SAS Programming Basics

Clarence Wm. Jackson, CSQA

Abstract

This paper will introduce SAS to new programmers by offering the very basic concepts of the SAS System. How to set up the SAS program with DATA and PROC steps, a review of the SAS processing cycle and step boundaries will be discussed. Examples of simple SAS programs to read a record using INPUT statements, simple logic processing, and use of PROCs SORT and PRINT will be presented.

Introduction

SAS as a programming language can be learned quickly and a user can began writing programs within hours of being introduced to SAS if there is the correct information being taught. Before I started using SAS, I programmed in many of the mainframe languages such as COBOL, ALC, FORTRAN, and others. One thing about those languages is the amount of things that need to be done before you even run the program.

My first encounter with SAS was in 1982 when I was hired as a systems analyst. The other analysts were using another mainframe based system, and my boss was using mainframe SAS version 79. A few months after I got started, we upgraded to version 82. Even though I'm now using SAS 9.x on my laptop with Enterprise Guide, I still like to code SAS, and the language didn't change, it just got better.

Was it easy to learn? You betcha! Once I got past the junk in my head about how complex computer languages are setup. How did I learn SAS? I wrote a program, ran it, and modified it, etc, until I was really having fun. And that was on day 1. Day 2 I learned how to deal with the log errors and notes, and then I pulled out the book, since there was only one SAS book at the time.

The objective of this paper is to give the reader just enough to get started with SAS. When you finish reading this paper, you too will be able to write a SAS program and run it. Then, you need to go get the books to improve your SAS skills.

Please consider the following before you start:

1. Define what the purpose of the program will be. In the old days, we would use requirements worksheets to document the purpose for programs, but those are rarely used outside of IT shops, and people tend to get the requirements loosely without doing the formal worksheets. But it is always a good idea to know what you want to get out of your program before you begin. So, just to help you in your efforts, you should note the major objectives for your program so you will know if you succeeded when you run it.
2. Review your data, or have an idea of the data structure, especially for raw data files. If you are using a SAS dataset/file that you didn't create as your input data, run a PROC CONTENTS on it. What's PROC CONTENTS? Now you need to read the book because I'm only going to say that it gives you

information about a SAS file that will really be useful, such as variable names, data types, and other details about the SAS file.

3. Keep it simple. SAS can be a simple programming tool, and new users can start using it faster than most any other programming language. But until you get more experience with SAS, keep it very simple so you can keep up with it.
4. When you write the program, test it. Use a dummy data file or set the program to read NOOBS so you can check the syntax and resolve coding errors quicker. By doing this, you can save time and resources. After you run it without data and get a clean run, add the data. If you are processing a lot of data, you might set up the program to only read a few records.

Understanding SAS DATA and PROC Steps and Step Boundaries

The 1st thing to understand about SAS is the concept of "Step Boundaries" denoted by 'DATA' and 'PROC' or procedure statements. SAS runs whatever code it finds within the step, processes the data, then goes to the next step in the next DATA or PROC step, repeats the cycle within the step, and so on until the end of the SAS job. When you began to write a SAS job, you will begin with either a DATA or PROC statement. A DATA step is where the real logic processing occurs within a SAS job. This is where you will define the data that could be processed by a PROC. Think of using the DATA step to get to your data and using the PROC step to process and present your data.

SAS Terms That You Need To Understand

SAS terminology that you find in the books can be confusing if you are new to SAS, even for a programmer. Some of the terms you will encounter in books and conversation with SAS users are

A. Dataset – table or collection of records that have been processed by a SAS DATA step

B. Observation – an individual data record within the dataset

C. Variable – a field or part of the data record

D. Data Values – the value of the variable

OK, that may not have cleared things up, so let me put it like this: When you go shopping, your entire list of purchases on the receipt relates to a 'dataset'. The individual items on the receipt are the observations. The item information such as description, quantity, and price are 'variables', and the actual description, quantity, and price are the 'data values'. Take for instance, the following line of data is considered an observation with five variables. The 1st variable's value is 'ALFRED', the 2nd variable's value is 'M", and so.

```
ALFRED     M 14 69 112
```

This will be illustrated further in the example program later in the paper. Other terms that you will hear spoken by experienced SAS users but not covered in this paper are

1. SAS Supervisor – This controls the SAS System from start to finish, based on preset options, your code, and the data.

2. Program Data Vector – This tracks the values of your data and the data that the SAS Supervisor uses.

A very important concept to understand is the computing cycle, which is simply 'input-process-output'.

