Chapter 6: Reading raw data

<SAS Certification Prep Guide - Base Programming for SAS 9>

Objectives : learn to

- read standard fixed-field data
- read nonstandard fixed-field data.
- read free-format data (data that is not organized in fixed fields)
- free-format data that is separated by nonblank delimiters, such as commas

Review of Column Input

you've learned how to use column input to
read raw data that is stored in an external file. You can use column input to read the values for Item, IDnum, InStock, and BackOrd from
the following raw data file.

```
input Item $ 1-13 IDnum $ 15-19 InStock 21-22 BackOrd 24-25;
```

Raw Data File Invent

```
1---+----10---+----20---+--
BIRD FEEDER    LG088  3 20
GLASS MUGS     SB082  6 12
GLASS TRAY     BQ049 12  6
PADDED HANGRS  MN256 15 20
JEWELRY BOX    AJ498 23  0
RED APRON      AQ072  9 12
CRYSTAL VASE   AQ672 27  0
PICNIC BASKET  LS930 21  0
```

■ Column Input Features

Column input has several features that make it useful for reading raw data.

It can be used to read character variable values that contain embedded blanks.

```
input Name $ 1-25;
```

```
1---+----10---+----20---+-
JOSEPH PAUL THACKERY JR.
```

No placeholder is required for missing data. A blank field is read as missing and does not
cause other fields to be read incorrectly.

```
input Item $ 1-13 IDnum $ 15-19 nstock 21-22 Backord 24-25;
```

```
1---+----10---+----20---+-
BIRD FEEDER    LG088  3 20
6 GLASS MUGS   SB082    12
GLASS TRAY     BQ049 12  6
```

Fields or parts of fields can be re-read.

```
input Item $ 1-13 IDnum $ 15-19 Supplier $ 15-16 InStock 21-22 BackOrd 24-25;
```

```
1---+----10---+----20---+--
PADDED HANGRS  MN256 15 20
JEWELRY BOX    AJ498 23  0
RED APRON      AQ072  9 12
```

Fields do not have to be separated by blanks or other delimiters.

```
input Item $ 1-13 IDnum $ 14-18 InStock 19-20 BackOrd 21-22;
```

```
1---+----10---+----20---+--
PADDED HANGRSMN2561520
JEWELRY BOX  AJ49823 0
RED APRON    AQ072 912
```

The following file contains commas (Salary). So, the values for Salary are considered to be nonstandard numeric values.

Raw Data File Empdata

```
1---+----10---+----20---+---
EVANS    DONNY 112 29,996.63
HELMS    LISA  105 18,567.23
HIGGINS  JOHN  111 25,309.00
LARSON   AMY   113 32,696.78
MOORE    MARY  112 28,945.89
POWELL   JASON 103 35,099.50
RILEY    JUDY  111 25,309.00
RYAN     NEAL  112 28,180.00
WILSON   HENRY 113 31,875.46
WOODS    CHIP  105 17,098.71
```

Using Formatted Input

Whenever you encounter raw data that is organized into fixed fields, you can use

- column input to read standard data only
- formatted input to read both standard and nonstandard data.

Formatted input is a very powerful method for reading both standard and nonstandard data in fixed fields.

```
INPUT <pointer-control> variable informat.;
```

where
- pointer-control positions the input pointer on a specified column
- variable is the name of the variable that is being created
- informat is the special instruction that specifies how SAS reads raw data.

you will learn to work with two column pointer controls.
- The @n moves the input pointer to a specific column number.
- The +n moves the input pointer forward to a column number that is relative to the current position.

■ Using the @n Column Pointer Control

Let's use the @n pointer control to locate variable values in the external file Empdata. As you can see, the values for LastName begin in column 1. We could start with the @1 pointer control. Next, the values for FirstName begin in column 9. To point to column 9, use an @ sign and the column number in the INPUT statement:

```
input @1 LastName $7.@9 FirstName $5.
```

However, the default column pointer location is column 1, so you do not need to use a column pointer control to read the first field.

```
input LastName $7. @9 FirstName $5.
```

```
↓---+----10---+----20---+---
EVANS   DONNY 112 29,996.63
HELMS   LISA  105 18,567.23
HIGGINS JOHN  111 25,309.00
LARSON  AMY   113 32,696.78
MOORE   MARY  112 28,945.89
POWELL  JASON 103 35,099.50
RILEY   JUDY  111 25,309.00
```

Column pointer controls are very useful. For example, you can use the @n to move a pointer forward or backward when reading a record.

```
input @9 FirstName $5. @1 LastName $7. @15 JobTitle 3. @19 Salary comma9.;
```

■ The +n Pointer Control

The +n pointer control moves the input pointer forward to a column number that is relative to the current position. The + moves the pointer forward n

columns.

```
input LastName $7. +1 FirstName $5. +5 Salary comma9. @15 JobTitle 3.;
```

```
1---+---↓-10---+----20---+---
EVANS   DONNY 112 29,996.63
HELMS   LISA  105 18,567.23
HIGGINS JOHN  111 25,309.00
```

You can use the notation +(-n) to move the +n pointer control backward.

