Josip Zohil, Koper, September, 2014

# Returning two parameters in VFP

In programming we mostly use functions, that return one value. There are various techniques how to return multiple values. In this article I shall show you how to return a pair of values using a function.

We return values to do something with them: display, store in a dbf file, elaborate them... We receive values, for example, from a function and pass them to another. If we don't pass values, we don't need a computer program.

## Two functions

Normally, we write programs in a sequential way: Insert data, elaborate , store or display and so on. This process may be described as a chain of functions: Compute the first function and pass the result to the next one and so on. We shall start from the second function. Let us say, we need two values x and y to display them on the screen, for example, a VFP function is:

FUNCTION twoparam(x,y)

?x,y

RETURN x+y

ENDFUNC

This function accepts two parameters x and y and displays them on the screen.

## Promise

Let us describe this function as an operation we shall do, when we receive the two values from a previous function. (This two values can arrive from everywhere, also from the WEB, or from a database). In practice, it may happens, this two values never arrives, so we look at this function as a promise: if the values arrive, a promise will be fulfilled, otherwise failed. Let us observe only the first possibility.

## Resolve

We are expecting two values. For example, they may be a result of a computation:

function f(x)

local y

y=x+3

return x,y && pseudo code

This function has to return two values x and y. A VFP return command returns only one value, so we shall modify it:

FUNCTION retTwo(x,twofun)

LOCAL y

y=x+3

RETURN twoFun+"("+STR(x)+","+STR(y)+")"

A new function accept as a parameter an expression (twofun) and a numeric value x. It returns an expression.

Visually we see a function retTwo is a composition of two functions. When we compute this composition (in our case in a successful way), we say, we have resolved a promise.

A function retTwo must be resolved (executed) in a successful way; in such a case it returns the two values and we can compute our promise function. Let us repeat: A promise is "waiting" for two values.

Note: In various languages we see a function promise as a done or then function.

## Resolved

Let us declare (not yet execute) a function:

res=retTwo(3,"twoparam")

We pass a value 3 as a first argument and "twoparam" as a second. A second argument is an expression, in our example a string containing a function name.

Note. VFP programmers use frequently this syntax: Bindevent, SQL query … This function is similar to a query declaration.

Now we execute (resolve) a declared function; it is written as an expression.

res1=EVALUATE(res)

This command display on the screen two numbers: 3 and 6.

We have passed (implicitly) two values from the first to the second function.

## Unfulfilled promise

Let us now pass to a more involved case, when we return two groups of parameters:

* for a resolved case
* in case we receive errors (unfulfilled promise).

We may receive errors notifications in various ways: Exception object, error message string,…. We shall receive an error as a string and publish it using a simple function:

FUNCTION perror(x)

?"Error:",x

RETURN x

endfunc

This is a promise in case of errors. It is a function that is waiting for a single argument ( a string).

Let us modify a function we have as a resolver:

FUNCTION retTwo(x,twofun,efun)

LOCAL y

try

y=x+3

z=t && generates an error

z=twoFun+"("+STR(x)+","+STR(y)+")"

CATCH TO ex

z=eFun+'("'+ex.message+'")' && returns an error value

endtry

RETURN z

A function retTwo accepts three parameters x, a value to be computed,

a success function funTwo and

an error function.

Visually we see it returns two expressions, for a resolved case and for a non resolved case (error).

Here is a declaration for a successful and unsuccessful case:

res=retTwo(3,"twoparam","perror") (A)

A declaration syntax is similar to a bindevent and raisevent functions.

We execute in this way:

res1=EVALUATE(res)

Inside a function retTwo are two functions: A compute function and a resolve function: If it resolved, it returns a two parameter value, otherwise a one parameter error.

## Conclusion

We have a function with variable output: two or one value. A declaration (A) is relatively simple (no if, do case or similar commands), all the complexity (with included error protection) is hidden in a function retTwo.

A better way would be, if we separate the function retTwo in two functions: computation and resolver.

May be we can write this composition using objects; for a VFP programmer a more familiar syntax.