The Power of SAS® Input Statements

Imelda C. Go, Richland County School District One, Columbia, SC

ABSTRACT

SAS can process data in a variety of forms. If the input data occurs in a predictable form or pattern governed by a set of rules, input statements can be written to process what would otherwise seem like irregular (and unusable) data. Examples include delimited data, text qualifiers, headers, multiple records per line, multiple lines per record, and conditional input statements.

INTRODUCTION

Input data can come from many sources and it is not always possible to control how input data are created. However, if there are rules that govern the structure of input data, these rules can be converted into input statements that read the data.

There is more than one way to write a functional input statement. Input statements can be written with multiple input styles present in the same statement. Important tools include column input, list input, formatted input, named input, informats, retain statements, conditional statements, column/line pointer controls, and line-hold specifiers.

COLUMN INPUT AND FORMATTED INPUT

The primary concern for column input is to know which columns correspond to a variable.

```sas
data stats1;
  input school $ 1-20 students 21-23 teachers 25-28;
cards;
  Arden Elementary    253 32
  Lyon Elementary     432 47
  Webber School       566 65;
PROC PRINT FOR DATA SET STATS1
OBS         SCHOOL         STUDENTS    TEACHERS
1     Arden Elementary       253         32
2     Lyon Elementary        432         47
3     Webber School          566         65
```

The previous input statement can be replaced by the following statement to produce the same result:

```sas
input school $20. students teachers;
cards;
  Arden Elementary  253 32
  Lyon Elementary  432 47
  Webber School  566 65
; PROC PRINT FOR DATA SET STATS1
OBS         SCHOOL         STUDENTS    TEACHERS
1                           253         32
2     Lyon Elementary       432         47
3     Webber School         566         65
```

Delimited Data

Use the INFILE statement’s DELIMITER= option to process data separated by commas. The ; argument below allows an informat to be used for reading the data. It says the variable school has character values and has a width of 20 characters.

```sas
data stats1;
  input school :$20. students teachers;
cards;
  Arden Elementary,253,32
  Lyon Elementary,432,47
  Webber School,566,65;
PROC PRINT FOR DATA SET ERROR
OBS    SCHOOL           STUDENTS    TEACHERS
1     253                  32          . _ERROR_=1 _N_=1
NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
NOTE: The data set WORK.ERROR has 2 observations and 3 variables.
```

LIST INPUT

List input involves reading data in the order in which they are listed. Data values are separated by at least one blank. For character values that have a blank as part of the value, the $ argument will allow a character value to have an embedded blank. For example, the $ argument below indicates the variable school may have embedded blanks and the value of school is read till two consecutive blanks occur. The variables students and teachers are read in list input style and their values are separated by a blank.

```sas
data stats1;
  input school :$20. students teachers;
cards;
  Arden Elementary 253 32
  Lyon Elementary,432,47
  Webber School 566 65;
PROC PRINT FOR DATA SET STATS1
OBS         SCHOOL         STUDENTS    TEACHERS
1     Arden Elementary       253         32
2     Lyon Elementary        432         47
3     Webber School          566         65
```

NOTE: Invalid data for TEACHERS in line 50 1-15.
RULE:----+----1----+----2----+----3----+----4----+----5----+----6
50   Lyon Elementary,432,47
SCHOOL=253 STUDENTS=32 TEACHERS=. _ERROR_=1 _N_=1
NOTE: SAS went to a new line when INPUT statement reached past the end of a line.

NOTE: The data set WORK.ERROR has 2 observations and 3 variables.
```

PROC PRINT FOR DATA SET ERROR
OBS    SCHOOL           STUDENTS    TEACHERS
1     253                  32          . _ERROR_=1 _N_=1
2     Webber School       566         65
```

Use the INFILE statement’s DSD option to correct this situation so that no value between delimiters is treated as a missing value. When the DSD option and no DELIMITER= option is used, the delimiter defaults to a comma.

```sas
data error;
  infile cards delimiter=',';
  input school :$20. students teachers;
cards;
  ,253,32
  Lyon Elementary,432,47
  Webber School,566,65;
PROC PRINT FOR DATA SET ERROR
OBS    SCHOOL           STUDENTS    TEACHERS
1     253                  32          . _ERROR_=1 _N_=1
2     Webber School       566         65
```

NOTE: Invalid data for TEACHERS in line 50 1-15.
RULE:----+----1----+----2----+----3----+----4----+----5----+----6
50   Lyon Elementary,432,47
SCHOOL=253 STUDENTS=32 TEACHERS=. _ERROR_=1 _N_=1
NOTE: SAS went to a new line when INPUT statement reached past the end of a line.

