Paper 087-2010

New Dogs and Old Tricks Part II: Using the SELECT statement and FLAGS to Streamline your Code

Stanley Fogleman, Harvard Clinical Research Institute, Boston, MA

ABSTRACT

Complex logic can make a program difficult to comprehend, analyze and maintain. Using the BASE language SELECT statement can provide a "Skeleton" for a well structured program. In addition, using FLAGS (a variable with only two possible values TRUE or FALSE) can make it easier to understand why a certain branch was taken. FLAGS can compartmentalize complex decisions to one area of the program and make later modification easier to locate and implement.

INTRODUCTION

Decision logic is usually the heart of any program. For anything more complex than a simple IF-ELSE, a SELECT statement is usually a better choice. Flags are a way to encapsulate complex logic. This paper introduces these two techniques with the hope that they will make your program more compact, easy to modify and easier to maintain. When IF statements grow more complex it is much harder to indent properly and follow the flow of logic. The beauty of the select statement is that only one branch will be taken per observation. An IF statement might take many different paths depending on the data passed through the program.

WHAT IS A SELECT STATEMENT?

If you are a SQL programmer, you may think you already know the answer to this. The SELECT statement I am referring to is a BASE SAS statement and IS NOT part of PROC SQL. The SELECT is basically a structured IF-ELSE statement. It has two forms.

The first is the simplest:

```
data one;
select;
 when (PanelName eq 'ENROLL')
  do;
  put 'some sample sas code';
   end;
   otherwise
   do;
   put 'some other code instead';
   end;
end; /* of SELECT statement */
```

The second form is a little trickier:

```
data one;
select(PanelName);
 when ('ENROLL')
  do;
  put 'some sample sas code';
   end;
   otherwise
   do;
   put 'some other code instead';
   end;
end; /* of SELECT statement */
```

In the first example, each 'When' statement is evaluated separately and in the second for, it is the resolved value of the variable that determines which branch is taken.

WHAT ARE FLAGS?

As stated earlier, flags are a "switch" variable with only two possible values. The point of creating them is to test for complex conditions once and store the value in a flag to be referenced later in the program.

source code:

```
data one;
set two;
if mix(var1, var2) > 0 then _THISCOND = 1;
```

Although this is a trivial example, for the rest of the data step, you can check _THISCOND instead of restating the condition.

Now imagine something more complicated:

```
If month in (10,11) and number_of_games_played > 162 then _playoffs = 1;
```

So "_playoffs" is a much more concise way of determining if we are in playoff season or not.

It is important to note that sas has a "built_in" feature to test for "true-false" where zero is false and any other value is true, Flags leverage this capability of sas, as in the following example:

```
if _merged or _multiple or _manyvis
then
    put 'true that';
else
    put 'no way jose';
```

Flags can be used in where clauses:

```
proc sort data=replicate(where=(_merged or _multiple or _manyvis))...
```

Just as an editorial comment, repeating complicated if clauses instead of using flags approaches madness.

WHAT ABOUT USING THEM BOTH?

The advantage that using both can bring are twofold:

1. a more tightly organized program.
2. one that is easier to diagnose and modify.
3. can catch error conditions by use of the built-in “otherwise” clause.

An example which uses both the select statement and flags:

```
select;
   when (_conditionA)
   do;
   put '*;';
   end;
   when (_conditionB)
   do;
   put '**;';
   end;
   otherwise
   do;
   put '***;';
   end;
end;
```

This is where the approach begins to pay dividends. In addition to having a flexible means of making decisions (the Select statement) you also have a good clue as to why a particular path was chosen (the use of flags in the condition statement). The otherwise clause should always be coded as a defensive measure in case faulty data is introduced to the program. Note: if all the conditions in the select statement are false and the otherwise statement is omitted, an error message is issued. A Select statement WITHOUT the benefit of flags can also be unwieldy. In addition the SELECT statement requires a condition to resolve to TRUE or FALSE and it is not always apparent which part of a complex condition is causing a problem, which can lead to lost time.

For example[1] (from the SAS Language manual):

```
  select;
     when (mon in ('JUN', 'JUL', 'AUG')
     and temp>70) put 'SUMMER ' mon=;
     when (mon in ('MAR', 'APR', 'MAY'))
     put 'SPRING ' mon=;
     otherwise put 'FALL OR WINTER ' mon=;
  end;
```

This is a prime example of “dense” coding and it is this style of programming that we wish to get away from.

CONCLUSION

Flags and the select statement are valuable tools to be added to any programmer's toolbox. Often the ability to maintain and modify programs is overlooked in the rush to get something functional out right away. Sometimes the ability to understand and make quick changes to a working program are as important as the "debut" of the program to begin with. As the maxim states, "Useful programs will always need to be modified" and the only thing worse that a program without all of the features you might like is a program that has had logic errors introduced as a result of faulty modification. Use of the SELECT statement requires some discipline, but the payoff is well worth it. Using Boolean logic can require a learning curve for some people, but again, it has the potential to make programs much simpler and easier to comprehend.

REFERENCES

1. SAS Language Reference 9.2 Dictionary – Second Edition Cary: SAS Institute p. 1699

ACKNOWLEDGMENTS

Two people who taught me to be a maintenance programmer: Aloys Verheggen and Patrick Macksey.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Stanley Fogleman
Harvard Clinical Research Institute
930 Commonwealth Ave
3rd Floor
Boston, MA 02215
Work Phone:
Fax:
E-mail:
Web:

sas | Certified Advanced Programmer for SAS 9