



Arrays Made Easy: An Introduction to Arrays and Array Processing

2nd Dimension	SALE_ARRAY		{r,1}	{r,2}	{r,3}	{r,4}	...	{r,12}
1st Dimension	Sales Variables	{1,c}	SALES1	SALES2	SALES3	SALES4	...	SALES12
	Expense Variables	{2,c}	EXP1	EXP2	EXP3	EXP4	...	EXP12
	Commission Variables	{3,c}	COMM1	COMM2	COMM3	COMM4	...	COMM12



SYSTEMS SEMINAR CONSULTANTS, INC.

Teresa Schudrowitz
2997 Yarmouth Greenway Drive
Madison, WI 53711
(608) 278-9964
train@sys-seminar.com

Introduction



- Many programmers often find arrays daunting
- A SAS array is a convenient way of temporarily identifying a group of variables for processing within a data step
- Arrays are a simple solution to many program scenarios

Topics



- Why do we need arrays?
- Basic array concepts
 - Definition
 - Elements
 - Syntax
 - Rules
- Using array indexes
- One dimension arrays
- Multi-dimension arrays
- Temporary arrays
- Explicit vs. implicit subscripting
- Sorting arrays
- When to use arrays
- Common errors and misunderstandings

What is An Array?



- Most mathematical and computer languages have some notation for repeating or other related values.
- Often called :
 - matrix
 - vector
 - dimension
 - in SAS data step, called an array

Natural Array



Everyone in this room is part of a natural array!

We can refer to each individual by name or by seat location.

Natural Array



Room

Column

1 2 3 4 5 6

Row

1

2

3

4

5

6

7

8

9

10

Natural Array



The winner is:

```
DATA SELECT_WINNER;  
  SET NAME_DS;  
  ARRAY AUDIENCE_ARRAY {50, 20} $ NAME1-NAME1000;  
  ...  
  WINNER = AUDIENCE_ARRAY{ROW, SEAT};  
RUN;
```

What is Different About SAS Arrays?



The SAS Array definition is a group of related variables that are already defined in a data step.

Differences:

- SAS array elements don't need to be contiguous
- SAS elements don't need the same length
- Elements don't need to even be related at all

Notes:

- All elements must be the same type, either all character or all numeric.

Why Do We Need Arrays?



The use of arrays may allow us to simplify processing.

Arrays can be used to:

- read data
- perform repetitive calculations
- perform table lookups
- create several related variables
- rotate datasets

Why Do We Need Arrays?



Convert temperatures from Fahrenheit to Celsius for all 24 temperatures.

```
DATA CONVERT_TEMP;  
  INPUT etc.  
  CELSIUS_TEMP1 = 5/9*(TEMP1 - 32);  
  CELSIUS_TEMP2 = 5/9*(TEMP2 - 32);  
  ...  
  CELSIUS_TEMP24 = 5/9*(TEMP24 - 32);  
RUN;
```



Defining an Array

An array and a loop can make the program smaller.

```
DATA CONVERT_TEMP;  
  INPUT etc.  
  ARRAY TEMPERATURE_ARRAY{24} TEMP1-TEMP24;  
  ARRAY CELSIUS_ARRAY{24} CELSIUS_TEMP1-CELSIUS_TEMP24;  
  DO I=1 TO 24;  
    CELSIUS_ARRAY{I} = 5/9*(TEMPERATURE_ARRAY{I} - 32);  
  END;  
RUN;
```



Defining an Array

- Arrays work for a few elements or hundreds of elements
- Variables used as elements do not need to be named consecutively

```
array sample_array {5} x a i r d;
```



Array Statement

The ARRAY statement defines variables as a group.

Syntax:

ARRAY *arrayname* {*n*} <\$> <*length*> *array-elements* <(initial values)>;

<i>arrayname</i>	Any valid SAS name
<i>n</i>	Number of elements or *
\$	Elements are character variables
<i>length</i>	Common length for array elements
<i>array-elements</i>	SAS variables to be part of array
<i>Initial values</i>	Initial values for each of the array elements

Array Statement



- The ARRAY statement is a compiler statement
- Array elements cannot be used in compiler statements like DROP or KEEP
- Arrays must be defined before array can be referenced
- Arrays must be defined in every step that uses them

Array Statement - Elements



Array elements must be all numeric or all character.

