Avoiding the "Ooh Nasty"

David Franklin, TheProgrammersCabin.com, Litchfield, New Hampshire, USA

ABSTRACT

Typically, when you build a small garden shed you should first do some planning, then do construction, and finally look it over and check what you have done and that it is going to do as intended. A similar process is used for writing programs – we should first do some planning, construct the program, and check it over to see that it producing what was asked. This paper takes a brief and lighthearted look at each of these stages, providing a few tips for avoiding some of the many pitfalls and gives a few pieces of SAS® code that are useful in the development of your program, hopefully avoiding having to say that dreaded phrase, "Ooh Nasty".

INTRODUCTION

When beginning to write SAS code there are three stages that are carried out:

- planning
- writing the program(s)
- testing and validating

This presentation will present some useful ideas in these three stages, hopefully to avoid any programming pitfalls. Also presented are two useful SAS macros that anyone understanding the data or programming will find useful.

PLANNING

There are five documents that you must have access to before programming. These are:

- General programming guidance and/or SOPs
- Study Protocol
- Study SAP
- Study Output Shells, e.g. table shells
- CRF (would be nice if this was annotated)

Also helpful is to do a PROC CONTENTS of any datasets. A macro is listed in Appendix A that produces an RTF document, combining a CONTENTS listing with a formats decode, an example shown below:

DM: DM Dataset Variables				
Variable Name	**Label**	**Type**	**Format**	**Comment**
STUDYID	Study Identifier	C20		
PATID	Patient ID	C8		
BRTH_DT	Date of Birth	C10		ISO date format.
AGE	Age of patient at randomization (CRF)	N8		
SEX	Gender	N3	(Format: SEX) 1=Male 2=Female	
RACE	Race	N3	(Format: RACE) 1=White 2=Black 3=Hispanic 4=Asian 9=Other	
RACE_OTH	Race, Other Specification	C200		
RAND_DT	Date of Randomization	N8		
AGE_CALC	Age of patient at randomization (DRVD)	N8		=FLOOR((RAND_DT – BRTH_DT)/365.25)

One of the features of the macro is that it is able to incorporate any comments to variables that the user may wish to add, an example of which is shown above for the AGE_CALC variable.

Another useful tip is to print out a few observations from each dataset using code similar to that given below:

```
proc sql noprint;
   select memname into :dslist separated by ' '
      from sashelp.vmember
      where libname in('RAWDATA');
   quit;
run;
%let i=1;
%do %while(%scan(&dslist,&i) ne );
   title1 "Dataset: %scan(&dslist,&i)";
   proc print data=rawdata.%scan(&dslist,&i) obs=5;
   run;
   %let i=%eval(&i+1);
%end;
```

Also consider using macros for “blocks” tables - if a group of tables is different only by a population and/or a variable cut then use a macro. A good example of this is the AE tables where they are the same layout and use the same code for counting but only differ in the variable used to select the Adverse Events for the count and/or a population. Another example is if the Demographics table is generated for Safety and ITT populations – the code is the same, the only difference is in the population selection. The advantages of using macros are:

- only one set of code needs to be written
- change is the difference in one or more parameter values
- only one set of code needs to be maintained
- only one set of code needs to be QC'ed

WRITING THE PROGRAM(S)

NO HARDCODES

The title says it all. A hardcode maybe obvious, for example:

```
if patid='001001' then height=178;
```

or less obvious

```
if stop_y>year("&sysdate9"d) then stop_y=year("&sysdate9"d);
```

If hardcodes are requested try to talk the requester out of it as it creates an integrity issue with the data. If hardcodes are still necessary then this should be indicated in the log with a very clear message. One good way of doing a hardcode is given in the following example:

```
if pt='001001' then do;
   if birthdt='25MAY2007'd then do;
      birthdt='25MAY1970'd;
      put 'WAR' 'NING: Hardcode for patient 001001 to correct birth date issue.';
   end;
   else put 'WAR' 'NING: Hardcode for patient 001001 should be reviewed as DOB '
             'in DM database has changed';
end;
```

Occasionally programmers will use a “time limited hardcode”, as the following example shows:

```
if pt='001001' and date()<="31OCT2007"d then birthdt='25MAY1970'd;
```

Unfortunately this type of hardcode is used far too often and does not allow for any data point change to reflect in the database, for example, what if the BIRTHDT value was corrected to 25MAY1971 before a run date of 29OCT2007? No hardcodes should exist in the final version of a table, listing or figure. A hardcode is different from a conversion as is done in vital signs and lab data, just to name a few.