Using Informats

An informat is an instruction that tells SAS how to read raw data. SAS provides many informats for reading standard and nonstandard data values.

See
http://support.sas.com/documentation/cdl/en/leforinforref/63324/HTML/default/viewer.htm#p13eggx7qc7l6in1oprinvadzubl.htm

■ Reading Character Values

The $w. informat enables you to read character data. The w represents the field width of the data value (the total number of columns that contain the raw data field).

In the example below, the $ indicates that FirstName is a character variable, the 5 indicates a field width of five columns, and a period ends the informat.

```
input @9 FirstName $5.;
```

```
          12345
1---+---↓10---+----20---+---
EVANS   DONNY 112 29,996.63
HELMS   LISA  105 18,567.23
HIGGINS JOHN  111 25,309.00
LARSON  AMY   113 32,696.78
MOORE   MARY  112 28,945.89
POWELL  JASON 103 35,099.50
RILEY   JUDY  111 25,309.00
RYAN    NEAL  112 28,180.00
WILSON  HENRY 113 31,875.46
WOODS   CHIP  105 17,098.71
```

■ Reading Standard Numeric Data

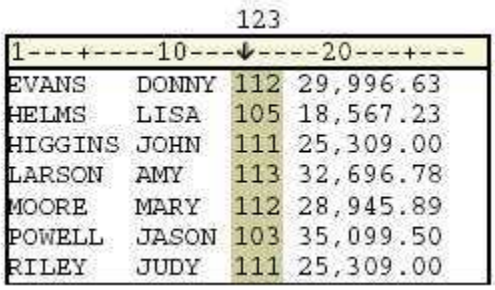
The informat for reading standard numeric data is the w.d informat.

The w specifies the field width of the raw data value, the period serves as a

delimiter, and the d optionally specifies the number of implied decimal places for the value. The w.d informat ignores any specified d value if the data already contains a decimal point.

In the example that is shown below, the values for JobTitle in columns 15-17 contain only numbers. Remember that standard numeric data values can contain only numbers, decimal points, scientific notation, and plus and minus signs. A d value is not necessary to read the values for JobTitle. Simply move the column pointer control forward 7 spaces to column 15, name the variable, and specify a field width of 3.

```
input @9 FirstName $5. @1 LastName $7. +7 JobTitle 3.;
```



Remember to specify the period in the informat name. For example, if you omit the period in the following INPUT statement, SAS assigns a length of 3 to JobTitle instead of reading JobTitle with the 3. informat.

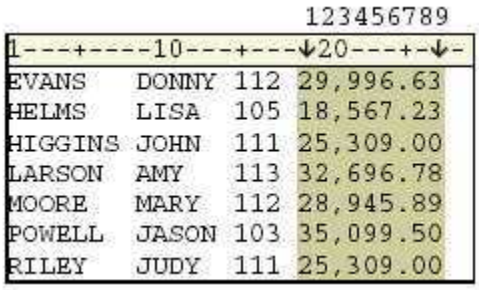
■ Reading Nonstandard Numeric Data

The COMMAw.d informat is used to read numeric values and to remove embedded

- blanks
- commas
- dashes
- dollar signs
- percent signs
- right parentheses
- left parentheses, which are converted to minus signs.

In the example below, the values for Salary contain commas, which means that they are nonstandard numeric values. The values for Salary begin in column 19, so use the @n or +n pointer control to point to column 19, and then name the variable. Now add the COMMAw.d informat and specify the field width. The values end in column 27, so the field width is 9 columns. Add a RUN statement to complete the DATA step.

```
data empinfo;
        infile empdata;
        input @9 FirstName $5. @1 LastName $7. +7 JobTitle 3.
        @19 Salary comma9.;
run;
proc print data=empinfo;
run;
```



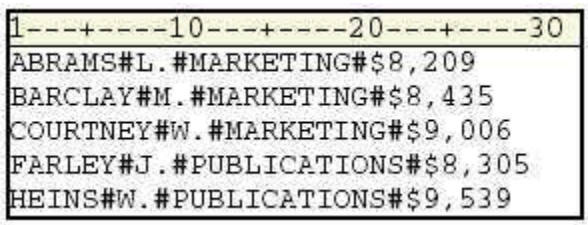
```
                          123456789
1---+----10---+---↓20---+-↓-
EVANS    DONNY 112 29,996.63
HELMS    LISA  105 18,567.23
HIGGINS  JOHN  111 25,309.00
LARSON   AMY   113 32,696.78
MOORE    MARY  112 28,945.89
POWELL   JASON 103 35,099.50
RILEY    JUDY  111 25,309.00
```

The COMMAw.d informat does more than simply read the raw data values. It removes special characters such as commas from numeric data and stores only numeric values in a SAS data set.

Reading Free-Format Data

You have already worked with raw data that is in fixed fields. In doing so, you used column input to read standard data values in fixed fields. You have also used formatted input to read both standard and nonstandard data in fixed fields.

