Standard subroutines

Mathematical functions

The following table contains examples and explanations of some mathematical functions in Delphi:

Delphi		Comment
iVal := **Round**(rX);		Rounds a real value and returns an integer value rounded to the nearest whole number. If the number is halfway between two integers, it will always be rounded to the nearest *even* number. E.g. Round(7.5) = 8 and also Round(8.5) = 8
Round(8.8)	9	
Round(8.3)	8	
Round(7.3)	7	
Round(6.5)	6	
iVal := **Trunc**(rX);		Truncates a real number to an integer. It 'chops' off the decimal part and returns only the integer part.
Trunc(8.5)	8	
Trunc(8.3)	8	
rVal := **Frac**(rX);		Returns only the decimal part of a real number.
Frac(74.89)	0.89	
Frac(0.3728)	0.3728	
iAnswer := **Ceil**(rX);		Returns the largest integer value nearest to the number. This function is stored in the Unit Math.
Ceil(2.1)	3	
Ceil(2.9)	3	
Ceil(-3.1)	-3	

Delphi		Comment
iAnswer := **Floor**(rX);		Returns the smallest integer value nearest to the number. This function is stored in the Unit Math.
Floor(2.1)	2	
Floor (2.9)	2	
Floor (-3.1)	-4	
rVal := **Sqrt**(rX);		Returns the square root of a number. The result is always of type real. Note: The square root of a negative number can not be calculated.
rVal := **Sqr**(rX); iVal := **Sqr**(iY);		Returns the square of a number – the result is of the same type as the number.
rVal := **Abs**(rX); iVal := **Abs**(iY);		Returns the absolute value of a number – the result is of the same type as the number.
rC:= 2 * **PI** * iR;		PI returns a value for the mathematical constant PI.
rAnswer := **Power**(iBase, iExponent);		Raises the first number to the power of the second number. All variables must be compatible with the data type *extended*. This function is stored in the Unit Math.
iVal := **Random**(iX);		Returns an *integer* value in the range [0..(iX-1)].
rVal := **Random**;		Returns a *real* number >=0 but < 1.
Tip • Random (100) has 100 possible answers (from 0 to 99). To generate a number between 1 and 100 (both included) do the following: iNo := Random (100) + 1; • To make the numbers generated by the Random function less predictable use the **Randomize** procedure. Randomize simply scrambles the results of the Random function. You only have to use this procedure once in your program, preferably when the program starts (use the OnActivate event of the Form).		
iVal := **RandomRange**(1,100);		Returns an integer value between 1 and 100 (both can be included)
Scratch had an equivalent function.		

Procedures that do calculations

Inc(iNumber, iAdd); Inc(iNumber); Inc(cChar,2);	Increases the value of iNumber with iAdd Increases the value of iNumber with 1 Changes the character to two places after this character in the ASCII table. (Increases the ordinal value of the character by 2)
Dec(iNumber,2); Dec(iNumber); Dec(cChar,2);	Decreases the value of iNumber with 2. Increases the value of iNumber with 1 Changes the character to two places before this character in the ASCII table. (Decreases the ordinal value of the character by 2)

String manipulation functions

A string is represented in memory as a sequence of numbered spaces.

```
Var sName : string;
sName := 'Peter';
```

```
sName [1] has the value 'P'
sName[4] has the value 'e'
```

Data in memory

P	e	t	e	r
[1]	[2]	[3]	[4]	[5]

We can do manipulations on strings such as to delete individual characters in a string, change some characters to upper case, add characters any place in a string, or extract a group of characters from the string. String manipulation functions can also provide us with information regarding strings, for example: How long is a string (how many characters does it consists of)? Where is the space in the string? Is the word 'house' part of the string?

Here are some examples of functions that can be used to manipulate strings:

Delphi function	Comment
lblOut.Caption:=**Concat**('Hello ',sName); lblOut.Caption := 'Hello ' + sName;	The Concat function can be used instead of the '+' operator.
iVal := **Length**(sName);	Returns the number of characters that a string value consist of as an integer.
iPos := **Pos**(sString, sName);	Returns the position of the first letter of sString in sName.
	Study the following examples, assume that sName := 'Software Application';

Example	Value returned
iPos := Pos('a',sName);	6
iPos := Pos('war',sName);	5
iPos := Pos(' ',sName);	9
iPos := Pos('A',sName);	10

sWord := **Copy**(sS, iStart, iLength);	Copies a substring from sS, from position iStart for iLength characters.
	Study the following examples, assume that sS := 'Software Application';

Example	String returned
sWord := Copy(sS,5,4);	ware
sWord := Copy(sS,2,3);	oft

sWord := **Uppercase**(sWord);	Converts all letters in sWord to uppercase and assigns it to sWord.
sWord[1] := **Upcase**(sWord[1]);	The number in brackets refers to the position of a character in a string. The Upcase function is used to convert one character to upper case. In this example the first character of the string is converted to upper case.

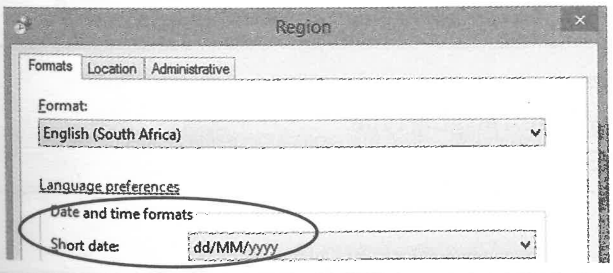
String manipulation procedures

We can also manipulate strings using the procedures *Insert* and *Delete*. These two procedures receive a number of values and then make changes to one of the values.