Special variables may be used to select all variables or all variables of a select type:

- `_NUMERIC_` - special variable to identify all numeric variables as array elements
- `_CHARACTER_` - special variable to identify all character variables as array elements
- `_ALL_` - special variable to identify all variables as array elements

Array Statement - N



- Defines the array subscript
- Refers to the number of elements in the array
- Values allowed for N are:
 - Numeric constant
 - Variable whose value is numeric
 - Numeric SAS expression
 - The asterisk (*)
- Array subscript must be enclosed within:
 - Braces {}
 - Square brackets []
 - Parentheses ()

Array Statement – N = *



When the array subscript is an asterisk (*) it is not necessary to know how many elements are within the array.

```
array allnums {*} _numeric_;
```

The DIM function can be used to return the count of elements:

```
do i = 1 to dim(allnums);  
  allnums{i} = round(allnums{i}, .1);  
end;
```



Array References

When an array is defined with an ARRAY statement, an array reference is created.

`array - name {n}`

- The value of n is the element's position within the array
- The variable name and the array reference are interchangeable
- An array reference may be used with the data step in almost any place other SAS variables may be used

Variable Name	Array Reference
temp1	temperature_array{1}
temp2	temperature_array{2}
temp3	temperature_array{3}
temp4	temperature_array{4}
temp5	temperature_array{5}

Array Indexes



- The array index is the range of array elements
- By default, the array subscript is 1-based
- The array index may be modified to begin with a lower bound other than 1

```
array temperature_array {12:24} temp12-temp24;
```



One Dimension Arrays

Logically a one dimension array appears as a single row

```
array temperature_array {24} temp1-temp24;
```

Program Data Vector:

temperature_array	{1}	{2}	{3}	{...}	{24}
Temperature variables	temp1	temp2	temp3	...	temp24

Reference to 9th element in the array:

```
y = temperature_array{9};
```



Multi-Dimensional Arrays

Two or more dimensions allows us to conceptually group variables as rows, columns, pages, etc.

Example: Create 12 amounts for Sales, expenses, and commissions in a single array.

data etc.

```
array sale_array {3, 12} sales1-sales12 exp1-exp12 comm1-comm12;  
x=sale_array{2,6}; /* refer to sixth expense variable (exp6). */
```

2nd Dimension	SALE_ARRAY		{r,1}	{r,2}	{r,3}	{r,4}	...	{r,12}
1st Dimension	Sales Variables	{1,c}	SALES1	SALES2	SALES3	SALES4	...	SALES12
	Expense Variables	{2,c}	EXP1	EXP2	EXP3	EXP4	...	EXP12
	Commission Variables	{3,c}	COMM1	COMM2	COMM3	COMM4	...	COMM12



Another Multi-Dimensional Array

```
DATA MONTHLY;
  INFILE SAMP;
  INPUT @ 1 ACCT          $CHAR19.
        @ 20  CURLIMIT    5.
        /* READING 6 MONTHLY FIGURES */
        /*   FOR 3 SETS OF VALUES   */
        @ 127 BALANCE1     PD6.2
        @ 133 BALANCE2     PD6.2
        @ 139 BALANCE3     PD6.2
        @ 145 BALANCE4     PD6.2
        @ 151 BALANCE5     PD6.2
        @ 157 BALANCE6     PD6.2
        @ 163 PAYMENT1     PD6.2
        @ 169 PAYMENT2     PD6.2
        @ 175 PAYMENT3     PD6.2
        @ 181 PAYMENT4     PD6.2
        @ 187 PAYMENT5     PD6.2
        @ 193 PAYMENT6     PD6.2
        @ 199 AMT_DUE1     PD6.2
        @ 205 AMT_DUE2     PD6.2
        @ 211 AMT_DUE3     PD6.2
        @ 217 AMT_DUE4     PD6.2
        @ 223 AMT_DUE5     PD6.2
        @ 229 AMT_DUE6     PD6.2
        @ 271 SSN         PD5.
        ;
        /* ARRAYS - VARIABLES FROM INPUT */
        /*   STATEMENT                   */
        ARRAY BAL{6} BALANCE1-BALANCE6;
        ARRAY PAY{6} PAYMENT1-PAYMENT6;

        ARRAY DUE{6} AMT_DUE1-AMT_DUE6;
```

**Try
this:**

```
DATA MONTHLY;
  INFILE SAMP;
  INPUT @ 1 ACCT          $CHAR19.
        @ 20  CURLIMIT    5.
        /* PLACE POINTER @127 */
        @127
        /* HOLD ROW(TRAILING @)*/
        @;
        /* 2 DIMENSIONAL ARRAY */
        /* 3 ROWS - AS FOLLOWS */
        /* 1 - BALANCE1-6       */
        /* 2 - PAYMENT1-6       */
        /* 3 - AMT_DUE1-6       */
        ARRAY AINFO{3,6}
                BALANCE1-BALANCE6
                PAYMENT1-PAYMENT6
                AMT_DUE1-AMT_DUE6;
        /* READ INTO ARRAYS */
        DO ROW = 1 TO 3;
          DO COL = 1 TO 6;
            * READ 6 MONTHS FOR 3 VARS;
            INPUT AINFO{ROW,COL} PD6.2 @;
          END;
        END;
        INPUT @271 SSN PD5. ; etc...
```

Temporary Arrays



When elements are constants needed only for duration of DATA step, you can omit variables from an array and instead use temporary array elements.

