**ARRAYS EXAMPLES**

**data** attractions;

input City $ **1**-**9** Museums **11** Galleries **13**

Other **15** TourGuide $ **17**-**24** YearsExperience **26**;

cards;

Rome 4 3 . D'Amico 2

Paris 5 . 1 Lucas 5

London 3 2 . Wilson 3

New York 5 1 2 Lucas 5

Madrid . . 5 Torres 4

Amsterdam 3 3 . .

;

**run**;

**proc** **print** data=attractions;

title 'Data Set ATTRACTIONS';

**run**;

**data** changes;

set attractions;

if Museums = **.** then Museums = **0**;

if Galleries = **.** then Galleries = **0**;

if Other = **.** then Other = **0**;

**run**;

**data** changes1;

set attractions;

array changelist{**3**} Museums Galleries Other;

do Count = **1** to **3**;

if changelist{Count} = **.** then changelist{Count} = **0**;

end;

**run**;

**proc** **print** data=changes1;

title 'Tour Attractions';

**run**;

**data** changes2 (drop=Count);

set attractions;

array changelist{**3**} Museums Galleries Other;

do Count = **1** to **3**;

if changelist{Count} = **.** then changelist{count} = **0**;

end;

**run**;

**proc** **print** data=changes2;

title 'Tour Attractions';

**run**;

**data** test (drop=i);

array a(**10**) A01-A10;

do i=**1** to **10**;

a(i)=i;

end;

**run**;

**proc** **print** noobs data=test;

**run**;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The DIM function is often used with the iterative DO statement to return

the number of elements in a dimension of an array, when the lower bound of the dimension is 1.

If you use DIM, you can change the number of array elements without changing the upper bound of the DO statement.

For example, because DIM(NEW) returns a value of 4, the following statements process all the elements in the array:

DIM function - Returns the number of elements in an array.

HBOUND Function - Returns the upper bound of an array.

HBOUND Function -Returns the lower bound of an array.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*/

**data** test1 (drop=i);

array new{\*} score1-score4 (**10** **20** **30** **40**);

do i=**1** to dim(new);

new{i}=new{i}+**10**;

end;

**run**;

**proc** **print** noobs data=test1;

**run**;

\*Example 1: Using Character Variables in an Array;

**data** text;

array names{\*} $ n1-n10;

array capitals{\*} $ c1-c10;

input names{\*};

do i=**1** to **10**;

capitals{i}=upcase(names{i});

end;

datalines;

smithers michaels gonzalez hurth frank bleigh

rounder joseph peters sam

;

**proc** **print** data=text;

title 'Names Changed from Lowercase to Uppercase';

**run**;

**data** score1(drop=i);

array test{**3**} t1-t3 (**90** **80** **70**);

array score{**3**} s1-s3;

input id score{\*};

do i=**1** to **3**;

if score{i}>=test{i} then

do;

NewScore=score{i};

output;

end;

end;

datalines;

1234 99 60 82

5678 80 85 75

;

**proc** **print** noobs data=score1;

title 'Data Set SCORE1';

**run**;

/\*Example 3: Creating an Array for Temporary Use in the Current DATA Step

When elements of an array are constants that are needed only for the duration of the DATA step,

you can omit variables from an array group and instead use temporary array elements. You refer to

temporary data elements by the array name and dimension. Although they behave like variables, temporary array elements

do not have names, and they do not appear in the output data set. Temporary array elements are automatically retained,

instead of being reset to missing at the beginning of the next iteration of the DATA step.

To create a temporary array, use the \_TEMPORARY\_ argument. The following example creates a temporary array named TEST:

\*/

**data** score2(drop=i);

array test{**3**} \_temporary\_ (**90** **80** **70**);

array score{**3**} s1-s3;

input id score{\*};

do i=**1** to **3**;

if score{i}>=test{i} then

do;

NewScore=score{i};

output;

end;

end;

datalines;

1234 99 60 82

5678 80 85 75

;

**proc** **print** noobs data=score2;

title 'Data Set SCORE2';

**run**;

/\*Example 4: Performing an Action on All Numeric Variables

This example multiplies all the numeric variables in array TEST by 3\*/

**data** sales;

infile datalines;

input Value1 Value2 Value3 Value4;

datalines;

11 56 58 61

22 51 57 61

22 49 53 58

;

**data** convert(drop=i);

set sales;

array test{\*} \_numeric\_;

do i=**1** to dim(test);

test{i} = (test{i}\***3**);

end;

**run**;

**proc** **print** data=convert;

title 'Data Set CONVERT';

**run**;

**DATA** TIMES;

INPUT TIME1 TIME2 TIME3 TIME4;

ARRAY T[**4**] TIME1-TIME4;

SUMTIME=**0**;

DO I=**1** TO **4**;

SUMTIME=SUMTIME+T(I);

END;

DATALINES;

22.3 25.3 28.2 30.6

22.8 27.5 33.3 35.8

18.5 26.0 29.0 27.9

22.5 29.3 32.6 33.7

;

**RUN**;

**PROC** **PRINT** DATA=TIMES;**RUN**;

|  |  |
| --- | --- |
| **Variable** | **Array reference** |
| c1t1 | temprg{1,1} |
| c1t2 | temprg{1,2} |
| c2t2 | temprg{2,2} |
| c2t5 | temprg{2,5} |

array-name {index-variable-1, ...,index-variable-n}

The following example creates an array that contains ten variables- five temperature measures (t1 through t5) from two cities (c1 and c2). The DATA step contains two DO loops.

* The outer DO loop (DO I=1 TO 2) processes the inner DO loop twice.
* The inner DO loop (DO J=1 TO 5) applies the ROUND function to all the variables in one row.

For each iteration of the DO loops, SAS substitutes the value of the array element corresponding to the current values of I and J.

options nodate pageno=1 linesize=80 pagesize=60;

data temps;

array temprg{2,5} c1t1-c1t5 c2t1-c2t5;

input c1t1-c1t5 /

c2t1-c2t5;

do i=1 to 2;

do j=1 to 5;

temprg{i,j}=round(temprg{i,j});

end;

end;

datalines;

89.5 65.4 75.3 77.7 89.3

73.7 87.3 89.9 98.2 35.6

75.8 82.1 98.2 93.5 67.7

101.3 86.5 59.2 35.6 75.7

;

proc print data=temps;

title 'Temperature Measures for Two Cities';

run;