PROGRAM TO THE CRF

The structure of the CRF is useful when writing the code. There are three types of variable – database, monitoring and data variables. Although eCRF has helped in trapping database errors don't assume that it works every time.

Try to use data variables where ever possible. An example of a database variable is

```
Record ID
```

and a monitoring variable is

```
Are there any Adverse Events? (Y/N)
```

and a data variable is

```
Adverse Event Description
```

ASSUME INVALID AND DUPLICATE DATA

Invalid or Duplicate data is something that should be considered when writing SAS code. If there are multiple records possible then discuss with a statistician which record to take for analysis and make a note of it in the program as a comment. The following code fragment shows the use for a check of duplicates and invalid data:

```
if n(hghtval) then do;
  select(hghtunt);
    when('CM') hghtsi=hghtval;
    when('IN') hghtsi=hghtval*2.54;
    otherwise
     put 'WAR' 'NING: Unexpected or unknown '
         'height unit: ' patid= hghtval= hghtunt=;
   end;
   if sum(first.patid,last.patid)<2 then
    put 'WAR' 'NING: Unexpected multiple result: '
         patid= hghtval= hghtunt=;
end;
```

If the variable being analyzed is a numeric but the value is stored as a character it is best to do any comparisons using numeric values. This can be best shown in the following example:

```
if ^missing(labvalue) and labvalue^='0';
```

This line of code is okay when zero means '0' but during the life of a database a value of '0.00' may included that will not be caught.

Do not be afraid to use the log or PRINT procedure to output data considered dubious, the latter usually going to an LST file with the same program name.

DATES

This is a special category of its own. If a full date is expected for a particular variable, e.g. Date of Birth, Treatment Date or Concomitant Medication date, then treat it as a full date and put out an exception to a log or LST file if a partial date is given. Only try to write code to deal with partial dates if partial dates are expected.

AVOID OVERWRITING DB VARIABLES

It is not good programming practice to write over data in database variables, even within a program where a temporary (work) dataset is being used. It makes it harder to track the flow of data when data from the stored dataset is being overwritten in the work area.

INITIALIZE VARIABLES

At the very least it is good programming practice to define the type and length of a new variable in a data step. This avoids the issue where SAS will assume a type and length of a newly created variable while the programmer may want another type and/or length.

USE COMMENTS

This makes the program easier to understand when someone else other than yourself is looking at the program – that program may be looked at again in five years time!

USE A PROGRAM HEADER

It is good practice to use a program header. From an informational view, the two most important fields in any program header are “WHY” and “NOTES” as the former say why the program was written the latter gives any

information that shows any particular oddities that may be needed by anyone looking at the program in the future. Even if you are just changing a line of code for whatever reason, keep a NOTES field up to date.

DON'T BE AFRAID TO USE THE HELP BUTTON

If you are not sure of a SAS statement, function or procedure do click the help button or ask someone. To use an example

```
select trtcd, count(*) as count
```

is different to

```
select trtcd, count(age) as count
```

The first line will count the number of observations in the dataset, but the second will count the number of observations with non-missing values.

TESTING AND VALIDATING

CHECK THE LOG

This cannot be emphasized enough. Attached in Appendix B is an example of a small QC macro that goes through the SAS Logs specified to check if there are programming issues that need resolution. As a general rule nothing should be found, however there may be instances where this may not be possible but these should be explained somewhere.

