省掉有默认值的参数：ByVal。

带默认值的参数必须位于参数列表的最后，如：

procedureByVal(I: Integer = 0; B: Boolean);

是不行的，应该改为：

procedureByVal(B: Boolean; I: Integer = 0);

因为默认值必须是一个常数表达式，所以dynamic-array、procedural、class、class-reference和interface等参数只能指定nil默认值；而record、variant、file和static-array等类型的参数则不能指定默认值。

按值传递时，形参和实参是两个变量；引用传递时，形参和实参是同一个变量。

# 四、**常用控件**

## 4.1 窗体

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Active | 确定窗体是否是活动的 |
| ActiveControl | 确定获得焦点的控件 |
| ActiveMDIChild | 确定在MDI应用程序中活动的子窗体 |
| ArrangeIcons | 排列在MDI应用程序中子窗体最小化图标 |
| AutoScroll | 确定滚动条是否自动显示在窗体边框内 |
| BorderIcons | 确定显示在标题栏中的按钮 |
| Canvas | 访问窗体的画布 |
| Caption | 确定窗体标题栏中的标题文本 |
| Cascade | 层叠MDI应用程序中的子窗体 |
| ClientHandle | 获取窗体客户区的句柄 |
| ClientHeight | 确定窗体客户区的高度 |
| ClientWidth | 确定窗体客户区的宽度 |
| Close | 关闭窗体 |
| Color | 确定窗体的颜色 |
| Create | 创建新窗体 |
| Ctl3D | 确定窗体是否用三维样式显示 |
| DefocusControl | 指定失去焦点的控件 |
| Destroy | 撤销窗体 |
| Enable | 确定窗体是否被激活（可操作） |
| FocusControl | 指定获得焦点的控件 |
| Font | 确定在窗体上绘制的文字的字体类型 |
| FormStyle | 确定窗体样式 |
| Height | 确定窗体高度 |
| Hide | 隐藏窗体 |
| HorzScrollBar | 窗体的水平滚动条 |
| Icon | 窗体图标 |
| KeyPreview | 确定窗体是否接收其所含控件的键盘事件 |
| MDIChildCount | 确定在MDI应用程序中打开的子窗体的数目 |
| MDIChildren | 访问在MDI应用程序中的子窗体 |
| Menu | 窗体的主菜单 |
| Next | 激活MDI应用程序中的下一子窗体 |
| OnActivate | 处理由窗体被激活所引起的事件 |
| OnClick | 处理由单击鼠标所引起的事件 |
| OnClose | 处理由关闭窗体所引起的事件 |
| OnCeate | 处理由窗体被创建所引起的事件 |
| OnDbClick | 处理由双击鼠标所引起的事件 |
| OnDeactivate | 处理由窗体被释放所引起的事件 |
| OnDestroy | 处理由窗体被撤销所引起的事件 |
| OnDragDrop | 处理由在窗体内拖放一控件所引起的事件 |
| OnDragOver | 处理由在窗体内拖动一控件所引起的事件 |
| OnHide | 处理由窗体被隐藏所引起的事件 |
| OnKeyDown | 处理键盘按键被按下不立即释放的事件 |
| OnKeyPress | 处理键盘按键被按下的事件 |
| OnKeyUp | 处理键盘按键按下后被释放的事件 |
| OnMouseDown | 处理鼠标按钮被按下不立即释放的事件 |
| OnMouseMove | 处理鼠标指针在窗体上移动的事件 |
| OnMouseUp | 处理鼠标按钮释放的事件 |
| OnPaint | 处理绘制窗体画布的事件 |
| OnResize | 处理由窗体大小改变引起的事件 |
| OnShow | 处理由窗体获得焦点引起的事件 |
| PopupMenu | 窗体弹出式菜单（快捷菜单） |
| Position | 确定窗体初始显示的位置和大小 |
| Previous | 使MDI应用程序中的前一子窗体得到焦点 |
| Print | 打印窗体图像 |
| PrintScale | 确定窗体被打印时是怎样绘制的 |
| Release | 处理完未决事件后撤销窗体 |
| SetFocus | 设置焦点 |
| Show | 显示窗体 |
| ShowHint | 确定是否显示工具提示 |
| ShowModal | 显示有模式窗体 |
| Tile | 平铺MDI应用程序中的子窗体 |
| VertScrollBar | 窗体的垂直滚动条 |
| Visible | 确定窗体是否可见 |
| Width | 确定窗体的宽度 |
| WindowMenu | 确定哪个菜单项是在MDI应用程序中的窗体菜单项 |
| WindowState | 确定窗体初始装入时是怎样显示的 |

