Hurray for Arrays that Handle Data from External Files

Steve Rhoades, IMS Health, Plymouth Meeting, PA

ABSTRACT:
There are often times when arrays can make your code easier to understand, much less verbose and faster to both code and debug when reading in external files. OK, OK I hear the uninitiated saying that’s great, but I don’t have time to learn how to do this. And frankly I understand this, I often have projects and very short windows to deliver them and that means almost no time to learn any new skills.

Here’s what I propose, what if you get a “How To” guide that shows examples with input and output files, the code, an explanation of how it all works along with an accompanying file so you can just type Submit and see it work at any time. And since there are variations for reading in arrays, this poster will provide some alternatives to see how to deal with some more complicated situations. It will also explain the concept of “column input”, and how it works with reading in data using arrays.

So if you have Base SAS®, know what a Data Step is and possess a rudimentary understanding of arrays you can add another tool to your arsenal of knowledge.

INTRODUCTION:
Have you ever had to write essentially the same input statements over and over again? It certainly makes the coding very long and rather tedious, error prone and more difficult to read and maintain. Unfortunately there are times that you have no choice and have to code this way, but then again there are a lot of times that you do have a choice. If portions of the data have patterns, such as twelve months of adjacent sales data or say multiple addresses, arrays can be a good alternative to reading in the data, or for that matter writing it out. This paper will use two datasets as input for all of the examples in this paper. The bulk of the examples will use the first input test dataset. In general I read the file in as twelve monthly sales figures, but this is not always the case. Since I am showing different techniques for arrays you will see some examples where the fields are defined differently. Just remember the dataset is the same whether I call the data 12 months of sales data, three years of quarterly data or even some other combination.

Personally I relate to examples in learning. Give me a summary of the examples so I know what it is doing and then show me the example and I’m very happy. This paper provides both and is intended to be used as a quick guide for using array in reading in data. We will cover a bare minimum of terms and then plunge right in.

TERMS AND DEFINITIONS USED FOR PROCESSING ARRAYS:

Single @ sign  On an INPUT line holds the line in place so that next INPUT line will read in the same line, but usually different columns.
+ n  On an INPUT statement this is used to advance the pointer a specified numbers of columns. For instance +2 would move the INPUT pointer two columns to the right.
truncover  Used on INFILE statement so when there are short lines it will fill out the rest of the record as if there were incoming spaces.

SYNOPSIS OF EXAMPLES CONTAINED IN THIS PAPER

• READING IN AN ARRAY – EXAMPLE 1 (@ sign not used reading in monthly sales data)
• READING IN AN ARRAY – EXAMPLE 2 (@ sign reading in monthly sales data)
• READING IN AN ARRAY – EXAMPLE 3 (Reading alternate fields into arrays)
• READING IN AN ARRAY – EXAMPLE 4 (Reading non-adjacent data into arrays)
• READING IN AN ARRAY – EXAMPLE 5 (Reading in multiple arrays, @ sign not used reading quarterly sales data)
• READING IN AN ARRAY – EXAMPLE 6 (Reading in multiple arrays, specifying the start position of each array)
• READING WITH ARRAYS WITHOUT A DO LOOP (Example 7 – Reading in a single array).
• READING WITH ARRAYS WITHOUT A DO LOOP (Example 8 – Reading in a multiple arrays).
• READING WITH ARRAYS AND TRANSPOSING THE DATA (Example 9 – Transpose by month).
• READING SPACE DELIMITED DATA WITH ARRAYS (Example 10 – Data has different widths and embedded spaces).
• WRITING OUT FLAT FILES USING ARRAYS (Example 11 – Avoiding Problems).

READING IN AN ARRAY – EXAMPLE 1 (@ sign not used reading in monthly sales data)

In this example we are reading in 12 months of sales data. We can specify the name, location and format of each sales data field or use an array to reduce the lines of coding and increase readability. Other than the array itself, other important pieces to be aware of is the @ sign to hold this record rather than reading in another record when the next input statement is encountered and the dim statement which contains the number of variables in an array. Notice the Input statement for the Sales Data has no @ sign for the location, the program starts reading data in column 37 and is incremented 6 bytes before it reads in the next month of sales data. In the below example I use the variable blankspace to get the column pointer to move to column 37, an alternate way would instead of using this variable increment the pointer counter on the previous line; the code would be:

