**String Functions**

The ability to modify and concatenate strings is a powerful feature of many programming languages, and Basic syntax doesn't disappoint. This section breaks out the different categories of string functions and summarizes how they work. The categories are: Analyzing a String, Parsing Strings, and Manipulating Strings.

Throughout this section, many of the functions listed use one or both of the arguments called compare and start. Rather than repetitively list their descriptions throughout the chapter, they are explained here for your reference.

The compare argument determines when string comparisons are supposed to be case sensitive. If compare is 0, the search is case-sensitive. If it is 1, the search is not case-sensitive. Case sensitivity means that even if two strings consist of the exact same letters, they will still be treated as different strings if one is upper-case and the other is lower-case. For example, �Joe� would not be the same as �joe�. The compare argument is optional. If it is left out, the comparison defaults to 0 (case sensitive).

The start argument tells the function to process characters starting at a specified position.[[1]](#footnote-1)[2] Any characters that are prior to that position are ignored. This argument is optional. If it is left out, then the function is performed for the entire string.

**Converting Data Types**

Basic syntax is a type safe language that requires all constants and variables in the same formula to be of the same data type. It also requires you to pass constants and variables as arguments using the exact data type that the formula expects. Even though the data types in Basic syntax are fairly simple, you still have to make sure that they are compatible. Fortunately, this shouldn't cause problems because there are functions to convert between the different data types. Table 6-5 lists these conversion functions.

Table 6-5. Conversion Functions

|  |  |
| --- | --- |
| Conversion Function | Description |
| CBool(number), CBool(currency) | Convert to Boolean. |
| CCur(number), CCur(string) | Convert to Currency. |
| CDbl(currency), CDbl(string),  CDbl(boolean) | Convert to Number. Equivalent to ToNumber(). See the section �Formatting Values for Output�. |
| CStr() | Convert to String. Equivalent to ToText(). |
| CDate(string), CDate(year, month, day), CDate(DateTime) | Convert to Date. |
| CTime(string), CTime(hour, min, sec), CDate(DateTime) | Convert to Time. |
| CDateTime(string),  CDateTime(date),  CDateTime(date, time),  CDateTime(year, month, day) | Convert to DateTime. |
| CDateTime(year, month, day, hour,  min, sec) | Convert to DateTime. |
| ToNumber(string), ToNumber(boolean) | Convert to a Number. |
| ToText() | Convert to String. Same as CStr(). |
| IsDate(string), IsTIme(), IsDateTime() | Test a string for being a valid date/time. |
| IsNumber(string) | Test a string for being a valid number. |
| ToWords(number),  ToWords(number, decimals) | Convert a number to its word equivalent. |

Most of the above functions are very simple. Pass the function a field/variable of one data type and it returns the equivalent in the other data type. The CBool() function takes a number or currency value and converts it to Boolean True or False. Any non-zero value is converted to True and zero is converted to False. When it is displayed on a report, it prints the words �True� or �False�.

The CCur() function takes a number or string and converts it to the Currency data type. When converting a string, it can have formatting characters in it (�$�, �,�, etc.) and it will still be converted properly.

The CDbl() and ToNumber() functions are equivalent. Pass each a value and it gets converted to a number.

The CDate(), CTime() and CDateTime() are all similar. Pass them a string and it gets converted to the proper data type. The string parser for this function is very sophisticated. It lets you pass strings as diverse as �Jan 19, 1991�, �5/26/1998� and �2002, Feb 04�. You can also pass numbers as individual arguments for representing the different parts of a date and time. See Table 6-5 for the various argument options.

When converting a string to a date or number, you run the risk of raising an error if the string isn't in the expected format. You can avoid this by testing the strings validity before converting it. The IsDate() and IsNumber() functions do this for you. They return True if the string can be properly converted. If not, they return False. For example, here is a function that converts a string to a date, but only if it is a valid date.

If IsDate({Invoice.ExpirationDate}) Then

��� Formula = CDate({Invoice.ExpirationDate})

End If

The ToWords() function takes a number and converts it to its equivalent in words. This is similar to writing a dollar amount on a check and then spelling out the full amount in words. It prints the decimal portion as �##/100�. You can set the number of decimals it displays by passing a number to the second argument, which is optional. Notice in the second example how it only displays one decimal place and it rounds it up to the next higher number.

'Demonstrate the ToWords() formula

Formula = ToWords(123.45)� 'Result is �one hundred twenty-three 45 / 100�

Formula = ToWords(123.45,1) 'Result is �one hundred twenty-three and 5 / 100

**Math Functions**

Table 6-8. Math Functions

|  |  |
| --- | --- |
| Function Name | Description |
| Abs(number) | Return the absolute value. |
| Fix(number, decimals) | Return a number with a specified number of significant digits. |
| Int(number),� numerator \ denominator | Return the integer portion of a fractional number. |
| Pi | 3.14... |
| Remainder(numerator, denominator), | Return the remainder of dividing the numerator by the denominator. |
| numerator Mod denominator | Return the remainder of dividing the numerator by the denominator. |
| Round(number, decimals) | Round up a number with a specified number of significant digits. |
| Sgn(number) | Return a number's sign. |
| Sqr(number), Exp(number), Log(number) | The standard arithmetic functions. |
| Cos(number), Sin(number), Tan(number), Atn(number) | The standard scientific functions. |

Most of these functions perform basic mathematical functionality. There are only a couple of interesting points to notice. Working with whole numbers and decimals is done numerous ways. The Fix() and Round() functions take a fractional number and truncate it to a specified number of digits. The Round() function will round up to the nearest decimal. The number of decimals to display is optional and the default is zero. The Int() function is similar to the Round() function except that it only returns the whole number and will round down to the nearest whole number. Table 6-9 shows how the three functions will return a different number depending upon the decimal portion and whether the number is positive or negative.

Table 6-9. Examples of Truncating Decimals

|  |  |  |
| --- | --- | --- |
| Function | 1.9 | -1.9 |
| Fix() | 1 | -1 |
| Round() | 2 | -2 |
| Int() | 1 | -2 |

If you want to get the whole number and you have the numerator and denominator available. You can use the \ operator to perform integer division. This does the division and only returns the integer portion of the result. The Mod operator and Remainder() function return the remainder after doing the division.

'Demonstrate the integer division and the Mod operator

Formula = 10 \ 3'Returns 3

Formula = 10 mod 3'Returns 1

Formula = Remainder(10, 3)'Returns 1

1. [↑](#footnote-ref-1)