## 4.2 标签控件(TLabel)

标签（Label）一般放在其他控件旁边，用来标记这些控件，从而对用户的操作进行提示，也可以用来

显示其它信息。

## 4.3 文本框控件(TEdit)

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| AutoSelect | 当编辑框获得输入焦点时，编辑框中的文字是否自动被选中 |
| AutoSize | 是否随字体变化而自动变动尺寸 |
| MaxLength | 最多可以输入的字符数，缺省时为0，表示可以输入任意多的字符 |
| OnChange | 由编辑框内容变化而引起的事件 |
| OnEnter | 由编辑框获取焦点而引起的事件 |
| OnKeyPress | 由编辑框中有按键按下而引起的事件 |
| PasswordChar | 密码掩码属性，为#0，没有掩码。为\*，则以\*掩码 |
| ReadOnly | 编辑框内容是否可以编辑 |
| Text | 显示和保存编辑框中的字符串 |
| Visible | 是否可见 |

## 4.4 Memo控件(TMEMO)

Memo 控件是编辑框的扩展，它也是应用程序最常用的控件之一，它的作用主要是让用户输入编辑多行文字。

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Font | 设定Memo中文字的字体，包括一系列子属性 |
| Lines | 设定Memo中文字的内容 |
| ReadOnly | 控制Memo中文字的内容是否只读 |
| ScrollBars | 设定Memo框的上下左右滚动条 |
| WantReturns | 控制Memo中文字的内容是否允许用户插入回车符号 |
| WordWrap | 控制Memo中文字的内容是否自动插入软回车换行 |
| SelLength | 设定Memo中被选中文字的长度 |
| SelStart | 设定Memo中被选中文字的开始位置 |
| SelText | 设置Memo中被选中文字内容 |
| Clear | 清除Memo中内容 |
| OnChange | 由改变Memo中内容而引起的事件 |

## 4.5 按钮控件(TButton,TBitBtn)

### 4.5.1 TButton控件

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Cancel | 按钮是否支持【ESC】键 |
| Caption | 按钮的标题 |
| Cursor | 按钮的鼠标 |
| Default | 按钮是否支持【Enter】键 |
| Enabled | 按钮是否被激活 |
| Hint | 按钮的提示信息 |
| ShowHint | 是否显示提示信息 |
| Name | 设置按钮的名字 |
| OnClick | 处理由单击鼠标按钮触发的事件 |

### 4.5.2 TBitBtn控件

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Cancel | 是否支持【ESC】键 |
| Caption | 标题 |
| Cursor | 鼠标 |
| Default | 是否支持【Enter】键 |
| Enabled | 是否被激活 |
| Glyph | 显示在按钮上的位图 |
| Kind | 按钮的类型 |
| Action | 动作 |
| Caption | 按钮的标题 |
| Cursor | 按钮的鼠标 |
| Enabled | 按钮是否被激活 |
| Glyph | 显示在按钮上的位图 |
| Name | 按钮的名字 |
| OnClick | 处理由单击鼠标按钮触发的事件 |
| Transparent | 按钮是否透明 |

## 4.6 选项按钮(TCheckBox,TRadioButton)

### 4.6.1 TCheckBox(复选框)

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Alignment | 设置文字和方框的位置关系 |
| Checked | 按钮是否被选中 |
| AllowGrayed | 按钮是否支持灰态 |
| Enabled | 按钮是否被激活 |
| State | 按钮状态 |

### 4.6.2 TRadioButton(单选框)

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Alignment | 设置文字和圆框的位置关系 |
| Checked | 确认按钮是否被选中 |
| OnClick | 处理按钮被单击所触发的事件 |

## 4.7 列表框(TListBox)

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Clear | 清楚列表框内容 |
| Columns | 设置列表框的列数 |
| Enabled | 确定列表框是否被激活 |
| ExtendedSelect | 确定是否可以使用Shift键、Ctrl键进行选择 |
| Items | 访问列表框中显示的字符串列表 |
| MultiSelect | 确定是否可以从列表框中选择多个项目 |
| OnClick | 处理单击鼠标所触发的事件 |
| Selected | 确定指定项是否被选中 |
| Sorted | 确定列表框中各项是否按字母顺序排列 |

