SAS Certification Prep Guide: Base Programming for SAS 9, Third Edition

by SAS Institute

Chapter 19: Creating a Single Observation from Multiple Records

Overview

Introduction

Information for one observation can be spread out over several raw data file records. You can write multiple INPUT statements when multiple input records comprise a single observation, as shown below.

```
input Lname $ 1-8 Fname $ 10-15;
input Department $ 1-12 JobCode $ 15-19;
input Salary comma10.;
```

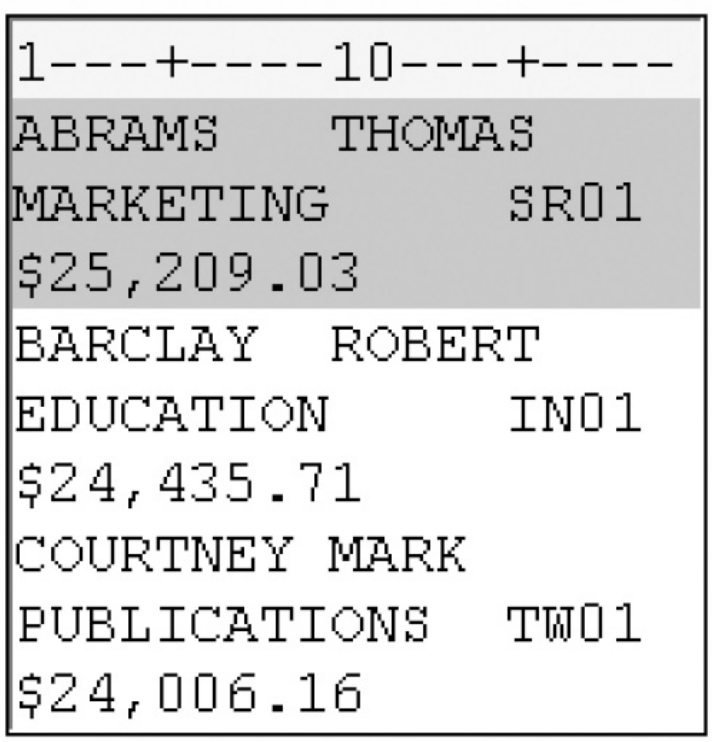

Figure 19.1: Multiple Records Comprising Each Observation

Alternatively, you can write one INPUT statement that contains a line pointer control to specify the record(s) from which values are to be read.

```
input #1 Lname $ 1-8 Fname $ 10-15
      #2 Department $ 1-12 JobCode $ 15-19
      #3 Salary comma10.;
```

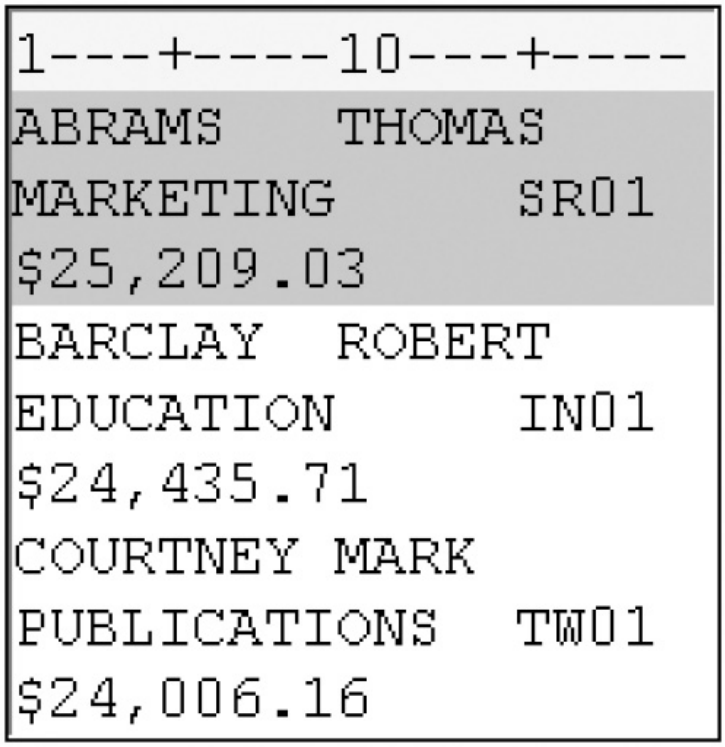

Figure 19.2: Multiple Records Comprising Each Observation

Objectives

In this chapter, you learn to

- read multiple records sequentially and create a single observation
- read multiple records non-sequentially and create a single observation.

Using Line Pointer Controls

You know that as SAS reads raw data values, it keeps track of its position with an input pointer. You have used column pointer controls and column specifications to determine the column placement of the input pointer.

Table 19.1: Input Statements for Column Specifications and Column Pointer Controls

Column Specifications	`input Name $ 1-12 Age 15-16 Gender $ 18;`
Column Pointer Controls	`input Name $12. @15 Age 2. @18 Gender $1.;`

But you can also position the input pointer on a specific record by using a line pointer control in the INPUT statement.

Table 19.2: INPUT Statement and Raw Data File Admit

```
input #2 Name $ 1-12 Age 15-16
Gender $ 18;
```

Raw Data File Admit

```
1---+----10---+----
S. Thompson   37 M
L. Rochester  31 F
M. Sabatello  43 M
```

There are two types of line pointer controls.

- The forward slash (/) specifies a line location that is relative to the current one.
- The #*n* specifies the absolute number of the line to which you want to move the pointer.

First we'll look at the forward slash (/). Later in the chapter, you'll learn how to use the #n, and you will see how these two

controls can be combined.

Reading Multiple Records Sequentially

The Forward Slash (/) Line Pointer Control

Use the forward slash (/) line pointer control to read multiple records sequentially. The / advances the input pointer to the next record. The / line pointer control moves the input pointer forward only and must be specified *after* the instructions for reading the values in the current record.

The single INPUT statement below reads the values for Lname and Fname in the first record, followed by the values for Department and JobCode in the second record. Then the value for Salary is read in the third record.

```
input Lname $ 1-8 Fname $ 10-15 /
      Department $ 1-12 JobCode $ 15-19 /
      Salary comma10.;
```

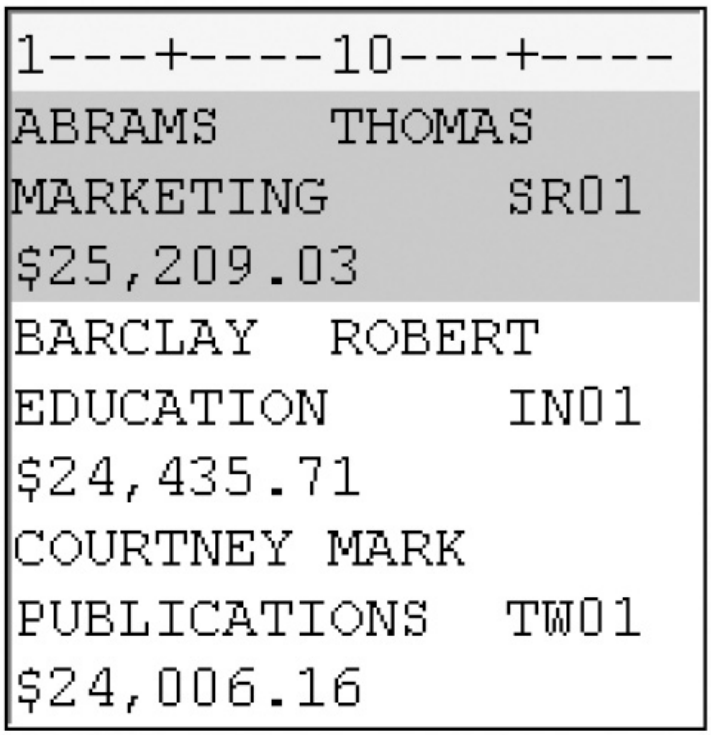

Figure 19.3: Multiple Records Comprising Each Observation

Using the / Line Pointer Control

Look at the forward slash (/) line pointer control in the following example.

The raw data file Memdata contains the mailing list of a professional organization. Your task is to combine the information for each member into a single observation. We'll begin by reading each member's name, followed by the street address, and finally the city, state, and zip code.