NOTE: The data set WORK.ERROR has 2 observations and 3 variables.
```
Delimited Data with Text Qualifiers

If the character values are enclosed in text qualifiers (single or double quotes), the DSD option removes the quotes from the value of the character value.

data stats2; 
infile cards dsd; 
input school : $20. students teachers; 
cards; 
** , 253, 32 
"Lyon Elementary", 432, 47 
"Webber School", 566, 65; 
To prevent the quotes from being removed from the values of variable school, add the ~ argument in the input statement as show below: 
input school ~$20. students teachers; 

HEADER RECORDS

Using headers eliminates the need to repeat data and reduces the size of raw data files. The following raw data can be transformed by using a school header.

**ORIGINAL DATA; 
data survey; 
input school $15. type $4. +1 
(item1-item10)(1.); 
cards; 
WEBBER SCHOOL K-6 1235324674 
WEBBER SCHOOL K-6 7498488367 
HALL INSTITUTE 9-12 3967026394 

*DATA WITH HEADERS; 
data survey; 
retain school type; 
input flag $ 1 @; 
if flag="*" then do; 
input #2 school $15. 
#3 type $4.; delete; end; 
else 
input #1 (item1-item10)(1.); 
#2 drop flag; 
cards; 
*WEBBER SCHOOL K-6 1235324674 
HALL INSTITUTE 9-12 3967026394; 

Header records have a * in the first column. The first character of a record (line) is read as variable flag. If flag="*", the current record is a header. The trailing @ holds the current record in case that record requires further processing. For records identified as headers, the school name (SCHOOL) and the school type (TYPE) are read in as variables (input #2 school $ 15. type $4.). The SCHOOL and TYPE values are retained and the header record is deleted. Subsequent records under a header will have the SCHOOL and TYPE values for that header. Records that are not headers contain data to be read by another input statement.

There are data that use various levels of headers. For example, there might be a school header, a grade header, and a classroom header that precedes student data records.

MULTIPLE RECORDS PER LINE

Instead of having only one record per line, it is possible to place multiple records on the same line by using the double trailing @.

MULTIPLE LINES PER RECORD

In the following example, 9th and 11th grade students have two lines per record and only have science and reading scores. On the other hand, 10th and 12th grade students have three lines per record and have science, math, and language scores.

**DATA WITH HEADERS; 
data survey; 
retain school type; 
input flag $ 1 @; 
if flag="*" then do; 
input #2 school $15. 
#3 type $4.; delete; end; 
else 
input #1 (item1-item10)(1.); 
#2 drop flag; 
cards; 
*WEBBER SCHOOL K-6 1235324674 
HALL INSTITUTE 9-12 3967026394; 

PROC PRINT FOR DATA SET GRADES 
   A   L 
   B   S   R   A 
   C   E   N 
   D   I   A   G   F 
   E   I   A   G   F 
   N   S   E   A   S   G   H   E   T   T 

Christopher Agustin 10 2.75 13 
Joy Recio 9 1.25 17 

The SCAN function isolated the first and last names in variable NAME. The FIRST and LAST name values are delimited by a space in the NAME variable.

The previous example shows a combination of input styles in the same DATA step: column input for STUDENT; the @32 column pointer control for GRADE; list input without pointer controls for GPA and ABSENCES; line pointer controls for the 2nd and 3rd lines of data; and named input for SCIENCE, MATH, LANGUAGE, and READING.

CONCLUSION

The use of input statements and other SAS statements and functions expands the definition of readable data.

REFERENCES


TRADEMARK NOTICE

SAS is a registered trademark or trademark of the SAS Institute Inc. in the USA and other countries. © indicates USA registration.