Suppose you have raw data that is free format; that is, it is not arranged in fixed fields. The fields are often separated by blanks or by some other delimiter, such as the pound sign (#) shown below. In this case, column input and formatted input that you might have used before to read standard and nonstandard data in fixed fields will not enable you to read all of the values in the raw data file.



```
1---+----10---+----20---+----30
ABRAMS#L.#MARKETING#$8,209
BARCLAY#M.#MARKETING#$8,435
COURTNEY#W.#MARKETING#$9,006
FARLEY#J.#PUBLICATIONS#$8,305
HEINS#W.#PUBLICATIONS#$9,539
```

Using List Input

■ Group Processing Using the CLASS Statement

Suppose you have an external data file like the one that follows. The file, which is referenced by the fileref Credit, contains the results of a survey on the use of

credit cards by males and females in the 18-39 age range.

Raw Data File Credit

```
1---+----10---+----20
MALE 27 1 8 0 0
FEMALE 29 3 14 5 10
FEMALE 34 2 10 3 3
MALE 35 2 12 4 8
FEMALE 36 4 16 3 7
MALE 21 1 5 0 0
MALE 25 2 9 2 1
FEMALE 21 1 4 2 6
MALE 38 3 11 4 3
FEMALE 30 3 5 1 0
```

You need to read the data values for
- gender
- age
- number of bank credit cards
- bank card use per month
- number of department store credit cards
- department store card use per month.

List input might be the easiest input style to use because, you simply list the variable names in the same order as the corresponding raw data fields. Remember to distinguish character variables from numeric variables.

Because list input, by default, does not specify column locations,
- all fields must be separated by at least one blank or other delimiter
- fields must be read in order from left to right
- you cannot skip or re-read fields.

■ Processing List Input

It's important to remember that list input causes SAS to scan the input lines for values rather than reading from specific columns. When the INPUT statement is submitted for processing, the input pointer is positioned at column 1 of the raw data file, as shown below.

```
data survey;
        infile credit;
        input Gender $ Age Bankcard FreqBank Deptcard FreqDept;
run;
```

■ **Working with Delimiters**

Most free-format data fields are clearly separated by blanks and are easy to imagine as variables and observations. But fields can also be separated by other delimiters, such as commas, as shown below.

Raw Data File Credit

```
1---+----10---+----20
MALE,27,1,8,0,0
FEMALE,29,3,14,5,10
FEMALE,34,2,10,3,3
MALE,35,2,12,4,8
FEMALE,36,4,16,3 7
MALE,21,1,5,0,0
MALE,25,2,9,2,1
FEMALE,21,1,4,2 6
MALE,38,3,11,4,3
FEMALE,30,3,5,1,0
```

When characters other than blanks are used to separate the data values, you can tell SAS which field delimiter to use. Use the DLM= option in the INFILE statement to specify a delimiter other than a blank (the default).

```
data survey;
        infile credit dlm=',';
        input Gender $ Age Bankcard FreqBank Deptcard FreqDept;
run;
```

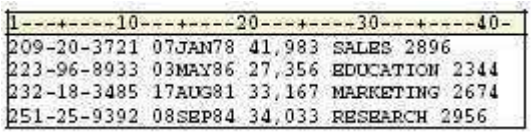
■ **Limitations of List Input**

In its default form, list input places several limitations on the types of data that can be read:

- Although the width of a field can be greater than eight columns, both character and numeric variables have a default length of 8. Character values that are longer than eight characters will be truncated.
- Data must be in standard numeric or character format.
- Character values cannot contain embedded delimiters.
- Missing numeric and character values must be represented by a period or some other character.

Mixing Input Styles

Look at the raw data file below and think about how to combine input styles to read these values.



- Column input is an appropriate choice for the first field because the values can be read as standard character values and are located in fixed columns.
- The next two fields are also located in fixed columns, but the values require an informat. So, formatted input is a good choice here.
- Values in the fourth field begin in column 28 but do not end in the same column. List input is appropriate here, but notice that some values are longer than eight characters. You need to use the : format modifier with an informat to read these values.
- The last field does not always begin or end in the same column, so list input is the best input style for those values.

The INPUT statement to read the data should look like this:

```
data mixed;
        infile rawdata;
        input SSN $ 1-11 @13 HireDate date7. @21 Salary comma6.
        Department : $9. Phone;
run;
proc print data=mixed;
run;
```

cf) : is used to read character values that are longer than 8 characters.

sas-code

```
data empinfo;
        infile empdata;
        input @9 FirstName $5. @1 LastName $7. +7 JobTitle 3.
        @19 Salary comma9.;
run;
proc print data=empinfo;
run;

data survey;
        infile credit;
        input Gender $ Age Bankcard FreqBank Deptcard FreqDept;
run;

data survey;
        infile credit dlm=',';
        input Gender $ Age Bankcard FreqBank Deptcard FreqDept;
run;

data mixed;
        infile rawdata;
        input SSN $ 1-11 @13 HireDate date7. @21 Salary comma6.
        Department : $9. Phone;
run;
proc print data=mixed;
run;
```