Chapter 20

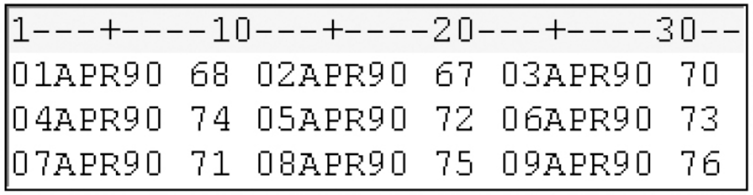
Creating Multiple Observations From a Single Record

Overview

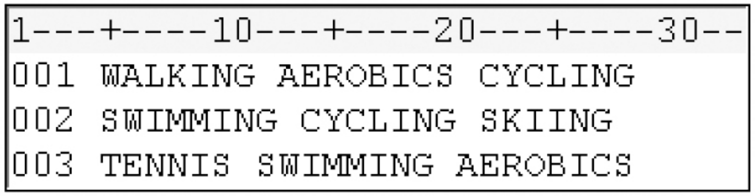
Raw data files may contain data for several observations in one record. Data can be stored in this manner to reduce the size of the entire data file. This chapter talks about how to create multiple observations from a single record of different situations.

Each record can contain

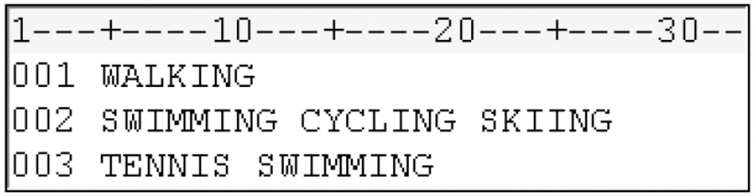
- repeating blocks of data that represent separate observations.
- an ID field followed by an equal number of repeating fields that represent separate observations.
- an ID field followed by a varying number of repeating fields that represent separate observations.



```
1---+----10---+----20---+----30--
01APR90 68 02APR90 67 03APR90 70
04APR90 74 05APR90 72 06APR90 73
07APR90 71 08APR90 75 09APR90 76
```



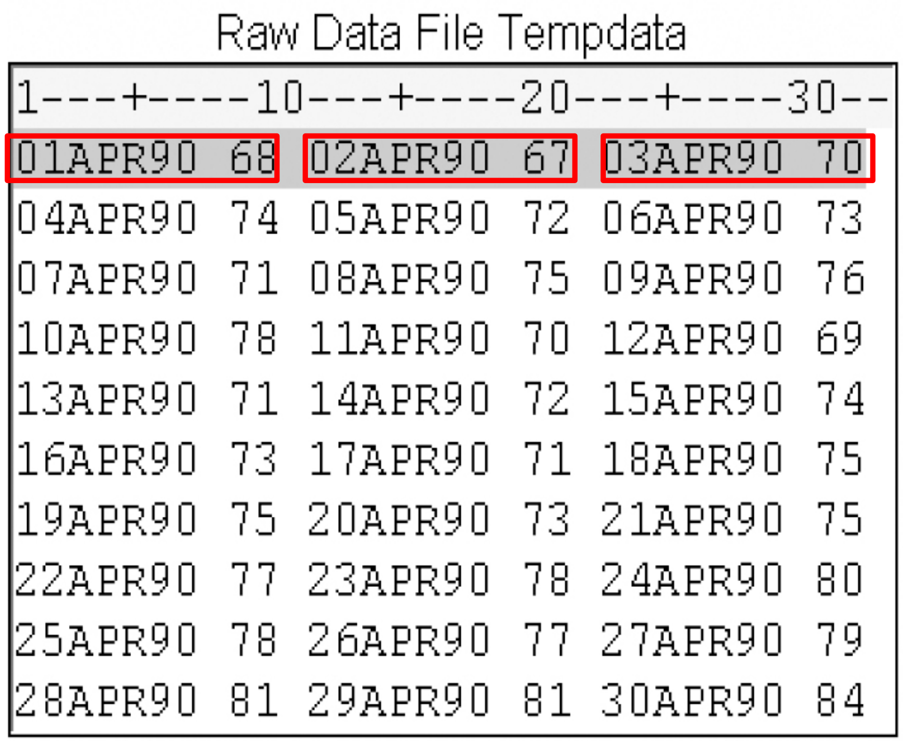
```
1---+----10---+----20---+----30--
001 WALKING AEROBICS CYCLING
002 SWIMMING CYCLING SKIING
003 TENNIS SWIMMING AEROBICS
```



```
1---+----10---+----20---+----30--
001 WALKING
002 SWIMMING CYCLING SKIING
003 TENNIS SWIMMING
```

Reading Repeating Blocks of Data

Each record in the file Tempdata contains three blocks of data. Each block contains a date followed by the day's high temperature in a city.


Raw Data File Tempdata

```
1---+----10---+----20---+----30--
01APR90 68 02APR90 67 03APR90 70
04APR90 74 05APR90 72 06APR90 73
07APR90 71 08APR90 75 09APR90 76
10APR90 78 11APR90 70 12APR90 69
13APR90 71 14APR90 72 15APR90 74
16APR90 73 17APR90 71 18APR90 75
19APR90 75 20APR90 73 21APR90 75
22APR90 77 23APR90 78 24APR90 80
25APR90 78 26APR90 77 27APR90 79
28APR90 81 29APR90 81 30APR90 84
```

You could write a DATA step that reads each record and creates three different Date and Temp variables.

SAS Date Set					
Date1	Temp1	Date2	Temp2	Date3	Temp3
11048	68	11049	67	11050	70

Alternatively, you could create a separate observation for each block of data in a record, a better structure for analysis and reporting with SAS procedures.

SAS Date Set	
Date	HighTemp
11048	68
11049	67
11050	70

Reading Repeating Blocks of Data

Holding the Current Record with a Line-Hold Specifier

In order to read repeating blocks of data, you need to hold the current record. To do so, you can use a line-hold specifier. SAS provides two line-hold specifiers.

- The single trailing at sign (@), which holds the input record for the execution of the next INPUT statement.
- The double trailing at sign (@@), which holds the input record for the execution of the next INPUT statement, even across iterations of the DATA step.

The term *trailing* indicates that @ or @@ must be the *last* item specified in the INPUT statement. For example,

```
input Name $20. @;
```

or

```
input Name $20. @@;
```

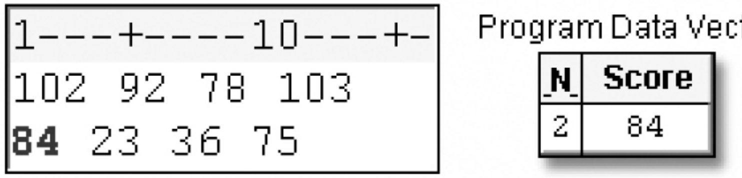
Reading Repeating Blocks of Data

Using the Double Trailing At Sign (@@) to Hold the Current Record

Normally, each time a DATA step executes, the INPUT statement reads the next record. But when the trailing @@ is used, the INPUT statement continues reading from the same record. After you have read the score 102, the following example shows the difference of using and without using the double trailing at sign @@.

- Without using the @@ sign

  Input Score;



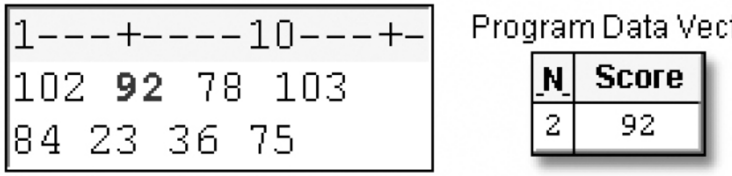
```
1---+----10---+-
102  92  78  103
84  23  36  75
```

Program Data Vector

N	Score
2	84

- Using the @@ sign

  Input Score @@;



```
1---+----10---+-
102  92  78  103
84  23  36  75
```

Program Data Vector

N	Score
2	92

Note: The @@ sign

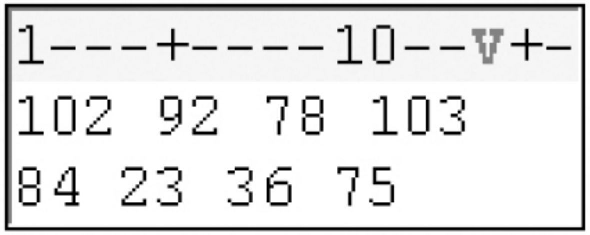
- holds the data line in the input buffer across multiple executions of the DATA step.
- typically is used to read multiple SAS observations from a single data line.
- <u>should not be used with the @ pointer control, with column input, nor with the MISSOVER option.</u>

Reading Repeating Blocks of Data

Using the Double Trailing At Sign (@@) to Hold the Current Record

A record that is held by the double trailing at sign (@@) is not released until either of the following events occurs:

- The input pointer moves past the **end of the record**. Then the input pointer moves down to the next record.



```
1---+----10--V+-
102 92 78 103
84 23 36 75
```

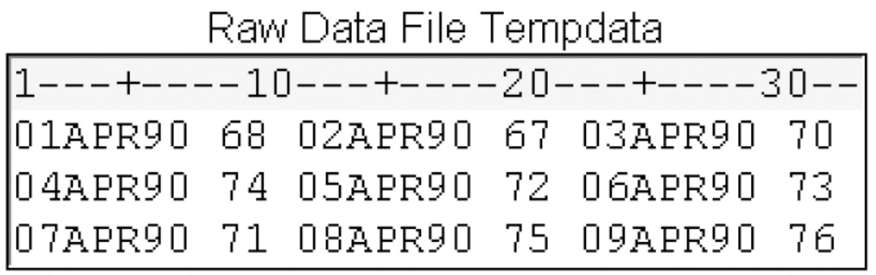
- An **INPUT** statement that has no trailing at sign executes.

```
input ID $ @@;
. . .
input Department 5.;
```

Reading Repeating Blocks of Data

Example: Using the Double Trailing At Sign (@@) to Hold the Current Record

This example requires only one INPUT statement to read the values for Date and HighTemp, but the INPUT statement must execute three times for each record by using the trailing @@.