REVIEW THE OUTPUT

Just have a quick look at the output and check if the numbers are as expected (n should always be less than or equal to N), that page breaks are okay and see that any statistical results are around what is anticipated. Maybe consider doing a quick PROC TABULATE to check that nothing was lost in the code to make the data useful for the REPORT procedure call. It is good practice to spell check the titles and footnotes, and make sure that footnote references actually align with references in the output.

CONCLUSION

It is not possible to eliminate all problems when writing programs for clinical output, but with some thought it is possible to prevent the majority of them. This paper has presented some ideas for achieving this goal so that you won't have say “Ooh Nasty” quite so often.

ACKNOWLEDGMENTS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

CONTACT

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

Author Name:	David Franklin
Company:	TheProgrammersCabin.com
Work Phone:	603-275-6809
Email:	dfranklin@TheProgrammersCabin.com
Web:	TheProgrammersCabin.com

APPENDIX A: CONTLST MACRO – DESCRIBING YOUR DATA

```
/*-----------------------------------------------------------------------------------------------
* PROGRAM NAME: ContLst.SAS
* DESCRIPTION: Create a document combining variable attributes for all variables within all
* datasets within a directory, list of format decodes associated with each format,
* and comments text that is user definable. Output is an RTF file.
*
* LOCATION: \MacroLib
*
* INPUTS: _datlib Data Library, REQUIRED
* _fmtlib Format Catalog
* _comfil Comments File
* OUTPUTS: _outfil Directory location and name of output file. File extension should be
* either RTF or DOC. REQUIRED
*
* NOTES: %ContLst(_datlib= , _outfil= <,_fmtlib=> <,_comfil=>);
*
* Parameters _DATLIB and _OUTFIL are required.
* In this release no check is made to determine that the directory referenced in
* _DATLIB exists, datasets exist with variables in _DATLIB, format catalog or
* comments file exist.
* If a format decode is longer than 400 characters then a note will appear below
* the format stating that the format was too long to desplay - another way should
* be used to decode the format if it is to be viewed by the user.
* The comments file has the following structure:
*
* DATASET_NAME~VARIABLE_NAME~COMMENT
*
* The three parameters are seperated by the ~ symbol.
* If no match is made between members in datasets referenced by _DATLIB and
* entries in the comments file then the comment is ignored in the final output. A
* comment can be up to 2000 characters in length and can include RTF codes.
* If a comment is longer than 150 characters then it is converted to a footnote
* and seen in the file as such.
* An example of a line in the COMMENTS file is
*
* DM~AGE~AGE = FLOOR((RAND_DT - DOB_DT)/365.25)
*
* Example:
* libname rawdat "E:\clin\tuai\data";
* libname library "E:\clin\tuai\formats";
* %ContLst(_datlib=RAWDAT,
* _fmtlib=LIBRARY.FORMATS,
* _comfil="E:\clin\tuai\docs\COMMENTS.TXT",
* _outfil="E:\clin\tuai\docs\DEFINE.RTF",);
*
* Call will create a description of the data in the library RAWDAT, decoding the
* formats using the format catalog FORMATS in the library LIBRARY, using comments
* from COMMENTS.TXT, generating document DEFINE.RTF.
*
* HISTORY
* WHO/WHEN: Original, David Franklin, 20-Oct-2007
* WHY: Based on production version 2.1 in MacroLib, for NESUG 2012 presentation.
