第3章 Object Pascal 编程基础

Delphi 是一个基于 Object Pascal 语言的 Windows 应用程序开发工具，要想掌握如何使用 Delphi 进行程序的开发，首先就必须掌握 Object Pascal 语言的语法基础。Object Pascal 是一种面向对象的程序设计语言，它是在 Pascal 语言的基础上发展起来的，既继承了 Pascal 语言语法严谨、数据结构丰富等优点，同时融入了面向对象编程的语法要素。

本章主要介绍 Object Pascal 的语法基础，例如什么是标识符、如何定义常量和变量、常用的数据类型、运算符和内部函数都有哪些，等等。语法是编程的基础，所以一定要深刻理解并掌握本章中的一些语法定义和规则，为以后程序代码的编写奠定良好的基础。

学习任务

☑ 了解 Object Pascal 中的保留字、标识符与注释语句。

※ 掌握 Object Pascal 中 3 种基本的数据类型。

☑ 了解数据类型之间的转换方式。

※ 掌握常量与变量的定义方法。

※ 掌握算术运算符与字符串运算符。

※ 掌握常用的内部函数。

☑ 牢记 Object Pascal 代码书写规则。

3.1 保留字与标识符

3.1.1 保留字

保留字是由系统规定的、具有特定意义的单词，在编程时不能被重新定义或用作其他用途。Object Pascal 语言中共规定了 65 个保留字，如表 3-1 所示。例如 begin、end、if、for 等都是保留字，它们都有各自的意义和作用，在应用程序中不允许对这些保留字进行修改。

表 3-1 Object Pascal 的保留字

保留字	保留字	保留字	保留字	保留字	保留字
and	array	as	const	begin	destructor
uses	var	while	with	end	constructor
do	downto	class	else	case	procedure

续表

保留字	保留字	保留字	保留字	保留字	保留字
not	object	of	or	out	resourcestring
string	then	until	for	try	initialization
to	goto	if	is	type	inherited
except	program	property	record	repeat	threadvar
set	shl	shr	unit	interface	dispinterface
exports	file	in	finally	function	implementation
label	library	mod	nil	raise	finalization
asm	xor	div	inline	packed	

注意：Object Pascal是不区分大小写的。因此，像And、AND、and都被看作是相同的保留字。在Delphi集成开发环境的代码编辑窗口中，保留字显示为黑体。

3.1.2 标识符

标识符是用来表示常量、变量、过程、函数、对象等语法实体的名称符号，它分为系统预定义的标准标识符和用户自定义标识符。

1. 系统预定义标识符

即标准标识符。它是Object Pascal语言系统预先配给标准函数、标准过程、标准常量以及标准文件等使用的标识符。例如标准的常量True和False、标准类型Integer和Real、标准函数Abs和Sqr等都是标准标识符。这些系统预定义的标识符在Object Pascal中都具有特定的含义，用户不能更改。

2. 自定义标识符

自定义标识符是程序员根据程序设计的需要自定义的标识符，可用来表示常量、变量、函数、过程等的名字。自定义的标识符命名规则如下：

(1) 必须由字母、数字、下画线组成，不能包含空格。

(2) 必须以字母或下画线开始，即首字符不能是数字。

(3) 不能与保留字和标准标识符相同。

注意：

(1) 自定义标识符须先定义或说明后，才能在程序中使用。例如，变量要先声明后使用。

(2) 自定义标识符长度不要超过255，否则只有前255个字符有效。

(3) 在Object Pascal中，标识符也不区分大小写。因此，标识符MyName和myname被认为是同一个标识符。

小贴示：一般情况下，使用有意义的英文单词来为变量、过程或函数等命名，以增强程序代码的可读性。

3. 标识符的限定

若要在当前程序中使用在其他地方定义的标识符,则需要使用限定标识符。

限定标识符的语法格式为:

```
标识符 1.标识符 2;
```

其中,标识符 1 限定标识符 2。例如,要修改 button1 对象上显示的文本,则在引用 caption 属性时需要指出 button1 标识符,如 button1.caption := '确定'。此外,限定标识符可以嵌套。例如,Form2.Edit1.Text 表示调用窗体 Form2 中 Edit1 对象的 Text 属性。

3.1.3 注释

在程序中使用注释是一个良好的编程习惯,注释能够增强程序的可读性和可维护性。因此,编程过程中常使用注释来对语句、代码块、命令等加以说明。注释内容仅供阅读程序时参考使用,它不会被编译器编译,不影响程序的执行与结果。此外,注释在程序调试过程中也有很大的作用。使用注释可以对部分代码作暂时的"删除",以缩小调试范围。待调试完毕之后,删除注释符号即可恢复被注释掉的代码块。

Object Pascal 中的注释有以下 3 种形式:

```
//注释内容  或  {注释内容}  或  (*注释内容*)
```

1. 行注释——//注释内容

在程序中,可以使用双斜杠"//"对该符号之后的内容进行注释。这种注释从"//"到本行结束有效,因此称为行注释。其语法格式为://注释内容。

行注释表示所在行的//符号后的内容为注释,且注释内容只能写在一行上。

例如:button1.text := '确定'; //将 button1 上的文本修改为确定

2. 块注释——{注释内容}

在程序中,可以使用一对花括号{}进行注释,{}中的内容在程序编译过程中不做任何处理。{}中的内容可以是单行文字也可以是多行文字,因此称为块注释。其语法格式为:{注释内容}。

例如:{接下来要声明一个函数,用来计算 n 的阶乘!}

3. 块注释——(*注释内容*)

