Cool Tool of the Month: Array Browser

By Whil Hentzen

[Sidebar text]

View the contents of any array in memory - at any time - with this standalone function. A FoxTalk exclusive.

[Article]

Displaying the contents of memory variables during program execution is a critical capability to have during debugging - but the tools that FoxPro provides - DISPLAY MEMORY, developer-created Wait Window messages, and the Debug Window - aren't always sufficient.

The Problem

This is especially the case if you're working with arrays. The options we have when working with regular memory variables are useful only in severely limited cases.

An array with more than four or five elements is nearly impossible to display in a Wait Window message. Using the Debug window doesn't have much more flexibility - you're limited to 16 expressions, and even with Cut and Paste, it's a nuisance to enter the array name and subscripts.

The Display Memory command is unlimited in how many array elements can be listed - but it shows the elements in a single column - making the examination of a two-dimensional array fall somwhere between fairly difficult (with small arrays) to virtually impossible (for arrays with more than about 50 elements). Display Memory is also not always available - requring you to insert it in your program code or to run it while your program is Suspended.

We need a better tool - a cool tool - to view the contents of an array, and we need to be able to use this tool in a variety of situations - interactively, while a program is Suspended, and within an application.

The Solution

Frustrated with the limitations discussed above, I sat down one day last fall with the intent of whipping up a quick little utility (Have you ever had one of those "Oh, this should just take me an hour or two" moments?) that would place the contents of an array into a cursor, and then browse that cursor.

An hour with the Language Reference convinced me that there was no easy way to do so. The APPEND FROM ARRAY command assumed the existence of a properly designed cursor or table, and there wasn't any quick way to build the cursor from the array.

Nonetheless, I threw together a little routine that determined the type and size of each column, built a cursor, and did the APPEND FROM MEMO. Worked like a champ. Until it broke. The more I used it, the more things I wanted it to do. And I eventually figured that other developers were probably having the same problems with viewing arrays. So I came up with a number of design parameters for a tool suitable for use by others. These include:

The Array Browser tool had to be easy - and nearly foolproof - to use. If you had to remember, and pass, five or six parameters to make it work, you wouldn't use it. Second, it had to trap for a number of typical errors - like forgetting to pass the name of an array, or passing a bad name. Third, since this would be used in the debugging process - oftentimes during Suspend - it had to be extremely careful about not stepping on the environment.

Finally, and this was important, it could make _no_ assumptions about the array it was going to view. After all, this was a debugging tool - and assuming that the array was properly created and populated would clearly be a mistake. In other words, I had to figure that the array might have mismatched data, missing elements, rows or columns, and the columns might not even contain the same type of data.

How to use it

Using the Array Browser couldn't be easier (trust me!). To try it out, create an array named XXX, say, and then from the Command Window, issue the command

```
=AB("XXX")
```

If the array name you passed is valid, Array Browser will start cranking and in a few (or a lot, depending on your environment and the size of the array) seconds, a browse window displays with the array.

The field heading for each column indicates the column number and the first column in the browse shows the row number - so that you can find a specific array element immediately.

The cursor is named ZYXWVUTS and is left open after you terminate the Browse, so you can pull it up again.

Special conditions

You don't have to specify what types of data, whether the array is one or two-dimensional, how many elements it has, or anything else. Array Browser automatically handles all cases.

If you forget to pass a name, you will be gently reminded. If you pass a bad array name, memory will be searched and a list of valid arrays will be presented for you to pick from. If no arrays exist, you will be notified and Array Browser will terminate.

Once Array Browser starts working on an array, it analyzes each column to determine if the entire column contains the same type of data. If not, the corresponding column in the cursor is created as Character. The size of the largest element in each column is also calculated so that each column of the cursor is created appropriately sized.

If a column type is changed to Character, the column is automatically sized to be a minimum of 10 characters wide so that numeric values later encountered are not truncated. This could have been checked for each row and a flag could be set, but it complicated the code and slowed the process way down. It shouldn't prove to be a hindrance - or even noticable - in most uses.

Possible enhancements

With the exception of possible bugs, I thought I was done with this tool when I sent it out for testing. Thanks to a loyal crew of AB Breakers - Frank Bosso, Dan Freeman, Geary Rachel, Vince Rice, Ceil Rosenfeld, Rick Strahl, Norm Strawser and Randy Wallin - the list of enhancements is already forming.

Hot on several people's list was the ability to write the contents of a modified browse back to the array. Holding us back on this is the possibility that the value in the array is not of the same type as the initial array - if a column had several types of data, the column would be created as Character, and data in that column shouldn't be written back to the array. I'm thinking about having a flag that indicates whether or not a column was created as it's original type.

Other requested enhancements include (1) the facility for a filter for the browse to be passed along with the array name, (2) the ability to browse two arrays at the same time (I think we'll have to wait for a better Browse to do this), and (3) a browse that is dynamically updated during program execution - like the values in the Debug Window are.

Your feedback

Array Browser is written in FoxPro and the verbosely commented source is included. I'd be interested in your feedback on the tool as well. Please do so via CompuServe's FoxForum or email at 70651,2270.

Where to find it

Array Browser is a FoxTalk exclusive - you can find it on this month's Companion Diskette. Bug fixes (gasp!) and updates (if any) will follow on future Companion Diskettes.