- As you write the instructions to read the values for Fname and Lname, notice that not all of the values for Lname begin in the same column. So, you should use standard list input to read these values.

```
data perm.members;
   infile memdata;
   input Fname $ Lname $
```

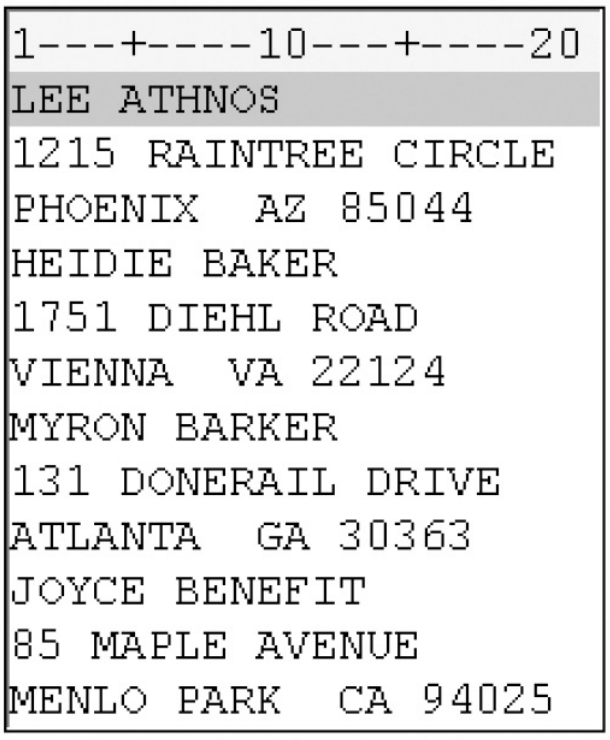

Figure 19.4: Raw Data File Memdata

- Now you want to read the values for Address from the second record. The / line pointer control advances the input pointer to the next record. At this point the INPUT statement is incomplete, so you should not place a semicolon after the line pointer control.

```
data perm.members;
   infile memdata;
   input Fname $ Lname $ /
```

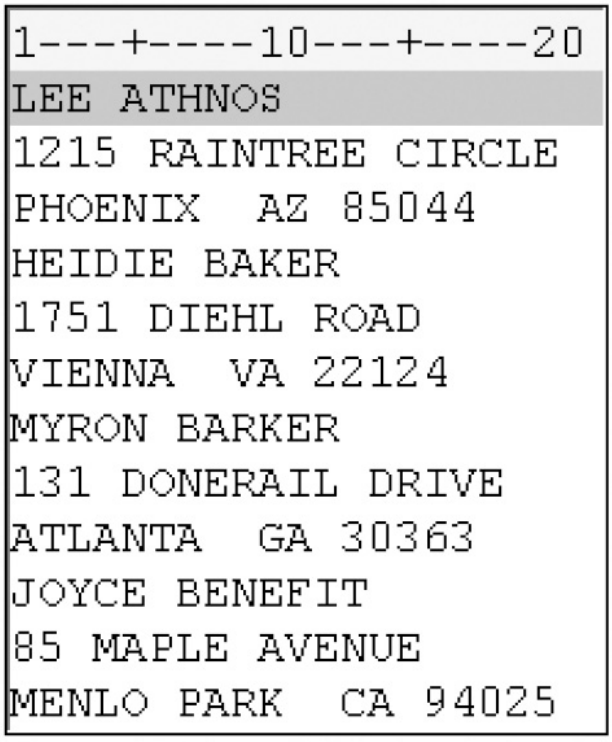

Figure 19.5: Raw Data File Memdata

- You can use column input to read the values in the next record as one variable named Address. Then add a line pointer control to move the input pointer to the next record.

```
data perm.members;
   infile memdata;
   input Fname $ Lname $ /
         Address $ 1-20 /
```

```
1---+----10---+----20
LEE ATHNOS
1215 RAINTREE CIRCLE
PHOENIX  AZ 85044
HEIDIE BAKER
1751 DIEHL ROAD
VIENNA  VA 22124
MYRON BARKER
131 DONERAIL DRIVE
ATLANTA  GA 30363
JOYCE BENEFIT
85 MAPLE AVENUE
MENLO PARK  CA 94025
```

Figure 19.6: Line Pointer Control Advanced One Record

- As you write the statements to read the values for City, notice that some of the values are longer than eight characters and contain embedded blanks. Also note that each value is followed by two consecutive blanks. To read these values, you should use modified list input with the ampersand (&) modifier.

  The values for State and the values for Zip do not begin in the same column. Therefore, you should use list input to read these values.

```
data perm.members;
   infile memdata;
   input Fname $ Lname $ /
         Address $ 1-20 /
         City & $10. State $ Zip $;
run;
```

```
1---+----10---+----20
LEE ATHNOS
1215 RAINTREE CIRCLE
PHOENIX  AZ 85044
HEIDIE BAKER
1751 DIEHL ROAD
VIENNA  VA 22124
MYRON BARKER
131 DONERAIL DRIVE
ATLANTA  GA 30363
JOYCE BENEFIT
85 MAPLE AVENUE
MENLO PARK  CA 94025
```

Figure 19.7: Line Pointer Control Advanced Another Record

Sequential Processing of Multiple Records in the DATA Step

Now that you've learned the basics of using the / line pointer control, let's look at the sequential processing of multiple records in the DATA step.

The values in the first record are read, and the / line pointer control moves the input pointer to the second record.

```
data perm.members;
   infile memdata;
   input Fname $ Lname $ /
         Address $ 1-20 /
         City & $10. State $ Zip $;
run;
```

```
>----+----10---+----20
LEE ATHNOS
1215 RAINTREE CIRCLE
PHOENIX  AZ 85044
HEIDIE BAKER
1751 DIEHL ROAD
VIENNA  VA 22124
```

Program Data Vector

N	Fname	Lname	Address	City	State	Zip
1	LEE	ATHNOS				

Figure 19.8: Reading the First Record of the First Observation

The values for Address are read, and the second / line pointer control advances the input pointer to the third record.

```
data perm.members;
   infile memdata;
   input Fname $ Lname $ /
         Address $ 1-20 /
         City & $10. State $ Zip $;
run;
```

```
>----+----10---+----20
LEE ATHNOS
1215 RAINTREE CIRCLE
PHOENIX  AZ 85044
HEIDIE BAKER
1751 DIEHL ROAD
VIENNA  VA 22124
```

Program Data Vector

_N_	Fname	Lname	Address	City	State	Zip
1	LEE	ATHNOS	1215 RAINTREE CIRCLE			

Figure 19.9: Reading the Second Record of the First Observation

The values for City, State, and Zip are read. The INPUT statement is complete.

```
data perm.members;
   infile memdata;
   input Fname $ Lname $ /
         Address $ 1-20 /
         City & $10. State $ Zip $;
run;
```

```
>----+----10---+----20
LEE ATHNOS
1215 RAINTREE CIRCLE
PHOENIX  AZ 85044
HEIDIE BAKER
1751 DIEHL ROAD
VIENNA  VA 22124
```

Program Data Vector

_N_	Fname	Lname	Address	City	State	Zip
1	LEE	ATHNOS	1215 RAINTREE CIRCLE	PHOENIX	AZ	85044

Figure 19.10: Reading the Third Record of the First Observation

The values in the program data vector are written to the data set as the first observation.

Program Data Vector

_N_	Fname	Lname	Address	City	State	Zip
1	LEE	ATHNOS	1215 RAINTREE CIRCLE	PHOENIX	AZ	85044

SAS Data Set Perm.Members

Fname	Lname	Address	City	State	Zip
LEE	ATHNOS	1215 RAINTREE CIRCLE	PHOENIX	AZ	85044

Figure 19.11: The Program Data Vector and the SAS Data Set Perm.Members

Control returns to the top of the DATA step. The variable values are reinitialized to missing.

```
data perm.members;
   infile memdata;
   input Fname $ Lname $ /
         Address $ 1-20 /
         City & $10. State $ Zip $;
run;
```

```
>----+----10---+----20
LEE ATHNOS
1215 RAINTREE CIRCLE
PHOENIX  AZ 85044
HEIDIE BAKER
1751 DIEHL ROAD
VIENNA  VA 22124
```

Program Data Vector

_N_	Fname	Lname	Address	City	State	Zip
2						

Figure 19.12: Reinitializing the Variable Values to Missing

During the second iteration, values for Fname and Lname are read beginning in column 1 of the fourth record.

```
data perm.members;
   infile memdata;
   input Fname $ Lname $ /
         Address $ 1-20 /
         City & $10. State $ Zip $;
run;
```

```
>----+----10---+----20
LEE ATHNOS
1215 RAINTREE CIRCLE
PHOENIX  AZ 85044
HEIDIE BAKER
1751 DIEHL ROAD
VIENNA  VA 22124
```

Program Data Vector

_N_	Fname	Lname	Address	City	State	Zip
2	HEIDIE	BAKER				

Figure 19.13: Reading the First Record of the Second Observation

The values for Address are read. The / line pointer control advances the input pointer to the fifth record.

```
data perm.members;
   infile memdata;
   input Fname $ Lname $ /
         Address $ 1-20 /
         City & $10. State $ Zip $;
run;
```

```
>----+----10---+----20
LEE ATHNOS
1215 RAINTREE CIRCLE
PHOENIX  AZ 85044
HEIDIE BAKER
1751 DIEHL ROAD
VIENNA  VA 22124
```

Program Data Vector

_N_	Fname	Lname	Address	City	State	Zip
2	HEIDIE	BAKER	1751 DIEHL ROAD			

Figure 19.14: Reading the Second Record of the Second Observation

The values for City, State, and Zip are read. The INPUT statement is complete.

```
data perm.members;
   infile memdata;
   input Fname $ Lname $ /
         Address $ 1-20 /
         City & $10. State $ Zip $;
run;
```

```
>----+----10---+----20
LEE ATHNOS
1215 RAINTREE CIRCLE
PHOENIX  AZ 85044
HEIDIE BAKER
1751 DIEHL ROAD
VIENNA  VA 22124
```

Program Data Vector

_N_	Fname	Lname	Address	City	State	Zip
2	HEIDIE	BAKER	1751 DIEHL ROAD	VIENNA	VA	22124

Figure 19.15: Reading the Third Record of the First Observation

The values in the program data vector are written to the data set as the second observation.

