Creating a Flexible Parameter Driven Reporting Program Using Global Variables

Karen Dudley, Blue Cross Blue Shield New Mexico, Albuquerque, NM

Abstract

By using global variables as parameters, a program can be created which will produce an infinite number of information variations from a single program. The parameters can be toggled on and off by setting up a test flag variable. The parameters may be a single value, a range of values, or a combination of values.

This approach requires only a single entry of the parameters for multiple usage. The global variables can be used to subset the data, create flexible merge statements, and generate meaningful report titles. Additionally, the values can be held in a permanent file to be used by other related programs.

Programs developed using this flexible structure can easily be used by business analysts and other non-programming users to create reports to meet their particular business requirements.

Introduction

This paper presents various parameter types and assorted usages of these parameters. The types of parameters described are: numeric, date, and character. The parameters are set up using %LET and PROC FORMAT. They can be used as a print flag, to subset data, and to create selective MERGE statements. Using PUT statements, CALL SYMPUT, and permanent files the parameters can also be used to set up report titles. Following are examples and suggested usage for various types of parameters.

Examples

Print Flag

A simple yet powerful way to use a parameter is to set a flag that determines if a report should be printed. This is helpful in a program that produces both detail and summary reports. Often times the detail report can be very lengthy and is not always needed. This parameter can be used to specify if the detail report should or should not be printed.

Create the parameter at the top of the program:

```
/* SET UP PRINT PARAMETER */
%LET PRNTDTL = 'N';
```

Use parameter in a WHERE statement:

```
/* PRINT DETAIL REPORT */
PROC REPORT;
   WHERE &PRNTDTL = 'Y';
   COLUMN.....
```

Setting the parameter to 'Y' will print the report. Setting the parameter to any other value (I prefer 'N' to be clear) will *NOT* produce the report. You will find this much easier than commenting the reporting section on or off.

Numeric

One usage of a numeric parameter is to subset data based on an amount. A common health care usage is to find claims that are over a specified amount, for example, claims with paid charges over $20,000. This parameter should also be used in the report title so that it is clear what the designated amount was.

Create the parameter at the top of the program:

```
/* SET UP PARAMETER */
%LET AMTPD = 20000;
```

Use parameter in a subsetting IF:

```
/* GET CLAIMS HISTORY INFO */
DATA CLAIMS;
INFILE HISTORY;
INPUT
   @105  PAIDAMT  7.2  @;
IF PAIDANT GE &AMTPD;
INPUT....
```

Create a formatted numeric variable to print in the report title:

```
DATA _NULL_;
   CALL SYMPUT ('AMT_PD',PUT
      (&AMTPD,DOLLAR7.2));
RUN;
```

Use the formatted variable in the report title:

```
PROC REPORT;
TITLE1 'REPORT TITLE';
TITLE2 "PAID CLAIMS OVER &AMT_PD";
COLUMN.....
```

Dates

Date parameters can also be used to subset data. A common health care usage is to find claims that were paid during a designated period of time. The dates should also be listed in the report titles so that the reporting time frame is specified.

First set up the date parameters at the top of the program:

```
/* SET UP PARAMETERS */
```

```
%LET PAID_BEG = '01JAN97'D;
%LET PAID_END = '31JAN97'D;
```

Subset the data:

```
/* GET CLAIMS HISTORY INFO */
DATA CLAIMS;
INFILE HISTORY;
INPUT
   @23  PAIDDTE   YYMMDD6.  @;
IF PAIDDTE GE &PAID_BEG
   AND PAIDDTE LE &PAID_END;
INPUT....
```

Create formatted date variables to be printed in the report title:

```
DATA _NULL_;
   CALL SYMPUT('STDATE',
      PUT(&PAID_BEG,MMDDYY10.));
   CALL SYMPUT('EDDATE',
      PUT(&PAID_END,MMDDYY10.));
RUN;
```

Use the formatted date variables in the report title:

```
PROC REPORT;
TITLE1 'REPORT TITLE';
TITLE2 "PAID DATES &STDATE TO
       &EDDATE";
COLUMN.....
```

Combination Character Values

Character parameters may be used in a number of ways. They can be used to test a single value, a range of values or a combination of values. A common health care usage is to look at a specific provider or a range of providers based on their provider identification numbers. This example uses a combination of values. Being able to subset data based on a combination of values is very dynamic. The testing parameter is created using PROC FORMAT.

