# Introduction to the SAS® Programming Language

**Thomas J. Winn, Jr.**
Texas State Auditor's Office, Austin, Texas



**Abstract**

This short paper is first in the "Introduction to SAS" sequence of papers. It includes some of a programmer's first steps toward learning SAS. The paper provides a brief overview of the SAS System, as well as an outline of the basic structure of the SAS programming language. In particular, it covers preliminary concepts regarding DATA and PROC steps, the different kinds of data, and SAS data files. This paper, and the associated presentation, is intended to provide a foundation for the next step of learning SAS, that of creating SAS data sets, using the INPUT statement and INFILE options.

**A Very Brief History of SAS**

While a graduate student in statistics at NCSU, Jim Goodnight wrote a computer program for analyzing agricultural data. After a few years, Jim's application had attracted a diverse and loyal following among its users, and the program's data management and reporting capabilities had expanded beyond Jim's original intentions. In 1976, Jim decided to work at developing and marketing his product on a full time basis, and SAS Institute was founded. Since its beginning, a distinguishing feature of the company has been its attentiveness to users of the software. Today, SAS Institute is the world's largest privately-held software company, and Dr. Jim Goodnight is its CEO. He continues to be actively involved as a developer of SAS System software.

**The SAS System**

The SAS System is an integrated suite of information delivery software products. It is a library of modular components, that are tied together by a central supervisory program. Applications of the SAS System include executive information systems; data entry, retrieval, and management; report writing and graphics; statistical and mathematical analysis; business planning; forecasting, and decision support; operations research and project management; statistical quality improvement; computer performance evaluation; and applications development.

Originally, "SAS" stood for "Statistical Analysis System", however, the applicability of SAS has grown far beyond that single purpose. Today, SAS is not an acronym for anything. Besides statistics, elements of the SAS System also include web enablement, data mining, data warehousing, and business intelligence solutions, a variety of industry-specific and business functional-area products (for managing organization, customers, and suppliers), as well as state-of-the-art applications development tools. Today, the SAS System is a comprehensive system for data management and analysis.

SAS software collects data from almost any platform and data format; it cleans and transforms the data into information which decision makers will understand; and it stores the information in an open and efficient data storage structure. To explore information, SAS software provides multidimensional data analysis, query and reporting, executive information systems, data mining, data visualization, and applications development capabilities. SAS solutions are client/server- and Web-enabled. SAS says about itself, "The Power to Know.™"

Components of the SAS System include (partial listing):

- Base SAS
- SAS/ACCESS
- SAS/AF
- SAS/ASSIST
- SAS/CONNECT
- SAS/EIS
- SAS/ETS
- SAS/FSP
- SAS/GIS
- SAS/GRAPH
- SAS/IML
- SAS/INSIGHT
- SAS/IntrNet
- SAS/LAB
- SAS/MDDB Server
- SAS/OR
- SAS/QC
- SAS/SHARE
- SAS/SPECTRAVIEW
- SAS/STAT
- SAS/TOOLKIT
- SAS/AppDev Studio
- SAS/Enterprise Guide
- SAS/Enterprise Miner
- SAS Universal ODBC Driver
- SAS/Warehouse Administrator

The "Introduction to SAS" section is designed to cover the fundamental elements of the SAS programming language. It includes elementary DATA step, and Base/SAS PROCedure programming.

**Introduction to SAS Programming**

All SAS jobs are a sequence of SAS steps, which are made up of instructions, which are called SAS statements. There are only two kinds of SAS steps: DATA steps are used to read, edit, and transform data (raw data or SAS data files), to prepare SAS data sets, PROC steps are ready-to-use procedures which analyze or process SAS data sets. In general, data must be in a SAS data file before they can be processed by SAS procedures.

Without going into the details at this time, here is a skeletal example of a SAS job:

```
DATA STUDENTS;
        INPUT  NAME $ 1-14  SEX $ 15
                SECTION $ 17-19 GRADE;
        DATALINES;
        . . . data lines . . .
        ;
PROC SORT DATA=STUDENTS;
        BY SECTION  DESCENDING GRADE;
 PROC PRINT  DATA=STUDENTS;
        BY SECTION;
 RUN;
```

There are two kinds of SAS data sets: SAS data files (or tables), and SAS data views. A SAS data file contains: the descriptor portion, which provides SAS procedures and some DATA step statements with descriptive information (data set attributes and variable attributes) about the data , and the data portion, a rectangular structure containing the data values, with rows (customarily called observations), and columns (customarily called variables); and which is passed to most procedures, observation by observation. A SAS catalog is a type of SAS file which stores many different types of information used by the SAS System. All SAS files reside in a SAS data library.