在程序中,可以使用圆括号和星号对包含在其中的文字或代码进行注释,其语法格式为:(*注释内容*)。

例如:(*接下来要声明一个函数,用来计算 n 的阶乘! *)

用户在编写代码的过程中,可以灵活采用多种注释形式,但要避免注释嵌套。下面代码段分别使用了注释的三种形式。

```
with edit1 do
```

```
{ 使用 with 语句为 edit1 对象的多个属性赋值。
其中,文本框中内容设置为 Hello,文本框中字体大小设为 24 磅,字体颜色设置为红色。}
// with 复合语句开始,使用赋值语句为对象属性赋值
begin
  text := 'hello! ';        //设置文本内容
  Font.Size := 24; {设置字号}
  Font.Color := clRed; ( * 设置字体的颜色 * )
end;
```

小贴示：行注释语句可以写在该行代码的后面,也可以另起一行单独写。行注释从符号//开始,到该行结束部分的内容都将被视为注释。块注释内容可以写在一行上,也可以写在多行上。

注意：注释不能嵌套。两种块注释语句可以相互替换,但要注意块注释符号必须成对出现,且圆括号和星号之间不能有空格。

小贴示：建议对于单行或少量几行注释使用符号//,对于大块注释使用{与}或(* 与 *)。

3.2 常用的数据类型

用户输入到计算机并被加工处理的符号的集合就是数据。数据的一个重要特征就是其所属的数据类型。数据的类型不仅确定了该类数据的表示形式和取值范围,而且还确定了该类数据所能参加的运算种类。

Object Pascal 提供了多种数据类型。常用的标准数据类型有：整型 Integer、实数型 Real、字符类型 Char、字符串类型 String、布尔类型 Boolean。标准数据类型属于 Object Pascal 内部约定的数据类型,无须定义就可以直接使用。另外还有一些数据类型,如数组类型 Array 等是一些高级的数据类型。高级数据类型体现了特殊的数据结构,使用前必须由用户自己定义,在后面的章节中会另作介绍。

3.2.1 数值型数据

数值型数据包括整型数据和实数型数据。

1. 整型数据

整型数据根据所表达的数的范围的不同又可分为：短整型 Shortint(8 位有符号整数)、整型 Integer(32 位有符号整数)、长整型 Longint(32 位有符号整数)以及字节型 Byte(8 位无符号整数)。在应用程序中,使用最为广泛的就是一般整型 Interger。

例如：var i:integer; i :=1; //声明一个 integer 型变量 i,并为其赋初值 1。

2. 实数型数据

实数型数据根据所表达的数的范围以及精度的不同可分为：单精度 Single(4 字节浮点数)、实数型 Real(6 字节浮点数)和双精度 Double(8 字节浮点数)。在实际应用中,使用较为广泛的是 Single 和 Real 型浮点数。

例如：var r,s:real; //声明两个 Real 型变量 r 和 s。

3.2.2 字符型数据

字符型数据包括单个字符类型 Char 和字符串类型 String。

1. 字符型 Char

字符型 Char 用来存储单个字符数据,用一对单引号括起来。例如,'a'、'?'、'2'等。

例如:var chr1:char; chr1='a'; //声明一个 Char 型变量 chr1,并为其赋初值。

小贴示:

(1) 系统函数 Chr( ):返回一个整型数据对应的字符,该整数就是该字符对应的 ASCII 码。例如,chr(13)返回回车键,chr(65)返回字符'A'。

(2) 系统函数 Ord( ):返回一个字符对应的序数即 ASCII 码。例如 ord('a')返回值为 97。

常用的特殊字符有:#9 或 chr(9)表示跳格键 Tab、#10 或 chr(10)表示换行键、#13 或 chr(13)表示回车键 Enter。

2. 字符串类型 String

字符串类型 String 用来存储一串字符,其中可包含 0 至多个字符,长度限制为 255。在 Delphi 中支持三种类型的字符串:短字符串 ShortString、长字符串 AnsiString 和宽字符串 WideString。一般情况下,当声明一个变量为字符串类型时,经常使用 String 保留字。默认情况下,String 类型就是 AnsiString 类型。

例如:var s1:string; s1=edit1.text; {将 edit1 中的文本内容存储于 String 型变量 s1 中}。

Object Pascal 提供了一些操作字符串的标准函数,如 Length( )函数可用来计算字符串的长度、Concat( )函数可以合并多个字符串、Insert( )函数可以将一个字符串插入到另一个字符串中等,这些函数在 3.6 节会详细介绍。

3. 字符串常量

在 Object Pascal 中,字符串常量是由一对单引号括起来的一串字符,其中可以包含 0～255 个字符。例如'a'、'Hello'、'123'等。

在 Delphi 中,许多对象的属性就是一个字符串类型的数据。例如,Button1.Caption :='确定'; Edit1.Text :='Admin'。又如,两个连续的单引号''表示一个空的字符串,其长度为 0。一对单引号中间添加一个空格' ',表示包含一个空格的字符串,其长度为 1。

此外,在字符串常量中允许使用嵌入控制符。所谓嵌入控制符就是由符号 # 和紧跟其后的 0～255 之间的整数构成的,该整数是控制字符对应的 ASCII 码值。例如,#9 等价于 chr(9),表示 Tab 键。#13 和 #10 分别表示回车符和换行符。字符串之间的连接用符号“+”,所以'object'+#13+#10+'pascal'等价于'object'+chr(13)+chr(10)+'pascal',表达的意思是字符串 object 显示完毕,换行在下一行显示字符串 pascal。