Program Data Vector

_N_	Fname	Lname	Address	City	State	Zip
2	HEIDIE	BAKER	1751 DIEHL ROAD	VIENNA	VA	22124

SAS Data Set Perm.Members

Fname	Lname	Address	City	State	Zip
LEE	ATHNOS	1215 RAINTREE CIRCLE	PHOENIX	AZ	85044
HEIDIE	BAKER	1751 DIEHL ROAD	VIENNA	VA	22124

Figure 19.16: The Program Data Vector and the SAS Data Set Perm.Members

After the data set is complete, PROC PRINT output for Perm.Members shows that each observation contains the complete information for one member.

```
proc print data=perm.members;
run;
```

Obs	Fname	Lname	Address	City	State	Zip
1	LEE	ATHNOS	1215 RAINTREE CIRCLE	PHOENIX	AZ	85044
2	HEIDIE	BAKER	1751 DIEHL ROAD	VIENNA	VA	22124
3	MYRON	BARKER	131 DONERAIL DRIVE	ATLANTA	GA	30363
4	JOYCE	BENEFIT	85 MAPLE AVENUE	MENLO PARK	CA	94025

Figure 19.17: PROC PRINT Output of the Complete Perm.Members Data Set

Number of Records per Observation

Note that the raw data file must contain the same number of records for each observation.

Suppose there are only two records for the second member. Remember, the INPUT statement is set up to read three records.

```
data perm.members;
```

```
infile memdata;
input Fname $ Lname $ /
Address $ 1-20 /
City & $10. State $ Zip $;
```

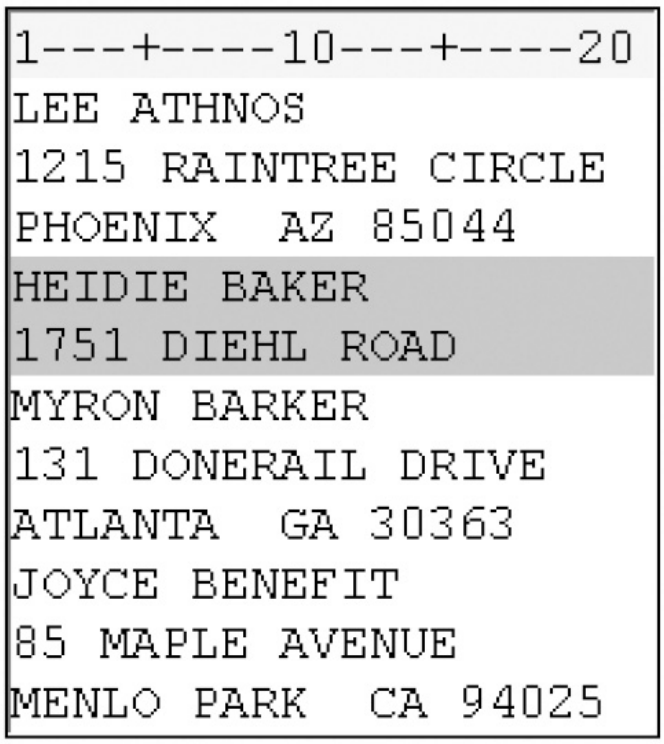

Figure 19.18: Raw Data File Memdata

The second member's name and address are read and assigned to corresponding variables. Then the input pointer advances to the next record, as directed by the INPUT statement, and the third member's name is read as a value for City.

The INPUT statement looks for a value for State and Zip, so the input pointer advances to the next record and reads the member's address.

The PROC PRINT output for this data set illustrates the problem.

Obs	Fname	Lname	Address	City	State	Zip
1	LEE	ATHNOS	1215 RAINTREE CIRCLE	PHOENIX	AZ	85044
2	HEIDIE	BAKER	1751 DIEHL ROAD	MYRON BARK	131	DONERAIL
3	ATLANTA	GA	JOYCE BENEFIT	85 MAPLE A	MENLO	PARK

Before you write the INPUT statement, check whether the raw data file contains the same number of records for each observation. In this raw data file there are now three records for each observation.

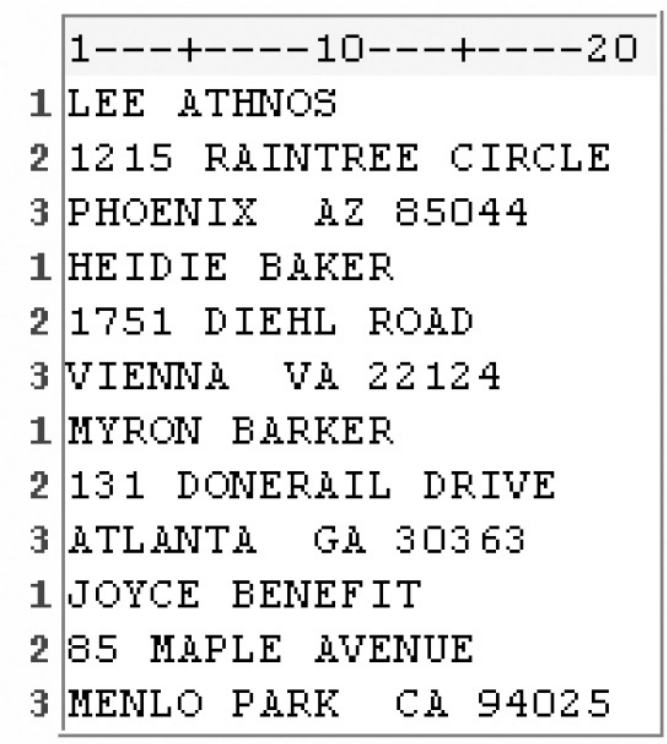

Figure 19.19: Verifying the Number of Records for Each Observation

Additional Note For more information about working with raw data files that contain missing records, see the SAS documentation.

Reading Multiple Records Non-Sequentially

The #n Line Pointer Control

You already know how to read multiple records sequentially by using the / line pointer control. Now let's look at reading multiple records non-sequentially by using the *#n* line pointer control.

The #n specifies the absolute number of the line to which you want to move the input pointer. The #n pointer control can read records in any order; therefore, it must be specified *before* the instructions for reading values in a specific record.

The INPUT statement below first reads the values for Department and JobCode in the second record, then the values for Lname and Fname in the first record. Finally, it reads the value for Salary in the third record.

```
input #2 Department $ 1-12 JobCode $ 15-19
      #1 Lname $ Fname $
      #3 Salary comma10.;
```

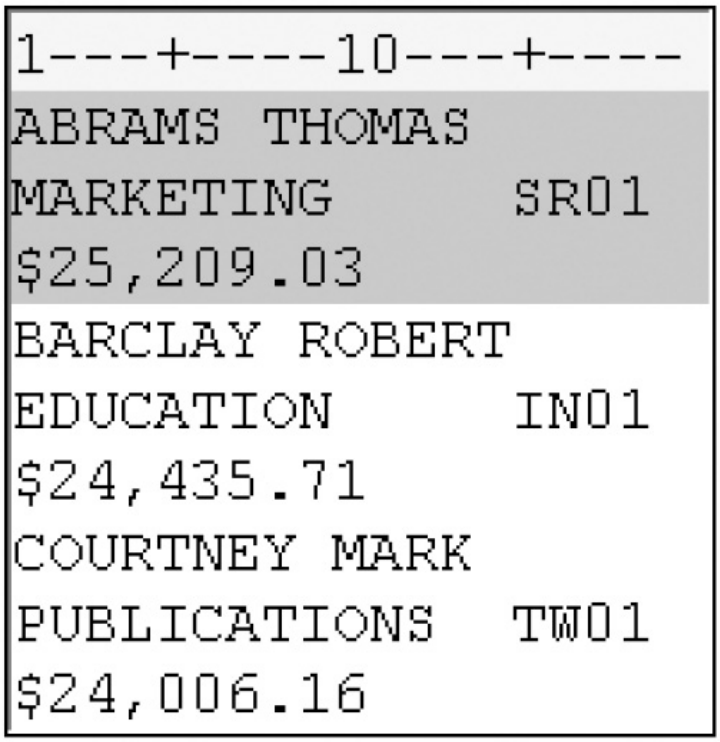

Figure 19.20: The Records of the First Observation

Using the #n Line Pointer Control

Look at the #n line pointer control in the following example.