- You refer to temporary data elements by the array name and dimension
- Temporary array elements behave like variables
- Temporary array elements do not have names
- Array elements do not appear in the output data set
- They are automatically retained
- Temporary arrays are slightly faster than those referencing variables

An Financial Example



Multiple interest rates are imbedded in a series of SAS code.

```
data etc.
```

```
  . . .  
if month_del eq 1 then balance = balance + (balance * 0.05);  
else if month_del eq 2 then balance = balance + (balance * 0.08);  
else if month_del eq 3 then balance = balance + (balance * 0.12);  
else if month_del eq 4 then balance = balance + (balance * 0.20);  
else if month_del eq 5 then balance = balance + (balance * 0.27);  
else if month_del eq 6 then balance = balance + (balance * 0.35);  
  . . .  
run;
```



Using a Temporary Array

Define the rates in a temporary array, then use it.

data etc.

```
. . .  
array rate {6} _temporary_ (0.05 0.08 0.12 0.20 0.27 0.35);  
if month_del ge 1 and month_del le 6 then  
    balance = balance + (balance * rate{month_del});  
. . .  
run;
```



Setting Initial Values Elsewhere

You can omit initial values on the array and set them elsewhere in the data step.

```
data etc.
```

```
  . . .  
array rateb {6} _temporary_;  
do i = 1 to 6;  
  rateb{i} = i * 0.5;  
end;  
  . . .  
run;
```

An Implicit Array Example



Earlier versions originally defined arrays more implicitly.

```
data temp;
  input x1$ x2$ x3$ x4$ x5$ x6$ x7$ x8$ x9$ x10$ x11$ x12$;
  infile datalines;
array item(j) $ 12 x1-x12;
do over item;
  put item;
end;
datalines;
a b c d e f g h i j k l
;
run;
```

Another Implicit Array Example



If you prefer to reference an index, the index is set in one statement and the array reference in another.

```
data temp;
  input x1$ x2$ x3$ x4$ x5$ x6$ x7$ x8$ x9$ x10$ x11$ x12$;
  infile datalines;
  array item(j) $ 12 x1-x12;
  do j=1 to 12;
    put item;
  end;
datalines;
a b c d e f g h i j k l
;
run;
```

An Explicit Array Example



A much simpler syntax.

```
data temp;
  input x1$ x2$ x3$ x4$ x5$ x6$ x7$ x8$ x9$ x10$ x11$ x12$;
  infile datalines;
  array item(*) $ 12 x1-x12;
  do j=1 to 12;
    put item{j};
  end;
datalines;
a b c d e f g h i j k l
;
run;
```

Sorting Arrays



Two SAS 9.1 experimental call routines can sort array values.

- CALL SORTN will sort numeric arrays
- CALL SORTQ will sort character arrays
- Syntax is not in 9.1 documentation
- No facility to sort two related arrays

Sorting Arrays



```
data temp;
input x1 x2 x3 x4 x5 x6;
infile datalines;
array xarray(*) x1-x6;
put '** before ** ' x1-x6;
call sortn(of x1-x6);
put '** After ** ' x1-x6;
datalines;
0.27 0.12 0.20 0.08 0.35 0.05
;
run;
```

```
** before ** 0.27 0.12 0.2 0.08 0.35 0.05
```

NOTE: The SORTN function or routine is experimental in 9.1.

```
** After ** 0.05 0.08 0.12 0.2 0.27 0.35
```



Common Array Errors

Can you spot the error in this program?

```
DATA REPEATS;
  ARRAY SEXARRAY{4} $ 1 SEX1-SEX4;
  ARRAY AGEARRAY{4} AGE1-AGE4;
  DO UNTIL (I > 4);
    I+1;
    INPUT SEXARRAY{I} AGEARRAY{I} @;
  END;I=0;DROP I;
DATALINES;
M 12 F 13 M 16 M 17
M 11 M 18 F 12 F 18
;
RUN;
PROC PRINT DATA=REPEATS; TITLE 'REPEAT DATASET'; RUN;
```



Common Array Errors – Invalid Index Range

- DO UNTIL checks if true at the end of the loop
- DO UNTIL checks for array bounds of 1-5, array only defined as 1-4

Modify the program to stay within bounds.