3.2.3 布尔型数据

布尔型数据 Boolean 用来表示关系运算或逻辑运算的结果，其取值由系统预定义的两个常量标识符 True 和 False 表示。适用于布尔类型的运算符有：

(1) 逻辑运算符：And(逻辑与)、Or(逻辑或)、Not(逻辑取反)

(2) 关系运算符：=(等于)、<>(不等于)、<(小于)、>(大于)、<=(小于等于)、>=(大于等于)

在 Delphi 中，组件的许多属性就是一个布尔类型的数据。如 Visible 属性、Readonly 属性、Enabled 属性等等，这些属性的取值都只能是 True 或者 False。

3.3 数据类型的转换

Object Pascal 对数据类型的要求非常严格，所有的变量都必须具有类型。即在声明一个变量时，须指出该变量所属的数据类型。

此外，Object Pascal 中赋值语句的语法格式为：

```
目标操作数 := 源操作数;
```

在一个赋值语句中，赋值符号左右两边的数据类型必须是兼容的。若源操作数和目标操作数的数据类型不一致，则在赋值之前需要进行数据类型的转换。

3.3.1 类型转换方式

在 Object Pascal 中，不同的数据类型之间是可以转换的。数据类型的转换方式大体可分为两种：强制类型转换和函数类型转换。

1. 强制类型转换

所谓强制类型转换是指将源操作数的数据类型强制转换为目标操作数的数据类型，但此种转换方法可能会丢失数据或者出现异常。

例如，将整型数据 10 强制转换为实数型时，系统采用强制类型转换方式将整型数据 10 强制转换为实型数据 10.0，不影响结果的正确性。但是反过来，如果将实型数据 10.1 赋值给一个整型变量时，若系统采用强制类型转换方式，则不得不舍掉小数位，会影响结果的精确度。

小贴示：

(1) 扩展转换：目标操作数的数据范围大于源操作数的数据范围，不影响结果的正确度。

(2) 收缩转换：目标操作数的数据范围小于源操作数的数据范围，可能会影响结果的精确度和准确度。

所以，建议大家在赋值时，一定要保证左右两边的数据类型匹配。如果不匹配，则必须使用函数类型转换，以保证结果的正确性。

2. 函数类型转换

所谓函数类型转换是指使用一定的转换函数将源操作数的数据类型转换为目标操作数的数据类型。

例如,本书例 1.1 中的代码 label1.Caption :='现在时间: '+DateTimetoStr(Now)。其中的 DateTimetoStr()就是一个转换函数,其作用是将日期时间型的数据 Now 转换成字符串型数据,然后与其他字符串进行连接,之后才可以赋值给 Label1 对象的 Caption 属性。因为,源操作数 Now 是一个日期型数据,而目标操作数 Caption 属性是一个 String 型数据,因而要使用函数类型转换以避免程序出现数据类型不匹配的异常情况。

3.3.2 常用的类型转换函数

常用的类型转换函数有:

(1) IntToStr:将整型数据转换为字符串类型的数据。

例如:IntToStr (10)返回值为'10'。

(2) StrToInt:将由纯数字组成的字符串转换为整数。

例如:StrToInt ('1024')返回值为 1024。

注意:使用此函数有个前提条件,即源字符串必须全部由数字组成,且去掉单引号之后必须是一个整数,否则使用该函数进行转换时将出现异常。

(3) FloatToStr:将实型数据转换为字符串类型的数据。

例如:FloatToStr (3.14)返回值为'3.14'。

(4) StrToFloat:将由数字和小数点组成的字符串转换为实数。

例如:StrToFloat('3.14')返回值为 3.14。

注意:使用此函数有个前提条件,即源字符串去掉单引号之后必须是一个实数,或可以转换为实数,否则将会出现转换异常。

(5) DateTimeToStr:将日期时间型的数据转换为字符串类型。

例如:在标签 Label1 上显示系统当前时间,代码为:Label1.Caption :=DateTimetoStr (Now)。

小贴示:Now 为系统预定义的函数,作用是返回当前的时间,其返回值类型是一个日期时间型的数据。Label1 对象的 Caption 属性是一个字符串类型的数据。所以,在赋值之前一定要进行数据类型的转换,使用转换函数 DateTimeToStr()将日期时间型的数据转换为字符串类型的数据。

3.4 常量和变量

3.4.1 常量

常量是指在程序的执行过程中其值不变的数据,可分为直接常量和符号常量。

1. 直接常量

直接常量是指在程序中直接引用的常数，包括：

(1) 整型常量：例如，2017、－10、0 等。

(2) 实型常量：包括定点数和浮点数。定点数即小数表示形式，例如 3.14、－10.5、7.0 等。浮点数即科学计数法表示形式，例如＋2.7E＋5、－6.8E－9 等。此外，Object Pascal 系统预定义了一个实数型常量 PI 用来表示圆周率，使用该常量时无须再定义。

(3) 字符型常量：由一对单引号括起来的单个字符。例如'a'、'?'、'8'等。

(4) 字符串常量：由一对单引号括起来的 1 个或多个字符。例如'Hello World!'、'欢迎大家使用 Delphi 7.0!'等。

(5) 布尔型常量：表示逻辑判断的结果。Object Pascal 中有两个布尔型常量，分别为 True 和 False。

(6) 常量表达式：常量也可以是一个计算结果，用表达式来表达。例如(2.5＋1)/(2.5－1)、'Borland'＋chr(13)＋chr(10)＋'Delphi'、Ord('z')－Ord('a')等。