## 4.8 组合框(TComboBox)

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| ItemHeight | 设置组合框的下拉高度 |
| Items | 访问组合框中显示的字符串列表 |
| MaxLength | 设置用户在组合框的编辑框中最多可输入的字符数 |
| Text | 设置或返回组合框上显示出来的文字内容 |
| Sorted | 确定组合框中各项是否按字母顺序排列 |
| Clear | 清楚组合框中内容 |
| OnChange | 处理由改变组合框内容所触发的事件 |

## 4.9 Panel 控件(TPanel)

TPanel是容器型的控件。

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Align | 设置在表单中的对齐方式 |
| Alignment | 设置标题的排列方式 |
| BevelInner | 设置内部斜面样式 |
| BevelOuter | 设置外部斜面样式 |
| BevelWidth | 设置斜面宽度 |
| BorderStyle | 设置边框样式 |
| OnResize | 处理由改变尺寸所触发的事件 |

## 4.10 对话框控件

### 4.10.1OpenDialog控件

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Title | 设置对话框标题 |
| InitialDir | 设置对话框初始目录 |
| FilterIndex | 设置对话框的过滤器索引 |
| Filter | 设置对话框的文件类型过滤器 |
| Options | 设置对话框的其他属性，控制对话框文件属性 |
| FileName | 设置对话框文件名 |

### 4.10.2 SaveDialog控件

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Title | 设置对话框标题 |
| InitialDir | 设置对话框初始目录 |
| FilterIndex | 设置对话框的过滤器索引 |
| Filter | 设置对话框的文件类型过滤器 |
| Options | 设置对话框的其他属性，控制对话框文件属性 |
| FileName | 设置对话框文件名 |
| Files | 多选文件名 |

### 4.10.3FontDialog控件

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Device | 设置字体显示设备 |
| Font | 设置字体 |
| Options | 设置字体的扩展属性 |

### 4.10.4 ColorDialog控件

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Color | 设置颜色 |
| Options | 设置颜色的扩展属性 |

### 4.10.5 PrintDialog控件

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| Collate | 设置对话框中分页选项 |
| Copies | 设置对话框中份数 |
| FromPage、ToPage | 设置打印范围 |
| Options | 设置对话框的扩展属性 |
| FindText | 设置所需查找的字符串 |
| Options | 设置对话框的扩展属性 |

### 4.10.6 ReplaceDialog控件

常用的属性、方法和事件

|  |  |
| --- | --- |
| 属性、方法和事件 | 功能解释 |
| FindText | 设置所需查找的字符串 |
| ReplaceText | 设置所需替换的字符串 |
| Options | 设置对话框的扩展属性 |

## 4.11 数据集(DataSet)

### 4.11.1 TADODataSet控件

TADODataSet是一个基于ADO Recordset的继承于DataSet的数据集对象。

ADO Recordset基于Windows标准数据组件，可容纳实现Ole驱动的任何数据库的表数据，也可用作于本地内存表。

常用的属性、方法(作内存表使用时)

|  |  |
| --- | --- |
| 属性、方法 | 功能解释 |
| FieldDefs | 字段定义列表属性 |
| Open | 打开和激活数据集控件，从而进行数据编辑 |
| First | 到首 |
| Prior | 向前 |
| Next | 向后 |
| Last | 到尾 |
| Edit | 编辑 |
| Cancel | 取消编辑 |
| Post | 保存 |
| Insert | 插入 |
| Append | 添加 |
| Delete | 删除 |
| Refresh | 刷新 |
| Filter,Filtered | 用于筛选指定条件的记录 |
| CanModify | 用于判断数据集中的数据是否可以修改 |

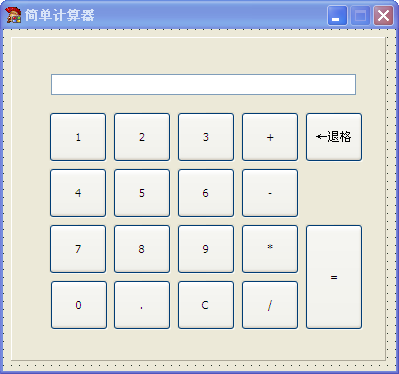
# 五、**例子**

## 5.1 简单计算器

1、启动Delphi，新建Delphi工程（File→New→VCL Forms Application），保存全部（File→Save All）为Unit1.pas和Project1.dpr；