Raw Data File Tempdata

```
1---+----10---+----20---+----30--
01APR90 68 02APR90 67 03APR90 70
04APR90 74 05APR90 72 06APR90 73
07APR90 71 08APR90 75 09APR90 76
```

```
data perm.april90;
      infile tempdata;
      input Date : date. HighTemp @@;
      format date date9.;
run;
```

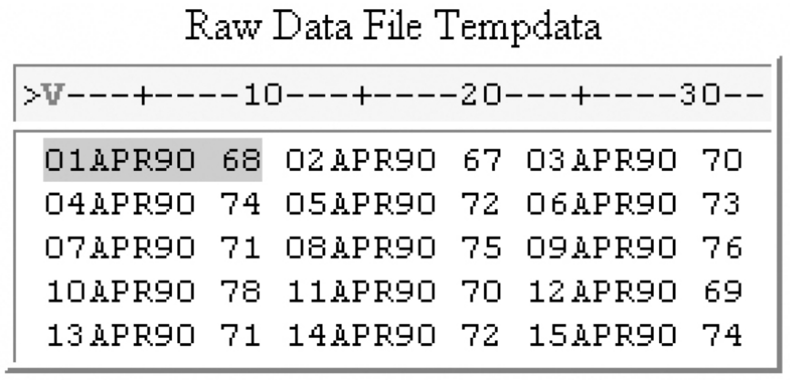
Obs	Date	HighTemp
1	01APR1990	68
2	02APR1990	67
3	03APR1990	70
4	04APR1990	74
5	05APR1990	72
6	06APR1990	73
7	07APR1990	71
8	08APR1990	75
9	09APR1990	76
10	10APR1990	78
11	11APR1990	70
12	12APR1990	69
13	13APR1990	71
14	14APR1990	71
15	15APR1990	74

Reading Repeating Blocks of Data

DATA Step Processing of Repeating Blocks of Data

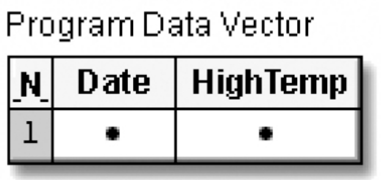
```
data perm.april90;
      infile tempdata;
      input Date : date. HighTemp @@;
run;
```

As the execution phase begins, the input pointer rests on column 1 of record 1.



Raw Data File Tempdata

```
>V---+----10---+----20---+----30--
01APR90 68 02APR90 67 03APR90 70
04APR90 74 05APR90 72 06APR90 73
07APR90 71 08APR90 75 09APR90 76
10APR90 78 11APR90 70 12APR90 69
13APR90 71 14APR90 72 15APR90 74
```

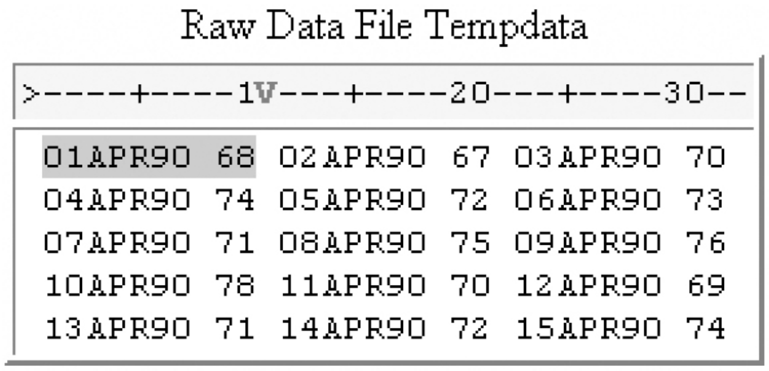


Program Data Vector

N	Date	HighTemp
1	•	•

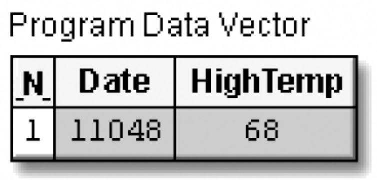
```
data perm.april90;
      infile tempdata;
      input Date : date. HighTemp @@;
run;
```

During the first iteration of the DATA step, the first block of values for Date and HighTemp are read into the program data vector. The @@ holds the current record.



Raw Data File Tempdata

```
>----+----1V---+----20---+----30--
01APR90 68 02APR90 67 03APR90 70
04APR90 74 05APR90 72 06APR90 73
07APR90 71 08APR90 75 09APR90 76
10APR90 78 11APR90 70 12APR90 69
13APR90 71 14APR90 72 15APR90 74
```



Program Data Vector

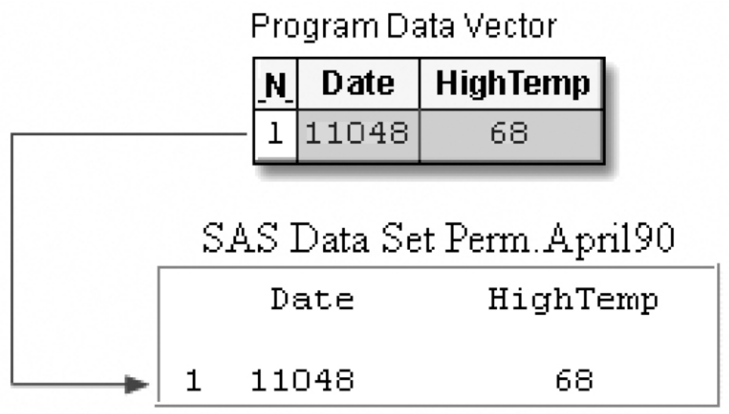
N	Date	HighTemp
1	11048	68

Reading Repeating Blocks of Data

DATA Step Processing of Repeating Blocks of Data

```
data perm.april90;
      infile tempdata;
      input Date : date. HighTemp @@;
run;
```

The first observation is written to the dataset.



```
_N_  Date   HighTemp
1    11048  68
```

```
      Date      HighTemp

1   11048         68
```

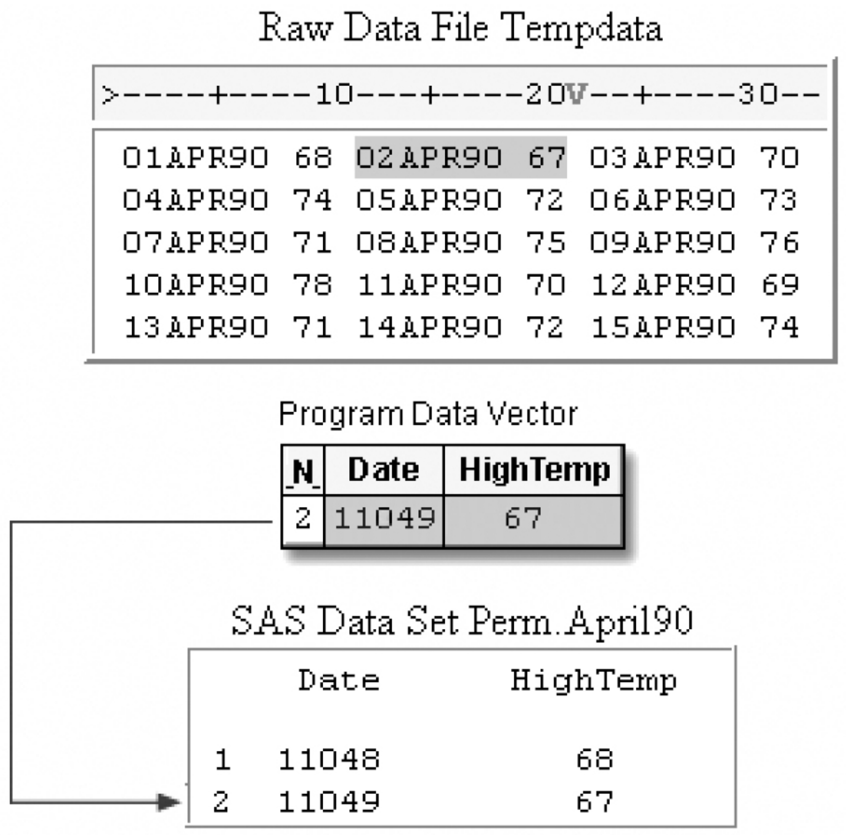
Control returns to the top of the DATA step, and the values of Date and HighTemp are reset to missing.