2. 符号常量

符号常量是用户根据程序所需自定义的常量。声明符号常量时不必指出其数据类型，但需要给该常量赋一个值。

声明常量的语法格式为：

```
const 常量名 = 常量值;
```

其中，const 是保留字，用来表明符号常量声明开始；常量名要符合标识符的命名规则。

例如，声明两个常量分别用来存放最小值和最大值，则代码为：

```
const Min = 0; Max = 100;
```

注意：

(1) 为常量赋值时，符号为＝，要注意区别赋值符号 :＝。

(2) 常量一经声明，其值在程序运行中不能再更改，即常量的值是只读的。

(3) 常量的数据类型就是为其所赋数值的数据类型。例如，上例中 Max 和 Min 两个常量都是整型。

(4) 每一条常量声明语句后都有一个分号。可使用 const 保留字同时声明多个常量，多个常量的声明之间用分号隔开。

3.4.2 变量

1. 变量的定义

变量是指在程序的执行过程中其值发生变化的数据。应用程序中的变量必须先声明后使用。声明变量的语法格式为：

```
var
变量名 1: 类型名 1;
```

```
⋮
变量名 n: 类型名 n;
```

其中,var 为保留字,表示变量的声明开始。变量名要符合标识符的命名规则。类型名用来指出变量的数据类型,它决定了变量中所存放的数据的范围、格式等。多个变量的声明之间用分号隔开。

当声明的多个变量具有相同的数据类型时,可以使用如下的紧凑格式:

```
var  变量名 1, … ,变量名 n:类型名;
```

例如:var i, j: integer; name: string; a, b, c: single;

注意:同类型的变量名之间用逗号隔开,不同类型的变量声明之间用分号隔开。

2. 变量的初始化

在函数或过程内部定义的变量称为局部变量,其余的称为全局变量。

例如:假设 project1 的单元代码清单如下:

```
unit Unit1;                    //unit1 中代码
Interface
Uses
⋮
type
⋮
var
  Form1: TForm1;
  i:integer;                   //全局变量的声明位置,声明在该位置的变量 i 就是一个全局变量
implementation
{ $R *.dfm}
procedure TForm1.Button1Click(Sender: TObject); //Button1 的 OnClick 单击事件
  var j:integer;               //局部变量的声明位置,该位置声明的变量 j 就是一个局部变量
begin
  j := 1;                      //为局部变量赋值
end;
end.
```

以上在应用程序中共声明了两个变量,其中变量 i 的声明放在了单元代码的接口部分 Interface 中 var 的位置,是一个全局变量。而变量 j 的声明放在了 Button1 的 OnClick 事件处理过程中,是一个局部变量。

在 Object Pascal 中,允许在声明全局变量时对变量赋初值。即,声明全局变量 i 的语句可修改为 var i: integer=10。只有全局变量才能在声明的同时赋初值,而在过程或函数内部声明的局部变量则不允许在声明的同时赋初值。也即,局部变量的声明和赋初值必须通过声明语句和赋值语句两个语句来分别完成。

注意:没有初始化的全局变量在编译时,编译器会自动为该变量赋值。例如,默认情况下,整型全局变量会被赋值为 0,实型全局变量会被赋值为 0.0。而局部变量的值在赋值前是随机的,系统不会为其分配默认的值。因此,局部变量使用之前必须赋初值。

小贴示:局部变量的声明语句放在程序主体代码中 begin 语句之前,局部变量的赋值语句放在 begin 语句之后。

3.5 运算符和表达式

运算符是代码中各种类型的数据进行运算的符号，而参与运算的数据称为操作数。表达式是一个求值的运算公式，它由运算符和配对的圆括号将常量、变量、函数等操作数以合理的形式组合而成。

Object Pascal 定义了多种运算符，常用的有赋值运算符、比较运算符、逻辑运算符、算术运算符、字符串运算符等。赋值运算符、比较运算符和逻辑运算符会分别放在本书第 4 章顺序结构程序设计和第 5 章选择结构程序设计中讲述。本节首先讲述另外的两种运算符，即算术运算符和字符串运算符。

3.5.1 算术运算符与算术表达式

算术运算符主要用于对浮点数和整数进行加、减、乘、除和取模运算。常用的算术运算符有＋(加法运算或取正数)、－(减法运算或负号)、＊(乘法运算)、/(浮点除法)、div(整除)、mod(求余数)。

注意：

(1) 参与算术运算的操作数可以是整数或实数，运算结果由系统自动向精度高的类型转化。

(2) 参与浮点除法运算/的操作数无论是整数还是实数，运算结果所得的商都是实数型数据。

(3) 参与 div 和 mod 运算的操作数必须是整数，运算结果也是整数，运算结果的符号类型与被除数相同。div 运算结果为两个操作数之商四舍五入之后所得的整数部分。mod 运算结果为两个操作数相除所得的余数部分。

(4) 在 a/b、a div b、a mod b 运算中，如果 b 为 0，将会发生异常。

Object Pascal 中的运算大部分是二元运算，即有两个操作数。只有一个操作数的运算称为一元运算或单目运算，一元运算符出现在操作数之前。其中＋和－运算符作为单目运算符时，放在浮点数或整数前，分别表示正数和负数。

在复杂的表达式中，各运算符的处理顺序是按照运算符的优先级来进行的。算术运算符的优先顺序按从高到低依次为：＋(取正)、－(取负)、＊、/、Div、Mod、＋(加法)、－(减法)。