2、选择Form1，在属性查看器中修改标题（caption）属性为【简单器】；

3、在窗体上拖放控件，修改它们的标题（caption）属性及位置,效果如下图



# 六、附录

## 6.1 Delphi 快捷键

|  |  |
| --- | --- |
| 快捷键 | 功能 |
| Alt+0（阿拉伯数字） | 窗口列表 |
| Alt+Backspace | 恢复 |
| Alt+F10 | 显示某对象的快捷菜单 |
| Alt+Shift+Backspace | 重做 |
| Backspace | 后退 |
| Ctrl+0 | 在代码编辑框中定位到书签0 |
| Ctrl+1 | 在代码编辑框中定位到书签1 |
| Ctrl+2 | 在代码编辑框中定位到书签2 |
| Ctrl+3 | 在代码编辑框中定位到书签3 |
| Ctrl+4 | 在代码编辑框中定位到书签4 |
| Ctrl+5 | 在代码编辑框中定位到书签5 |
| Ctrl+6 | 在代码编辑框中定位到书签6 |
| Ctrl+7 | 在代码编辑框中定位到书签7 |
| Ctrl+8 | 在代码编辑框中定位到书签8 |
| Ctrl+9 | 在代码编辑框中定位到书签9 |
| Ctrl+Alt+F11 | 显示工程管理器 |
| Ctrl+C | 复制 |
| Ctrl+Down | 在表单上向下移动当前控件 |
| Ctrl+E | 跳到代码编辑框编辑断点 |
| Ctrl+End | 移动到文末 |
| Ctrl+Enter | 在当前光标中打开文件 |
| Ctrl+Enter | 在对象查看器中编辑对象属性 |
| Ctrl+F | 显示查找窗口 |
| Ctrl+F12 | 查看单元文件 |
| Ctrl+F2 | 程序重置（中止程序运行并初始化） |
| Ctrl+F3 | 显示堆栈窗口 |
| Ctrl+F4 | 关闭当前文件（在代码编辑框中） |
| Ctrl+F5 | 增加观察值 |
| Ctrl+F7 | 求值/修改窗口 |
| Ctrl+F9 | 编译 |
| Ctrl+Home | 移动到文首 |
| Ctrl+I | 在代码编辑框中插入一个制表符 |
| Ctrl+J | 显示代码模板窗口 |
| Ctrl+K+0 | 在代码编辑框中设定/取消书签0 |
| Ctrl+K+1 | 在代码编辑框中设定/取消书签1 |
| Ctrl+K+2 | 在代码编辑框中设定/取消书签2 |
| Ctrl+K+3 | 在代码编辑框中设定/取消书签3 |
| Ctrl+K+4 | 在代码编辑框中设定/取消书签4 |
| Ctrl+K+5 | 在代码编辑框中设定/取消书签5 |
| Ctrl+K+6 | 在代码编辑框中设定/取消书签6 |
| Ctrl+K+7 | 在代码编辑框中设定/取消书签7 |
| Ctrl+K+8 | 在代码编辑框中设定/取消书签8 |
| Ctrl+K+9 | 在代码编辑框中设定/取消书签9 |
| Ctrl+K+B | 在代码编辑框中标记块开始 |
| Ctrl+K+C | 在代码编辑框中拷贝块 |
| Ctrl+K+H | 在代码编辑框中隐藏/显示选定块 |
| Ctrl+K+I | 在代码编辑框中把选定块按设定的缩进值进行右缩进 |
| Ctrl+K+K | 在代码编辑框中标记块结尾 |
| Ctrl+K+L | 在代码编辑框中标记本行为块 |
| Ctrl+K+N | 在代码编辑框中把块转化为大写 |
| Ctrl+K+O | 在代码编辑框中把块转化为小写 |
| Ctrl+K+R | 在代码编辑框中以块的方式读入文件 |
| Ctrl+K+T | 在代码编辑框中标记本词为块 |
| Ctrl+K+U | 在代码编辑框中把选定块按设定的缩进值进行左缩进 |
| Ctrl+K+V | 在代码编辑框中移动选定块 |
| Ctrl+K+W | 在代码编辑框中把选定块写入文件 |
| Ctrl+K+Y | 在代码编辑框中删除选定块 |