The SAS System processes the program in two steps: (1) it compiles the program, and (2) it executes the program. When the program is compiled, a program data vector (PDV) is constructed for each DATA step. It is an area of memory which includes all variables which are referenced either explicitly or implicitly in the DATA step.

At execution time, the PDV is the location where the current working values are stored as they are processed by the DATA step. Variables are added to the PDV sequentially as they are encountered during parsing and interpretation of SAS source statements. Each step (DATA or PROC) is compiled and executed separately, in sequence. And at execution time within each DATA step, each observation is processed iteratively through all of the SAS programming statements of the DATA step.

SAS procedures (PROCs) are programs that are designed to perform specific data processing and analysis tasks on SAS data sets. Base/SAS procedures fall into the following categories:
**SAS Utilities** -- APPEND, CATALOG, CIMPORT, COMPARE, CONTENTS, COPY, CPORT, DATASETS, DBCSTAB, DISPLAY, EXPLODE, EXPORT, FORMAT, FSLIST, IMPORT, OPTIONS, PMENU, PRINTTO, RANK, REGISTRY, SORT, SQL, STANDARD, TRANSPOSE, TRANTAB;
**Descriptive Statistics** -- CORR, FREQ, MEANS, SQL, SUMMARY, TABULATE, UNIVARIATE;
**Reporting** -- CALENDAR, CHART, FORMS, MEANS, PLOT, PRINT, REPORT, SQL, SUMMARY, TABULATE, TIMEPLOT.

**Creating SAS Data Files**

Since SAS procedures can operate only on SAS data sets, then the first step in processing any raw data using SAS will be to transform them into a SAS data set. Whenever the SAS System creates a SAS DATA file, it does the following:

1. it reads the DATA statement, creates the structure of a SAS data set, and marks the statement as the place to begin the processing of each line of data;
2. it uses the description of the data in the INPUT statement to read the data line, and to produce an observation;
3. it uses the observation to execute any other SAS statements that are in the DATA step;
4. it adds the observation to the data set being created; and
5. it returns to the beginning of the DATA step for the processing of the next observation.

All SAS DATA step statements are executed once for each observation.
All SAS statements begin with an identifying keyword, and end with a semicolon. SAS statements are free-format.

They can begin anywhere, and end anywhere. A single statement may continue over several lines. Several statements may be on a single line. Blanks (as many as desired) are used to separate fields. Other special characters also may be used to separate fields.

The data portion of a SAS data file is a collection of data values arranged in a rectangular table. The rows in the table are called observations. The columns in the table are called variables. There are <u>two</u> kinds of variables: character variables, and numeric variables. Each variable has a name.

There are rules for naming SAS data sets and variables: 1 to 32 characters in length (8 character maximum in Version 6 and earlier versions), start with A-Z or _ (underscore), continue with letters, numbers, or underscores. It is recommended that you choose meaningful names.

Character data can consist of up to 32,767 characters (max. of 200 characters in Version 6 and earlier versions). Character values may include letters, numbers, blanks and special characters, although generally you should not include any semicolons within the data. Numeric data values must be numbers, and they may be preceded by a + or -. Unless the data are being read using a special SAS informat, do not include commas or dollar signs in numeric data values. SAS assigns the value of a decimal point (“.”) to missing numeric values, and a blank (“ ”) to missing character values. You can enter these values into your data to indicate missing values.

Every SAS data set has a name and is physically stored on some type of media (disk, tape, etc.). In simple jobs, the SAS data sets are stored on temporary space, but they can be stored “permanently”. A temporary SAS data set exists only for the duration of the current SAS job, or interactive SAS session. A permanent SAS data set exists after the end of the current SAS job or interactive SAS session. Both types of SAS data sets have two-level names, of the form libref.data-set-name, where libref is a reference to the name of a SAS data library (a collection of SAS files).

