PharmaSUG2011 – Paper CC06

A SAS® Macro to Indent the un-indented SAS programs

Sreekanth Reddy Middela, Novartis Pharmaceuticals Corporation, East Hanover, New Jersey
Jyothsna Samala, Percept Pharma Services, Bridgewater, New Jersey
Anirudh Bhetala, Percept Pharma Services, Bridgewater, New Jersey

ABSTRACT

The ease of understanding any SAS program written by a programmer depends on parameters like Indentation of the SAS code, amount of commenting used and so on. Though, usage of proper comments in the program is highly essential, proper indentation also plays an equally important role to make the code look comfortable and unambiguous to understand in less time by the reviewer. This paper will provide an open code for any SAS programmer across domains, to use it as a tool to indent the un-indented program, thus making it reviewer-friendly.

INTRODUCTION

This macro was programmed out of necessary and I think lot of us would have faced the situation of reviewing and/or modifying SAS programs/macros written by some other programmer which are not formatted well and not indented well. This issue becomes serious when the program spans for couple of hundred pages especially if the programs being modified are standard macros.

I was caught in a similar situation when I was supposed to review and then modify a complex standard reporting macro which spanned for more than couple of hundred pages and I was in a real shock when I tried to debug and understand the program because the program was in free style i.e no proper commenting, no step boundaries, no indentation. To make the situation more interesting the program had many nested DO loops.

I had two options to proceed further to understand the program and modify as required. First one, being manually editing the program to indent the compound do loops and insert blank lines between each data step or new procedure or new macro definition or call, tedious and very time consuming. I opted for the second option to write a macro which would automatically indent and insert the appropriate blank lines etc .. for readability. Once the macro successfully performs indentation I should be able to review and understand the program easily when compared to reviewing the un-indented program. This macro helped me finish the review and modification of the program in much less time than expected.

FUNCTIONALITY OF %INDENT MACRO

- Incorporate 3 blank lines between the step boundary and the new Data step or new PROC or new Macro
- Incorporate 1 blank lines between the DO statement and the previous statement
- Incorporate 2 blank lines between the END statement and the next statement
- Insert 4 blanks spaces for the body of any DO Block
- Insert comment to the END statement for additional readability to explain the user which DOES loop or which IF condition this END statement corresponds to

FUTURE ENHANCEMENTS

- Indent macro can be updated to read any program and list all the local and global macros that are being used in any program
- All the keywords and variables can be upcased

%INDENT MACRO ALGORITHM

Last in First Out (LIFO) algorithm is used for the counters. Counter gets incremented for each new DO Loop and the corresponding DO Block gets shifted by 4 spaces.

EXAMPLE TO THE%INDENT MACRO

The following un-indented code from the input program needs to be fed into the %INDENT Macro and the sample partial input code is as follows

INPUT TO THE %INDENT MACRO

```
data new ;
set old ;
if a > 1 then do ;
result = 'a > 1' ;
output ;
end ;
if b > 3 then do ;
result = 'b > 3' ;
output ;
end ;
do while (x>100) ;
result = 'x > 100' ;
output ;
if b > 30 then do ;
result = 'b > 30' ;
if b > 300 then do ;
result = 'b > 300' ;
output ;
end ;
output ;
end ;
end ;
run ;
```

%INDENT SAMPLE MACRO CALL

```
%indent(pgm_in=%str(C:\Documents and Settings\sree\My
Documents\Papers\INDENT\test.sas),
        pgm_out=%str(C:\Documents and Settings\sree\My
Documents\Papers\INDENT\test_out.sas)) ;
```

OUTPUT TO THE %INDENT MACRO

```
data new ;
   set old ;

   if a > 1 then do ;
       result = 'a > 1' ;
       output ;
   end ; /*if a > 1 then do ; */


   if b > 3 then do ;
       result = 'b > 3' ;
       output ;
   end ; /*if b > 3 then do ; */


   do while (x>100) ;
       result = 'x > 100' ;
       output ;

       if b > 30 then do ;
           result = 'b > 30' ;
```

```
            if b > 300 then do ;
                result = 'b > 300' ;
                output ;
            end ; /*if b > 300 then do ;*/

            output ;
        end ; /*if b > 30 then do ; */

    end ; /*do while (x>100) ; */

    run ;
```

