15. Processing variables with arrays
SAS Arrays

A SAS array is a temporary grouping of SAS variables under a single name. It exists only for the duration of the DATA step.

Useful for processing several variables at the same time:

- Perform repetitive calculations
- Create new variables
- Initialize variables with values
ARRAY statement

\[
\text{ARRAY array-name\{dimension\} <elements>;} \\
\text{- array-name specifies the name of the array} \\
\text{- dimension describes the number and arrangement} \\
\text{of array elements} \\
\text{- elements lists the variables to include in the array.} \\
\text{They must be either all numeric or all character}
\]

It is a compile-time only statement
Specifying the name of the array

```
ARRAY array-name{dimension} <elements>;
```

- Do not give an array the same name as a variable in the same DATA step.
- Avoid using the name of a SAS function: you won’t be able to use the function in the same DATA step.
- You cannot use array names in LABEL, FORMAT, DROP, KEEP, or LENGTH statements.
Specifying the dimension

```
ARRAY array-name{dimension} <elements>;
```

The dimension of the array can be specified as:

- The number of array elements (in a one-dimensional array) (e.g.: \{4\})
- A range of values (e.g.: \{96-99\})
- * : SAS determines the dimensions of the array by counting the number of elements
- Dimensions can be enclosed in parentheses, (), braces {}, or brakets []
Specifying array elements

The order of the element of the array is the one specified in the ARRAY statement.

The elements of the array can be specified as:
- A set of elements separated by a space (e.g. qtr1 qtr2 qtr3 qtr4 gio1 gio2 luc marie)
- A numbered range of variables
  - The variable must have the same name except for the last character or characters
  - The last character of each variable must be numeric
  - The variables must be numbered consecutively (e.g. qtr1-qtr4 tot92-tot96)
- _NUMERIC_: all numeric variables that have already been defined in the current DATA step
- _CHARACTER_: all character variables that have already been defined in the current DATA step
- _ALL_: all variables that have already been defined in the current DATA step (they must be of the same type)

If you do not specify the elements of the array, SAS automatically creates new variables
Referencing the elements of an array

When you define an array in a DATA step, a subscript is assigned to each array element:

```
ARRAY quarter{4} jan apr jul oct ;
```

To reference an array element in the DATA step, specify the name of the array followed by a subscript value enclosed in parentheses:

```
array-name(subscript)
```

Where `subscript`:
- Is enclosed in parentheses
- Specifies a variables, a SAS expression, or an integer
- Is within the lower and the upper bounds of the dimension of the array

**e.g:** `quarter(2)` refers to `apr`
The DIM function

```
DIM(array-name) ;
```

where `array-name` is the name of the array

This function returns the number of the elements in the array

e.g.:
```
DO i=1 TO DIM(quarter);
```
Using the arrays

The company will contribute augmenting employees donations of 25%. Calculate the total contributions.

Partial SAS Data Set Employee_Donations

<table>
<thead>
<tr>
<th>Employee_ID</th>
<th>Qtr1</th>
<th>Qtr2</th>
<th>Qtr3</th>
<th>Qtr4</th>
</tr>
</thead>
<tbody>
<tr>
<td>120265</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>25</td>
</tr>
<tr>
<td>120267</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>120269</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>120270</td>
<td>20</td>
<td>10</td>
<td>5</td>
<td>.</td>
</tr>
</tbody>
</table>

DATA charity;
    SET Lib9_3.employee_donations;
    KEEP Employee_ID Qtr1-Qtr4;
    ARRAY Contrib{4} Qtr1-Qtr4;
    DO i=1 TO DIM(Contrib);
        Contrib{i}=Contrib{i}*1.25;
    END;
RUN;
PROC PRINT DATA=charity (OBS=10) ;
RUN;

PROC PRINT DATA=Lib9_3.employee_donations;
RUN;

PROC PRINT DATA=Lib9_3.employee_donations;
KEEP Employee_ID Qtr1-Qtr4;
Qtr1=Qtr1*1.25;
Qtr2=Qtr2*1.25;
Qtr3=Qtr3*1.25;
Qtr4=Qtr4*1.25;
RUN;
PROC PRINT DATA=charity (OBS=10) ;
RUN;

ARRAY contrib[4] Qtr1 Qtr2 Qtr3 Qtr4;
ARRAY contrib{*} Qtr1 Qtr2 Qtr3 Qtr4;
ARRAY contrib(4) Qtr1-Qtr4;
How SAS processes the DATA step: the compilation phase