注意：在书写算术表达式时，原来的数学表达式中省略的内容必须重新写上。例如，2x 必须写成 2＊x。此外，所有括号都用小括号()，且括号必须成对出现。例如，数学表达式 3[x＋2(y＋z)]应该写成 3＊(x＋2＊(y＋z))。

3.5.2 字符串运算符与字符串表达式

字符串运算符只有一种，即字符串连接符“＋”，用于连接两个或更多的字符串成为一个字符串，操作数可以是字符型或字符串型。例如，'delphi'＋'应用程序开发工具'连接结果为'delphi 应用程序开发工具'。又如，label1.caption :＝ '祖国'＋chr(13)＋chr(10)＋'万岁'；则程序运行结果，对象 label1 上第一行显示“祖国”字样，第二行显示“万岁”字样。默认情况下，

字符串连接之后只有前面 255 个字符有效。

3.6 常用的内部函数

过程和函数都是能够完成某个或某些特定功能的程序段,都可以在程序中被调用以实现特定的功能。两者的区别在于:函数具有返回值,而过程则没有返回值。Object Pascal 本身提供了大量的内部函数,这些内部函数已经被 Object Pascal 系统所定义,因而程序员在程序中可以直接使用。

3.6.1 数学函数

(1) Abs(x):返回 x 的绝对值。

例如:Abs(3)返回值为 3,Abs(-10.5)返回值为 10.5。

(2) Sqr(x):返回 x 的平方。

例如:Sqr(1.5)返回值为 2.25,Sqr(-4)返回值为 16。

(3) Sqrt(x):返回 x 的平方根。

例如:Sqrt(25)返回值为 5.0,Sqrt(1.44)返回值为 1.2。

注意:使用 Sqrt 函数求平方根时,函数返回值的类型为 Real 型,而不是 Integer 型。此外,使用平方根函数时,操作数 x 必须大于等于 0,否则会出现运算异常。

(4) Int(x):返回一个实型数据 x 的整数部分,且不四舍五入。

例如:Int(4.6)返回值为 4.0,Int(-4.6)返回值为-4.0。

注意:函数 Int(x)的返回值类型是 Real 型,而不是 Integer 型。

(5) Frac(x):返回一个实型数据 x 的小数部分。

例如:Frac(4.6)返回值为 0.6。

注意:函数 Frac(x)返回的是 x 的小数部分,返回值类型是 Real 型,而不是 Integer 型。

(6) Trunc(x):返回一个实数 x 的整数部分,且不四舍五入。

例如:Trunc(12.6)返回值为 12,Trunc(-4.6)返回值为-4。

注意:函数 Trunc(x)的返回值类型是 Integer 型,而不是 Real 型。

(7) Round (x):返回一个实数 x 四舍五入后所得的整数。

例如:Round (12.6)返回值为 13,Round(-4.6)返回值为-5。

注意:函数 Round(x)的返回值类型是 Integer 型,而不是 Real 型。

(8) Random(x):返回一个 0 至 x 之间的随机整数。

例如:Label1.left:=Random(form1.width)。

(9) Pi:返回圆周率的值。

例如:var S, r:single; r:=2; S:=Pi*r*r。

(10) Max(x,y):返回 x、y 中较大的一个。

例如:Max(3,6)返回值为 6。

(11) Min(x,y):返回 x、y 中较小的一个。

例如:Min(12,14)返回值为 12。

(12) Sin(x)：返回 x 对应的正弦值。其中，x 用弧度数表示。

例如：Sin(pi/6)返回值为 0.5。

注意：使用正弦、余弦、反正切等函数时，操作数都要用弧度表示，而不用度数表示。

(13) Cos(x)：返回 x 对应的余弦值。

(14) ArcTan(x)：返回 x 对应的反正切值。

(15) Exp(x)：返回以 e 为底的指数函数，等价于 e^x。

(16) Ln(x)：返回以 e 为底的对数函数。

小贴示：在代码中引用数学函数时，若代码编辑窗口中不识别函数的名称，则在代码段 interface 部分的 uses 中加入名称为 math 的单元即可。

3.6.2 字符串函数

(1) function Concat(s1 [, s2,…, sn]: string): string;

函数 Concat(s1,s2,…,sn)返回字符串 s1,s2,…,sn 连接所得的字符串。

例如：Concat('aa','bb', 'cc')返回值为'aabbcc'。

(2) function Pos(Substr: string; S: string): integer;

函数 Pos(Substr,S)返回子串 Substr 在源字符串 S 中第一次出现的位置。若 Substr 不是 S 的子串，则函数返回值为 0。

例如：Pos('ll', 'hello')返回值为 3，Pos('a', 'hello')返回值为 0。

(3) function Length(S): integer;

函数 Length(S)返回字符串 S 的长度。

例如：Length('hello')返回值为 5。

(4) function Copy(S; Index, Count: Integer): string;

函数 Copy(S, Index, Count)返回字符串 S 从第 index 位开始，复制 count 个字符所得的字符串。

例如：Copy('Goodness',5,4)返回值为'ness'。

(5) procedure Delete(var S: string; Index, Count:integer);

过程 Delete(S, Index, Count)用于删除字符串变量 S 中的字符，从第 Index 位开始删除，删除长度为 Count。删除之后所得的字符串存于 S 中。

例如：

```
var S:string;        //声明字符串 S
S := 'Goodness';     //为变量 S 赋值
Delete (S,5,4);      //执行 Delete 过程，则变量 S 最终值为'Good'
```