- Input is the start of any compute cycle. There must be something coming in to be processed before any output can occur. Input for SAS programs can be from raw data files, in stream cards or data lines, or SAS dataset/files. Input is what you want to process.

- Process is the part of the cycle that turns the input into something that can be use, a very important part of the cycle.

- Output is the part of the cycle that gives you what you need from the computer.

Every program requires input, processes the input, then returns the results as output. SAS does that in so many different ways. For the purpose of this paper, DATA steps are the part of SAS that does the input and processing, and PROC steps are processing and output. However, you will find that both DATA and PROC steps can do input-process-output as you advance in your SAS learning.

SAS Programming Statements

SAS programs are done by using SAS statements that describe the action to be taken by SAS. SAS statements are delimited by ';' or a semi-colon. This denotes then end of a statement segment. SAS is a free form programming language, which means there are no rules regarding where on the line of code statements need to be. Each statement contains a SAS keyword, at least one blank between words, and a semi-colon. You can write SAS where one statement can span multiple lines of code, which makes it possible to write code that is easy to read and maintain.

SAS DATA steps usually create SAS datasets or datafiles from raw input data. SAS uses the following keyword statements to perform the input process within the DATA step:

- INFILE statements defines the raw source file of data to be read into SAS

- INPUT statements defines the location of fields on the record that will become variables

- CARDS/DATALINES tells SAS that data follows in the job stream. The ';' is used to tell SAS to stop reading the data as data and start back to reading SAS statements.

SAS also provides processing in the DATA Step with statements that does some things that allow logic to occur with **IF, IF-THEN, IF-THEN-ELSE, DO-END,** and **SELECT** statements, which will not be covered in this paper, but are really covered in the SAS Help files.

For this paper, a small file of information on kids in a class will be used. The file contains the name, sex, age, height, and weight of each child. The task is to set these up in a SAS dataset and create a listing. A simple SAS program is needed that reads a record in the DATA step from card input, and prints the output in the PROC PRINT step. There are 2 steps in this program.

```
DATA CLASS;
  INFILE CARDS;
  INPUT     NAME  $ 1-10
            SEX   $ 12
            AGE     14-15
            HEIGHT  17-18
            WEIGHT  20-22;

CARDS;
ALFRED      M 14 69 112
ALICE       F 13 56 84
BERNADETT   F 13 65 98
BARBARA     F 14 63 102
HENRY       M 14 63 102
JAMES       M 12 57 83
JANE        F 12 59 85
JANET       F 15 62 112
JEFFREY     M 13 62 84
JOHN        M 12 59 99
JOYCE       F 11 51 50
JUDY        F 14 64 90
LOUISE      F 12 56 77
MARY        F 15 66 112
PHILLIP     M 16 72 150
ROBERT      M 12 64 128
RONALD      M 15 67 133
THOMAS      M 11 57 85
WILLIAM     M 15 66 112
;


PROC PRINT;
```

The 1st step marked A consists of 4 statements, DATA INFILE INPUT, and CARDS. The DATA statement creates a SAS dataset/file called 'CLASS', defines the input file with the INFILE statement, defines the location of the variables NAME, SEX, AGE, HEIGHT, and WEIGHT and what columns in the raw data record the values for

those variables will be, then defines the in stream input records with the CARDS statement B. So for the 1st record:

```
ALFRED      M 14 69 112
```

The NAME variable will have the value 'ALFRED', SEX would have the value 'M', AGE would be '14', HEIGHT would be '69', and WEIGHT would be '112'. That would be the 1st observation, and SAS will loop thru the data assigning values to variables until the data file is read completely. Notice the semi-colon at the end of the data - this lets SAS know that the data is at an end.

Then PROC PRINT step is used to present the data as a list C.

Notice that NAME and SEX have a '$' between the variable name and the column number. SAS uses 2 basic data types, character and numeric, to read the values into the variables. SAS will assume that all values are numeric unless you designate character values. You tell SAS to handle characters by using the '$' format. SAS will set a default length for characters at 8 unless you define how long the string is.