Partial SAS log:

```
173 DATA REPEATS;
174     ARRAY SEXARRAY{4} $ 1 SEX1-SEX4;
175     ARRAY AGEARRAY{4} AGE1-AGE4;
176     DO UNTIL (I > 4);
177         I+1;
178         INPUT SEXARRAY{I} AGEARRAY{I} @;
179     END;I=0;DROP I;
180 DATALINES;
ERROR: Array subscript out of range at line 178 column 5
SEX1=M SEX2=F SEX3=M SEX4=M AGE1=12 AGE2=13 AGE3=16 AGE4=17 I=5 _ERROR_=1 _N_=1
ERROR: Array subscript out of range at line 178 column 5
SEX1=M SEX2=F SEX3=M SEX4=M AGE1=12 AGE2=13 AGE3=16 AGE4=17 I=5 _ERROR_=1 _N_=1
183     PROC PRINT DATA=REPEATS;TITLE 'REPEAT DATASET';RUN;
```



Function Name as Array Name

Using the same array name as a function can cause problems.

```
DATA REPEATS;  
  ARRAY SEXARRAY{4} $ 1 SEX1-SEX4;  
  ARRAY MEAN{4} AGE1-AGE4;  
  DO I=1 TO 4;  
    INPUT SEXARRAY{I} MEAN{I} @;  
  END;DROP I;  
  MEANAGE=MEAN(OF AGE1-AGE4);  
DATALINES;  
M 12 F 13 M 16 M 17  
M 11 M 18 F 12 F 18  
;  
RUN;  
PROC PRINT DATA=REPEATS;  
  TITLE 'REPEAT DATASET';  
RUN;
```



Function Name as Array Name

Partial SAS Log:

```
151+DATA REPEATS;
152+  ARRAY SEXARRAY{4} $ 1 SEX1-SEX4;
153+  ARRAY MEAN{4} AGE1-AGE4;
      ----

WARNING: An array is being defined with the same name as a SAS-supplied or
user-defined function. Parenthesized references involving this name will
be treated as array references and not function references.

154+  DO I=1 TO 4;
155+    INPUT SEXARRAY{I} MEAN{I} @;
156+  END;DROP I;
157+  MEANAGE=MEAN(OF AGE1-AGE4);
      ----

ERROR: Too many array subscripts specified for array MEAN.

NOTE: SAS STOPPED PROCESSING THIS STEP BECAUSE OF ERRORS.
```



Reference Array in Multiple Steps

```
DATA REPEATS;
  INPUT SEX1 $ AGE1 SEX2 $ AGE2 SEX3 $ AGE3 SEX4 $ AGE4;
  ARRAY AGEARRY{4} AGE1-AGE4;
  DO I=1 TO 4;
    AGEARRY{I}=AGEARRY{I}*12;
  END;DROP I;
DATALINES;
M 12 F 13 M 16 M 17
M 11 M 18 F 12 F 18
;
RUN;
DATA REPEAT2;
  SET REPEATS;
  SUM=0;
  DO I=1 TO 4;
    SUM+AGEARRY{I};
  END;
RUN;
PROC PRINT DATA=REPEAT2;
  TITLE 'REPEAT DATASET';
RUN;
```



Reference Array in Multiple Steps

Partial SAS Log:

```
206     DATA REPEATS;
207         INPUT SEX1 $ AGE1 SEX2 $ AGE2 SEX3 $ AGE3 SEX4 $ AGE4;
208         ARRAY AGEARRAY{4} AGE1-AGE4;
209         DO I=1 TO 4;
210             AGEARRAY{I}=AGEARRAY{I}12;
211         END;DROP I;
212     DATALINES;
NOTE: The data set WORK.REPEATS has 2 observations and 8 variables.
215     DATA REPEAT2;
216         SET REPEATS;
217         SUM=0;
218         DO I=1 TO 4;
219             SUM+AGEARRAY{I};
ERROR: Undeclared array referenced : AGEARRAY.
ERROR: Variable AGEARRAY has not been declared as an array.
```

Arrays need to be declared in every data step.



Know When to Use Arrays

When should an array be used in a data step?

- Data could be put into a row and operations performed across the data
- Data with lots of similar values
- Natural array
- Repetitive calculations
- Table lookup

Contact Us



SYSTEMS SEMINAR CONSULTANTS, INC.

SAS® Training, Consulting, & Help Desk Services

Teresa Schudrowitz and Steven J. First

(608) 278-9964

FAX (608) 278-0065

sfirst@sys-seminar.com

tschudrowitz@sys-seminar.com

www.sys-seminar.com

2997 Yarmouth Greenway Drive • Madison, WI 53711