注意：Delete 是一个过程而不是函数，使用该过程之前需要定义要执行删除操作的字符串变量 S，该变量的值在执行 Delete 过程的前后发生了变化。

(6) procedure Insert(Source: string; var S: string; Index: Integer);

过程 Insert(Source, S, Index)将源字符串 Source 插入字符串 S 中，插入位置为 Index。

例如：

```
var S:string;            //定义变量 S
```

```
S := 'Good';              //为变量 S 赋值
Insert ('ness', S,5);     //执行过程 Insert,则变量 S 返回值为'Goodness'
```

注意：Insert 是一个过程而不是函数。使用该过程之前需要预先定义被插入字符的源字符串变量 S,该变量的值在执行 Insert 过程前后发生了变化。

3.6.3 日期时间函数

(1) function Now: TDateTime;

函数 Now 用来返回系统当前的日期和时间。

(2) function Date: TDateTime;

函数 Date 用来返回系统当前的日期。

(3) function Time: TDateTime;

函数 Time 用来返回系统当前的时间。

(4) function DayOfWeek(tt: TDateTime): integer;

函数 DayOfWeek(tt)用来返回日期 tt 所对应的星期数,返回值是 1～7 之间的一个整数。当返回值为 1 时,表示 tt 日期对应的是星期日。当返回值为 2～7 时,表示星期 1～星期 6 星期。

例如：DayOfWeek(now)用来返回当前日期对应的是星期几。

3.6.4 转换函数

(1) function LowerCase(const S: string): string;

函数 LowerCase(S)将字符串 S 中所有的字符全部转换为小写。

例如：LowerCase('OBJECT')返回值为'object'。

(2) function UpperCase(const S: string): string;

函数 UpperCase(S)将字符串 S 中所有的字符全部转换为大写。

例如：UpperCase ('Delphi')返回值为'DELPHI'。

(3) procedure Str(x;var S:string);

过程 Str(x,S)将数值型变量 x 转换为字符串类型后,储存于字符串型变量 S 中。

例如：var s:string; //声明字符串型变量 s

str(120,s); //调用过程 str 后,变量 s 返回值为'120'。

str(3.15,s) //调用过程 str 后,变量 s 返回值为'3.15000000000000E+0000'。

注意：Str 是一个过程而不是一个函数,使用该过程之前需要预先定义一个字符串类型的变量用来存储转换结果。

小贴示：在 Object Pascal 中,过程作为一条独立语句出现,而函数则往往作为表达式的一部分出现在语句代码中。

(4) procedure Val(s:string;var v;varcode:integer);

过程 Val(s,v,code)将字符串型变量 s 转换为数值类型数据,并存储在数值型变量 v 中。若字符串 s 可以转换为数值型数据,则参数 code 返回值为 0。若字符串 s 中含有非数字的字符,则转换失败,同时参数 code 返回值为第一个非数字字符在字符串 s 中出现的位置。

例如：var a,code：integer；

Val('123',a,code)； //调用 Val 过程后，变量 a 返回值为 123，code 返回值为 0。

Val('312qq',a,code)； //调用 Val 过程后，变量 a 返回值为 312，code 返回值为 4。

注意：Val 是一个过程而不是函数。使用该过程之前需要预先定义两个变量分别用来存储转换后的数值型数据和非法字符所在的数字位。

(5) function IntToStr(x：Integer)：string；

函数 IntToStr(x)将整型数据 x 转换成对应的字符串。

例如：IntToStr(2017)返回值为'2017'。

(6) function FloatToStr(x)：string；

函数 FloatToStr(x)将实型数据 x 转换成对应的字符串。

例如：FloatToStr(3.14)返回值为'3.14'。

(7) function StrToInt(const S：string)：integer；

函数 StrToInt(S)将全部由数字组成的字符串 S 转换成对应的整型数据。

例如：StrToInt('100')返回值为 100。

注意：使用 StrToInt( )函数时，操作数去掉一对单引号之后必须是整数，否则将出现转换异常。

(8) function StrToFloat(const S：string)：extended；

函数 StrToFloat(S)将数字字符串 S 转换成对应的实数。

例如：StrToFloat('3.14')返回值为 3.14。

注意：使用 StrToFloat( )函数时，操作数去掉一对单引号之后必须是实数，否则将出现转换异常。

(9) function DateToStr(dd：TDateTime)：string；

函数 DateToStr(dd)将日期型数据 dd 转换成字符串型数据。

例如：Label1.caption：= '现在日期是：'+DateToStr(Date)； //只显示日期

(10) function TimeToStr(tt：TDateTime)：string；

函数 TimeToStr(tt)将日期型数据 tt 转换成字符串型数据。

例如：Label1.caption：= '现在时间是：'+TimeToStr(Time)； //只显示时间

(11) function DateTimeToStr(dt：TDateTime)：string；

函数 DateTimeToStr(dt)将日期时间型数据 dt 转换成字符串型数据。

例如：Label1.caption：= '当前是：'+DateTimeToStr(Now)； //显示日期和时间

(12) function Chr(X：Byte)：char；

函数 Chr(X)返回 ASCII 码值为 X 的字符。

例如：Chr(66)返回值为'B'，Chr(97)返回值为'a'。

(13) function Ord(C:Char)：Byte；

函数 Ord(C)返回字符型数据 C 对应的 ASCII 码值。

例如：Ord ('d')返回值为 100。

3.6.5 格式化函数

function Format(const Format：string；const Args：array of const)：string；

函数 Format 用来将数据按照指定的格式转换为字符串。第一个参数为格式化字符串,该字符串由字符%开头,格式为%[(所占宽度)][.(小数位数)](类型)。第二个参数为要格式化的数据,放在一对方括号[]中。