Looking at the example data above, notice that the name 'BERNADETT' has a length of 9, so at execution of the program, SAS will truncate or shorten it to 8, causing the value to lose the last 'T'. To correct this, we should use an input format of '$9.' Or as noted, include the columns so SAS will allocate a length of 9. The changed code will be

```
NAME  $9. 1-10
```

The output would now look like this:

```
                                The SAS System         19:07 Tuesday, August
25, 2009   1

                        Obs    NAME         SEX    AGE    HEIGHT    WEIGHT

                          1    ALFRED        M      14      69        112
                          2    ALICE         F      13      56         84
                          3    BERNADETT     F      13      65         98
                          4    BARBARA       F      14      63        102
                          5    HENRY         M      14      63        102
                          6    JAMES         M      12      57         83
                          7    JANE          F      12      59         85
                          8    JANET         F      15      62        112
                          9    JEFFREY       M      13      62         84
                         10    JOHN          M      12      59         99
                         11    JOYCE         F      11      51         50
                         12    JUDY          F      14      64         90
                         13    LOUISE        F      12      56         77
                         14    MARY          F      15      66        112
                         15    PHILLIP       M      16      72        150
                         16    ROBERT        M      12      64        128
                         17    RONALD        M      15      67        133
                         18    THOMAS        M      11      57         85
                         19    WILLIAM       M      15      66        112
```

PROC PRINT and PROC SORT

Now that we have created a SAS dataset, we can use PROCs to process it. For the most part, PROCs operates only on SAS files, so the data must be in SAS data format. PROC PRINT as used in the example only needs one statement to produce a listing, however there are many options that can be used to make your listing look better.

If you wanted to sort the data by SEX, then you would need to use PROC SORT. A simple usage would look like the following:

```
PROC SORT;
    BY SEX;
```

That's all that would be needed to sort the data. When you sort data using PROC SORT, SAS will consider SEX as the 'BY' variable, and set an index for you. You can then use the BY variable in other PROCs that support it, like PROC PRINT. An example of using the BY variable in PROC PRINT after a sort would look like the following:

```
PROC PRINT;
    BY SEX;
```

This will produce a listing that is grouped by SEX.

```
                                The SAS System         18:37 Tuesday, August 25, 2009   1

------------------------------------------- SEX=F -------------------------------------------

                        Obs    NAME         AGE    HEIGHT    WEIGHT

                          1    ALICE         13      56         84
                          2    BERNADETT     13      65         98
                          3    BARBARA       14      63        102
```

Now let's do a few things with the listing. First, there are many ways to get rid of the 'OBS' column, one of which is to use the 'ID' statement to be the 1st variable in the list. We can also easily add group totals with the 'N' option on the PROC PRINT statement. To do so, we'll modify the PROC PRINT code to look like the following:

```
PROC PRINT N;
        BY SEX;
        ID NAME;
```

And the output results would be:

```
                                        The SAS System         18:37 Tuesday, August
25, 2009   2

------------------------------------------- SEX=F
-------------------------------------------

                                NAME          AGE    HEIGHT    WEIGHT

                                ALICE          13      56         84
                                BERNADETT      13      65         98
                                BARBARA        14      63        102
                                JANE           12      59         85
                                JANET          15      62        112
                                JOYCE          11      51         50
                                JUDY           14      64         90
                                LOUISE         12      56         77
                                MARY           15      66        112

                                               N = 9


------------------------------------------- SEX=M
-------------------------------------------

                                 NAME        AGE    HEIGHT    WEIGHT

                                 ALFRED       14      69        112
                                 HENRY        14      63        102
                                 JAMES        12      57         83
                                 JEFFREY      13      62         84
                                 JOHN         12      59         99
                                 PHILLIP      16      72        150
```

Now, to add a title that replaces 'The SAS System' to something we might like, we simply can add a TITLE statement. TITLE and FOOTNOTE statements are global, meaning that they can be anywhere in the program. You can have up to 10 of each as 'TITLE1,..TITLE10'. We'll add the TITLE statement right before the PROC PRINT as:

```
TITLE 'SAS Programming Basics – CJac IT Consulting';

PROC PRINT N;
        BY SEX;
```

```
ID NAME;
```