* Removed checks for existence of inputs and other data driven checks, multiple
* formats, and checks for existence of inputs, paper and font size options for
* space limitation reasons.
*-----------------------------------------------------------------------------------------------*/
%macro ContLst(_datlib=WORK /*Data Library - REQUIRED*/
              ,_outfil= /*Location and Name of Output File - REQUIRED*/
              ,_fmtlib= /*Format Library*/
              ,_comfil= /*Location and Name of Comments File*/
              );

options nofmterr nobyline;

*----------------------------;
*Get contents of datasets;
*----------------------------;
proc contents data=&_datalib.._all_ noprint
              out=dat0 (keep=libname memname memlabel name type length varnum label format);
run;
data dat1;
   set dat0;
```

```
   name=upcase(name);
   format=upcase(format);
   if format='$' then format='';
run;

*----------------------------;
*Get formats;
*----------------------------;
%if (&_fmtlib ne ) %then %do;
   proc format library=&_fmtlib cntlout=fmt0;
   run;
   data fmt1;
      retain lastfmt fmtlistn fmtlistl fmtlistf;
      length lastfmt fmtlistn $9 fmtlistl code $200 fmtlistf 3;
      set fmt0 end=eof;
      keep fmtlistl fmtlistn;
      if _n_=1 or lastfmt^=fmtname then do;
         if _n_>1 then output;
         lastfmt=fmtname;
         fmtlistf=0;
      end;
      if lastfmt=fmtname then do;
         if missing(fmtlistn) then do;
            if type='C' then fmtlistn='$'||trim(left(fmtname));
            else fmtlistn=trim(left(fmtname));
         end;
         if fmtlistf=0 then do;
            if (verify(start,end)=0) then code=cats(start,'=',label);
            else code=cats(start,'-',end,'=',label);
            if missing(fmtlistl) then do;
               if length(code)>200 then do;
                  fmtlistf=1;
                  fmtlistl='Format list too long to display';
               end;
               else fmtlistl=cats(code);
            end;
            else do;
               if sum(length(fmtlistl),length(code),1)>200 then do;
                  fmtlistf=1;
                  fmtlistl='Format list too long to display';
               end;
               else fmtlistl=cats(fmtlistl)||'\line '||cats(code);
            end;
         end;
      end;
      if eof then output;
   run;
   proc sort data=dat1;
      by format;
   run;
   proc sort data=fmt1;
      by fmtlistn;
   run;
   data dat1;
      merge dat1 (in=a) fmt1 (rename=(fmtlistn=format));
      by format;
      if a;
   run;
%end;

*----------------------------;
*Get comments;
*----------------------------;
%if (&_comfil ne ) %then %do;
   data usrcom;
      length memname $40 name $40 comment $2000;
      infile "&_comfil" length=len lrecl=2082;
      input txt $varying2082. len;
      memname=upcase(scan(txt,1,"~"));
      name=upcase(scan(txt,2,"~"));
      comment=scan(txt,3,"~");
      drop txt;
      if sum(missing(memname),missing(name),missing(comment))=0;
   run;
   proc sort data=dat1;
```

```
      by memname name;
   run;
   proc sort data=usrcom nodupkey;
      by memname name;
   run;
   data dat1;
      merge dat1 (in=a) usrcom;
      by memname name;
      if a;
   run;
%end;

*----------------------------;
*Get data together for report;
*----------------------------;
data dat2;
   attrib comment length=$2000
   fmtlistl length=$200;
   set dat1;
