ByRef in Functions and Subroutines

from the Liberty BASIC 4 helpfile

Table of Contents

ByRef in Functions and Subroutines

Passing Arguments into Subroutines and Fuctions - by value and BYREF

Example of passing by value

Example of passing by reference

More about pass by reference

Passing Arguments into Subroutines and Fuctions - by value and BYREF

Through Liberty BASIC 3, values can only be passed into a user subroutine or function by value. "Passing by value" is a common term meaning that once a value is passed into a subroutine or function, it is just a copy of the passed value and has no relationship to the original value. If the value is changed in the called subroutine or function, it does not change in the main program.

Now Liberty BASIC 4 allows us to pass variables by reference. This means that if a subroutine or function so declares, it can modify the value of a variable passed in, and when the subroutine or function ends and execution returns to the caller the change will be reflected in the variable that was used as a parameter in the call.

By default and without any direct instruction by the programmer, parameters passed into user defined functions and subroutines in QBasic and Visual Basic are passed by reference. That is not true in Liberty BASIC, where values are passed by value as the default. Passing by reference is only done when the BYREF keyword is used.

Example of passing by value

This example shows how passing by value works. The only value that comes back from the function is the one assigned to result$ when the function call returns.

```
    'this is the way it always worked
    x = 5.3
    y = 7.2
    result$ = formatAndTruncateXandY$(x, y)
    print "x = "; x
    print "y = "; y
    print result$
    end

function formatAndTruncateXandY$(a, b)
    a = int(a)
    b = int(b)
    formatAndTruncateXandY$ = str$(a)+", "+str$(b)
end function
```

Example of passing by reference

In contrast to the "passing by value" example, each of the parameters in the function in this example are to be byref (pass by reference). This means that when the value of a and b are changed in the function that the variables used to make the call (x and y) will also be changed to reflect a and b when the function returns. Try stepping through this example in the debugger.

```
    'now you can pass by reference
    x = 5.3
    y = 7.2
    result$ = formatAndTruncateXandY$(x, y)
    print "x = "; x
    print "y = "; y
    print result$
    'and it works with subroutines too
    wizard$ = "gandalf"
    call capitalize wizard$
    print wizard$
    end

function formatAndTruncateXandY$(byref a, byref b)
    a = int(a)
    b = int(b)
    formatAndTruncateXandY$ = str$(a)+", "+str$(b)
end function

sub capitalize byref word$
    word$ = upper$(left$(word$, 1))+mid$(word$, 2)
```

```
end sub
```

More about pass by reference

Passing by reference is only supported using string and numeric variables as parameters. You can pass a numeric or string literal, or a computed number or string, or even a value from an array, but the values will not come back from the call in any of these cases. Step through the example in the debugger to see how it works!

```
    'you can also call without variables, but the changes
    'don't come back
    result$ = formatAndTruncateXandY$(7.2, 5.3)
    print result$
    'and it works with subroutines too
    call capitalize "gandalf"
    a$(0) = "snoopy"
    call capitalize a$(0)
    end

function formatAndTruncateXandY$(byref a, byref b)
    a = int(a)
    b = int(b)
    formatAndTruncateXandY$ = str$(a)+", "+str$(b)
end function

sub capitalize byref word$
    word$ = upper$(left$(word$, 1))+mid$(word$, 2)
end sub
```