Program Date Vector		
_N_	Date	HighTemp
2	•	•

Reading Repeating Blocks of Data

DATA Step Processing of Repeating Blocks of Data

```
data perm.april90;
        infile tempdata;
        input Date : date. HighTemp @@;
run;
```



Raw Data File Tempdata

```
>----+----10---+----20V--+----30--
 01APR90 68 02APR90 67 03APR90 70
 04APR90 74 05APR90 72 06APR90 73
 07APR90 71 08APR90 75 09APR90 76
 10APR90 78 11APR90 70 12APR90 69
 13APR90 71 14APR90 72 15APR90 74
```

Program Data Vector

_N_	Date	HighTemp
2	11049	67

SAS Data Set Perm.April90

```
     Date       HighTemp

1    11048         68
2    11049         67
```

During the second iteration, the INPUT statement reads the second block of values for Date and HighTemp from the first record, and the @@ prevents the input pointer from moving down to the next record. The second observation is written to the dataset at the end of the data step.

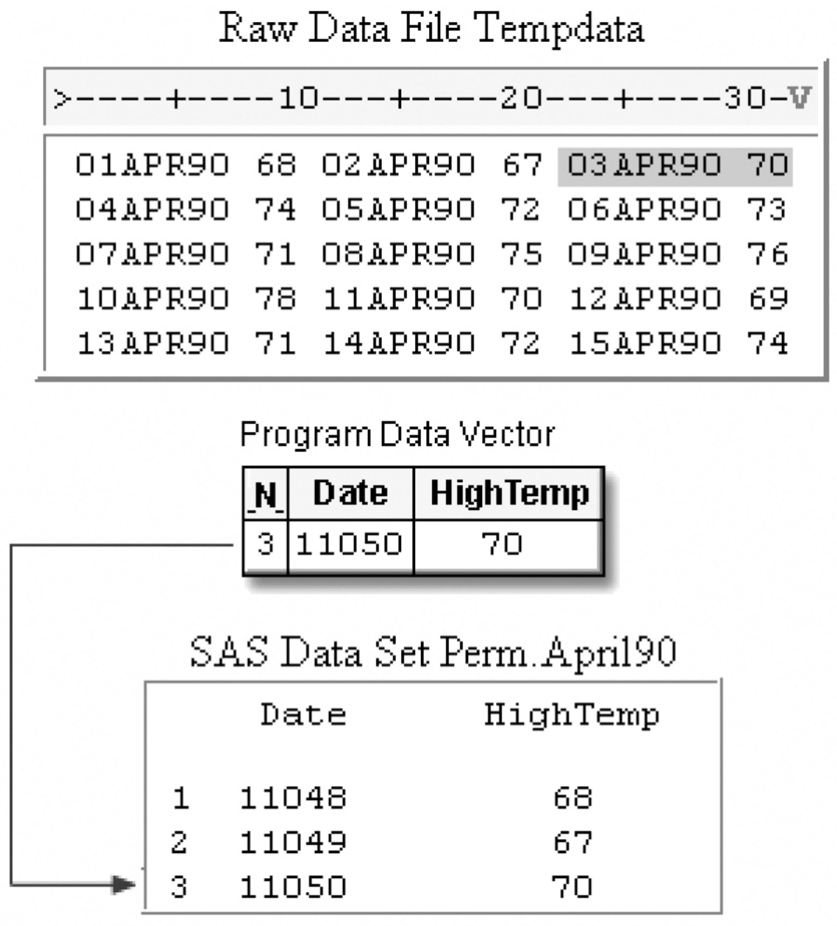
Control returns to the top of the DATA step, and the values of Date and HighTemp are reset to missing.

Program Date Vector		
_N_	Date	HighTemp
3	•	•

Reading Repeating Blocks of Data

DATA Step Processing of Repeating Blocks of Data

During the third iteration, the last block of values is read and written to the dataset as the third observation, and then @@ is released.



Raw Data File Tempdata

```
>----+----10---+----20---+----30-V
 01APR90 68 02APR90 67 03APR90 70
 04APR90 74 05APR90 72 06APR90 73
 07APR90 71 08APR90 75 09APR90 76
 10APR90 78 11APR90 70 12APR90 69
 13APR90 71 14APR90 72 15APR90 74
```

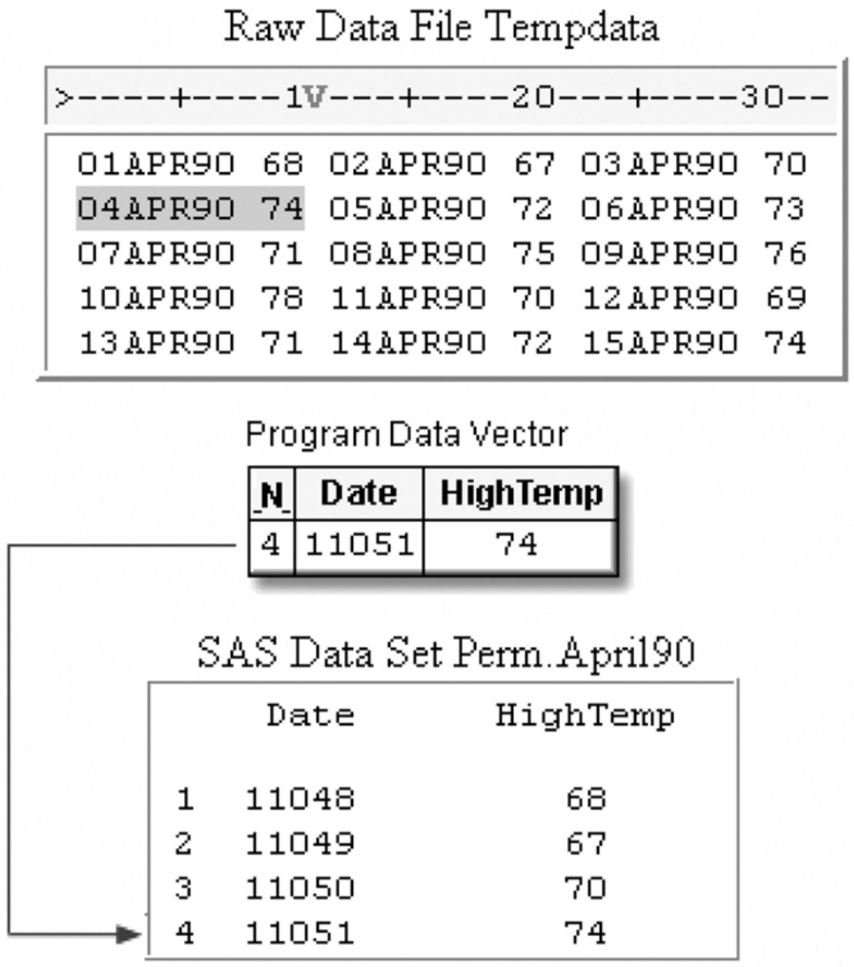
Program Data Vector

_N_	Date	HighTemp
3	11050	70

SAS Data Set Perm.April90

```
        Date        HighTemp

 1     11048           68
 2     11049           67
 3     11050           70
```

During the fourth iteration, the first block of values in the second record is read and written as the fourth observation.



Raw Data File Tempdata

```
>----+----1V---+----20---+----30--
 01APR90 68 02APR90 67 03APR90 70
 04APR90 74 05APR90 72 06APR90 73
 07APR90 71 08APR90 75 09APR90 76
 10APR90 78 11APR90 70 12APR90 69
 13APR90 71 14APR90 72 15APR90 74
```

Program Data Vector

_N_	Date	HighTemp
4	11051	74

SAS Data Set Perm.April90

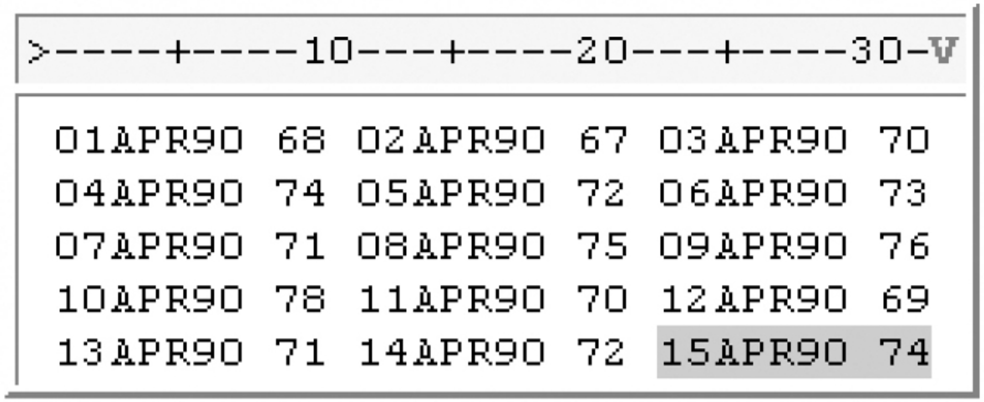
```
        Date        HighTemp

 1     11048           68
 2     11049           67
 3     11050           70
 4     11051           74
```