The raw data file Patdata contains information about the patients of a small group of general surgeons.

The first three records contain a patient's name, address, city, state, and zip code. The fourth record contains the patient's ID number followed by the name of the primary physician.

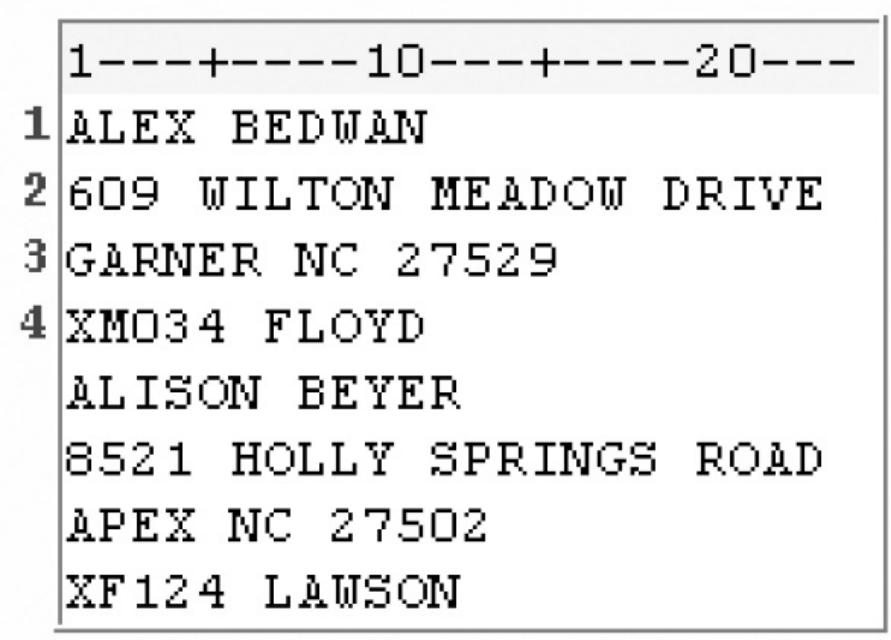

Figure 19.21: Observation Records in a Raw Data File

Suppose you want to read each patient's information in the following order:

1. ID number (ID)
2. first name (Fname)
3. last name (Lname)
4. address (Address)
5. city (City)
6. state (State)
7. zip (Zip)
8. doctor (Doctor)

- To read the values for ID in the fourth record, specify #4 before naming the variable and defining its attributes.

```
data perm.patients;
   infile patdata;
   input #4 ID $5.
```

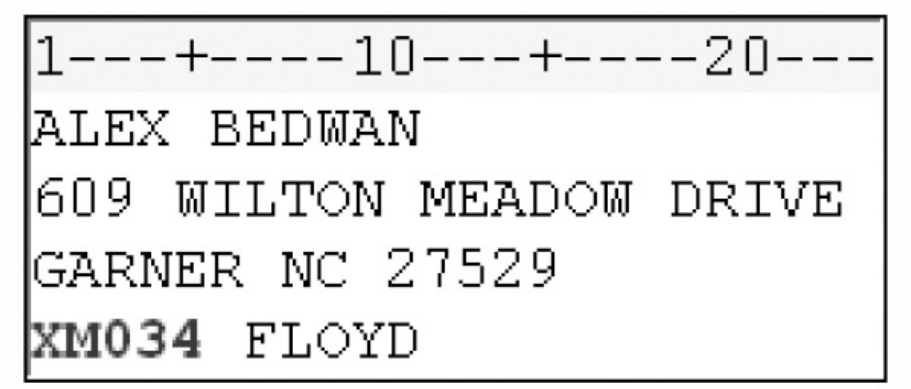

Figure 19.22: Specifying the ID Value in the Fourth Record of an Observation

- To read the values for Fname and Lname in the first record, specify #1 before naming the variables and defining their attributes.

```
data perm.patients;
   infile patdata;
   input #4 ID $5.
         #1 Fname $ Lname $
```

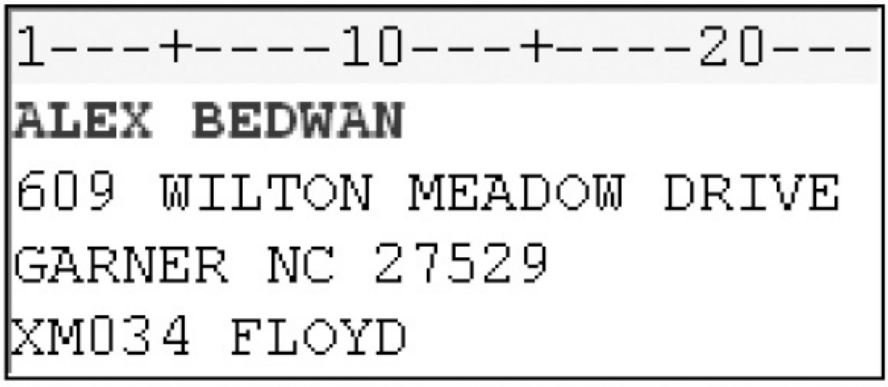

Figure 19.23: Specifying the First Record of an Observation

- Use the #n line pointer control to move the input pointer to the second record and read the value for Address.

```
data perm.patients;
```

```
infile patdata;
input #4 ID $5.
      #1 Fname $ Lname $
      #2 Address $23.
```

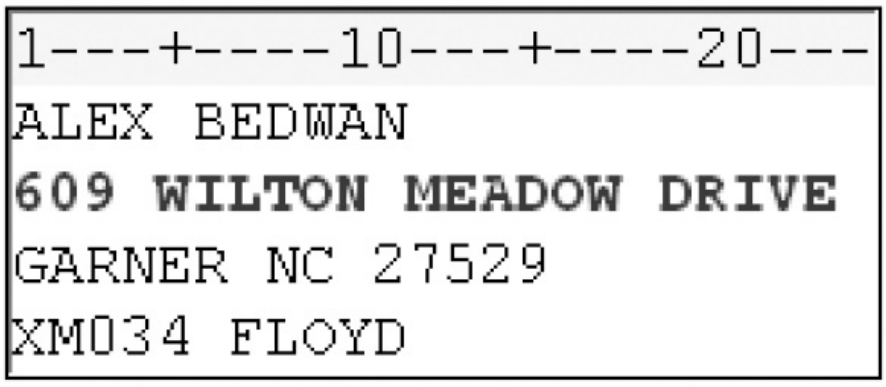

Figure 19.24: Specifying the Second Record of an Observation

- Now move the input pointer to the third record and read the values for City, State, and Zip, in that order.

  Additional Note In this raw data file, the values for City contain eight characters or less and do not contain embedded blanks, so you can use standard list input to read these values.

```
data perm.patients;
   infile patdata;
   input #4 ID $5.
         #1 Fname $ Lname $
         #2 Address $23.
         #3 City $ State $ Zip $
```

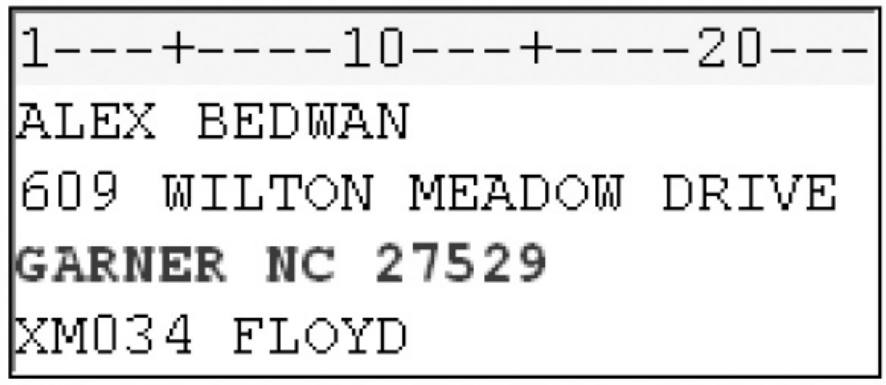

Figure 19.25: Specifying the Third Record of an Observation

- Now you need to move the input pointer down to the fourth record to read the values for Doctor, which begin in column 7. Don't forget to add a semicolon at the end of the INPUT statement. A RUN statement completes the program.

```
data perm.patients;
   infile patdata;
   input #4 ID $5.
         #1 Fname $ Lname $
         #2 Address $23.
         #3 City $ State $ Zip $
         #4 @7 Doctor $6.;
run;
```