SAS creates the PDV (only some of the variables are showed here)

<table>
<thead>
<tr>
<th>Employee_ID</th>
<th>Qtr1</th>
<th>Qtr2</th>
<th>Qtr3</th>
<th>Qtr4</th>
</tr>
</thead>
</table>

SAS creates the subscripts

<table>
<thead>
<tr>
<th>Employee_ID</th>
<th>contrib{1}</th>
<th>contrib{2}</th>
<th>contrib{3}</th>
<th>contrib{4}</th>
</tr>
</thead>
</table>

SAS creates the index-variable i

<table>
<thead>
<tr>
<th>Employee_ID</th>
<th>Qtr1</th>
<th>Qtr2</th>
<th>Qtr3</th>
<th>Qtr4</th>
<th>i</th>
</tr>
</thead>
</table>
How SAS process the DO loop in the execution phase

SAS executes the SET statement and puts variable values in the PDV:

(KEEP and ARRAY are compile-time statements...)

SAS starts executing the DO loop, from i=1...

... to i=4

When the index-variable is incremented to 5, the index is out of range; DO loop stops processing.
Creating new variables with the ARRAY statement

- If you do not specify the elements of the array, SAS automatically creates new variables:
  E.g.: ARRAY Giorgio{3} ;

- The names of the new variables are obtained by concatenation of the name of the array
  and numbers 1,2,3... The example above creates variables Giorgio1 Giorgio2 Giorgio3

- If you prefer, you can specify new variable names. E.g. : ARRAY Giorgio{3} jan feb mar ;

- If you want to create an array of character variables, add a $ symbol after the dimension.
  E.g. : ARRAY Giorgio{5} $; The default length is 8, but you can specify the length you prefer.
  E.g.: ARRAY Giorgio{5} $ 20;

- You can assign initial values to the elements of an array when you define the array by
  placing the initial values after the elements, enclosed in parentheses and separated by
  blanks (if characters, enclose them in quotation marks). E.g.:
  • ARRAY Giorgio{3} g1 g2 g3 (0 800 90);
  • ARRAY Giorgio{3} $ g1 g2 g3 (‘White’ ‘Red’ ‘Green’);
Exercice 1

I want to create a dataset called TotConstr with 2 variables, Employee_ID and Total, the second one corresponding to the total annual contribution for each employee.

```
DATA TotConstr (KEEP= Employee_ID Total);
  SET Lib9_3.Employee_donations;
  Total=SUM(Qtr1,Qtr2,Qtr3,Qtr4);
RUN;
PROC PRINT DATA=TotConstr (OBS=10);
RUN;
```

```
DATA TotConstr (KEEP= Employee_ID Total);
  SET Lib9_3.Employee_donations;
  ARRAY Contrib{*} Qtr1-Qtr4;
  Total=SUM(OF Contrib{*});
RUN;
PROC PRINT DATA=TotConstr (OBS=10);
RUN;
```
Exercice 2

I want to create a dataset called Percent with 6 variables: Employee_ID, Total and Pct1, Pct2, Pct3, Pct4, corresponding to the percent contribution of each trimester (with respect to the annual contribution) for each observation.

```sas
DATA percent (KEEP= Employee_ID Total Pct1-Pct4);
  SET Lib9_3.Employee_donations;
  ARRAY Contrib{*} Qtr1-Qtr4;
  ARRAY Pct{4} ;
  Total=SUM(OF Contrib{*} ) ;
  DO i=1 TO 4;
    Pct[i]=Contrib[i]/Total;
  END;
RUN;
PROC PRINT DATA=percent;
  FORMAT Pct1-Pct4 PERCENT6.;
RUN;
```
I want a new dataset, called Change, with the variables of the dataset Employee_donation and 3 more variables (Diff1, Diff2, Diff3) corresponding to the differences between donations in each successive quarter.

DATA change (DROP= i);
   SET Lib9_3.Employee_donations;
   ARRAY Contrib{*} Qtr1-Qtr4;
   ARRAY Diff{3};
   DO i=1 TO 3;
      Diff(i)=Contrib(i+1)-Contrib(i);
   END;
RUN;
PROC PRINT DATA=Change;
RUN;
Exercice 4

