July, 1996

Advisor Answers

FoxPro 2.x and Visual FoxPro

Q: I am a relatively new user. I am trying to figure out the correct syntax for copying a column from a multidimensional array into another array. I would appreciate any advice you could give.

–Matthew Anderson (via CompuServe)

A: FoxPro's ACOPY() function copies an entire array or a set of rows from one array to another, but there's no native way to copy one column of an array to another array. In fact, some of the other array manipulation functions seem to give columns short shrift, too.

The reason columns are second-class citizens in FoxPro is because two-dimensional arrays aren't really two-dimensional. They're stored sequentially in what's known as row-major order, with all the data from the first row, then all the data from the second row and so forth. In fact, any two-dimensional array can be addressed as if it were one-dimensional: aMyArray[1,1] is the same as aMyArray[1]. The AELEMENT() and ASUBSCRIPT() functions let you convert between the two forms of addressing an array element.

The upshot of this storage technique is that it's easy for FoxPro to grab a row's worth of data, but much harder to grab a column. So neither ACOPY() nor ASORT() address columns.

Of course, there is a way to copy one or more columns. It's just tedious. Here's a function called ACOLCOPY that does the trick. It accepts the same parameters as ACOPY() with a couple of differences. First, the arrays have to be passed by reference (that is, preceded with "@"), since this isn't a built-in function. In addition, the third parameter which specifies the first column to copy is required. If you need to copy the entire array, you can use ACOPY(), so you have to indicate which columns are to be copied.

The function returns the number of elements copied or -1, if something prevents the copying. The destination array is redimensioned, if necessary. If there aren't enough columns or rows, it's enlarged to have enough. If the array is too large, however, the extra data locations are left alone.

```
*FUNCTION aColCopy
* Copy one or more columns from a source array
* to a destination array.

LPARAMETERS aSource, aDest, nStartCol, nColCount, nDestCol
  * aSource = array to be copied
  * aDest = destination array
  * nStartCol = first column to copy - required
  * nColCount = number of columns to copy - optional.
  *             Go to end of aSource, if omitted
  * nDestCol = first column of destination to receive
```

```
  *            copied data - optional
  *            1 if omitted

LOCAL nOldRows, nOldCols, nOldCount, nItem

* Check source array
IF TYPE("aSource[1]")="U" OR ALEN(aSource,2)=0
  * not a 2-d array, can't do it
  RETURN -1
ENDIF

* Check for starting column
IF TYPE("nStartCol")<>"N"
  RETURN -1
ENDIF

* Check number of columns. Compute if necessary
IF TYPE("nColCount")<>"N" OR nStartCol+nColCount>ALEN(aSource,2)
  nColCount=ALEN(aSource,2)-nStartCol+1
ENDIF

* Check destination column.
IF TYPE("nDestCol")<>"N"
  nDestCol=1
ENDIF

* Check destination array for size. It must exist to be
* passed in.
* First, make sure it's an array.
* Then, see if it's shaped right for all the data.
* Two cases - if enough cols, but not enough rows, just add
* If not enough cols, have to move data around.
IF TYPE("aDest[1]")="U"
  DIMENSION aDest[ALEN(aSource,1),nColCount+nDestCol-1]
ELSE
  IF ALEN(aDest,2)>=nColCount+nDestCol-1  && enough columns
    IF ALEN(aDest,1)<ALEN(aSource,1)      && not enough rows
      * add some
      DIMENSION aDest[ALEN(aSource,1),ALEN(aDest,2)]
ENDIF
  ELSE
    * now the hard one
    * not enough columns, so need to add more
    * (and maybe rows, too)
    nOldRows=ALEN(aDest,1)
    nOldCols=ALEN(aDest,2)
    nOldCount=ALEN(aDest)
    DIMENSION aDest[MAX(nOldRows,ALEN(aSource,1)), ;
                    nColCount+nDestCol-1]

    * DIMENSION doesn't preserve data location,
    * so we need to adjust the data
    * We go backward from the end of the array toward
    * the front, moving data down, so we don't overwrite
    * any data by accident

    FOR nItem=nOldCount TO 2 STEP -1
      * Use new item number and old dimensions to determine
      * new item number for each element
      IF nOldCols<>0
        nRow=CEILING(nItem/nOldCols)
```

```
        nCol=MOD(nItem,nOldCols)
        IF nCol=0
          nCol=1
        ENDIF
      ELSE
        nRow=nItem
        nCol=1
      ENDIF

      aDest[nRow,nCol]=aDest[nItem]
    ENDFOR
  ENDIF
ENDIF

* finally ready to start copying
FOR nCol=1 TO nColCount
  FOR nRow=1 TO ALEN(aSource,1)
    aDest[nRow,nDestCol+nCol-1]= ;
         aSource[nRow,nStartCol+nCol-1]
  ENDFOR
ENDFOR

RETURN nColCount*ALEN(aSource,1)
```

This function should let you do whatever copying of columns you need without having to worry about the details.

–Tamar