Reading Repeating Blocks of Data

DATA Step Processing of Repeating Blocks of Data

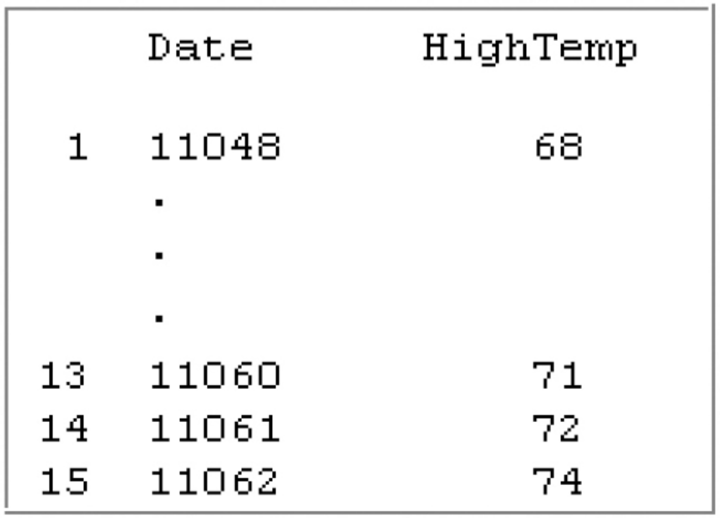
The execution phase continues until the last block of data is read and written.

Raw Data File Tempdata



```
>----+----10---+----20---+----30-V
 01APR90 68 02APR90 67 03APR90 70
 04APR90 74 05APR90 72 06APR90 73
 07APR90 71 08APR90 75 09APR90 76
 10APR90 78 11APR90 70 12APR90 69
 13APR90 71 14APR90 72 15APR90 74
```

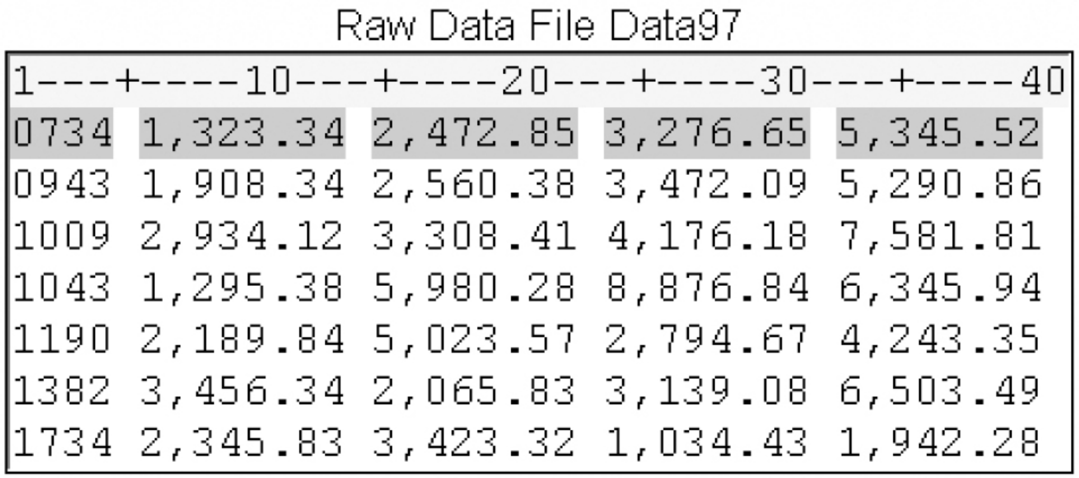
SAS Data Set Perm.April90



```
       Date       HighTemp

 1     11048         68
       .
       .
       .
13     11060         71
14     11061         72
15     11062         74
```

Reading the Same Number of Repeating Fields

Each record in the file Data97 contains a sales representative's **ID number**, followed by four repeating fields that represent his or her quarterly sales totals of 1997. You want to pair each employee ID number with one quarterly sales total to produce a single observation. Four observations are generated from each record.


Raw Data File Data97

```
1---+----10---+----20---+----30---+----40
0734 1,323.34 2,472.85 3,276.65 5,345.52
0943 1,908.34 2,560.38 3,472.09 5,290.86
1009 2,934.12 3,308.41 4,176.18 7,581.81
1043 1,295.38 5,980.28 8,876.84 6,345.94
1190 2,189.84 5,023.57 2,794.67 4,243.35
1382 3,456.34 2,065.83 3,139.08 6,503.49
1734 2,345.83 3,423.32 1,034.43 1,942.28
```

ID	Quarter	Sales
0734	1	1323.34
0734	2	2472.85
0734	3	3276.65
0734	4	5345.52
0943	1	1908.34
0943	2	2560.38
0943	3	3472.09
0943	4	5290.86

To accomplish this, you must execute the DATA step once for each record, repetitively reading and writing values in one DATA step iteration. This means that the DATA step must

- read the value for ID and hold the current record,
- create a new variable named Quarter for each sales figure,
- read a new value for Sales and write the values to the dataset as an observation,
- continue reading a new value for Sales and writing values to the dataset 3 more times.

Reading the Same Number of Repeating Fields

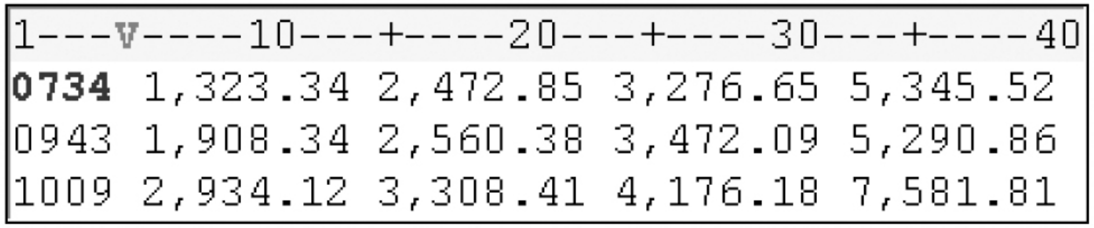
Using the Single Trailing At Sign (@) to Hold the Current Record

First, you need to read the value for ID and hold the record so that subsequent values for Sales can be read.

In this case, you want to hold the record with the single trailing @, so that a second INPUT statement can read the multiple sales values from a single record within the same iteration of the DATA step.

```
data perm.sales97;
  infile data97;
  input ID $ @;
```

Raw Data File Data97



```
1---V----10---+----20---+----30---+----40
0734 1,323.34 2,472.85 3,276.65 5,345.52
0943 1,908.34 2,560.38 3,472.09 5,290.86
1009 2,934.12 3,308.41 4,176.18 7,581.81
```

Like @@, the single trailing @

- enables the next INPUT statement to continue reading from the same record.
- releases the current record when control returns to the top of the DATA step or when a subsequent INPUT statement executes without a line-hold specifier.

Reading the Same Number of Repeating Fields

Using the Single Trailing At Sign (@) to Hold the Current Record

The second INPUT statement reads a value for Sales and holds the record with the single trailing at sign @. An OUTPUT statement writes the observation to the SAS dataset, and the DATA step continues processing. When all of the repeating fields have been read and output, control returns to the top of the DATA step, and the record is released.

```
data perm.sales97;
  infile data97;
  input ID $ @;
  Quarter=1;
  input Sales : comma. @;
  output;
  Quarter+1;
  input Sales : comma. @;
  output;
  Quarter+1;
  input Sales : comma. @;
  output;
  Quarter+1;
  input Sales : comma. @;
  output;
run;
```

A neater program using a DO loop

```
data perm.sales97;
  infile data97;
  input ID $ @;
  do Quarter=1 to 4;
    input Sales : comma. @;
    output;
  end;
run;
```

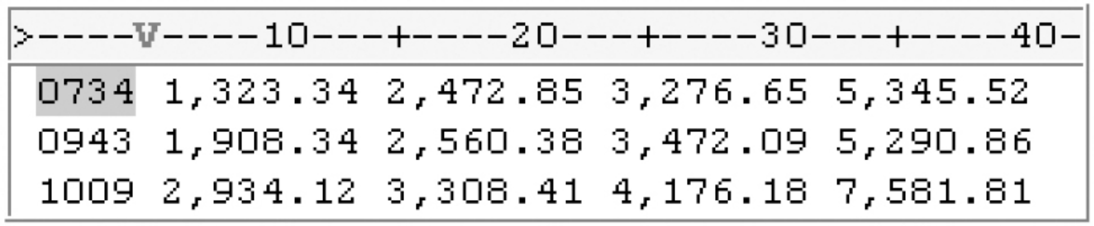
Reading the Same Number of Repeating Fields

Processing a DATA Step That Contains an Iterative DO Loop

```
data perm.sales97;
  infile data97;
  input ID $ @;
  do Quarter=1 to 4;
    input Sales : comma. @;
    output;
  end;
run;
```

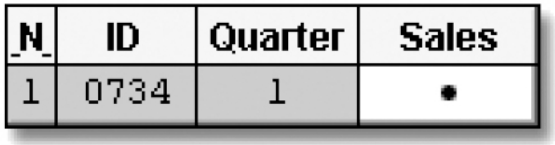
During the first iteration, the value for ID is read and Quarter is initialized to **1** as the loop begins to execute.