%INDENT MACRO CODE

```
/*************************************************************************************
*************
PROTOCOL NAME   : XXXXXXXXX
PROGRAM NAME    : INDENT.SAS
AUTHOR          : Sreekanth Middela
DATE            : 20OCT2010
SAS VERSION     : V.8.2
OPERATING
SYSTEM          : Win XP
PURPOSE         : This macro indents an un-indented program
INPUT MACRO PARAMETERS :
PGM_IN          : Input program name that needs to be indented along with the
path and extension
PGM_OUT         : Output program name that needs to be created along with the
path and extension
COMMENT :
Sample macro call :
%indent(pgm_in=%str(C:\Documents and Settings\jyothsna\My
Documents\Papers\INDENT\test.sas),
        pgm_out=%str(C:\Documents and Settings\jyothsna\My
Documents\Papers\INDENT\test_out.sas)) ;

*************************************************************************************
************/;


%macro indent(pgm_in=,pgm_out=) ;

%let name1 = &pgm_in. ;
%let name2=  &pgm_out. ;


*********************************
Read the input program
Create counters for DO loops
For every DO loop start the flag doend set to 1
For every END (End Of Do Loop) doend flag is set 2
For everything else doend flag is set to missing
*********************************;
data indent1 (keep = name name1 doend);
  infile "&name1"   truncover ;
  input name $ 300. ;
  name1 = name ;
  name = upcase(left(trim(name))) ;

  i = 1 ;
  do while(scan(name,i) ne ' '  ) ;
        temp = scan(name,i) ;
        if temp = 'DO' or temp = '%DO' then  doend = 1 ;
```

```
        i = i + 1 ;
  end ;

  i = 1 ;
  do while(scan(name,i) ne ' ' ) ;
        temp = scan(name,i) ;
        if temp = 'END' or temp = '%END' then  doend = 2 ;
        i = i + 1 ;
  end ;

run ;

/*title 'Only doend flags' ;*/
/*proc print ;*/
/*  run;*/


**********************************
Creating the do loop counter docnt,

Line value is also set. If a DATA or PROC or MACRO stats the ln value is set to
1
  otherwise ln value is set to 4 (pushing this code 4 spaces inside)


  docnt = 1 from 1st do loop start to until it reaches 1 2nd do loop. Once it
reaches the 2nd do loop docnt value is
  incremented by 1 and so on .. Once the nth do loop starts docnt value will be
n.
  When the nth do loop ends with an END statement the value of docnt is
decremented by 1 and so on ..

**********************************;

data indent2 ;
  set indent1 ;

    if indexw(upcase(name),'DATA') or indexw(upcase(name),'PROC') or
       indexw(upcase(name),'%MACRO')  then do ;
       dpm = 1 ;
       ln = 1 ;
       end ;
    else ln = 4 ;


    if doend = 1 then  do ;
       docnt+1 ;
    end ;


    if doend = 2 then  do ;
       docnt = docnt - 1 ;
    end ;

run ;

/*title 'Only docnt (do loop counters) doend flag and ln values' ;*/
/*proc print ;*/
/*  var name docnt doend ln;*/
/*  run;*/


**********************
```

```
As the start of DO loop needs to start from the same line as previous
statement,
  we need to re-adjust the do loop counter docnt to docnt - 1 for the do loop
start statement
Adjusting the counters
**********************;
data indent3 ;
  set indent2 ;

 if doend = 1 then  do ;
    docnt = docnt - 1 ;
 end ;

 run ;


/*title 'Do loop counters re-adjusted for the start of each DO loop' ;*/
/*proc print ;*/
/*  run;*/

***********************
*** LOCF the docnt values to docnt2 for the purpose of creating segments,
segmnt variable  ***;
***********************;
data indent4 ;
  set indent3 ;
  docnt2 = lag(docnt) ;
  run ;

/*title 'Do loop counters re-adjusted for the start of each DO loop' ;*/
/*proc print ;*/
/*  run;*/


***********************
*** Segments
*** Segment gets a value of 1 until it reached the begining of the Do loop.
    from the next observation onwards Segment's value is incremented by 1 until
a next DO loop or END statement is encounted.
    and then from the next observation onwards Segment's value is incremented
by 1 until a next DO loop or END
    and so on ... So now the input program is devided into segments along with
docnt and ln value ***;
***********************;
data indent5 ;
  retain segmnt  ;
  set indent4 ;

  if _n_ = 1 then segmnt = 1 ;

  else do ;
      if docnt - docnt2 ne 0 then segmnt + 1 ;
  end ;

  recnum = _n_ ;

  if docnt < 0 then docnt = 0 ;


  run ;

/*title 'Do loop counters re-adjusted for the start of each DO loop' ;*/
/*proc print ;*/
```

```
/*  run;*/


**********************
End of - Adjusting the counters
**********************;


proc sort data = indent5 ;
  by segmnt recnum ;
  run ;

proc sql noprint ;
  select max(docnt) into :num
  from indent5 ;
  quit ;

%let num = &num ;
%let num1 = %eval(&num + 1) ;
%put num1 = &num1 ;

*********************************
Implmenting the LIFO Algorithm
LIFO - Last In First Out
*********************************;

*********************************;
*** Attach the automatically generated Comment(begining of the Do Loop) along
with the END statement ***;
*** Remember the last.segmnt has DO statements and first.segmnt has END
statements ***;
*** at the begining of every DO loop, assign the compleet statement to the
Array(docnt+1), in order to
    auto generate the comment at the end of the Do loop i.e at the END statment
***;
*********************************;

data indent6 (drop = r:) ;
  length  r0 - r&num. temp $ 20 ;
  set indent5 ;
  array  r(&num1.) $ r0 - r&num. ;
  retain r:  ;
  by segmnt ;

  if _n_ ne 1 then do ;

      if last.segmnt then do ;

          i = 1 ;
          do while(scan(name,i) ne ' '  ) ;
                 temp = scan(name,i) ;
                 if temp = 'DO' or temp = '%DO' then  doend = 1 ;
                 i = i + 1 ;
          end ;

          if doend = 1 then do ;
               r(docnt+1) = name1 ;
               *put _all_ ;
          end ;

      end ;

      if first.segmnt then do ;
```

```
          i = 1 ;
          do while(scan(name,i) ne ' '  ) ;
                 temp = scan(name,i) ;
                 if temp = 'END' or temp = '%END' then  doend = 2 ;
                 i = i + 1 ;
          end ;

          temp = r(docnt+1 ) ;
          if doend = 2 then do ;
               name1 = compbl (name1 || '/*' || temp  || '*/')  ;
               *put _all_ ;
               end ;

      end ;

  end ;

  run ;


/*title 'Auto Generate Comments' ;*/
/*proc print ;*/
/*  run;*/

*********************************;
*** using the docnt and the ln values to determine the line number values to
write the output file ***;
*********************************;
data indent7 ;
  set indent6 ;
  ln = ln +  4 * docnt ;
  run ;


/**proc print ;*/
/*  run;*/


***********************
Writing the Output file
***********************;


data _null_ ;
  set indent7 ;
 * file print ;
  file "&name2" ;
  if doend = 1 or dpm = 1 then do ;
    if dpm = 1 and _n_ ne 1 then do ;
        put ;
        put ;
    end ;
    put ;
    put @ln  name1 ;
  end ;

else if doend = 2 then do ;
    put @ln  name1 ;
    put ;
end ;

else put @ln  name1 ;
```

```
  run ;

%mend indent ;
```