| Ctrl+L | 再次查找 |
| Ctrl+Left | 在表单上向左移动当前控件 |
| Ctrl+Left Arrow | 向左移一个词 |
| Ctrl+N | 在代码编辑器中插入新行 |
| Ctrl+O+C | 在代码编辑框中设定一个列块 |
| Ctrl+O+L | 在代码编辑框中设定一个行块 |
| Ctrl+Q+0 | 在代码编辑框中定位到书签0 |
| Ctrl+Q+1 | 在代码编辑框中定位到书签1 |
| Ctrl+Q+2 | 在代码编辑框中定位到书签2 |
| Ctrl+Q+3 | 在代码编辑框中定位到书签3 |
| Ctrl+Q+4 | 在代码编辑框中定位到书签4 |
| Ctrl+Q+5 | 在代码编辑框中定位到书签5 |
| Ctrl+Q+6 | 在代码编辑框中定位到书签6 |
| Ctrl+Q+7 | 在代码编辑框中定位到书签7 |
| Ctrl+Q+8 | 在代码编辑框中定位到书签8 |
| Ctrl+Q+9 | 在代码编辑框中定位到书签9 |
| Ctrl+Q+B | 在代码编辑框中移动到块开头 |
| Ctrl+Q+K | 在代码编辑框中移动到块结尾 |
| Ctrl+R | 显示替换窗口 |
| Ctrl+Righ | 在表单上向右移动当前控件 |
| Ctrl+Right Arrow | 向右移一个词 |
| Ctrl+Shift+C | 通过函数或过程的声明生成其代码编辑框架 |
| Ctrl+Shift+F11 | 显示工程配置窗口 |
| Ctrl+Space | 显示代码自动完成窗口（与汉字输入法的快捷键冲突） |
| Ctrl+Tab | 在代码编辑框的各页中切换 |
| Ctrl+UP | 在表单上向上移动当前控件 |
| Ctrl+V | 粘贴 |
| Ctrl+X | 剪切 |
| Del | 删除 |
| Down Arrow | 光标下移 |
| End | 在代码编辑器中光标移动到行末 |
| F1 | 上下文敏感帮助 |
| F11 | 显示对象查看器 |
| F12 | 在表单和代码编辑框间切换 |
| F3 | 再次查找 |
| F4 | 运行至光标处 |
| F5 | 设置/取消断点 |
| F7 | 进入式运行（进入子函数） |
| F8 | 跨越式运行（跳过子函数） |
| F9 | 运行（编译并运行） |
| Home | 在代码编辑器中光标移动到行首 |
| Ins | 打开/关闭插入模式 |
| Left Arrow | 光标左移 |
| Page Down | 向下翻页 |
| Page Up | 向上翻页 |
| Right Arrow | 光标右移 |
| Shift+Ctrl+0 | 在代码编辑框中设定/取消书签0 |
| Shift+Ctrl+1 | 在代码编辑框中设定/取消书签1 |
| Shift+Ctrl+2 | 在代码编辑框中设定/取消书签2 |
| Shift+Ctrl+3 | 在代码编辑框中设定/取消书签3 |
| Shift+Ctrl+4 | 在代码编辑框中设定/取消书签4 |
| Shift+Ctrl+5 | 在代码编辑框中设定/取消书签5 |
| Shift+Ctrl+6 | 在代码编辑框中设定/取消书签6 |
| Shift+Ctrl+7 | 在代码编辑框中设定/取消书签7 |
| Shift+Ctrl+8 | 在代码编辑框中设定/取消书签8 |
| Shift+Ctrl+9 | 在代码编辑框中设定/取消书签9 |
| Shift+Down | 在表单上增加当前控件的高度 |
| Shift+Left | 在表单上减小当前控件的宽度 |
| Shift+Righ | 在表单上增加当前控件的宽度 |
| Shift+Space | 插入空格 |
| Shift+UP | 在表单上减小当前控件的高度 |
| Space | 插入空格 |
| Tab | 插入制表符 |
| Up Arrow | 光标上移 |