例如:format('%4d',[35])将整型数据 35 转换为长度为 4 的字符串,返回值为' 35'。

例如:format('%.2f',[1.467])将实型数据 1.467 四舍五入后小数点之后保留 2 位,并转换为字符串,返回值为'1.47'。

【例 3.1】 常用的内部函数使用示例。

新建工程,向窗体中添加 4 个 Button 组件,各按钮上显示的文字信息如图 3-1 所示。单击各个按钮,练习相应的系统函数的使用。

图 3-1 界面设计及运行情况

程序主要代码清单如下:

```
procedure TForm1.Button1Click(Sender: TObject);     //"数学函数"按钮的 OnClick 事件
begin
  showmessage(inttostr(abs(-10)));                  showmessage(inttostr(sqr(-10)));
  showmessage(floattostr(sqrt(16)));                showmessage(inttostr(max(10,15)));
  showmessage(inttostr(random(200)));               showmessage(floattostr(pi));
  showmessage(floattostr(int(10.6)));               showmessage(floattostr(frac(10.6)));
  showmessage(inttostr(trunc(10.6)));               showmessage(inttostr(round(-10.6)));
end;

procedure TForm1.Button2Click(Sender: TObject);   //"字符串函数"按钮的 OnClick 事件
var  str1, str2:string;
begin
  str1 := 'hello'; str2 := 'china!';
  showmessage(concat('hello ','china!'));          //等价于 showmessage(concat(str1,str2));
  showmessage(inttostr(pos('ll',str1)));
  showmessage('字符串 str1 的长度为'+inttostr(length(str1)));
  showmessage(copy('asdfesd',2,4));
  delete(str1,2,3); showmessage(str1);
  insert(str1,str2,6); showmessage(str2);
end;

procedure TForm1.Button3Click(Sender: TObject);   //"时间函数"按钮的 OnClick 事件
var i:integer;
begin
  showmessage('现在是:'+ datetimetostr(now)) ;
  showmessage('当前日期为:'+ datetostr(date));
  showmessage('当前时间为:'+ timetostr(time));
  i := dayofweek(now);
  if i=1 then showmessage('今天是星期天')
  else showmessage('今天是星期'+ inttostr(i-1));
end;
```

```
procedure TForm1.Button4Click(Sender: TObject);   //“转换函数”按钮的 OnClick 事件
begin
  showmessage('字符 a 的 ascii 码为' + inttostr(ord('a')));
  showmessage('ascii 码为 88 的字符是' + chr(88));
  showmessage(Lowercase('DELPHI'));
  showmessage(Uppercase('pascal'));
end;
```

小贴示：showmessage(S)是一个过程，其作用是弹出一个消息对话框，给用户一些提示信息。其中参数 S 是字符串类型的数据，便是对话框中显示的文字信息。有关对话框的使用详见第 9 章。

3.7 Object Pascal 语句书写规则

Object Pascal 程序语句的书写比较自由，可以把多条语句放在一个程序行中，也可以将一个语句分为多行书写，但在书写语句时要注意以下几点规则：

(1) Object Pascal 不区分大小写。

(2) 除了字符串中的汉字外，其余字符均在英文半角输入法状态下输入。

(3) 每条语句必须以分号“;”结束。

(4) 一行上可以写多条语句，也可以将一条语句写在多行上。

(5) 每行代码最好不超过 80 个字符(包括空格)。如果某条语句过长，应另起一行书写。同一条语句换行时不需要续行符，只要没有出现分号，就表明语句尚未结束。

(6) 逻辑操作符与操作数之间必须留空格，否则识别为非法字符。

例如：if (a > 0) and (b > 1)语句中操作符 and 前后必须留有一个空格。

(7) 复合语句以 Begin 开始，以 End 结束。语法格式为：

```
begin
  语句 1;
  ⋮
  语句 n;
end ;
```

3.8 基本组件及系统函数综合应用

【例 3.2】 用户在文本框中输入三角形三边的长度，根据海伦公式，计算并在标签上输出三角形的面积。程序界面设计及运行结果如图 3-2 所示。

新建工程，向窗体中添加 6 个 Label 组件，3 个 Edit 组件和 2 个 Button 组件，各组件主要属性设置如图 3-2 所示，其中显示面积的标签的 name 属性修改为 lbl_area。当单击“确定”按钮时，将三角形的面积显示在标签中。当单击“重新计算”按钮时，清空文本框和显示面积的标签中的内容。