@31 RouteNbr $Char05. +1 Incrementing the column pointer

EXAMPLE 1

```sas
DATA Ex1_ArrayProcess;
Infile CandySls TRUNCOVER;
Array MthlySales {*} Mth1-Mth12; /*read in 12 mths of sales data*/
Input @1   Candy      $Char25.
@26   CandyNbr   $Char04.
@31   RouteNbr   $Char05.
@36   BlankSpace $Char01. /* Monthly sales starts in pos 37*/
@       /* Do not advance to next record*/
Do I = 1 to dim(mthlysales); /*Dimension of this is 12*/
   Input MthlySales {I} 6.2
   @ /* Do not advance to the next record */
End;
Drop I BlankSpace; /* These are not needed, drop them */
Run;
```

PROGRAM

--[[---------------------------Program Example1.SAS--------------------------]]
/*- ITEMS TO NOTE: */
/*- There is a blank space between RouteNbr field and beginning of */
/*- monthly sales data. The var BlankSpace causes the col ptr to */
/*- move one space so First Sales Data file line up properly. */
/*- Notice single @ signs. They tell SAS not to move to next rec. */
/*- Instead the col ptr is positioned to read in next field. The */
/*- first month's Sales Data field begins in col 37, ptr is then */
/*- shifted to column 43 for second mth. Each time a mth is read */
/*- ptr shifts 6 positions. If you forget the @ signs expect */
/*- to see Missing Values and LOST CARDS statements in SAS log. */
--[[---------------------------Program Example1.SAS--------------------------]]

```sas
FileName CandySls 'U:\My Documents\NESUG\Nesug 2007\DataFiles\Example1.txt';

Data Ex1_ArrayProcess;
Infile CandySls TRUNCOVER;
Array MthlySales {*} Mth1-Mth12; /*read in 12 mths of sales data*/

Input @1 Candy      $Char25.
   @26 CandyNbr   $Char04.
   @31 RouteNbr   $Char05.
   @36 BlankSpace $Char01. /* Monthly sales starts in pos 37*/
   @ /* Do not advance to next record*/
;
Do I = 1 to dim(mthlysales); /*Dimension of this is 12*/
   Input MthlySales {I} 6.2
   @ /* Do not advance to the next record */
;
End;
Drop I BlankSpace; /* These are not needed, drop them */
Run;
```
READING IN AN ARRAY – EXAMPLE 2 (@ sign reading in monthly sales data)

In this example 12 months of sales data is again being read into SAS. This time the column pointer is initially set to 37 (see variable MthCol) so I no longer have to keep track of the preceding columns. The input now reads as ‘Input @MthCol MthlySales {I} 6.2’ and the column pointer must be incremented by 6 for each iteration. Again @ signs are used to prevent another record from being read in for each Input statement.

EXAMPLE 2
DATA (Same as example 1)

PROGRAM
/*-------------------------------------------------------------------*/
/*- Program Example2.SAS                                            */
/*- Using explicit pointer in DO LOOP to read in mthly sales data   */
/*- Accounting for the blank space between the RouteNbr field &     */
/*- Specify the col to start reading mthly Sales Data using MthCol-*/
/*-       and increment it by 6 positions on each loop.             */
/*- Notice single @ signs. These tell SAS not to move to next rec-*/
/*-       First mth’s Sales Data field begins in col 37, ptr is then-*/
/*-       see Missing Values and LOST CARDS statements in the SAS log.*/
/*-------------------------------------------------------------------*/

FileName CandySls 'U:\My Documents\NESUG\Nesug 2007\DataFiles\Example1.txt';
Data Ex2_ArrayProcess;
Infile CandySls TRUNCOVER;
Array MthlySales {*} Mth1-Mth12; /*read in 12 mths of sales data*/
MthCol=37;

Input @1   Candy      $Char25.
   @26   CandyNbr   $Char04.
   @31   RouteNbr   $Char05.
   @     /* Do not advance to next record*/;
Do I = 1 to DIM(MthlySales);
   Input @MthCol MthlySales {I} 6.2 /*shifts 6 bytes ea iteration*/
      @     /* Do not advance to next record*/;
   MthCol+6; /*Add 6 bytes to the column Pointer*/
End;
Drop I MthCol; /* These are not needed, drop them */
Run;

PRINT OF EX2_ARRAYPROCESS SAS OUTFILE is identical to EX1_ARRAYPROCESS file.
READING IN AN ARRAY – EXAMPLE 3 (Reading alternate fields into arrays)

This technique demonstrates how to read in patterns of similar data. For example suppose you had six years of monthly data stored by month in a pattern such as 2002, mth1; 2003, mth1; to 2007, mth1; and then the same pattern repeated for mth2 to mth12. This technique could be used to read all 6 years of data in 6 yearly arrays containing the 12 months of data pertaining to each year. In this example we want all odd months written to one array and all even months to another array. Since the variable blankfield was read in column 36, the column pointer is now positioned at 37 when the do loop starts. Since each input field is defined as a length of 6, the column pointer moves six bytes to read in the next variable.