   length desc $200 vtype $5 vfmt $250;
   if ^missing(memlabel) then desc=catt(memname)||': '||catt(memlabel);
   else desc=catt(memname)||': '||catt(memname)||' Dataset Variables';
   if type=1 then vtype=compress('N'||put(length,5.));
   else if type=2 then vtype=compress('C'||put(length,5.));
   if ^missing(format) then do;
      if missing(fmtlistl) then vfmt=catt('(Format:',format,')');
      else vfmt=catt('(Format:',format,')\line',fmtlistl);
   end;
run;

*----------------------------;
*Generate the report;
*----------------------------;
proc sort data=dat3;
   by desc varnum;
run;
data _null_;
   file "&_outfil" LRECL=1048576;
   set all1 end=eof;
   by dat3;
   if _n_=1 then do;
      put "{\rtf1\ansi\ansicpg1252\uc1";
      put "\deff0\deflang1033\deflangfe1033{\fonttbl{\f0\froman\fcharset0" @;
      put "\fprq2{\*\panose 02020603050405020304}Times New Roman;}}";
      put "{\colortbl;\red0\green0\blue0;\red0\green0\blue255;\red0" @;
      put "\green255\blue255;\red0\green255\blue0;\red255\green0\blue255;" @;
      put "\red255\green0\blue0;\red255\green255\blue0;\red255\green255" @;
      put "\blue255;\red0\green0\blue128;\red0\green128\blue128;" @;
      put "\red0\green128\blue0;\red128\green0\blue128;\red128\green0" @;
      put "\blue0;\red128\green128\blue0;\red128\green128\blue128;\red192\green192\blue192;}";
      put "{\stylesheet{\widctlpar\adjustright \fs20\cgrid \snext0 " @;
      put "Normal;}{\*\cs10 \additive Default Paragraph Font;}" @;
      put "{\s15\widctlpar\tqc\tx4320\tqr\tx8640\adjustright \fs20" @;
      put "\cgrid \sbasedon0 \snext15 header;}" @;
      put "{\s16\widctlpar\tqc\tx4320\tqr\tx8640\adjustright \fs20" @;
      put "\cgrid \sbasedon0 \snext16 footer;}" @;
      put "{\*\cs17 \additive \sbasedon10 page number;}}\margl1440" @;
      put "\margr1440 \widowctrl\ftnbj\aenddoc\hyphcaps0\formshade" @;
      put "\viewkind4\viewscale92\viewzk2\pgbrdrhead\pgbrdrfoot \fet0" @;
      put "\sectd \psz1\linex0\headery709\footery709\colsx709\endnhere\sectdefaultcl";
      put "{\header \pard\plain \s15\widctlpar\brdrb\brdrs\brdrw10" @;
      put "\brsp20 \tqc\tx4680\tqr\tx9360\adjustright \fs20\cgrid " @;
      put "{\b\lang2057 Vertex\tab Data Definitions\tab &_prot_id\par }}";
      put "{\footer \pard\plain \s16\widctlpar\brdrt\brdrs\brdrw10\brsp20 \tqc" @;
      put "\tx4680\tqr\tx9360\adjustright \fs20\cgrid " @;
      put "{\b\lang2057 %upcase(&_fname)\tab Page }{\field{\*\fldinst {\cs17" @;
      put "\b PAGE }}{\fldrslt {\cs17\b\lang1024 1}}}{\cs17\b of }" @;
      put "{\field\flddirty{\*\fldinst {\cs17\b NUMPAGES }}{\fldrslt {\cs17\b" @;
      put "\lang1024 1}}}{\cs17\b \tab \chdpl}{\b\lang2057 \par }}";
      put "{\fet \ftnbj \ftnstart}";
      put "{\*\pnseclvl1\pnucrm\pnstart1\pnindent720\pnhang{\pntxta .}}";
      put "{\*\pnseclvl2\pnucltr\pnstart1\pnindent720\pnhang{\pntxta .}}";
      put "{\*\pnseclvl3\pndec\pnstart1\pnindent720\pnhang{\pntxta .}}";
      put "{\*\pnseclvl4\pnlcltr\pnstart1\pnindent720\pnhang{\pntxta )}}";
      put "{\*\pnseclvl5\pndec\pnstart1\pnindent720\pnhang{\pntxtb ({\pntxta )}}";
```

```
      put "{\*\pnseclvl6\pnlcltr\pnstart1\pnindent720\pnhang{\pntxtb (}{\pntxta )}}";
      put "{\*\pnseclvl7\pnlcrm\pnstart1\pnindent720\pnhang{\pntxtb (}{\pntxta )}}";
      put "{\*\pnseclvl8\pnlcltr\pnstart1\pnindent720\pnhang{\pntxtb (}{\pntxta )}}";
      put "{\*\pnseclvl9\pnlcrm\pnstart1\pnindent720\pnhang{\pntxtb (}{\pntxta )}}";