%INDENT MACRO CALL

```
%indent(pgm_in=%str(C:\Documents and Settings\Jyo\My
Documents\Sree\Papers\INDENT\21022011\test.sas),
        pgm_out=%str(C:\Documents and Settings\Jyo\My
Documents\Sree\Papers\INDENT\21022011\test_out.sas)) ;
```

CONCLUSION

The Indent macro was designed on SAS v8.2. As expected the macro meets the expected functionality to indent the un-indented program automatically and inserting blank lines where needed thus not only improving the readability but also the saving the time.

CONTACT INFORMATION:

Your comments and suggestions are valued and encouraged.
Contact the authors at:

Author Name : Sreekanth Middela
Company : Novartis Pharmaceuticals
Address : One Health Plaza
City state ZIP :East Hanover NJ 07936
Work Phone : 862-778-3006
E-mail : sreekanth.middela@novartis.com

Author Name : Jyothsna Samala
Company : Percept Pharma Services
Address :991 US Highway 22, Suite 200
City state ZIP : Bridgewater, NJ , 08807
Work Phone : 908-731-5357
Fax : 866-591-4635
E-mail : jyothsna.samala@perceptservices.com

Author Name : Anirudh Bhetala
Company : Percept Pharma Services
Address :991 US Highway 22, Suite 200
City state ZIP : Bridgewater, NJ , 08807
Work Phone : 908-731-5357
Fax : 866-591-4635
E-mail : anirudh.bhetala@perceptservices.com