Raw Data File Data97



```
>----V----10---+----20---+----30---+----40-
 0734 1,323.34 2,472.85 3,276.65 5,345.52
 0943 1,908.34 2,560.38 3,472.09 5,290.86
 1009 2,934.12 3,308.41 4,176.18 7,581.81
```

Program Data Vector



_N_	ID	Quarter	Sales
1	0734	1	•

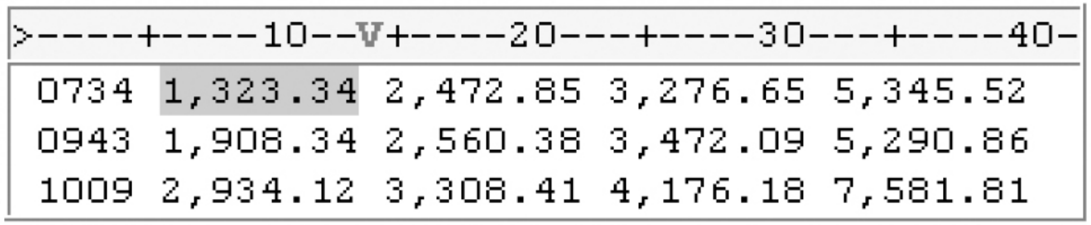
Reading the Same Number of Repeating Fields

Processing a DATA Step That Contains an Iterative DO Loop

```
data perm.sales97;
  infile data97;
  input ID $ @;
  do Quarter=1 to 4;
    input Sales : comma. @;
    output;
  end;
run;
```

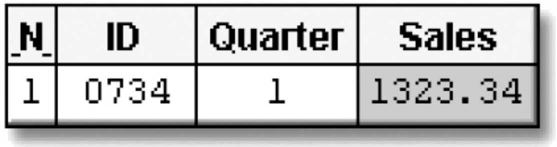
The INPUT statement reads the first repeating field and assigns the value to Sales in the program data vector. The @ holds the current record.

Raw Data File Data97



```
>----+----10--V+----20---+----30---+----40-
 0734 1,323.34 2,472.85 3,276.65 5,345.52
 0943 1,908.34 2,560.38 3,472.09 5,290.86
 1009 2,934.12 3,308.41 4,176.18 7,581.81
```

Program Data Vector



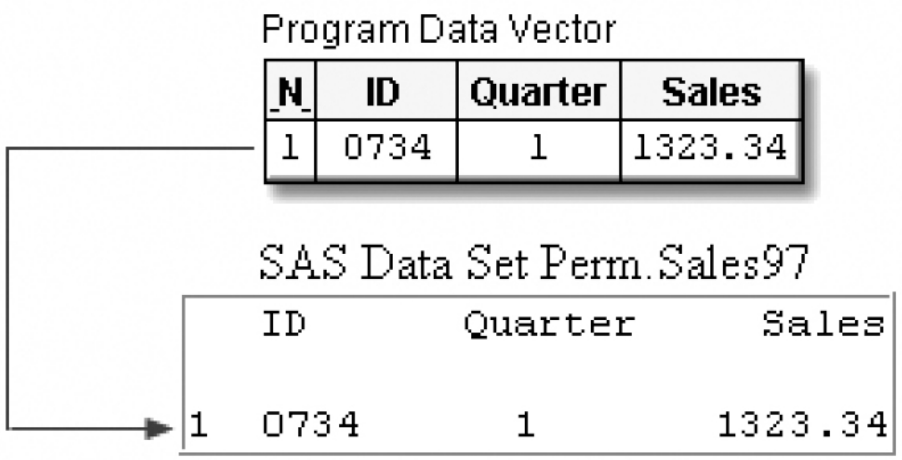
_N_	ID	Quarter	Sales
1	0734	1	1323.34

Reading the Same Number of Repeating Fields

Processing a DATA Step That Contains an Iterative DO Loop

```
data perm.sales97;
  infile data97;
  input ID $ @;
  do Quarter=1 to 4;
    input Sales : comma. @;
    output;
  end;
run;
```

The OUTPUT statement writes the values in the program data vector to the dataset as the first observation.



Program Data Vector

_N_	ID	Quarter	Sales
1	0734	1	1323.34

SAS Data Set Perm.Sales97

```
      ID          Quarter        Sales

1   0734            1          1323.34
```

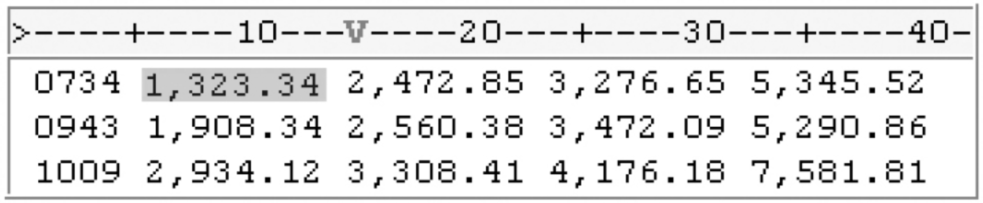
Reading the Same Number of Repeating Fields

Processing a DATA Step That Contains an Iterative DO Loop

The END statement indicates the bottom of the loop, but control returns to the DO statement, not to the top of the DATA step. Now the value of Quarter is incremented to **2**, and the Sales values is still the same.

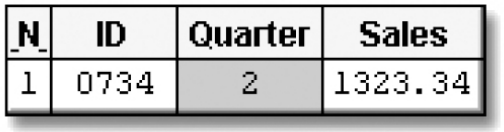
```
data perm.sales97;
  infile data97;
  input ID $ @;
  do Quarter=1 to 4;
    input Sales : comma. @;
    output;
  end;
run;
```

Raw Data File Data97



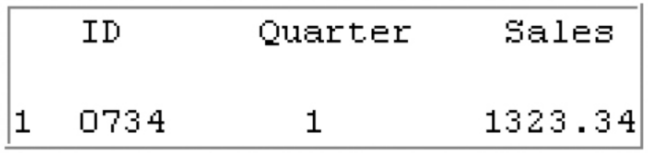
```
>----+----10---V----20---+----30---+----40-
 0734 1,323.34 2,472.85 3,276.65 5,345.52
 0943 1,908.34 2,560.38 3,472.09 5,290.86
 1009 2,934.12 3,308.41 4,176.18 7,581.81
```

Program Data Vector



_N_	ID	Quarter	Sales
1	0734	2	1323.34

SAS Data Set Perm.Sales97



```
    ID          Quarter       Sales

1   0734           1         1323.34
```

Reading the Same Number of Repeating Fields

Processing a DATA Step That Contains an Iterative DO Loop

```
do Quarter=1 to 4;
     input Sales : comma. @;
     output;
  end;
```

The INPUT statement executes again, reading the second repeating field and storing the value for Sales in the program data vector. The @ holds the record.

Raw Data File Data97

```
>----+----10---+----20-V-+----30---+----40-
 0734 1,323.34 2,472.85 3,276.65 5,345.52
 0943 1,908.34 2,560.38 3,472.09 5,290.86
 1009 2,934.12 3,308.41 4,176.18 7,581.81
```

Program Data Vector

N	ID	Quarter	Sales
1	0734	2	2472.85

SAS Data Set Perm.Sales97

```
    ID          Quarter       Sales

1   0734          1         1323.34
```

The OUTPUT statement writes the values in the program data vector as the second observation.

Raw Data File Data97

```
>----+----10---+----20-V-+----30---+----40-
 0734 1,323.34 2,472.85 3,276.65 5,345.52
 0943 1,908.34 2,560.38 3,472.09 5,290.86
 1009 2,934.12 3,308.41 4,176.18 7,581.81
```

Program Data Vector

N	ID	Quarter	Sales
1	0734	2	2472.85

SAS Data Set Perm.Sales97

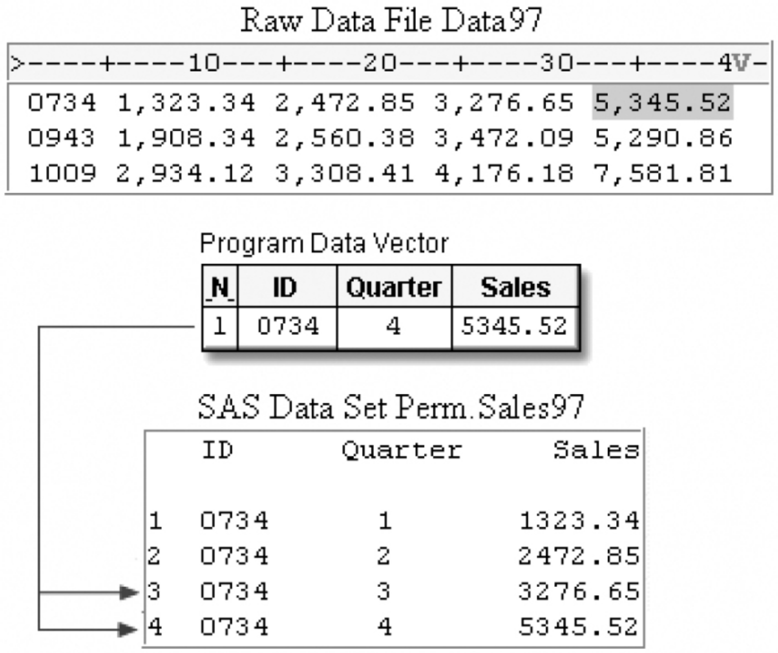
```
    ID          Quarter       Sales

1   0734          1         1323.34
2   0734          2         2472.85
```

Reading the Same Number of Repeating Fields

Processing a DATA Step That Contains an Iterative DO Loop

```
do Quarter=1 to 4;
     input Sales : comma. @;
     output;
  end;
```

The loop continues executing while the value for Quarter is **3**, and then **4**. In the process, the third and fourth observations are written.