Create a parameter at the top of program:

```
&LET PROV_NUM =
   'K123','K456','2220'-'2229';
```

Create testing values using PROC FORMAT:

```
PROC FORMAT;
   VALUE $PROVOK
   &PROV_NUM = 'Y'
   OTHER = 'N';
RUN;
```

Create a subsetting variable by "formatting" the input variable:

```
/* READ PROVIDER INFO */
DATA TEMPPROV;
INFILE PROVIDER;
   @10   PROVID   $CHAR4.   @;
GOODPROV = PUT(PROVID,$PROVOK.);
IF GOODPROV = 'Y';
INPUT.....
```

This is accomplished by creating a testing variable using a PUT statement with the input variable and the user defined format. In this example, if the current value of PROVID matches the format (e.g. it is K123 or K456 or between 2220 and 2229) then GOODPROV will be 'Y'. Otherwise GOODPROV will be 'N'. GOODPROV is then used in the subsetting IF statement. If GOODPROV is 'Y' the record will be written to TEMPPROV.

Test Flags

The above example can be made more powerful by combining it with a test flag which allows you to toggle the parameter testing on or off. This enables you to turn the test on and off simply by changing the testing flag.

Create variables at the top of program and include the test flag:

```
%LET TESTPROV = 'Y';
%LET PROV_NUM =
   'K123','K456','2220'-'2229';
```

Create testing values using PROC FORMAT:

```
PROC FORMAT;
   VALUE $PROV
   &PROV_NUM = 'Y'
   OTHER = 'N';
RUN;
```

Test input values:

```
/* READ PROVIDER INFO */
DATA TEMPPROV;
INFILE PROVIDER;
   @10   PROVID   $CHAR4.   @;
GOODPROV = PUT(PROVID,$PROV.);
IF (TESTPROV = 'Y'
      AND GOODPROV = 'Y')
   OR TESTPROV = 'N';
INPUT.....
```

In the above example, if the value of PROVID is K123 or K456 or between 2220 and 2229 then the record would be written to TEMPPROV because TESTPROV is 'Y' and GOODPROV is 'Y'.

If TESTPROV is set to 'N' then value of PROVID would not matter and all records would be written to TEMPPROV. Having the test flag creates more initial code but eliminates maintenance work.

Selective MERGE

If you have multiple test flags and parameters in your program the test flags can be used in MERGE statements.

Create variables at top of program:

```
%LET TESTSPEC = 'Y';
%LET SPEC_NUM = '55';
```

```
%LET TESTPROV = 'N';
%LET PROV_NUM = 'K1234';
```

Use test flags to determine MERGE:

```
DATA HOLD;
MERGE TEMPPROV(IN=FILEA)
    TEMPSPEC(IN=FILEB);
BY PROVID;
IF TESTSPEC = 'Y' AND TESTPROV = 'N'
    THEN IF FILEA;
ELSE IF TESTSPEC = 'N'
        AND TESTPROV = 'Y'
    THEN IF FILEB;
ELSE IF TESTSPEC = 'Y'
        AND TESTPROV = 'Y'
    THEN IF FILEA AND FILEB;
RUN;
```

Write Parameters to a Permanent File

If the parameters are going to be used in a series of programs they can be written to a permanent file to be read by the other programs.

Write parameters to a perm file:

```
DATA PERM.PARMS(KEEP=LINE2 AMTPD);
        AMT_PD = &AMTPD;
STDATE = (PUT,&PAID_BEG,MMDDYY10.);
EDDATE = (PUT,&PAID_END,MMDDYY10.);
LINE2 = 'PAID DATES '||STDATE||
    ' TO '||EDDATE;
RUN;
```

To retrieve and use parameters in another program:

```
DATA _NULL_;
    SET PERM.PARMS;
    CALL SYMPUT('RTITLE2',
        PUT(LINE2,$35.));
    CALL SYMPUT('AMTPAID',
        PUT(AMT_PD,2.0));
RUN;
```

To use parameters:

```
PROC REPORT;
TITLE1 ' REPORT TITLE';
TITLE2 "&RTITLE2";
TITLE3 "PAID OVER &AMTPAID";
COLUMN....
```

Conclusion

These examples illustrate a variety of uses for parameter variables. The examples have included selectively printing a report using a parameter, using a numeric parameter to subset data and include that parameter in the report title, and using dates to subset data and include the date parameters in the report title.

Character parameters can be used in a number of ways. An example shows how to set up a parameter with a combination of single values and a range of values. This parameter is used to subset data. The test flag example builds on the combination value example to illustrate how a test flag can be used to toggle the parameter testing on or off by using the test flag's value.

Parameters can even be used in MERGE statements. This is another usage of a test flag parameter. The final example describes how parameters can be written to a permanent file so the parameters can be used by another program. This is just a starting point for creating inventive and imaginative reports to suit your business needs.

For more information contact:

Karen Dudley
Blue Cross Blue Shield of New Mexico
505-237-5196
E-mail: kkdudle@bluemail.com

References

SAS Institute Inc. (1990), "SAS Language Reference, Version 6, First Edition," Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1990), "SAS Procedures Guide, Version 6, Third Edition," Cary, NC: SAS Institute Inc.