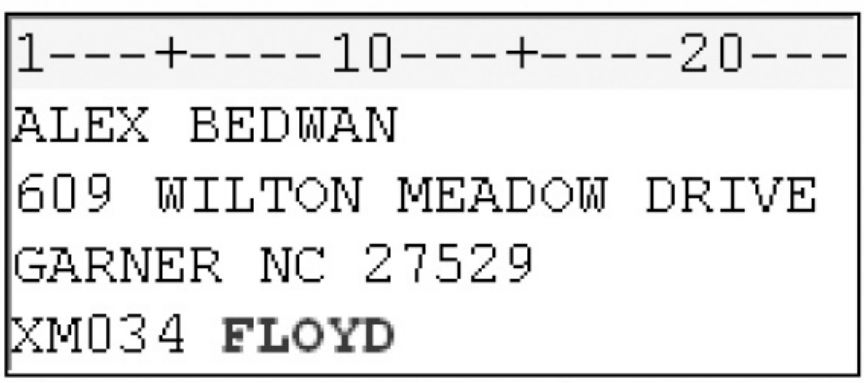

Figure 19.26: Specifying the Doctor Value in the Fourth Record of an Observation

Execution of the DATA Step

The *#n* pointer controls in the program below cause four records to be read for each execution of the DATA step.

```
data perm.patients;
   infile patdata;
   input #4 ID $5.
         #1 Fname $ Lname $
         #2 Address $23.
         #3 City $ State $ Zip $
         #4 @7 Doctor $6.;
run;
```

The first time the DATA step executes, the first four records are read, and an observation is written to the data set.

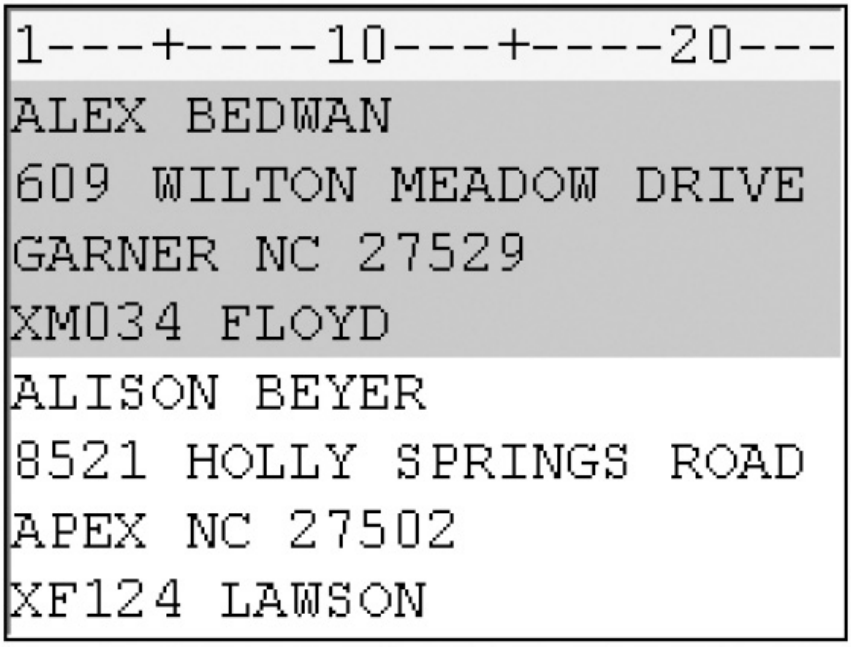

Figure 19.27: Raw Data File with the First Four Records Highlighted

During the second iteration, the next four records are read, and the second observation is written to the data set, and so on.

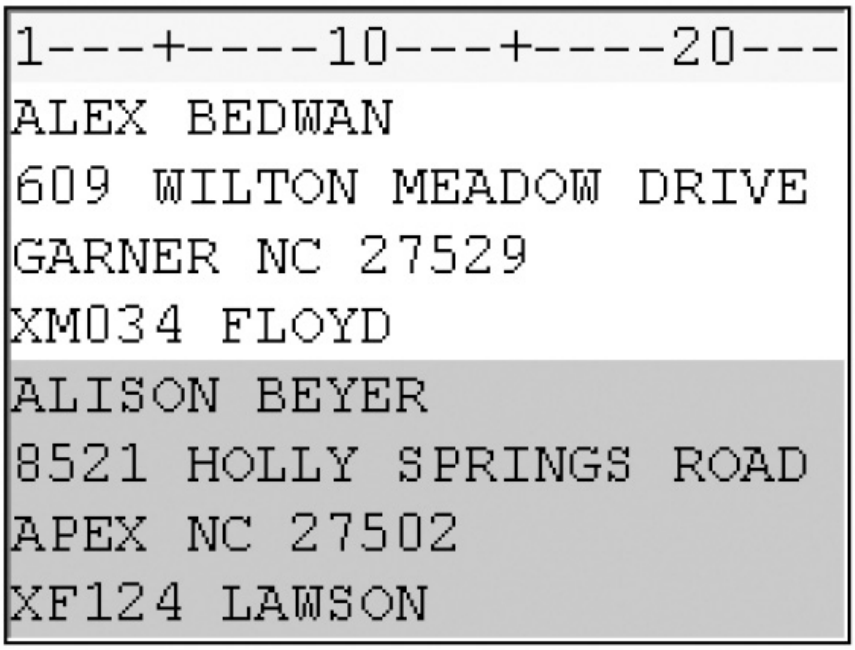

Figure 19.28: Raw Data File with the Next Four Records Highlighted

The PROC PRINT output of the data set shows how information that was spread over several records has been condensed into one observation.

```
proc print data=perm.patients noobs;
run;
```

ID	Fname	Lname	Address	City	State	Zip	Doctor
XM034	ALEX	BEDWAN	609 WILTON MEADOW DRIVE	GARNER	NC	27529	FLOYD
XF124	ALISON	BEYER	8521 HOLLY SPRINGS ROAD	APEX	NC	27502	LAWSON
XF232	LISA	BONNER	109 BRAMPTON AVENUE	CARY	NC	27511	LAWSON
XM065	GEORGE	CHESSON	3801 WOODSIDE COURT	GARNER	NC	27529	FLOYD

Figure 19.29: PROC PRINT Output of Data Set Perm.Patients

Combining Line Pointer Controls

The forward slash (/) line pointer control and the #*n* line pointer control can be used together in a SAS program to read multiple records both sequentially and non-sequentially.

For example, you could use both the / line pointer control and the #n line pointer control to read the variables in the raw data file Patdata in the following order:

---

1. ID
2. Fname
3. Lname
4. Address
5. City
6. State
7. Zip

8. Doctor

```
data perm.patients;
   infile patdata;
   input #4 ID $5.
         #1 Fname $ Lname $ /
            Address $23. /
            City $ State $ Zip $ /
            @7 Doctor $6.;
run;
```

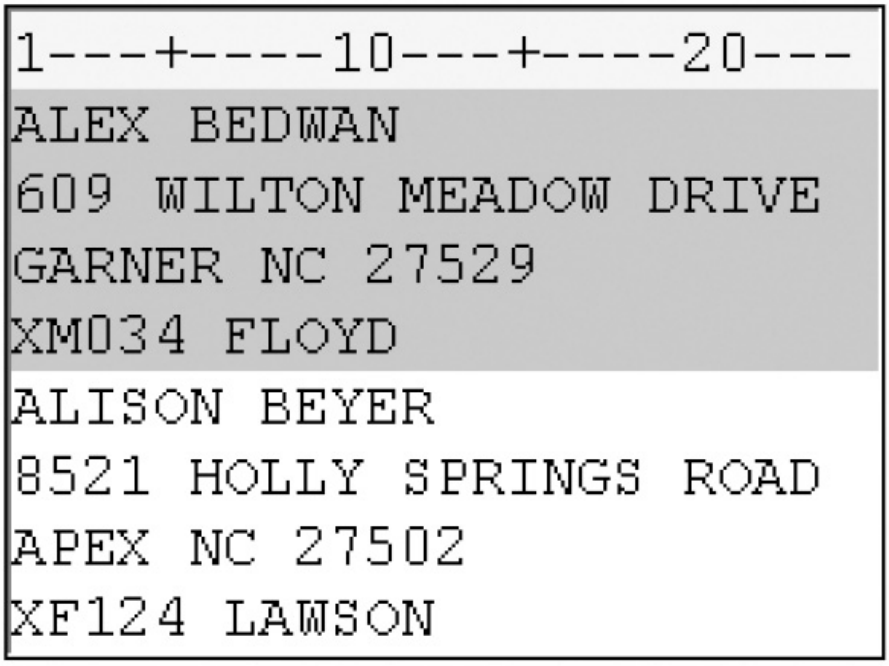

Figure 19.30: Raw Data File with the First Four Records Highlighted

- To read the values for ID in the fourth record, specify #4 before naming the variable and defining its attributes.
- Specify #1 to move the input pointer back to the first record, where the values for Fname and Lname are read.
- Because the next record to be read is sequential, you can use the / line pointer control after the variable Lname to move the input pointer to the second record, where the value for Address is read.
- The / line pointer control in the next line directs the input pointer to the third record, where the values for City, State, and Zip are read.
- The final / line pointer control moves the input pointer back to the fourth record, where the value for Doctor is read.