      put "\pard\plain \widctlpar\adjustright \fs20\cgrid {\lang2057 \par }";
      put "\ul\b\fs24 Table of Contents\fs20\b0\ulnone\par";
   end;
   if first.desc then put @1 "\par " desc;
run;
data _null_;
   file "&_outfil" LRECL=1048576 mod;
   set all1 end=eof;
   by desc;
   if first.desc then do;
      put "\page\par";
      put "\trowd \trgaph108\trleft-108\trkeep\trhdr\trbrdrt\brdrs\brdrw10 \trbrdrl" @;
      put "\brdrs\brdrw10 \trbrdrb\brdrs\brdrw10 \trbrdrr\brdrs\brdrw10 " @;
      put "\trbrdrh\brdrs\brdrw10 \trbrdrv\brdrs\brdrw10 \clvertalt\clbrdrt" @;
      put "\brdrs\brdrw10 \clbrdrl\brdrs\brdrw10 \clbrdrr\brdrs\brdrw10 \cltxlrtb " @;
      put "\cellx9360\pard \widctlpar\intbl\adjustright {\b\lang2057 " desc "\cell }" @;
      put "\pard \widctlpar\intbl\adjustright " @;
      put "{\lang2057 \row }";
      put "\trowd \trgaph108\trleft-108\trhdr\trbrdrt\brdrs\brdrw10 \trbrdrl\brdrs" @;
      put "\brdrw10 \trbrdrb\brdrs\brdrw10 \trbrdrr\brdrs\brdrw10 " @;
      put "\trbrdrh\brdrs\brdrw10 \trbrdrv\brdrs\brdrw10 \clvertalt\clbrdrt\brdrs" @;
      put "\brdrw10 \clbrdrl\brdrs\brdrw10 \clbrdrb\brdrs\brdrw45 " @;
      put "\clbrdrr\brdrs\brdrw10 \cltxlrtb \cellx1320\clvertalt\clbrdrt\brdrs" @;
      put "\brdrw10 \clbrdrl\brdrs\brdrw10 \clbrdrb\brdrs\brdrw45 " @;
      put "\clbrdrr\brdrs\brdrw10 \cltxlrtb \cellx3600\clvertalt\clbrdrt\brdrs" @
      put "\brdrw10 \clbrdrl\brdrs\brdrw10 \clbrdrb\brdrs\brdrw45 " @;
      put "\clbrdrr\brdrs\brdrw10 \cltxlrtb \cellx4320\clvertalt\clbrdrt\brdrs" @;
      put "\brdrw10 \clbrdrl\brdrs\brdrw10 \clbrdrb\brdrs\brdrw45 " @;
      put "\clbrdrr\brdrs\brdrw10 \cltxlrtb \cellx6650\clvertalt\clbrdrt\brdrs" @;
      put "\brdrw10 \clbrdrl\brdrs\brdrw10 \clbrdrb\brdrs\brdrw45 " @;
      put "\clbrdrr\brdrs\brdrw10 \cltxlrtb \cellx9360\pard \widctlpar\intbl" @;
      put "\adjustright {\b\lang2057 Variable Name\cell Label\cell Type" @;
      put "\cell Format\cell Comment\cell }\pard \widctlpar\intbl\adjustright {\lang2057 \row }";
   end;
   put "\trowd \trgaph108\trleft-108\trkeep\trbrdrt\brdrs\brdrw10 \trbrdrl" @;
   put "\brdrs\brdrw10 \trbrdrb\brdrs\brdrw10 \trbrdrr\brdrs\brdrw10 " @;
   put "\trbrdrh\brdrs\brdrw10 \trbrdrv\brdrs\brdrw10 \clvertalt\clbrdrl\brdrs" @;
   put "\brdrw10 \clbrdrb\brdrs\brdrw10 \clbrdrr\brdrs\brdrw10 " @;
   put "\cltxlrtb \cellx1320\clvertalt\clbrdrl\brdrs\brdrw10 \clbrdrb\brdrs" @;
   put "\brdrw10 \clbrdrr\brdrs\brdrw10 \cltxlrtb " @;
   put "\cellx3600\clvertalt\clbrdrl\brdrs\brdrw10 \clbrdrb\brdrs\brdrw10 \clbrdrr" @;
   put "\brdrs\brdrw10 \cltxlrtb \cellx4320\clvertalt\clbrdrl\brdrs\brdrw10 " @;
   put "\clbrdrb\brdrs\brdrw10 \clbrdrr\brdrs\brdrw10 \cltxlrtb \cellx6650\clvertalt" @;
   put "\clbrdrl\brdrs\brdrw10 \clbrdrb\brdrs\brdrw10 " @;
   put "\clbrdrr\brdrs\brdrw10 \cltxlrtb \cellx9360\pard \widctlpar\intbl\adjustright " @;
   put "{\lang2057 " name +(-1) "\cell " label +(-1) "\cell " vtype +(-1) "\cell " @;
   if ^missing(vfmt) then put vfmt @;
   put "\cell " @;
   if ^missing(comment) then do;
      if length(comment)>150 then
        put "{\super \chftn{\footnote {\super \chftn} \fs20 " comment "}}" @;
      else put comment @;
   end;
   put "\cell }\pard " @;
   put "\widctlpar\intbl\adjustright {\lang2057 \row }";
   if last.desc then put @1 "\pard \widctlpar\adjustright\par";
   if eof then put @1 "\par }";
run;
%mend ContLst;
%ContLst;
run;
/*End of File*/
```