EXAMPLE 3

DATA
Same as previous examples

PROGRAM
 /*------------------------------------------------------------------*/  
/*- Program Example3.SAS                                           */  
/*- Assumes mths begin with mth 1 in col 37, followed by mth 2 in */  
/*- column 43 and continues in this sequence until mth 12.         */  
/*- When the program runs it reads the first input field into the-*/  
/*- MthlySalesOdd array, moves 6 cols to right, reads second --*/  
/*- input field into MthlySalesEven. Seq repeats six times.      */  
 /*------------------------------------------------------------------*/
 FileName CandySls 'C:\Documents and Settings\srhoades\My Documents\NESUG07\Nesug 2007\DataFiles\Example1.txt';

 Data Ex3_ArrayProcess; 
 Infile CandySls TRUNCOVER; 
 Array MthlySalesOdd {*} Mth1 Mth3 Mth5 Mth7 Mth9 Mth11; /*odd*/ 
 Array MthlySalesEven {*} Mth2 Mth4 Mth6 Mth8 Mth10 Mth12; /*even*/ 

 Input @1 Candy $Char25. 
 @26 CandyNbr $Char04. 
 @31 RouteNbr $Char05. 
 @36 blankfield $Char01. 
 @ /* Do not advance to next record*/ ;

 Do I = 1 to 6; 
 Input MthlySalesOdd {I} 6.2 
 MthlySalesEven {I} 6.2 
 @ /* Do not advance to next record*/ ;
 End; 
 Drop I blankfield ; /* These are not needed, drop them */ 
 Run;

PRINT OF EX3_ARRAYPROCESS SAS OUTFILE
 Print of Input Example3 Data 14:21 Saturday, July 28, 2007 20

 Obs Mth1 Mth3 Mth5 Mth7 Mth9 Mth11 Mth2 Mth4 Mth6 Mth8 Mth10 Mth12
 1 110.25 345.78 789.00 333.45 777.88 343.55 200.50 456.88 754.11 555.55 999.11 111.11
 2 440.25 445.78 669.00 883.45 700.88 333.55 330.50 556.88 774.11 995.55 911.11 441.11
 3 444.55 777.88 777.54 667.89 711.76 345.66 555.66 666.98 800.66 111.22 432.12 770.01

 Candy Route
 BERTIE BOTTS BEANS 4432 12345
 Hershey Bar 12oz. 4444 22225
 Peanut Chew 5556 34567


READING IN AN ARRAY – EXAMPLE 4 (Reading non-adjacent data into arrays)

In this example we are still reading in the same file, but suppose we only want to read in the first month of each quarter. Well there is a regular pattern to this, but there are two months between the start of one quarter and the next. Here’s a technique to handle this. By now you know that these fields are 6 bytes long, so how do we get past the intervening two months to get to the start of the next quarter. Notice the +12; this increments the column pointer another 12 bytes and now we are ready to read in the first month of the next quarter. Here’s a program that does this.

EXAMPLE 4

DATA

Same as previous examples

PROGRAM

/*---------------------------------------------------------------*/
/*- Program Example4.SAS                                      -*/
/*- Read in the mth1, 4, 7, 10 into one array using add to ptr to*/
/*- skip intervening mths. Using +12 the ptr skips over the mths*/
/*- that we do not want to read in.                           */
/*---------------------------------------------------------------*/

FileName CandySls 'C:\Documents and Settings\srhoades\My Documents\NESUG07\Nesug 2007\DataFiles\Example1.txt';

Data Ex4_ArrayProcess;
Infile CandySls TRUNCOVER;
Array MthlySales {*} Mth1 Mth4 Mth7  Mth10;/*read 1st qtr mth*/
Input @1   Candy      $Char25.
@26   CandyNbr   $Char04.
@31   RouteNbr   $Char05.
@36   blankfield $Char01. /*Or use +1 above*/
@       /* Do not advance to next record*/
;
Do I = 1 to dim(Mthlysales);
   Input MthlySales {I}  6.2 +12
   @                  /* Do not advance to next record*/
   ;
End;
Drop I blankfield ; /* These are not needed, drop them */
Run;