And the resulting report now looks like:

```
                          SAS Programming Basics - CJac IT Consulting                          3
                                                                  18:37 Tuesday, August 25, 2009

---------------------------------------------- SEX=F ----------------------------------------------

                                NAME          AGE    HEIGHT    WEIGHT

                                ALICE          13      56         84
                                BERNADETT      13      65         98
                                BARBARA        14      63        102
                                JANE           12      59         85
                                JANET          15      62        112
                                JOYCE          11      51         50
                                JUDY           14      64         90
                                LOUISE         12      56         77
                                MARY           15      66        112

                                             N = 9


---------------------------------------------- SEX=M ----------------------------------------------

                                NAME        AGE    HEIGHT    WEIGHT

                                ALFRED       14      69        112
                                HENRY        14      63        102
                                JAMES        12      57         83
                                JEFFREY      13      62         84
                                JOHN         12      59         99
                                PHILLIP      16      72        150
                                ROBERT       12      64        128
                                RONALD       15      67        133
                                THOMAS       11      57         85
                                WILLIAM      15      66        112

                                             N = 10
```

There are many other things we could do to the report and the program to make it work better, but this will be enough to get you started. Please refer to the SAS HELP or other documentation on the complete list of options in PROC PRINT.

Reading the SAS Log for Errors

If your SAS program stops running (abnormal ending, or ABEND), you will need to review the SAS log. The SAS Log report is your buddy, and tells you everything that SAS is doing for you, and will be your source of feedback regarding the execution of the SAS program. It will tell you if you have an error, missed a record read, how many records were read and processed, and other useful information. Most times, you will have a clean log with a bunch of 'NOTE:' without errors, but in those times that you don't, you will be glad to have the log to guide you in fixing

your code. Always check your log. Here is the log for the last program. Notice that cards code are numbered, feedback from SAS are noted. If there were errors, it would be 'ERROR:' followed by the type of error encounted.

```
NOTE: Copyright (c) 2002-2003 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) 9.1 (TS1M3)
      Licensed to The SAS Learning Edition 4.1, Site 0042234001.
NOTE: This session is executing on the XP_PRO  platform.



NOTE: SAS 9.1.3 Service Pack 4

NOTE: This version of the Learning Edition is limited to 1500 input observations.
NOTE: SAS initialization used:
      real time           30.20 seconds
      cpu time            0.81 seconds

1
2    DATA CLASS;
3      INFILE CARDS;
4    INPUT NAME $ 1-10 SEX $ 12   AGE 14-15 HEIGHT 17-18 WEIGHT 20-22;
5
6    CARDS;

NOTE: The data set WORK.CLASS has 19 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.88 seconds
      cpu time            0.00 seconds


26   ;
27
28   PROC SORT;
29    BY SEX;
30   TITLE 'SAS Programming Basics - CJac IT Consulting';
31

NOTE: There were 19 observations read from the data set WORK.CLASS.
NOTE: The data set WORK.CLASS has 19 observations and 5 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.41 seconds
      cpu time            0.01 seconds


32   PROC PRINT N;
NOTE: Writing HTML Body file: sashtml.htm
33     BY SEX;
34     ID NAME;
35   run;

NOTE: There were 19 observations read from the data set WORK.CLASS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           3.39 seconds
      cpu time            0.30 seconds
```

One thing that the log will not show you are logic errors. If the results are not what you intended, read the log to see what went on with your program, and compare it to what you wanted to do with the program.

One of the most common errors is the missing semi-colon at the end of a statement. SAS uses the semi-colon as a statement delimiter, so you'll know you

missed one when parts of your code didn't run when SAS includes the next statement in sequence with the statement that has the missing semi-colon.

Creating Variables and Assigning Values

By using simple assignments, you can add variables, such as using the following statement in the DATA step to create a 'KILO' variable:

```
KILO=WEIGHT*.45;
```

KILO will have a numeric value. The '=' is the key, and the new value will be named first, followed by the value to be assigned to it. The value can be the result of a calculation, function, or other operation.

If I wanted to create a variable with a character value, I could do so like this:

```
IF SEX='M' THEN GENDER='BOY '; ELSE GENDER = 'GIRL';
```

GENDER would then be assigned a value of 'BOY ' or 'GIRL' based on the value of SEX. Notice that I included an extra blank on the first assignment of 'BOY '. This is because SAS will assign a length of the variable on the first assignment encounter, so if I didn't do this and used 'BOY', then the value for GENDER if SEX='F' would be 'GIR'.

Keep this in mind when assigning values – make sure they are big enough to hold the result. When in doubt, use a LENGTH statement.

Conclusion

Although this is a 2 step SAS program, all SAS programs are written a step at a time, and statement by statement. Use the help files, and read other books that will help you gain more knowledge in how to use SAS to solve your needs. So go for it and write your programs.

Author info

Clarence Wm. Jackson, CSQA
CJac.csqa@sbcglobal.net