Additional Note Alternatively, you can use just the #n line pointer control (as shown earlier in this chapter and below) to read the variables in the order shown above.

```
infile patdata;
input #4 ID $5.
      #1 Fname $ Lname $
      #2 Address $23.
      #3 City $ State $ Zip $
      #4 @7 Doctor $6.;
run;
```

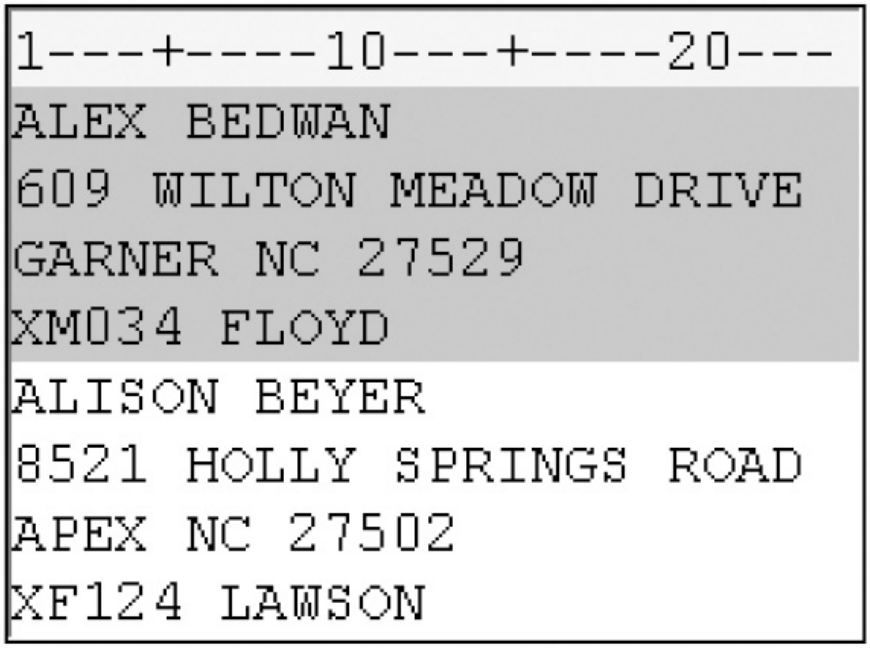

Figure 19.31: Raw Data File with the First Four Records Highlighted

Chapter Summary

Text Summary

Multiple Records per Observation

Information for one observation can be spread out over several records. You can write one INPUT statement that contains line pointer controls to specify the record(s) from which values are read.

Reading Multiple Records Sequentially

The forward slash (/) line pointer control is used to read multiple records sequentially. Each time a / pointer is encountered, the input pointer advances to the next line.

Reading Multiple Records Non-Sequentially

The #n line pointer control is used to read multiple records non-sequentially. The #n specifies the absolute number of the line to which you want to move the pointer.

Combining Line Pointer Controls

The / line pointer control and the #n line pointer control can be combined within a SAS program to read multiple records both sequentially and non-sequentially.

Syntax

```
LIBNAME libref 'SAS-data-library';
FILENAME fileref 'filename';
DATA SAS-data-set;
     INFILE file-specification;
     INPUT #n variable …;
        #n @n variable … / variable … / variable …;
RUN;
PROC PRINT DATA=SAS-data-set;
RUN;
```

Sample Program

```
libname perm 'c:\records\empdata';
filename personel 'c:\records\empdata\new.dat';
data perm.emplist3;
```

```
   infile personel;
   input #2 Department $ 5-16
         #1 @16 ID $4. @1 Name $14. /
            JobCode 3. /
            Salary comma9.;
run;
proc print data=perm.emplist3;
run;
```

Points to Remember

- When a file contains multiple records per observation, depending on the program, the file might need to contain the same number of records for each observation.
- Because the / pointer control can move forward only, the pointer control is specified *after* the values in the current record are read.
- The #n pointer control can read records in any order and must be specified *before* the variable names are defined.
- A semicolon should be placed at the end of the *complete* INPUT statement.

Chapter Quiz

1. Select the best answer for each question. After completing the quiz, check your answers using the answer key in the appendix. ?

You can position the input pointer on a specific record by using

a. column pointer controls.

b. column specifications.

c. line pointer controls.

d. line hold specifiers.

2. Which pointer control is used to read multiple records sequentially? ?

a. *@n*

b. *+n*

c. /

d. all of the above

3. Which pointer control can be used to read records non-sequentially? ?

a. *@n*

b. *#n*

c. *+n*

d. /

4. Which SAS statement correctly reads the values for Fname, Lname, Address, City, State and Zip in order? ?

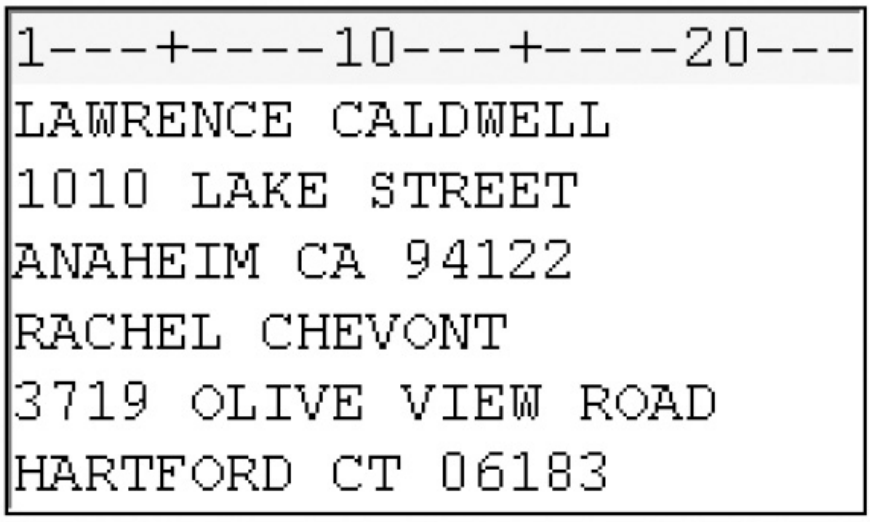

Figure 19.32: Raw Data File

a.
```
input Fname $ Lname $ /
      Address $2 0. /
      City $ State $ Zip $;
```

b.
```
input Fname $ Lname $ /;
      Address $2 0. /;
      City $ State $ Zip $;
```

c.
```
input / Fname $ Lname $
      / Address $2 0.
      City $ State $ Zip $;
```

d.
```
input / Fname $ Lname $;
      / Address $2 0.;
      City $ State $ Zip $;
```

5. Which INPUT statement correctly reads the values for ID in the fourth record, and then returns to the first record to read the values for Fname and Lname? ?