PRINT OF EX4_ARRAYPROCESS SAS OUTFILE

Print of Input Example4 Data

<table>
<thead>
<tr>
<th>Obs</th>
<th>Mth1</th>
<th>Mth4</th>
<th>Mth7</th>
<th>Mth10</th>
<th>Candy</th>
<th>Nbr</th>
<th>Nbr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>110.25</td>
<td>456.88</td>
<td>333.45</td>
<td>999.11</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
</tr>
<tr>
<td>2</td>
<td>440.25</td>
<td>556.88</td>
<td>883.45</td>
<td>911.11</td>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
</tr>
<tr>
<td>3</td>
<td>444.55</td>
<td>666.98</td>
<td>667.89</td>
<td>432.12</td>
<td>Peanut Chew</td>
<td>5556</td>
<td>34567</td>
</tr>
</tbody>
</table>
READING IN AN ARRAY – EXAMPLE 5 (Reading in multiple arrays, @ sign not used reading in 3 years of quarterly sales data)

In this example we are reading in 3 years of quarterly data. Some of you may realize that I am reading the exact same file as the monthly examples. Well yes that is true, but then again I do reserve the right to change what the file actually contains – it now contains quarterly data. Again the Input statement for the Sales Data has no @ sign for the location, the program starts reading data in column 37 and is incremented 6 bytes before it reads in the next month of sales data.

EXAMPLE 5
DATA
Same as previous examples

PROGRAM
/*-------------------------------------------------------------------*/
/*- Program Example5.SAS (READ IN MULTIPLE ARRAYS)                 -*/
/*- Since an explicit ptr not being used to read sales data, we need-*/
/*- to be sure col 36 is read in before the sale data. Suppose      -*/
/*- instead of 12 mths, we are reading 3 yrs of qtrly data. The data-*
/*- is arranged in year qtr sequence where all of the qtr1 data is  -*/
/*- grouped together, followed by quarter 2, then 3 and finally 4.  */
/*-------------------------------------------------------------------*/

FILENAME CandySls 'U:\My Documents\NESUG\Nesug 2007\DataFiles\Example1.txt';

Data Ex5_ArrayProcess;
Infile CandySls TRUNCOVER;
Array QtrYr1 {*} YrQ11-YrQ14; /*first yr of qtrs*/
Array QtrYr2 {*} YrQ21-YrQ24; /*second yr of qtrs*/
Array QtrYr3 {*} YrQ31-YrQ34; /*third yr of qtrs*/
Input @1 Candy $Char25.
@26 CandyNbr $Char04.
@31 RouteNbr $Char05.
@36 BlankSpace $Char01. /* Mthly sales starts pos 37*/
@ /* Do not advance to next record*/
;
Do I = 1 to 4; /*Four quarters of data */
Input QtrYr1 {I} 6.2
QtrYr2 {I} 6.2
QtrYr3 {I} 6.2
@ /* Do not advance to the next record */
;
End;
Drop I BlankSpace; /* These are not needed, drop them */
Run;

PRINT OF EX5_ARRAYPROCESS SAS OUTFILE
Print of Input Example5 Data 05:56 Saturday, July 28, 2007 4

Obs YrQ11 YrQ12 YrQ13 YrQ14 YrQ21 YrQ22 YrQ23 YrQ24 YrQ31 YrQ32 YrQ33 YrQ34
1 110.25 456.88 333.45 999.11 200.50 789.00 555.55 343.55 345.78 754.11 777.88 111.11
2 440.25 556.88 883.45 911.11 330.50 669.00 995.55 333.55 445.78 774.11 700.88 441.11
3 444.55 666.98 667.89 432.12 555.66 777.54 111.22 345.66 777.88 800.66 711.76 770.01

Candy          Route
Candy          Nbr        Nbr
BERTIE BOTTS BEANS 4432  12345
Hershey Bar 12oz.  4444  22225
Peanut Chew 5556  34567

6
READING IN AN ARRAY – EXAMPLE 6 (Reading in multiple arrays, specifying the start position of each array)

In this example we are reading in 3 years of quarterly data. However the data will now be interpreted as year 1, quarters 1 to 4, followed by years 2 and 3. Since there are three starting points we need to set up three different column pointers and increment each one every time through the Do Loop. Pay attention to the pointers below.

