

Multi-Dimensional Arrays in 4th Dimension

Djundi Karjadi • Joshua Wachs

© Natural Intelligence Consulting. All Rights Reserved.

If used commercially, please give credit in comments of procedures.

Have you ever wondered if you can do Multi-Dimensional arrays in 4th Dimension? Well...now you can (do them that is, not wonder about them).

All you need are the nice tricks shown below:

The process consists of two procedures and function written directly in a 4D procedure.

Procedures:

Create2D_Array (ArrayName; sizeX; sizeY)

This creates a "pseudo" 2-dimensional array called ArrayName.

Set2D_Array (ArrayName; locX; locY; Value)

This procedure will put "Value" into ArrayName (locX; locY) .

Function:

Get2D_Array (ArrayName; locX; locY)

This function will return the value in ArrayName (locX; locY)

Let's let the source code speak for itself (paste it into 4D for correct formatting):
(NOTE: Print out the procedures or do a copy/paste for complete accuracy!)

-
- ` Procedure Create2D_Array
- ` This Procedure takes 3 Parameters
- ` 1) ArrayName : String
- ` 2) X dimension : Integer
- ` 3) Y Dimension : Integer
- ` This procedure will create a 2-Dimensional array of size X by Y
- ` And create 1 global variable named: ArrayNameY for mapping purposes
- ` Example usage:
- ` Create2D_Array("List";10;30)

```

ArrayName:=$1 ` Copying to a temp Global Variable
xCoord:=$2    ` Simply Because Execute Can not process local Vars
yCoord:=$3
EXECUTE(ArrayName+"0:=(xCoord*yCoord)") ` ArrayVar0:=Xdim*Ydim
EXECUTE(ArrayName+"Y:=yCoord") ` Saves the value Y Dimension of the array to
                                ` a global var ArrayNameY

CLEAR VARIABLE("ArrayName")
CLEAR VARIABLE("xCoord")
CLEAR VARIABLE("yCoord") ` We don't need you anymore

```

-
- ` Procedure Set2D_Array
- ` This procedure takes 4 Parameters
- ` 1) ArrayName
- ` 2) X Coord
- ` 3) Y Coord
- ` 4) The value to store
- ` Example Usage:
- ` Set2D_Array("List";5;7;"Natural Intelligence Consulting meets Acius")

```

ArrayName:=$1
xCoord:=$2
yCoord:=$3
toStore:=$4
EXECUTE(ArrayName+"{{{(xCoord-1)*"+ArrayName+"Y+yCoord}}}:=toStore")
CLEAR VARIABLE("ArrayName";"xCoord";"yCoord";"toStore")

```

-
- ` Function Set2D_Array
- ` This Function takes 3 Parameters
- ` 1) ArrayName
- ` 2) X Coord
- ` 3) Y Coord
- ` and it will return the value in \$0
- ` Example Usage:
- ` Current:=Get2D_Array("List";5;9)

```

ArrayName:=$1
xCoord:=$2
yCoord:=$3
EXECUTE("tempResult:="+ArrayName+"{{{(xCoord-1)*"+ArrayName+"Y+yCoord}}}")
` Maps the 2D array to the 1D array
$0:=tempResult
CLEAR VARIABLE("ArrayName";"xCoord";"yCoord";"tempResult")

```

You can create as many Arrays as your Macintosh's memory will allow. Since this procedure is actually a 1-Dimensional array mapped to a 2-Dimensional array, the maximum # of elements allowed is the Maximum # allowed in a 4D one dimensional array. I think is somewhere about 32000, so you probably can create an array of 32 by 1000, memory permitting.

Note: I used some global variables because the **Execute** command can not process a local variable. So... make sure you are not using the same global names or something is going to get clobbered!

Any comments, criticisms, improvements or otherwise, please contact us at the addresses/mailboxes below. Also available from Natural Intelligence Consulting:

- 4D Say 1.0.....The Macintalk speech driver for 4th Dimension
-• 4D Play 1.0 Plays SoundCap, SoundWave, SoundEdit, and snd resources from 4D. [Coming soon.]
-• And more! Look for these and future great utilities from Natural Intelligence Consulting on your favorite BBS or on-line network.

Djundi Karjadi
Joshua Wachs
Natural Intelligence Consulting

MacNet: JUNDI, NATURAL
Genie:XTH57382
Delphi: NATURAL
CIS: 72427,177