Raw Data File Data97

```
>----+----10---+----20---+----30---+----4V-
 0734 1,323.34 2,472.85 3,276.65 5,345.52
 0943 1,908.34 2,560.38 3,472.09 5,290.86
 1009 2,934.12 3,308.41 4,176.18 7,581.81
```

Program Data Vector

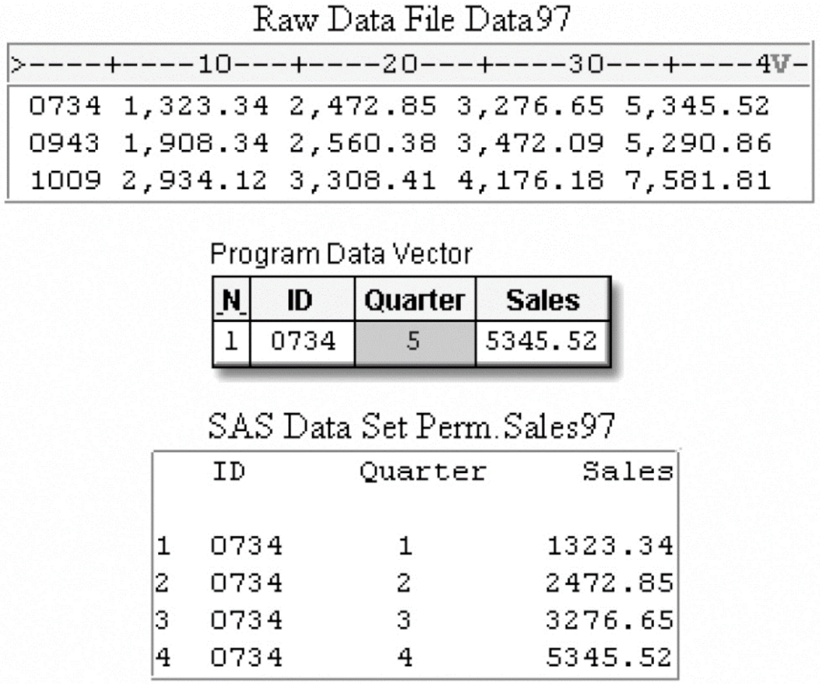
N	ID	Quarter	Sales
1	0734	4	5345.52

SAS Data Set Perm.Sales97

```
    ID         Quarter       Sales

1   0734          1        1323.34
2   0734          2        2472.85
3   0734          3        3276.65
4   0734          4        5345.52
```

```
do Quarter=1 to 4;
     input Sales : comma. @;
     output;
  end;
```

After the 4th observation is written, control returns to the top of the loop and Quarter is incremented to **5.** The loop stops.



Raw Data File Data97

```
>----+----10---+----20---+----30---+----4V-
 0734 1,323.34 2,472.85 3,276.65 5,345.52
 0943 1,908.34 2,560.38 3,472.09 5,290.86
 1009 2,934.12 3,308.41 4,176.18 7,581.81
```

Program Data Vector

N	ID	Quarter	Sales
1	0734	5	5345.52

SAS Data Set Perm.Sales97

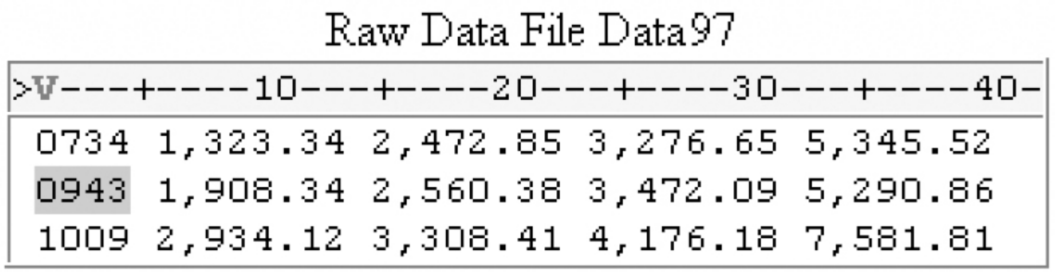
```
    ID         Quarter       Sales

1   0734          1        1323.34
2   0734          2        2472.85
3   0734          3        3276.65
4   0734          4        5345.52
```

Reading the Same Number of Repeating Fields

Processing a DATA Step That Contains an Iterative DO Loop

```
data perm.sales97;
  infile data97;
  input ID $ @;
  do Quarter=1 to 4;
    input Sales : comma. @;
    output;
  end;
run;
```

Raw Data File Data97



```
>V---+----10---+----20---+----30---+----40-
0734 1,323.34 2,472.85 3,276.65 5,345.52
0943 1,908.34 2,560.38 3,472.09 5,290.86
1009 2,934.12 3,308.41 4,176.18 7,581.81
```

When the execution phase is complete, you can display the dataset with the PRINT procedure.

Obs	ID	Quarter	Sales
1	0734	1	1323.34
2	0734	2	2472.85
3	0734	3	3276.65
4	0734	4	5345.52
5	0943	1	1908.34
6	0943	2	2560.38
7	0943	3	3472.09
8	0943	4	5290.86
9	1009	1	2934.12
10	1009	2	3308.41
11	1009	3	4176.18
12	1009	4	7581.81

Control returns to the top of the DATA step, and the input pointer moves to column 1 of the next record. The variable values in the program data vector are reset to missing, and then the process continues…

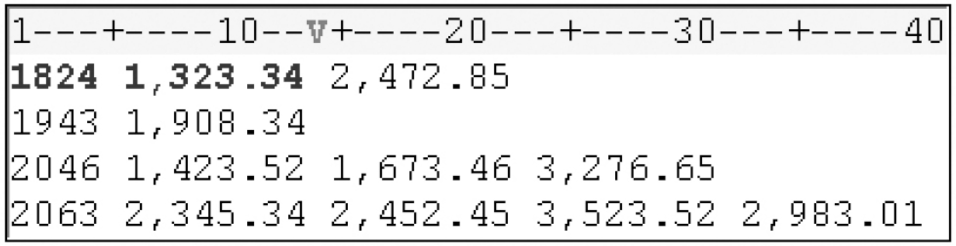
Program Date Vector			
_N_	ID	Quarter	Sales
2	•	•	•

Reading a Varying Number of Repeating Fields

Using the MISSOVER Option

The following dataset contains records of a varying number of repeating fields (i.e., with missing values). The DATA step must read the same record repeatedly; however, you need to prevent the input pointer from moving to the next record when there are missing Sales values by using the MISSOVER option in the INFILE statement . A DO loop reads the rest of Sales values conditionally.

Raw Data File Data97



```
1---+----10--V+----20---+----30---+----40
1824 1,323.34 2,472.85
1943 1,908.34
2046 1,423.52 1,673.46 3,276.65
2063 2,345.34 2,452.45 3,523.52 2,983.01
```

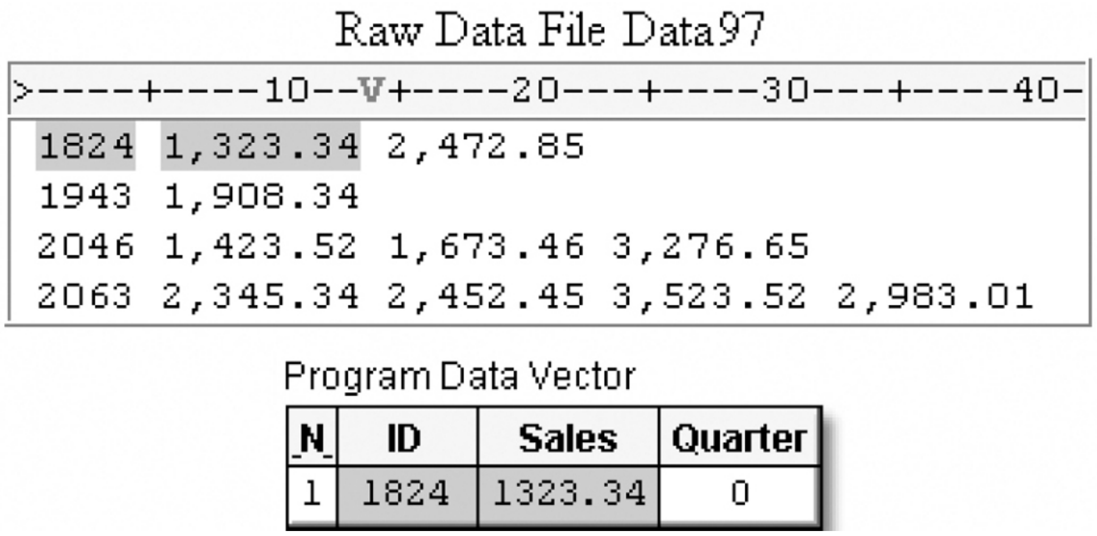
```
data perm.sales97;
  infile data97 missover;
  input ID $ Sales : comma. @;
  Quarter=0;
  do while (sales ne .);
    quarter+1;
    output;
    input sales : comma. @;
  end;
run;
```

Because there is at least one value for the repeating field, Sales, in each record, the first INPUT statement reads both the value for ID and the first Sales value for each record. The trailing @ holds the record so that any subsequent repeating fields can be read.