I want a new dataset, called Change, with the variables of the dataset Employee_donation and 4 more variables (Diff1, Diff2, Diff3, Diff4) corresponding to the differences between donations in each quarter and a target to join. Targets in the 4 trimesters are 20$, 20$, 10$, 15$.

```
DATA Change (DROP=i Goal1-Goal4);
  SET Lib9_3.Employee_donations;
  ARRAY Contrib{*} Qtr1-Qtr4;
  ARRAY Diff{4} ;
  ARRAY Goal{4} (20 25 10 15);
  DO i=1 TO 4;
    Diff(i)=Contrib(i)-Goal(i);
  END;
RUN;
PROC PRINT DATA=Change;
VAR Employee_ID Diff1-Diff4;
RUN;
```

How to handle MD as zeros?

```
DATA change (DROP=i);
  SET Lib9_3.Employee_donations;
  ARRAY Contrib{*} Qtr1-Qtr4;
  ARRAY Diff{4} ;
  ARRAY Goal{4} (20 25 10 15);
  DO i=1 TO 4;
    Diff(i)=SUM(Contrib(i),-Goal(i));
  END;
RUN;
PROC PRINT DATA=Change;
VAR Employee_ID Diff1-Diff4;
RUN;
```
Temporary arrays

To create temporary array elements for DATA step processing without creating new variables, specify _TEMPORARY_ after the array name and dimension.

E.g.: we don’t want variables Goal1-Goal4 in the change dataset:

```
DATA change (DROP=i);
  SET Lib9_3.Employee_donations;
  ARRAY Contrib{*} Qtr1-Qtr4;
  ARRAY Diff{4} ;
  ARRAY Goal{4} _TEMPORARY_ (20 25 10 15);
  DO i=1 TO 4;
    Diff(i)=Contrib(i)-Goal(i);
  END;
RUN;
PROC PRINT DATA=Change;
RUN;
```

Temporary arrays are useful for improving performance time.
Multidimensional arrays

You can also group variables into table-like structures called multidimensional arrays.

To define a multidimensional array, you specify the number of elements in each dimension, separated by a comma.

In the case of two dimensions, you can think to the array as a table.

```
ARRAY array-name{r,c} elements;
```

-  
  -  
  -  

The elements are listed in the table by rows.

E.g.: ARRAY Lili{3,4} x1-x12;

```
X1   X2   X3   X4
X5   X6   X7   X8
X9   X10  X11  X12
```

You can reference any element of the array by referencing the 2 dimensions.

```
Lili(2,3)
```
Exercise 5 (a)

Creation of the dataset « Monthly » in library « Work »

```sas
DATA Monthly;
  INPUT Year Month1 Month2 Month3 Month4 Month5 Month6 Month7 Month8 Month9
  Month10 Month11 Month12;
DATALINES;
2010 23000 21500 24600 23300 20000 21100 23000 21600 24600 24300 22700 24800
2011 23400 24400 25300 25100 22500 24400 28700 27400 27100 28700 26500 27600
2012 23500 24600 25300 28700 25500 27600 28700 28400 28100 29700 28800 29300
;
PROC PRINT DATA=Monthly;
RUN;
```

Imagine that your company’s sales in the period 2010 - 2012 are stored by month in this dataset.
Exercise 5 (b)

Your task is to generate a new data set, called « Quarters » of \textit{ quarterly sales rather than monthly sales }.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Obs & Year & Qtr1 & Qtr2 & Qtr3 & Qtr4 \\
\hline
1 & 2010 & 69100 & 64400 & 69200 & 71800 \\
2 & 2011 & 73100 & 72000 & 83200 & 82800 \\
3 & 2012 & 73400 & 81800 & 85200 & 87800 \\
\hline
\end{tabular}
\end{table}

\begin{code}
DATA Quarters (KEEP = Year Qtr1-Qtr4);
SET Monthly;
ARRAY Mois{4,3} Month1-Month12;
ARRAY Qtr{4};
DO i=1 TO 4;
  Qtr{i} = 0;
  DO j=1 TO 3;
    Qtr{i}+Mois{i,j};
  END;
END;
RUN;

PROC PRINT DATA=Quarters;
RUN;
\end{code}
Rotating a data set

Suppose you want to rotate the « Quarters » dataset to create four observations of each input observation. In the new data set, called « Rotate », we must have \((4 \times 3 =)\) **12 observations** for which the **year**, the **quarter** and the **amount of sales** are considered as variables.

```sas
DATA Rotate (DROP = Qtr1-Qtr4);
  SET Quarters;
  ARRAY Sales{4} Qtr1-Qtr4;
  DO Trimestre=1 TO DIM(Sales);
    Amount = Sales{Trimestre};
    OUTPUT;
  END;
RUN;

PROC PRINT DATA=Rotate;
RUN;
```