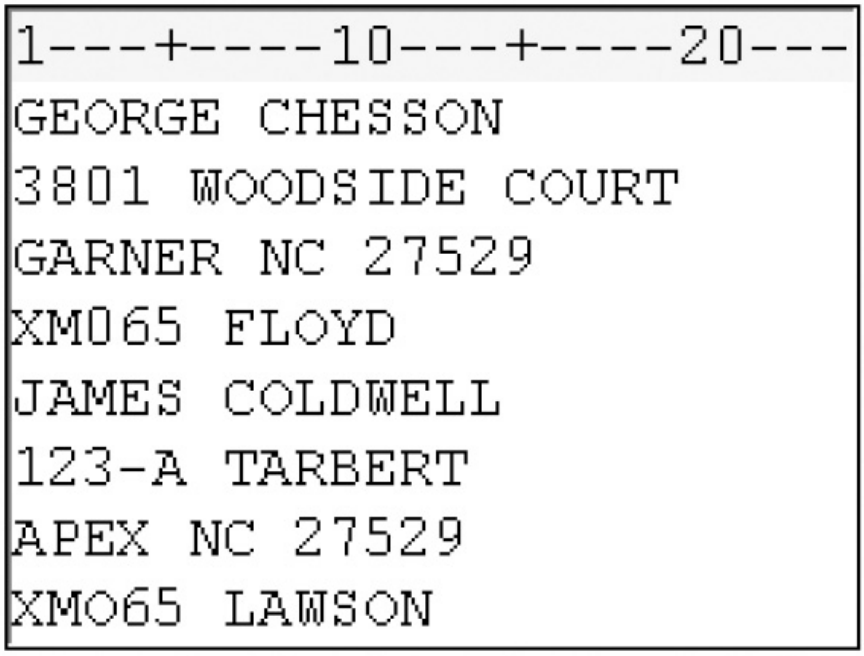

Figure 19.33: Raw Data File

a.
```
input #4 ID $5.
      #1 Fname $ Lname $;
```

b.
```
input #4 ID $ 1-5
      #1 Fname $ Lname $;
```

c.
```
input #4 ID $
      #1 Fname $ Lname $;
```

d. all of the above

6. How many records will be read for each execution of the DATA step? ?

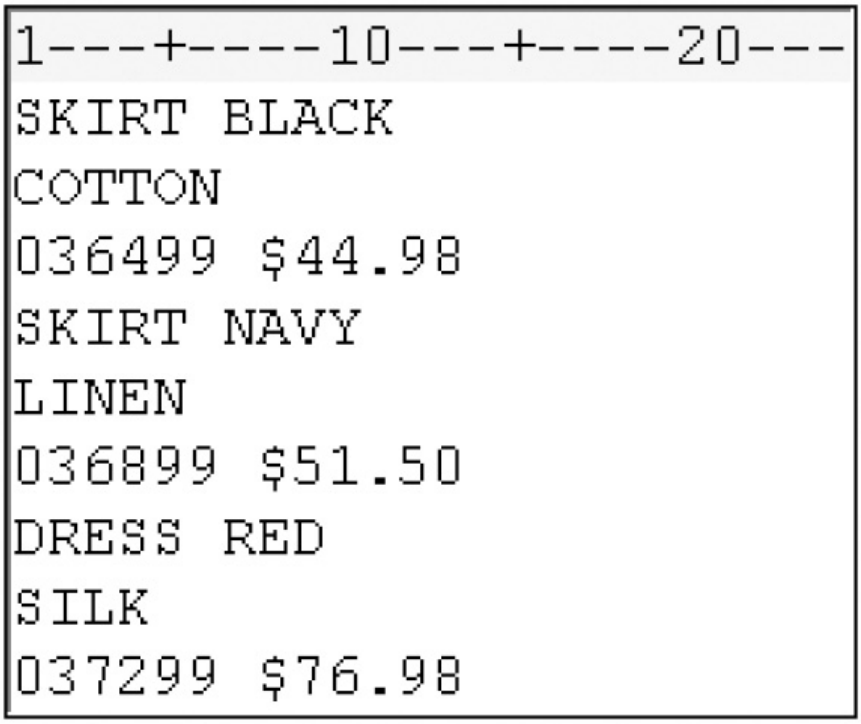

Figure 19.34: Raw Data File

```
data spring.sportswr;
   infile newitems;
   input #1 Item $ Color $
         #3 @8 Price comma6.
         #2 Fabric $
         #3 SKU $ 1-6;
run;
```

a. one

b. two

c. three

d. four

7. Which INPUT statement correctly reads the values for City, State, and Zip? ?

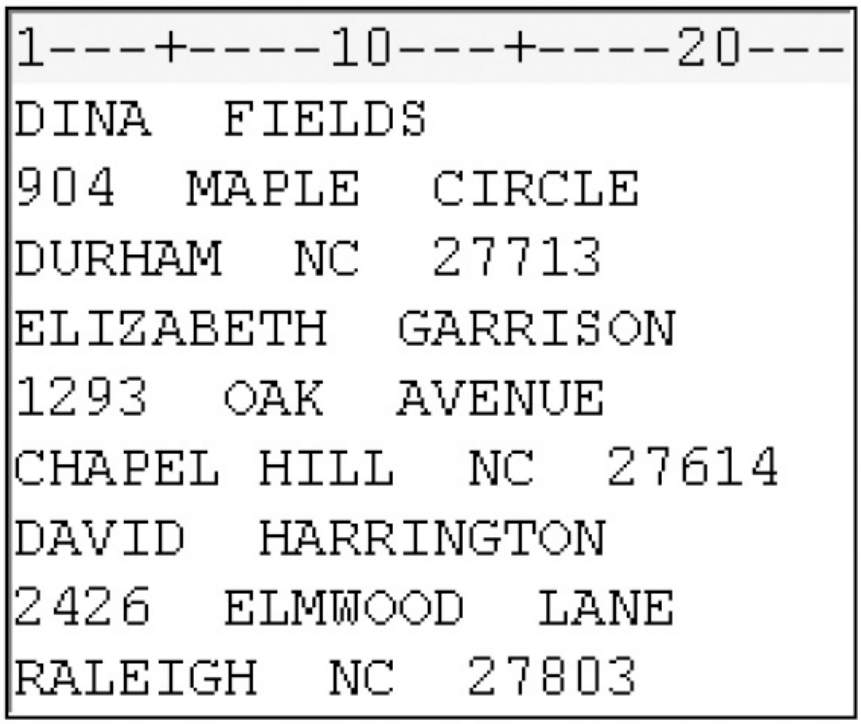

Figure 19.35: Raw Data File

a. `input #3 City $ State $ Zip $;`

b. `input #3 City & $11. State $ Zip $;`

c. `input #3 City $11. +2 State $2. + 2 Zip $5.;`

d. all of the above

8. Which program does *not* read the values in the first record as a variable named Item and the values in the second record as two variables named Inventory and Type? ?

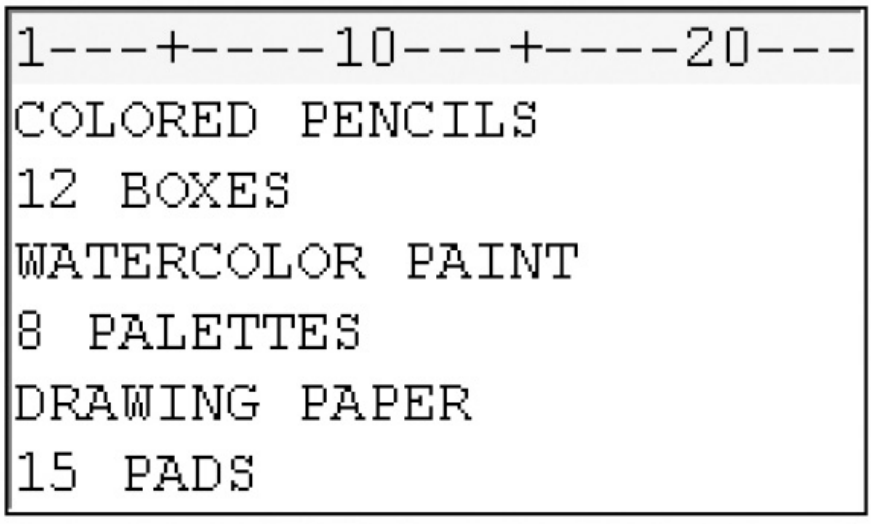

Figure 19.36: Raw Data File

a.
```
data perm.supplies;
     infile instock pad;
     input Item & $16. /
           Inventory 2. Type $8.;
```

```
   run;
```

b.
```
data perm.supplies;
   infile instock pad;
   input Item & $16.
         / Inventory 2. Type $8.;
 run;
```

c.
```
data perm.supplies;
   infile instock pad;
   input #1 Item & $16.
         Inventory 2. Type $8.;
run;
```

d.
```
data perm.supplies;
   infile instock pad;
   input Item & $16.
         #2 Inventory 2. Type $8.;
run;
```

9. Which INPUT statement reads the values for Lname, Fname, Department, and Salary (in that order)? ?

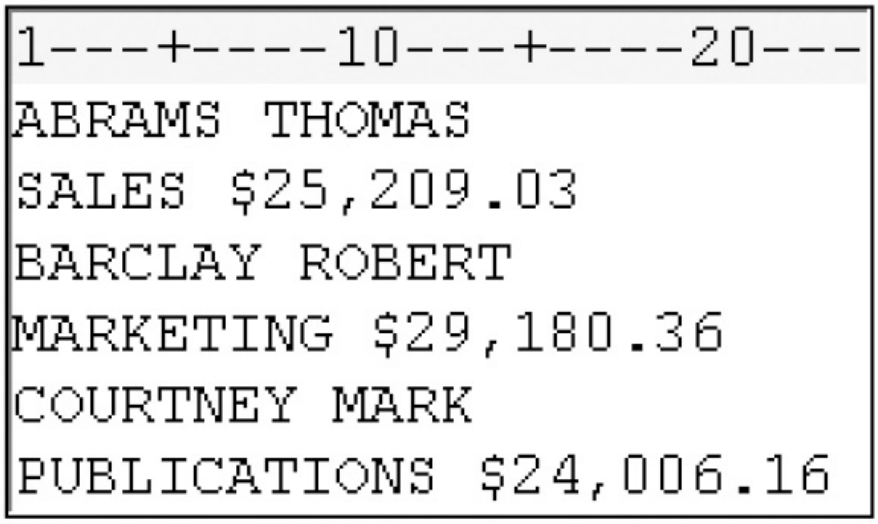

Figure 19.37: Raw Data File

a.
```
input #1 Lname $ Fname $ /
       Department $12. Salary comma10.;
```

b.
```
input #1 Lname $ Fname $ /
       Department : $12. Salary : comma.;
```

c.
```
input #1 Lname $ Fname $
      #2 Department : $12. Salary : comma.;
```

d. both b and c

10. Which raw data file poses potential problems when you are reading multiple records for each observation? ?

a.