Delete(sString, iStart, iLength); Deletes a substring from a string. sString: The string that should be changed. iStart: The position of the first character that should be deleted. iLength: The number of characters to be removed from the string – the starting one included.		
Example:	sS := 'Lets start from scratch'; **Delete**(sS,12,5);	Value of sS is now: 'Lets start scratch'
Insert(sSub, sString, iPosition); Inserts a substring into a string in a given position. sSub: the string that should be inserted. sString: The string that should be changed. iPosition: The position where the substring should start in the changed string.		
Example:	sS := 'Bread tastes good'; **Insert**('Hot ',sS,1);	'Hot Bread tastes good'
Val(sString, iNumber,iCode);		Converts sString to a number and stores the number in variable iNumber. If the variable iCode is 0, the conversion was successful, if not, the value of iCode shows the position of the character in the string that caused the problem.
Str(rNumber,sString:5:2);		Converts the value stored in rNumber to a string value and stores the string in sString. In this case the number should consist of 5 places including 2 decimals.

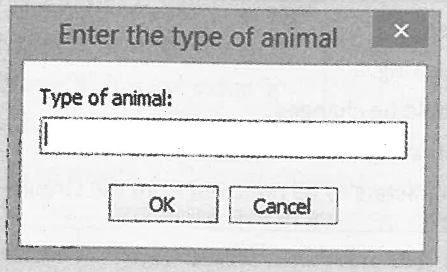
Time and date functions

sTime := **TimeToStr**(Time);	Converts the system time to a string.
Date	Returns the current system date.
pnlDate.Caption := **DateToStr**(Date);	Converts the system date to a string. The format will be determined by the *Short Date* format in the *Region and Language* settings. For example, 23 March 2013 will be displayed as 23/03/2013 if the settings is as follows:



bLeap := IsLeapYear(2000);	The *IsLeapYear* function returns true if a given calendar value is a leap year. The year supplied as the argument can be a value between 0 and 9999.

Other functions

sAnimal := **InputBox**('Enter the type of animal', 'Type of animal: ', '');	A function to input a string that creates a pop up: 
frmScreen.Color := **RGB**(iRed, iGreen, iBlue);	Sets the colour of the screen using a combination of red, green and blue.
bAnswer := **Odd** (iValue);	Determines whether the argument is odd or not. Returns a Boolean value.

iNum := **Ord**('A');	The function *Ord* is usually used to determine the ordinal value (ASCII code) of a character.
	iNum := Ord('A') ; //Returns the value 65
	iNum := Ord('1') ; //Returns the value 49

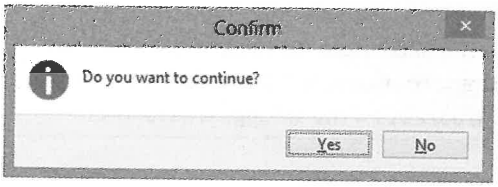
cChar := **Chr**(65);	The function *Chr* can return the character associated with an ASCII code.
	cChar := Chr(65) ; //Returns the character 'A'
	cChar := Chr(50) ; //Returns the character '2'

if MessageDlg('Are you sure you want to delete these records?',
mtWarning, [mbOk, mbCancel], 0) = mrOK **then**

The function *MessageDlg* displays a pop-up message to the user. The programmer can specify:

- which type of message it should be (a warning, an error, just information etc.). This determines the symbol and heading which will be displayed in the pop-up message.
- which Buttons should be displayed (a 'Yes' Button, an 'OK' Button etc).

Examples of the pop-up message produced by the MessageDlg function:



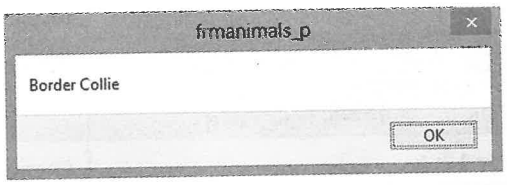
The parameters of the function *MessageDlg* are as follows:

function **MessageDlg(const Message**:string; **DialogType**:TMsgDlgType;
Buttons:TMsgDlgButtons; **HelpContext**:Longint):Integer;

The *Message* can be any string you wish to display. Arguments may have the following values:			
DialogType (one of the following)	mtWarning	mtConfirmation	mtInformation
	mtError	mtCustom	
Buttons (one or more of the following)	mbYes	mbNo	mbOK
	mbCancel	mbAbort	mbRetry
	mbIgnore	mbAll	mbNoToAll
	mbYesToAll	mbHelp	
HelpContext	This value is used in conjunction with the Help Button, but it will not be discussed in this Appendix. Use the value 0 (zero).		

The function returns an integer value, but Delphi also provides the option to compare the return value to a Delphi enumerated value. For example, when the user clicks on the [Yes] Button, the function returns a value mrYes (it is not a string – therefore no quotes should be used).			
Values returned by the function	mrYes	mrNo	mrOK
	mrCancel	mrAbort	mrRetry
	mrIgnore	mrAll	mrNoToAll
	mrYesToAll	The Help Button provides no return value.	

Other procedures

ShowMessage(sAnimal);	A procedure to display a message, without using a specific object on the form: 
Randomize;	Scrambles the results of the Random function to make the results less predictable
Beep;	Makes a beep noise
Sleep(200);	Delays the program for (200 milli seconds)