EXAMPLE 6
DATA

Same as previous examples

PROGRAM
/*- Prog Example6.SAS (READ IN MULTIPLE ARRAYS WITH EXPLICIT PTRS)-*/
/*- Since an explicit ptr is being used to read sales data, need to*/
/*- specify where arrays start. In this ex I have decided that my-*/
/*- qtrly data is actually stored as yr1, qtrs 1-4, yr2, the four -*/
/*- qtrs and finally yr3 with qtrs 1-4. To do this setting    */
/*- up 3 col ptrs and then increment each one */
/*- of these pointers every time the DO LOOP is incremented. */
/*- The 1st yr’s sales data begins in col 37, the 2nd in col 61 & */
/*- the 3rd in col 85. Each time a qtr the ptrs must be updated to */
/*- point to the next quarter.*/

FileName CandySls 'U:\My Documents\NESUG\Nesug 2007\DataFiles\Example1.txt';

Data Ex6_ArrayProcess;
Infile CandySls TRUNCOVER;
Array QtrYr1   {*} YrQ11-YrQ14;  /*first yr of qtrs*/
Array QtrYr2   {*} YrQ21-YrQ24;  /*second yr of qtrs*/
Array QtrYr3   {*} YrQ31-YrQ34;  /*third yr of qtrs*/

/* Specify where each year starts */
Yr1Ptr=37;
Yr2Ptr =61;
Yr3Ptr =85;

Input @1   Candy      $Char25.
@26   CandyNbr   $Char04.
@31   RouteNbr   $Char05.
@;

Do I = 1 to 4;
Input @Yr1Ptr QtrYr1 {I}  6.2
@Yr2Ptr QtrYr2 {I}  6.2
@Yr3Ptr QtrYr3 {I}  6.2
@;  /* Do not advance to next record*/

Yr1Ptr+6;  /*Increment the ptrs for next quarter*/
Yr2Ptr+6;
Yr3Ptr+6;
End;

Drop I Yr1Ptr Yr2Ptr Yr3Ptr;  /*These not needed, drop them*/
Run;

PRINT OF EX6_ARRAYPROCESS SAS OUTFILE
Print of Input Example6 Data                 05:56 Saturday, July 28, 2007   8

Obs YrQ11  YrQ12  YrQ13  YrQ14  YrQ21  YrQ22  YrQ23  YrQ24  YrQ31  YrQ32  YrQ33  YrQ34
1  110.25 200.50 345.78 456.88 789.00 754.11 333.45 555.55 777.88 999.11 343.55 111.11
2  440.25 330.50 445.78 556.88 669.00 774.11 883.45 995.55 700.88 911.11 333.55 441.11
3  444.55 555.66 777.88 666.98 777.54 800.66 667.89 111.22 711.76 432.12 345.66 770.01

Candy Route
Candy               Nbr   Nbr
BERTIE BOTTS BEANS 4432  12345
Hershey Bar 12oz.  4444  22225
Peanut Chew        5556  34567
READING WITH ARRAYS WITHOUT A DO LOOP (Example 7 – Reading in a single array).

You can use an array to read in data even without using a Do Loop. The code is very straightforward, but not very intuitive. Here is an example of an input statement to read in an array:

```sas
@13 (ArrayABC{*}) (10.);
```

Here's the translation, starting at column 13 read in ArrayABC. The {*} basically says to count the number of variables in the array and insert it here for the number of times to read in this field, and the (10.) indicates that the field is a numeric that is 10 wide, if the parenthesis is omitted, an error occurs. In this example I will use similar code for reading in the four quarters of a single year.

Quirk to be aware of:

- `@37 (QtrYr1{1}) (6.2)` Reads in only 1st qtr of data and puts it YrQ11
- `@37 (QtrYr1{2}) (6.2)` Reads in only 1st qtr of data and puts it YrQ12.
- `@37 (QtrYr1{3}) (6.2)` Reads in only 1st qtr of data and puts it YrQ13.
- `@37 (QtrYr1{4}) (6.2)` Reads in only 1st qtr of data and puts it YrQ14.

EXAMPLE 7

DATA

<table>
<thead>
<tr>
<th>Qtr1</th>
<th>Qtr2</th>
<th>Qtr3</th>
<th>Qtr4</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>110.25</td>
</tr>
<tr>
<td>HERSEY BAR 12OZ.</td>
<td>4444</td>
<td>22225</td>
<td>440.25</td>
</tr>
<tr>
<td>PEANUT CHEW</td>
<td>5556</td>
<td>34567</td>
<td>444.55</td>
</tr>
</tbody>
</table>

PRINT OF EX7_ARRAYPROCESS SAS OUTFILE

Print of Input Example7 Data 14:21 Saturday, July 28, 2007 7

READING WITH ARRAYS WITHOUT A DO LOOP (Example 8 – Reading in a multiple arrays).