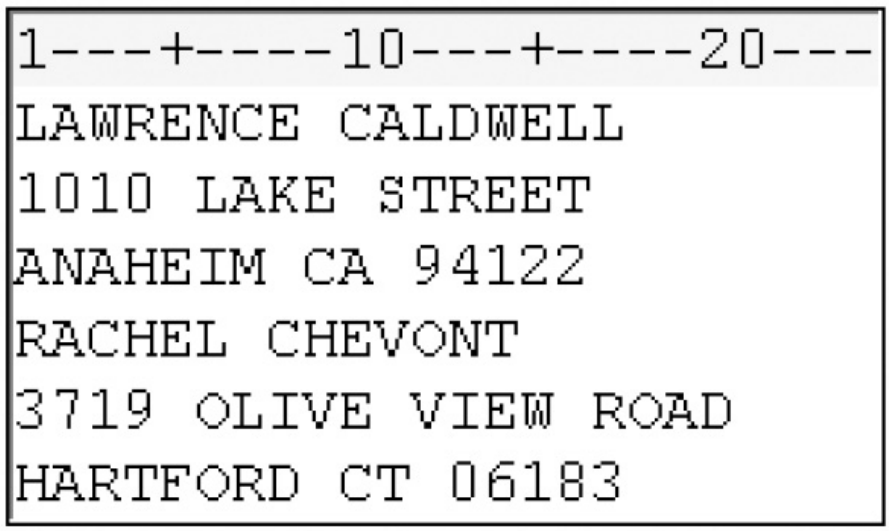

Figure 19.38: Raw Data File

b.

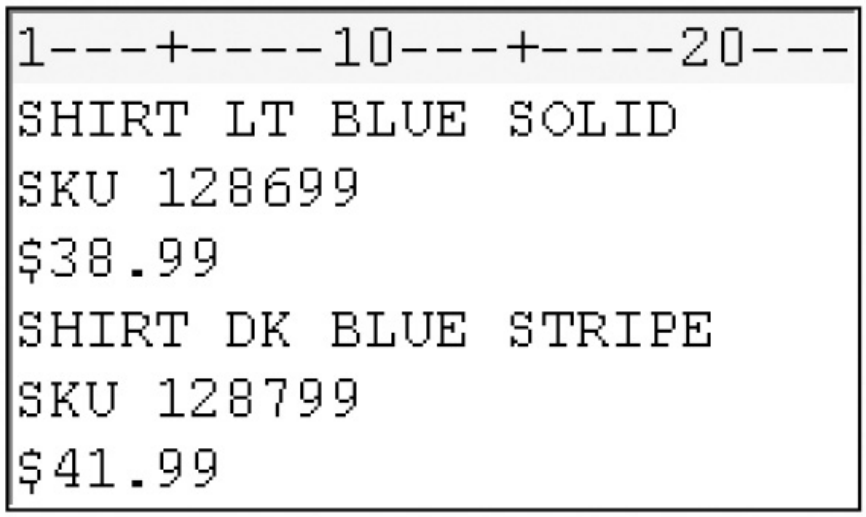

Figure 19.39: Raw Data File

c.

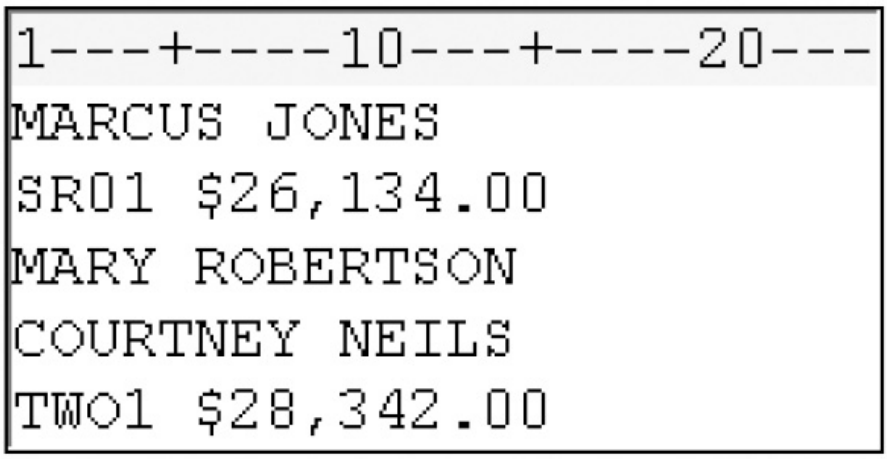

Figure 19.40: Raw Data File

d.

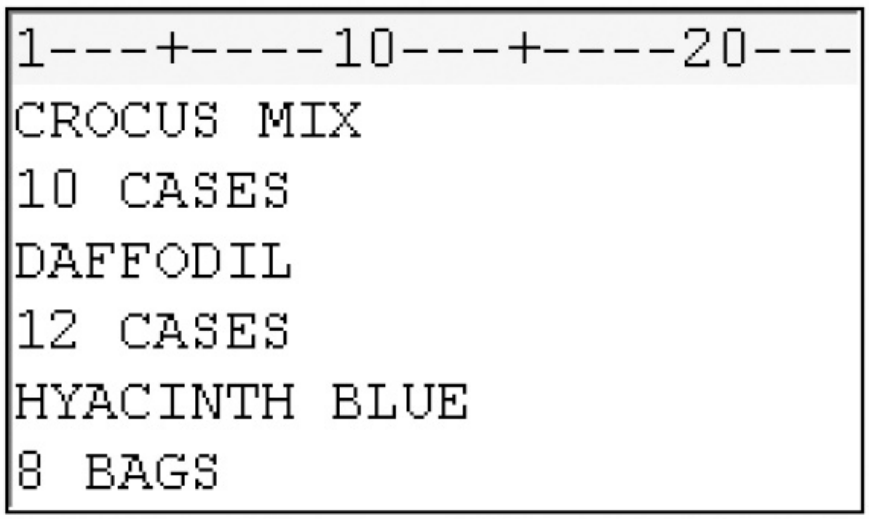

Figure 19.41: Raw Data File

Answers

1. Correct answer: c

   Information for one observation can be spread out over several records. You can write one INPUT statement that contains line pointer controls to specify the record(s) from which values are read.

2. Correct answer: c

   The forward slash (/) line pointer control is used to read multiple records sequentially. Each time a / pointer is encountered, the input pointer advances to the next line. @n and +n are column pointer controls.

3. Correct answer: b

   The #n line pointer control is used to read records non-sequentially. The #n specifies the absolute number of the line to which you want to move the pointer.

4. Correct answer: a

   The INPUT statement uses the / line pointer control to move the input pointer forward from the first record to the second record, and from the second record to the third record. The / line pointer control only moves the input pointer forward and must be specified after the instructions for reading the values in the current record. You should place a semicolon only at the end of a complete INPUT statement.

5. Correct answer: d

   The first #n line pointer control enables you to read the values for ID from the fourth record. The second #n line pointer control moves back to the first record and reads the values for Fname and Lname. You can use formatted input, column input, or list input to read the values for ID.

6. Correct answer: c

   The first time the DATA step executes, the first three records are read, and an observation is written to the data set. During the second iteration, the next three records are read, and the second observation is written to the data set. During the third iteration, the last three records are read, and the final observation is written to the data set.

7. Correct answer: b

A combination of modified and simple list input can be used to read the values for City, State, and Zip. You need to use modified list input to read the values for City, because one of the values is longer than eight characters and contains an embedded blank. You cannot use formatted input, because the values do not begin and end in the same column in each record.

8. Correct answer: c

The values for Item in the first record are read, then the following / or #n line pointer control advances the input pointer to the second record, to read the values for Inventory and Type.

9. Correct answer: d

You can use either the / or #n line pointer control to advance the input pointer to the second line, in order to read the values for Department and Salary. The colon (:) modifier is used to read the character values that are longer than eight characters (Department) and the nonstandard data values (Salary).

10. Correct answer: c

The third raw data file does not contain the same number of records for each observation, so the output from this data set will show invalid data for the ID and salary information in the fourth line.