APPENDIX B: QC MACRO – CHECK YOUR LOG(S) FOR ISSUES

```
/*-------------------------------------------------------------------------------------------
* PROGRAM NAME: QC.SAS
* DESCRIPTION: Check the logs or a specified selection of LOG files and put output to a single
* user defined file.
*
* LOCATION: \MacroLib
*
* INPUTS: _infil LOG Files to review. This may be a single file, a wildcard selection of
* files or a list of files. Examples are:
* "E:\CLIN\TUAI\Tables\DEMOG.LOG" , or
* "E:\CLIN\TUAI\Tables\D*.LOG", or
* "E:\CLIN\TUAI\Tables\DEMOG.LOG" "E:\CLIN\TUAI\Tables\VS.LOG
* of any combination or them. Note each item must be surrounded by quotes
* and a space my exist between them. REQUIRED
* OUTPUTS: _outfil Directory location and name of output file containing finds found. REQUIRED
*
* NOTES: %QC(_infil= , outfil= )
*
* Both parameters are required.
* In this release no check is made to determine that the directory or any files
* referenced are present.
*
* Example:
* %QC(_infil="E:\CLIN\TAUI\TABLES\ECG.LOG"
* "E:\CLIN\TAUI\TABLES\DM*.LOG",
* _outfil="E:\clin\tuai\TABLES\LOGQCFINDING.LOG");
* Call will create a description of the data in the library RAWDAT, decoding the
* formats using the format catalog FORMATS in the library LIBRARY, using comments
* from COMMENTS.TXT, generating document DEFINE.RTF.
*
* HISTORY
* WHO/WHEN: Original, David Franklin, 20-Oct-2007
* WHY: Based on production version 3.7 in MacroLib, for NESUG 2012 presentation.
* Removed all but three checks for space limitation reasons.
*-------------------------------------------------------------------------------------------*/
%macro qc(_infil= /*Log Files to Review*/
         ,_outfil= /*Outfile for QC*/
         );
   filename indat (&_infil);
   data fil0 (keep=fn) err0;
      length filename fn $200 linenum ln 8;
      infile indat length=len filename=fn line=ln;
      input txt $varying200. len;
      filename=fn;
      linenum=ln;
      if linenum=1 then output fil0;
      if (index(txt,'ERROR') and index(txt,'_ERROR_')=0) or
          index(txt,'WARNING') or
          index(txt,'character to numeric') or
          index(txt,'numeric to character') then output err0;
   run;
   proc sort data=fil0; by filename; run;
   proc sort data=err0; by filename linenum; run;
   data all0;
      merge fil0 (in=a) err0 (in=b);
      by filename;
      if a; if ^b then txt='No Issues Found';
   run;
   data _null_;
      file "&_outfil" lrecl=200;
      set all0 end=eof;
      by fn;
      if _n_=1 then put "LOG REPORT" / "Generated: &sysdate9.:&systime" /;
      if first.fn then
        put '************************************************' / "Log File:" +1 fn /;
      put txt $200.;
      if last.fn then put;
      if eof then
        put '************************************************' / "/*EOF*/";
   run;
%mend QC;
/*End of File*/
```