You can read multiple arrays without using a Do Loop. It is very similar to reading in a single array and the input statements are the same. You have to declare all of the arrays and specify where they start. Remember on the input lines to include the parenthesis. Also on the input line the same quirk that applies to single arrays also applies here. I am repeating the details of the quirk below; it applies the same way to QtrYr3 in the following program.
Quirk to be aware of:

@37 (QtrYr1{1}) (6.2) Reads in only 1st qtr of data and puts it YrQ11.
@37 (QtrYr1{2}) (6.2) Reads in only 1st qtr of data and puts it YrQ12.
@37 (QtrYr1{3}) (6.2) Reads in only 1st qtr of data and puts it YrQ13.
@37 (QtrYr1{4}) (6.2) Reads in only 1st qtr of data and puts it YrQ14.

EXAMPLE 8

DATA

<table>
<thead>
<tr>
<th></th>
<th>Qtr1</th>
<th>Qtr2</th>
<th>Qtr3</th>
<th>Qtr4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qtr</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BERTIE BOTTS BEANS
HERSHEY BAR 12OZ.
PEANUT CHEW

PROGRAM

/*------------------------------------------------------------------*/
/*- Prog Example8.SAS (READ IN MULTIPLE ARRAYS WITHOUT A DO LOOP) -*/
/*- The qtrly sales data begins in col 37 for first year and col 85-*/
/*- for 3rd yr. Be sure to include parenthesis or it will not work. -*/
/*----------------------------------------------------------------*/

FileName CandySls 'C:\Documents and Settings\srhoades\My Documents\NESUG07\Nesug2007\DataFiles\Example1.txt';

Data Ex6_ArrayProcess;
Infile CandySls TRUNCOVER;
Array QtrYr1 {*} YrQ11-YrQ14; /*first yr of qtrs*/
Array QtrYr3 {*} YrQ31-YrQ34; /*third yr of qtrs*/
Input @1   Candy      $Char25.
@26   CandyNbr $Char04.
@31   RouteNbr $Char05.
@37   (QtrYr1{*}) ( 6.2)
@85   (QtrYr3{*}) ( 6.2)
Run;

PRINT OF EX8_ARRAYPROCESS SAS OUTFILE
Print of Input Example8 Data                 14:21 Saturday, July 28, 2007  10

Obs YrQ11  YrQ12  YrQ13  YrQ14  YrQ31  YrQ32  YrQ33  YrQ34  Candy
1  110.25  200.50  345.78  456.88  777.88  999.11  343.55  111.11  BERTIE BOTTS BEANS
2  440.25  330.50  445.78  556.88  700.88  911.11  333.55  441.11  Hershey Bar 12oz.
3  444.55  555.66  777.88  666.98  711.76  432.12  345.66  770.01  Peanut Chew

Candy   Route
Nbr     Nbr
4432   12345
4444   22225
5556   34567
READING WITH ARRAYS AND TRANSPOSING THE DATA (Example 9 – Transpose by month).

The input records contain 12 months of data. Suppose that instead of a single record with 12 months, you want a record for each month. This can easily be done by using arrays as you read in the data. However, the same technique can also be used on SAS files as well. In this example the Do Loop is used to identify the month being written out and then it is simply assigning the array values to a single variable name and writing the file out.

EXAMPLE 9 Transposing Data Using Arrays

**DATA**

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bertie Botts Beans</td>
<td>Hershey Bar 12oz.</td>
</tr>
<tr>
<td>4432 12345 110.25200.50345.788456.88789.00754.11333.45</td>
<td>4444 22225 440.25330.50445.78556.88669.00774.11883.45</td>
</tr>
<tr>
<td>Peanuts Chew</td>
<td></td>
</tr>
<tr>
<td>5556 34567 444.55555.66777.88666.98777.54800.66667.89</td>
<td></td>
</tr>
</tbody>
</table>

**PROGRAM**

```sas
/*-----------------------------------------------*/
/* Program Example9_SAS Transpose data */
/* Each rec of input dataset contains 12 mths of data. This example-*/
/* will write a single rec out for ea mth. Using technique multi */
/* fields can be transposed simultaneously. Depending on */
/* requirement it can be used as an alternative to Proc Transpose */
/*-----------------------------------------------*/

FileName CandySls 'C:\Documents and Settings\srhoades\My Documents\NESUG07\Nesug 2007\DataFiles\Example1.txt';

Data Ex9_ArrayProcess;
Infile CandySls TRUNCOVER;

Array MthlySales {*} Mth1-Mth12; /*read in only first mth of qtr*/

Input @1 Candy $Char25.
@26 CandyNbr $Char04.
@31 RouteNbr $Char05.
@36 blankfield $Char01.
/* Do not advance to next record*/
;

Do I = 1 to dim(Mthlysales);
  Input MthlySales {I} 6.2
       /* Do not advance to next record*/
       ;
  Mth=I;
  MthSale=MthlySales {I};
  Output;
End;

Drop I blankfield mth1-mth12; /*These not needed, drop them*/
Run;
```
### Partial PRINT OF EX9_ARRAYPROCESS SAS OUTFILE