With temporary SAS data sets, the SAS System automatically assigns the libref WORK and you specify the data set name. When you create a permanent SAS data set, you must specify both the libref and the data set name. The SAS System does not assign the libref for you. Although there are other methods for certain operating environments, the LIBNAME statement is the most universal method of assigning a libref. The general form is

```
LIBNAME libref ‘SAS-data-library’;
```

Here is an example of a LIBNAME statement in the Windows environment:

```
LIBNAME mylib1 ‘C:\mySASlib’;
```

Here is an example of a LIBNAME statement in certain mainframe environments:

```
LIBNAME mylib2 'data.set.name';
```

Here is an example of reading data from a "permanent" SAS data set (in the SAS data library whose previously defined libref is 'MYLIB'):

```
DATA EXAMPLE;
      SET MYLIB.STUFF;
```

Here is an example of creating a "permanent" SAS data set (also in the SAS data library whose previously defined libref is 'MYLIB'):

```
DATA MYLIB.TESTDATA;
      SET SAMPLE1;
```

A SAS DATA statement instructs the SAS System to create and name a SAS data set. It has the general syntax:

```
DATA data-set-name-1  (options-1)
      data-set-name-2  (options-2)
            . . .
      data-set-name-k  (options-k) ;
```

Many SAS data sets can be created in a single DATA step. DATA step options include such things as: DROP= , IN= , FIRSTOBS= , KEEP= , OBS= , RENAME= , WHERE= , and others.

The two major functions of the DATA statement are: to signal the beginning of the DATA step, and to name the data set(s) being created.

When creating temporary SAS data sets, the data set name can be supplied by the programmer:

```
DATA STUDENTS;
      INPUT  NAME $ 1-14  SEX $ 15
             SECTION $ 17-19 GRADE;
      DATALINES;
      . . . data lines . . .      ;
```

or, if the name is omitted in the DATA statement, the SAS System will provide a name (DATA1, DATA2, etc.):

```
DATA ;
      INPUT  NAME $ 1-14  SEX $ 15
             SECTION $ 17-19 GRADE;
      DATALINES;
      . . . data lines . . .              ;
```

**A Few Words About Working With Dates and Times Using SAS**

Whenever SAS reads date value inputs, it converts them into integers. SAS dates are positive or negative integers representing the number of elapsed days between January 1, 1960 and the specified date. Similarly, SAS converts time values into the number of seconds since midnight of the current day. SAS datetime values are the number of seconds since midnight on January 1, 1960. Since dates and times are numeric entities, one may use ordinary arithmetic to determine elapsed time, or future/past dates and times.

For examples,

```
AGEDAYS = THISDATE - BIRTHDATE;
AGEYRS = AGEDAYS / 365.25;
```

You may use date constants or time constants in a SAS expression by writing the date or time enclosed in quotes, and followed by a D (date), a T (time), or DT (date:time).

```
THISDATE = '20Aug2001'D;
SLEEPTIME = '23:59:59.9'T;
FAISDODO = '21Aug2001 20:30'DT;
```

To read data that are date or time values, SAS has a variety of informats. To write date or time values in reports, SAS has numerous formats. SAS also has several special functions for working with date or time values. We'll learn more about informats, formats, and functions in another presentation.

**Summary**

This short paper included some of a programmer's first steps toward learning about the SAS programming language. In particular, it covered the following items: an overview of the SAS System, a few fundamental ideas regarding SAS data sets, some preliminary concepts regarding DATA and PROC steps, and the different kinds of data in SAS.

**Suggested References:**


Ronald P. Cody & Raymond Pass, SAS Programming By Example (1995)

Lora D. Delwiche & Susan J. Slaughter, The Little SAS Book: A Primer, Second Edition (1998)

Frank DiIorio, SAS Applications Programming: A Gentle Introduction

SAS Institute Inc., SAS OnlineDoc, Version 8

SAS Institute Inc., Getting Started With the SAS System, Version 8

SAS Institute Inc., SAS Language Reference: Concepts, Version 8

SAS Institute Inc., SAS Language Reference: Dictionary, Version 8, Volumes 1 and 2

SAS Institute Inc., SAS Procedures Guide, Version 8, Volumes 1 and 2


**Author Information.**


Tom Winn
Texas State Auditor's Office
P.O. Box 12067
Austin, TX 78711-2067

phone: 512 / 936-9735
e-mail: twinn@sao.state.tx.us