Processing a DATA Step That Has a Varying Number of Repeating Fields

During the first iteration of the DATA step, values for ID and Sales are read. The trailing @ holds the record, and variable Quarter is initialized to 0.

```
data perm.sales97;
  infile data97 missover;
  input ID $ Sales : comma. @;
  Quarter=0;
  do while (sales ne .);
    quarter+1;
    output;
    input sales : comma. @;
  end;
run;
```


Raw Data File Data97

```
>----+----10--V+----20---+----30---+----40-
 1824 1,323.34 2,472.85
 1943 1,908.34
 2046 1,423.52 1,673.46 3,276.65
 2063 2,345.34 2,452.45 3,523.52 2,983.01
```

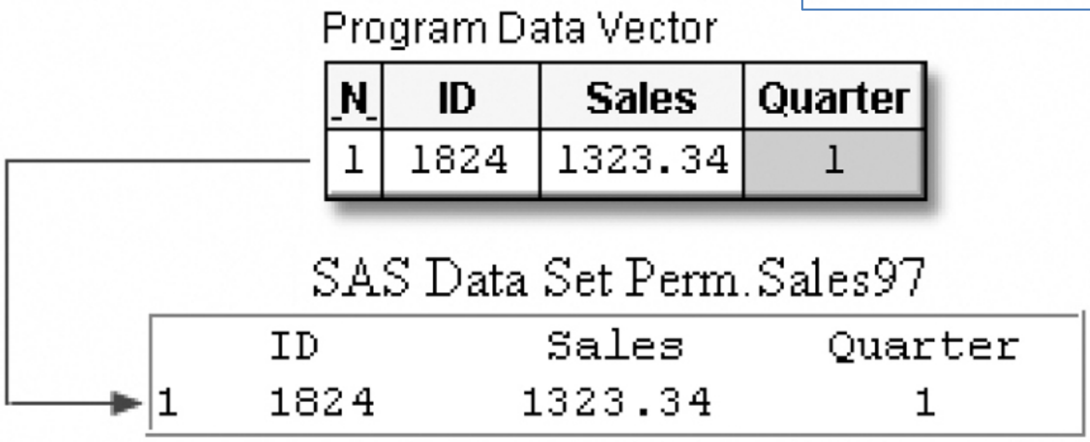
Program Data Vector

N	ID	Sales	Quarter
1	1824	1323.34	0

Processing a DATA Step That Has a Varying Number of Repeating Fields

The DO WHILE statement checks to see if Sales has a value, which it does, so the other statements in the DO loop execute. The value of Quarter is incremented by 1 and OUTPUT statement writes the current observation to the dataset.

```
data perm.sales97;
  infile data97 missover;
  input ID $ Sales : comma. @;
  Quarter=0;
  do while (sales ne .);
    quarter+1;
    output;
    input sales : comma. @;
  end;
run;
```


Program Data Vector

_N_	ID	Sales	Quarter
1	1824	1323.34	1

SAS Data Set Perm.Sales97

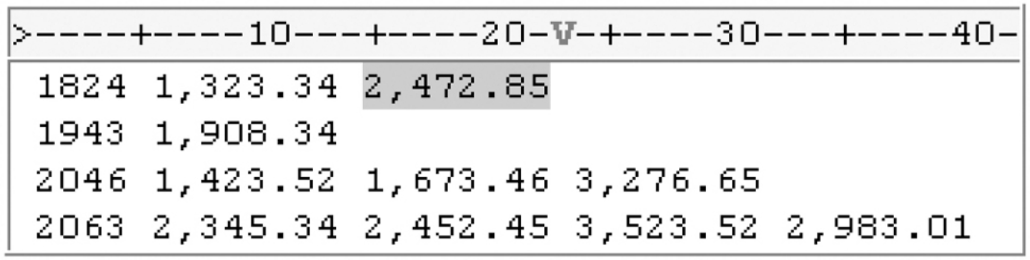
```
      ID          Sales       Quarter
1     1824        1323.34        1
```

Processing a DATA Step That Has a Varying Number of Repeating Fields

The INPUT statement reads the next value for Sales, the end of the loop is reached, and control returns to the DO WHILE statement.

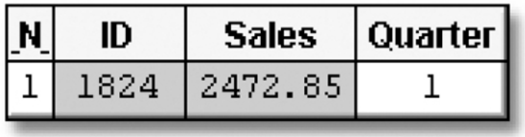
```
data perm.sales97;
  infile data97 missover;
  input ID $ Sales : comma. @;
  Quarter=0;
  do while (sales ne .);
    quarter+1;
    output;
    input sales : comma. @;
  end;
run;
```

Raw Data File Data97



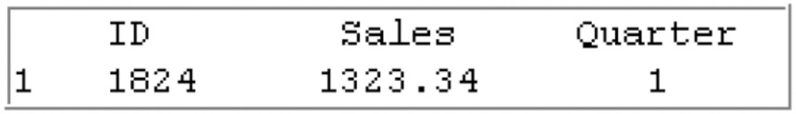
```
>----+----10---+----20-V-+----30---+----40-
 1824 1,323.34 2,472.85
 1943 1,908.34
 2046 1,423.52 1,673.46 3,276.65
 2063 2,345.34 2,452.45 3,523.52 2,983.01
```

Program Data Vector



_N_	ID	Sales	Quarter
1	1824	2472.85	1

SAS Data Set Perm.Sales97

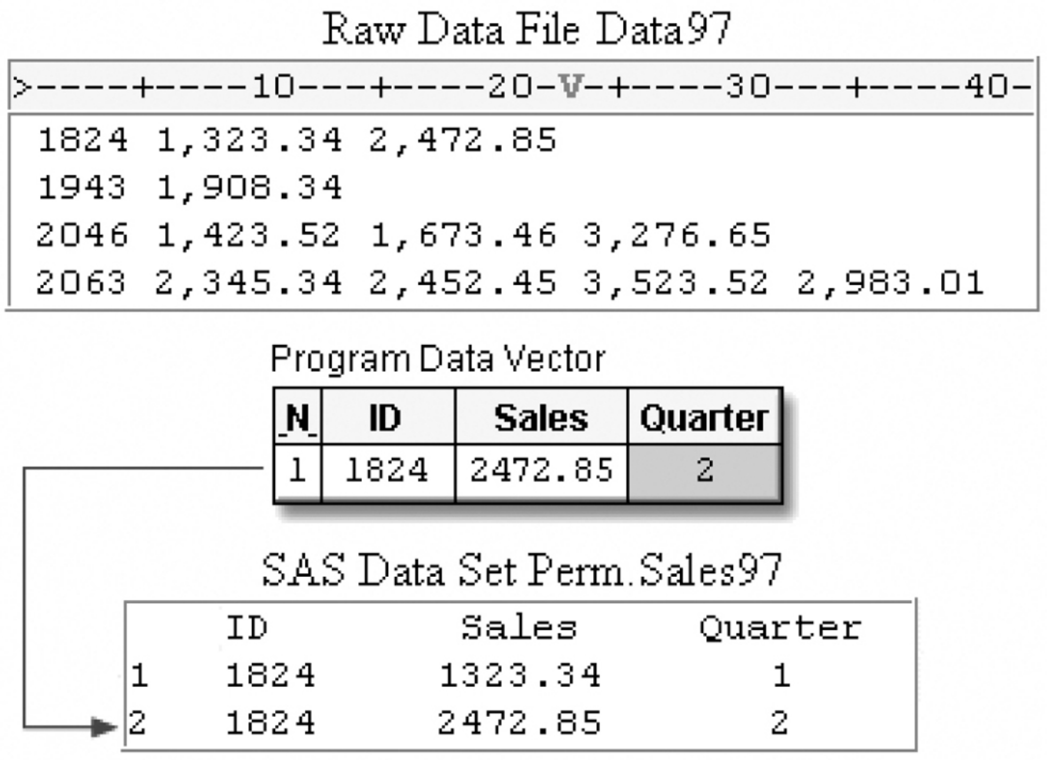


	ID	Sales	Quarter
1	1824	1323.34	1

Processing a DATA Step That Has a Varying Number of Repeating Fields

The condition is checked and Sales still has a value, so the loop executes again. Quarter is incremented to 2, and the values in the program data vector are written out as the second observation.

```
data perm.sales97;
  infile data97 missover;
  input ID $ Sales : comma. @;
  Quarter=0;
  do while (sales ne .);
    quarter+1;
    output;
    input sales : comma. @;
  end;
run;
```