Print of Input Example9 Data 14:21 Saturday, July 28, 2007 23

<table>
<thead>
<tr>
<th>Obs</th>
<th>Candy</th>
<th>Nbr</th>
<th>Nbr</th>
<th>Mth</th>
<th>Sale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>1</td>
<td>110.25</td>
</tr>
<tr>
<td>2</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>2</td>
<td>200.50</td>
</tr>
<tr>
<td>3</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>3</td>
<td>345.78</td>
</tr>
<tr>
<td>4</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>4</td>
<td>456.88</td>
</tr>
<tr>
<td>5</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>5</td>
<td>789.00</td>
</tr>
<tr>
<td>6</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>6</td>
<td>754.11</td>
</tr>
<tr>
<td>7</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>7</td>
<td>333.45</td>
</tr>
<tr>
<td>8</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>8</td>
<td>555.55</td>
</tr>
<tr>
<td>9</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>9</td>
<td>777.88</td>
</tr>
<tr>
<td>10</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>10</td>
<td>999.11</td>
</tr>
<tr>
<td>11</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>11</td>
<td>343.55</td>
</tr>
<tr>
<td>12</td>
<td>BERTIE BOTTS BEANS</td>
<td>4432</td>
<td>12345</td>
<td>12</td>
<td>111.11</td>
</tr>
<tr>
<td>13</td>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
<td>1</td>
<td>440.25</td>
</tr>
<tr>
<td>14</td>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
<td>2</td>
<td>330.50</td>
</tr>
<tr>
<td>15</td>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
<td>3</td>
<td>445.78</td>
</tr>
<tr>
<td>16</td>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
<td>4</td>
<td>556.88</td>
</tr>
<tr>
<td>17</td>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
<td>5</td>
<td>669.00</td>
</tr>
<tr>
<td>18</td>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
<td>6</td>
<td>774.11</td>
</tr>
<tr>
<td>19</td>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
<td>7</td>
<td>883.45</td>
</tr>
<tr>
<td>20</td>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
<td>8</td>
<td>995.55</td>
</tr>
<tr>
<td>21</td>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
<td>9</td>
<td>700.88</td>
</tr>
<tr>
<td>22</td>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
<td>10</td>
<td>911.11</td>
</tr>
<tr>
<td>23</td>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
<td>11</td>
<td>333.55</td>
</tr>
</tbody>
</table>

### READING SPACE DELIMITED DATA WITH ARRAYS (Example 10 – Data has different widths and embedded spaces)

Things to notice with this example:

- This file is partially fixed, particularly the first twenty-five bytes of it. Also the prize fields contain room for ten characters.
- If you are defining a character array you should specify the length of the variables as the default is 8 characters. Also if there can be embedded spaces you need to repeat the length on the Input statement to pick up variables with embedded spaces.
- The colon is used in this program where fields may not be as wide as specified.
- The parenthesis and colon and bracket are equivalents, on the Input statement I intermixed them.

### EXAMPLE 10

```plaintext
DATA
--------1--------2--------3--------4--------5--------6--------7
BERTIE BOTTS BEANS 4432 12345 910.25 920.50 934.78 945.88 789.00
HERSHEY BAR 12OZ. 4444 22225 440.25 330.50 445.78 556.88 669.00
PEANUT CHEW 5556 34567 444.55 555.66 777.88 666.98 777.54
--------8--------9--------0--------1--------2--------3--------4
754.11 333.45 150.42 JAGUAR GOLD TOOTHTWINKIE GOLF TEES
774.11 883.45 199.53 APPLE CORECHewingGUMOLD COFFEE LIP BALM
800.66 667.89 1234.56 CRUISE ROAD KILL PUP TENT FLEA BATH
```
FILENAME CandySls 'C:\Documents and Settings\srhoades\My Documents\NESUG07\Nesug2007\DataFiles\Example2.txt';

Data Ex10_ArrayProcess;
Infile CandySls TRUNCOVER;
Array Qtrs {*} QTR1-QTR4;
Array Commissions {*} Comm1-Comm4;
Array BonusPrize {*} $10 Prize1-Prize4;