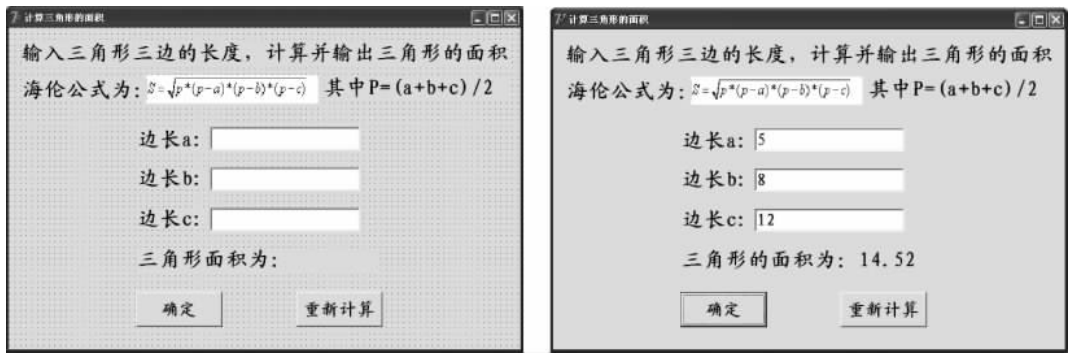

图 3-2 系统界面设计及运行情况

程序主要代码清单如下：

```
procedure TForm1.Button1Click(Sender: TObject);   //“确定”按钮的 OnClick 事件
var  a,b,c,p,s:real;              //定义变量 a、b、c、s 分别存储三角形三边的长度和面积
begin
  a := strtofloat(edit1.text); //从文本框中获得三边的长度值
  b := strtofloat(edit2.text); //使用 strtofloat()函数将字符串型数据强制转换为实型数据
  c := strtofloat(edit3.text); //判定用户输入的三边长度是否可以构成一个三角形
  if (a + b > c) and (a + c > b) and (b + c > a) and (abs(a - b)< c) and (abs(a - c)< b) and (abs(b - c)
< a)  then
  begin
    p := (a + b + c)/2;           //根据海伦公式计算三角形的面积
    s := sqrt(p * (p - a) * (p - b) * (p - c));
    lbl_area.Caption := '三角形的面积为: ' + format('%.2f',[s]);
                                  //将面积的值保留 2 位小数
  end else
  begin
    showmessage('您输入的数据无法构成三角形的三条边,请重新输入!') ;
    edit1.Clear;                  //若输入的三边长度不合法,则清空内容重新输入
    edit2.Clear;  edit3.Clear;   edit1.SetFocus;
    lbl_area.Caption := '三角形的面积为: ';
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);  //“重新计算”按钮的 OnClick 事件
begin
  edit1.Clear;   edit2.Clear;   edit3.Clear;     edit1.SetFocus;
  lbl_area.Caption := '三角形的面积为: ';
end;
```

注意：为保证程序的合理性,本例对于用户输入的三角形三边的长度必须验证其是否合法,能否组成三角形。若不合法,则需要提示用户重新输入。

本章小结

语法是编程的基础，要想在代码编写过程中得心应手，必须首先熟练掌握该编程语言的语法基础。本章重点介绍了 Object Pascal 编程语言的语法基础，让读者学习并掌握基本数据类型、常量和变量的定义、运算符和表达式以及常用的一些内部函数。这些语法的理论知识虽然较为枯燥，但是大部分语言虽语法格式各不相同，其编程中的程序设计思想却是相通的。所以，在实际的操作练习中，应该多注意不同语言在语法定义方面的异同点，灵活运用，在实践中巩固本章所学的知识，为复杂应用程序代码的编写奠定良好的基础。

习题 3

3.1 下列符号中哪些可以作为用户自定义的标识符？

(1) Student (2) And (3) Function (4) _3sum (5) 3ab

(6) xyz (7) X + y (8) Hello! (9) 'HAITa' (10) sqr

3.2 Delphi 定义的数据类型中，哪些是标准数据类型？

3.3 下列(　　)符号不能作为 Delphi 7 中的变量名。

(A) ABCD_EFG (B) _P000000 (C) 89TWDDFF (D) xyz

3.4 下列符号(　　)是 Delphi 7 中的合法变量名。

(A) AB7 (B) 7A B (C) 7_AB (D) A[B]7

3.5 下列数据哪些是变量？哪些是常量？是什么类型的常量？

(1) name (2) 'name' (3) False (4) '120' (5)100 (6)12.345 (7) num

3.6 下列常量声明语句中(　　)是合法的。

(A) const c := (a > 3.4); (B) const c : 3.4;

(C) const c = 3.4; (D) const c := 10 and 20;

3.7 下列常量的声明语句是否合法？不合法的请说明原因。

(1) const I = maxint +1; (2) const zero = 0.001;

(3) const pi =π; (4) const ss = "pascal";

3.8 下列实数中哪些是合法的，哪些是不合法的？不合法的请说明理由。

(1) 0.25E + 02 (2) .25 + 2 (3) 25E + 2 (4) 34.5 (5) .123 (6) −3E−4

3.9 指出下列变量声明语句的错误。

```
var
  a1; a2;  a3 : real;  a3 : integer;
```

3.10 计算下列表达式的值。

(1) 10 + 11 div 2 (2) (30 + 23 div 5) mod 3 (3) ord('a') + ord('d')

(4) (10 > 3) and (3 <> 7) (5) 'Good'+'123' (6) 25/2+sqr(1.2)

3.11 对于一元二次方程的根的表达式中(　　)是 Object Pascal 中合法的表达式。

(A) (− b + sqr(b * b − 4ac)) / (2 * a)

(B) (－ b ＋ sqrt(b * b － 4 * a * c)) div (2a)

(C) (－ b ＋ sqrt(b * b － 4 * a * c)) / (2 * a)

(D) (－ b ＋ sqrt(b － 4 * a * c)) / (2 * a)

3.12 数学式子 sin30°写成 Delphi 表达式是下列(　　)。

(A) Sin30　　(B) Sin(30)

(C) SIN(30°)　　(D) Sin(30 * Pi / 180)

3.13 计算下列表达式的值。

(1) 6＋12 * sqr(5)/4　　(2)14 mod 3 * 2＋round(3.7)

(3) 'abc'＋format('%3d',[23])　　(4)12.4＋frac(3.56) * 3