Raw Data File Data97

```
>----+----10---+----20-V-+----30---+----40-
 1824 1,323.34 2,472.85
 1943 1,908.34
 2046 1,423.52 1,673.46 3,276.65
 2063 2,345.34 2,452.45 3,523.52 2,983.01
```

Program Data Vector

_N_	ID	Sales	Quarter
1	1824	2472.85	2

SAS Data Set Perm.Sales97

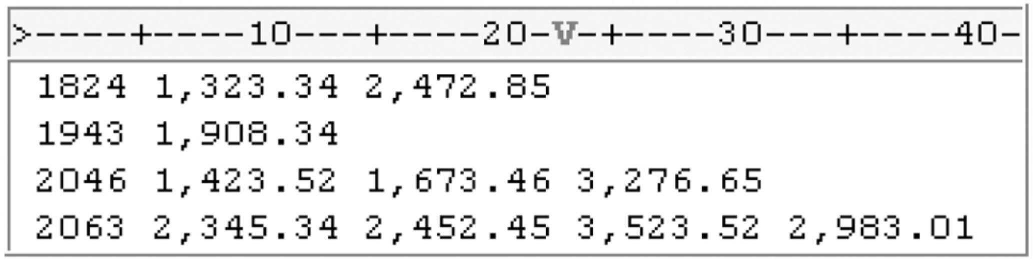
```
     ID          Sales        Quarter
1    1824       1323.34          1
2    1824       2472.85          2
```

Processing a DATA Step That Has a Varying Number of Repeating Fields

The INPUT statement executes again, but there is no value left in the 1st record. The **MISSOVER** option prevents the input pointer from moving to the next record in search of another value for Sales. Therefore, SALES receives a missing value.

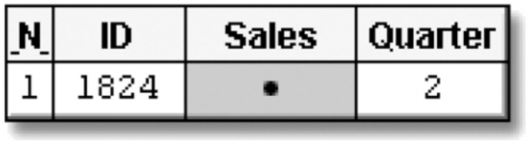
```
data perm.sales97;
  infile data97 missover;
  input ID $ Sales : comma. @;
  Quarter=0;
  do while (sales ne .);
    quarter+1;
    output;
    input sales : comma. @;
  end;
run;
```

Raw Data File Data97



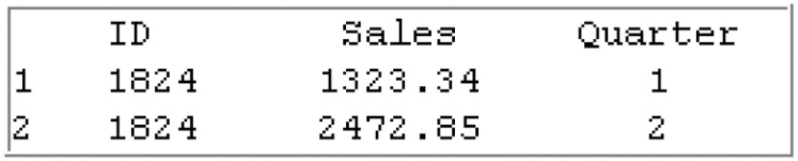
```
>----+----10---+----20-V-+----30---+----40-
 1824 1,323.34 2,472.85
 1943 1,908.34
 2046 1,423.52 1,673.46 3,276.65
 2063 2,345.34 2,452.45 3,523.52 2,983.01
```

Program Data Vector



_N_	ID	Sales	Quarter
1	1824	•	2

SAS Data Set Perm.Sales97



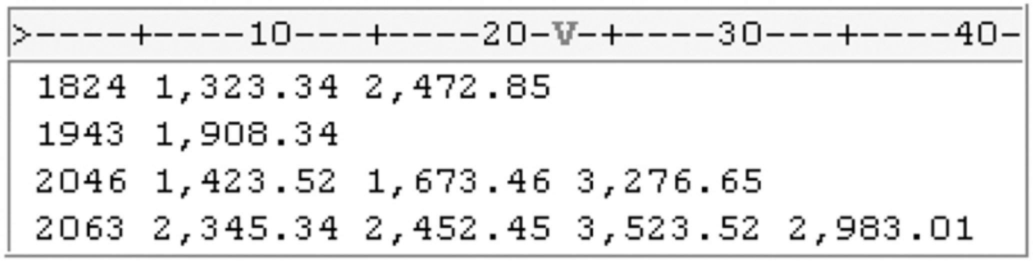
```
      ID         Sales       Quarter
1     1824       1323.34        1
2     1824       2472.85        2
```

Processing a DATA Step That Has a Varying Number of Repeating Fields

The end of the loop is reached, and control returns to the DO WHILE statement. Because the condition is now false, the statements in the loop are not executed and the values in the PDV are not output.

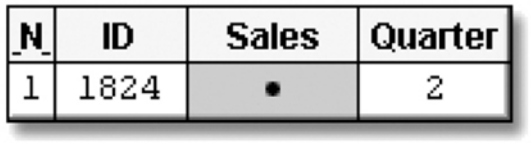
```
data perm.sales97;
  infile data97 missover;
  input ID $ Sales : comma. @;
  Quarter=0;
  do while (sales ne .);
    quarter+1;
    output;
    input sales : comma. @;
  end;
run;
```

Raw Data File Data97



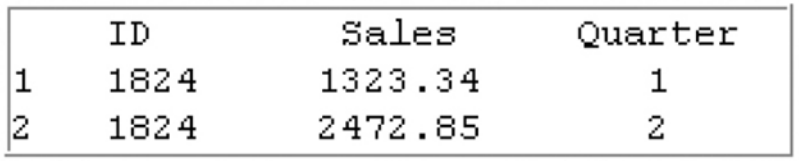
```
>----+----10---+----20-V-+----30---+----40-
 1824 1,323.34 2,472.85
 1943 1,908.34
 2046 1,423.52 1,673.46 3,276.65
 2063 2,345.34 2,452.45 3,523.52 2,983.01
```

Program Data Vector



N	ID	Sales	Quarter
1	1824	•	2

SAS Data Set Perm.Sales97

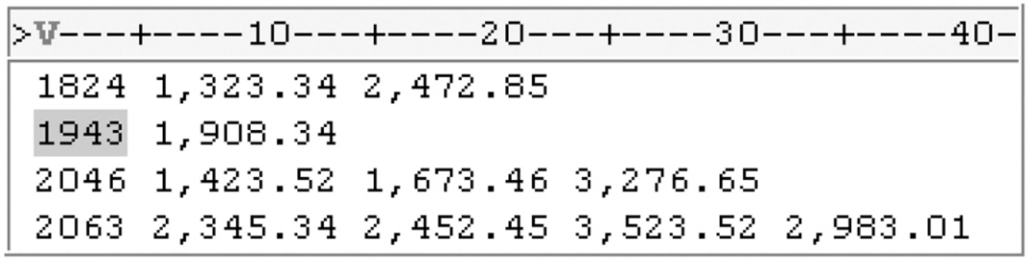


```
      ID          Sales      Quarter
1     1824      1323.34         1
2     1824      2472.85         2
```

Processing a DATA Step That Has a Varying Number of Repeating Fields

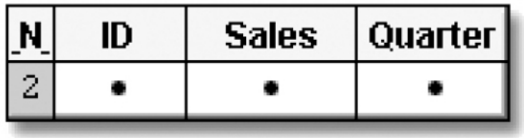
Now control returns to the top of the DATA step, the values in the program data vector are reset to missing, and the input statement reads the next record. The DATA step continues executing until the end of the file.

Raw Data File Data97



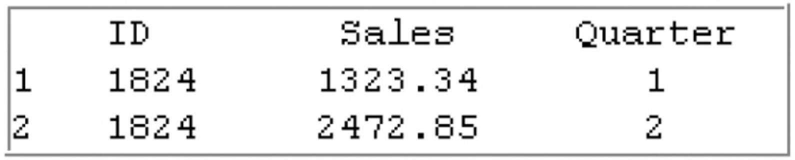
```
>V---+----10---+----20---+----30---+----40-
 1824 1,323.34 2,472.85
 1943 1,908.34
 2046 1,423.52 1,673.46 3,276.65
 2063 2,345.34 2,452.45 3,523.52 2,983.01
```

Program Data Vector



_N_	ID	Sales	Quarter
2	•	•	•

SAS Data Set Perm.Sales97



```
      ID          Sales      Quarter
1     1824        1323.34       1
2     1824        2472.85       2
```

```
data perm.sales97;
  infile data97 missover;
  input ID $ Sales : comma. @;
  Quarter=0;
  do while (sales ne .);
    quarter+1;
    output;
    input sales : comma. @;
  end;
run;
```

Obs	ID	Sales	Quarter
1	1824	1323.34	1
2	1824	2472.85	2
3	1943	1908.34	1
4	2046	1423.52	1
5	2046	1673.46	2
6	2046	3276.65	3
…	…	…	…