Input Candy      $Char25.
                CandyNbr $
                RouteNbr $
                Qtrs{*}: 6.2 Commissions{*}: 6.2 BonusPrize{*} $Char10.
;
Run;

Partial PRINT OF EX10_ARRAYPROCESS SAS OUTFILE
Print of Input Example10 Data 14:21 Saturday, July 28, 2007 52
Obs QTR1 QTR2 QTR3 QTR4 Comm1 Comm2 Comm3 Comm4
1  910.25 920.50 934.78 945.88 89.00 154.11  33.45 150.42
2  440.25 330.50 445.78 556.88 69.00  174.11  83.45  199.53
3  444.55 555.66 777.88 666.98 77.54  100.66 167.89  234.56

Obs Prize1 Prize2 Prize3 Prize4 Candy Nbr Nbr
1  Jaguar Gold Tooth Twinkie Golf Tees BERTIE BOTTS BEANS 4432 12345
2  Apple Core ChewingGum Old Coffee Lip Balm Hershey Bar 12oz. 4444 22225
3  Cruise Road Kill Pup Tent Flea Bath Peanut Chew 5556 34567
WRITING OUT FLAT FILES USING ARRAYS (Example 11 – Avoiding Problems).
Writing out a flat file can result in either a single output record or all of the file records being written out which is most likely what you want. The code is the same as Example 10 except there are some additional files to write out a flat file.

**EXAMPLE 11**

**DATA**

Same as example 10

**PROGRAM**

```sas
 /*------------------------------------------------------------------*/ /*- Prog Example11.SAS   Repeat of Ex 10 except I am writing out a-*/ /*- flat file. The differences are bolded.                        */ /*------------------------------------------------------------------*/

FileName CandySls 'C:\Documents and Settings\srhoades\My Documents\NESUG07\Nesug 2007\DataFiles\Example2.txt';
FileName O1Out "C:\TEMP\Ex11.txt";
Data Ex11_ArrayProcess;
Infile CandySls TRUNCOVER;
Array Qtrs {*}           QTR1-QTR4;
Array Commissions {*}    Comm1-Comm4;
Array BonusPrize {*} $10 Prize1-Prize4;
Input Candy      $Char25.
CandyNbr   $;
RouteNbr   $;
Qtrs{*}: 6.2 Commissions(*): 6.2 BonusPrize{*}  $Char10.
;File O1OUT;
Put @1 Candy
@26 CandyNbr
@32 RouteNbr
@;
Do I = 1 to 4;
   Put ' ' BonusPrize (i) $Char10.
@;
End;
put; ←Prevents writing single output line
Drop I;
Run;
```

Display of the output flat file

Example of output when the last put statement is not included, all observations written to single output observation:

<table>
<thead>
<tr>
<th>Peanut Chew 12oz. S</th>
<th>5556</th>
<th>34567</th>
<th>Cruise</th>
<th>Road Kill</th>
<th>Pup Tent</th>
<th>Flea Bath</th>
</tr>
</thead>
</table>

Example of output when the last put statement is included:

<table>
<thead>
<tr>
<th>BERTIE BOTTS BEANS</th>
<th>4432</th>
<th>12345</th>
<th>Jaguar</th>
<th>Gold Tooth</th>
<th>Twinkie</th>
<th>Golf Tees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hershey Bar 12oz.</td>
<td>4444</td>
<td>22225</td>
<td>Apple Core</td>
<td>ChewingGum</td>
<td>Old Coffee</td>
<td>Lip Balm</td>
</tr>
<tr>
<td>Peanut Chew</td>
<td>5556</td>
<td>34567</td>
<td>Cruise</td>
<td>Road Kill</td>
<td>Pup Tent</td>
<td>Flea Bath</td>
</tr>
</tbody>
</table>


CONCLUSIONS:
Base SAS® with its array capabilities can reduce code and provide easier program maintenance when reading external data. This paper provided examples to show how to write the code and pointed out potential coding problems and how to avoid them. These examples can be used as templates in developing your programs. The paper has a number of examples, use the examples to see how to use arrays in reading in data. The accompanying zipped files and programs are intended as references for you.

REFERENCES:
From support.sas.com Sample 8: Use arrays to read and restructure a data set

CONTACT INFORMATION:
Steve Rhoades
IMS HEALTH Inc.
660 W. Germantown Pike
Plymouth Meeting, Pa. 19462
Work Phone (610) 834-4614
Email: SRhoades@US.IMSHEALTH.COM

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are trademarks of their respective companies.