BUILDING AND USING MACRO VARIABLE LISTS

Clark Roberts, Decision Analytics

ABSTRACT

It is often necessary, in writing programs, to process the same logic over a set of values. Everyone is familiar with the purpose of ARRAYs and DO loops, to apply similar logic to several different variables without repeating the code for each one. The SAS® System provides several varieties of DO loop syntax, one in particular, the DO iterative syntax has several flavors, including DO index = list, which is especially useful if the values of the index variable are not contiguous. Unfortunately, this syntax is limited to data step processing and there’s no equivalent to this in the macro facility. This paper will discuss how to use the SAS Macro facility to emulate the DO list statement for processing a list of character variables that can provide an application with dynamic data driven capability. It will also discuss different ways that the macro variable lists can be constructed. Several examples will be presented to demonstrate the creation of the lists, including examples that use information from the SAS Help Dictionary views, and applications that use these lists to control process flow.

USING MACRO VARIABLE LISTS

The heart of the approach is the use of the %DO %WHILE construct combined with the %SCAN function. The basic syntax for the loop is:

```
%let i = 1;
%do %while(%scan(Amacclist,&i,%str( ) ) = %str( ) )
    %let var = %scan(Amacclist,&i,%str( ) )
    ...
    more SAS code
    ...
    %let i = %eval(&i + 1)
%end;
```

The ALLCAPS.SAS macro (figure 1) in the examples section demonstrates the use a macro variable list named CVARLIST in a %DO %WHILE loop to capitalize all the variables in a given SAS data set. The CVARLIST macro variable is built using the GETCVAR macro (figure 1) discussed in the following section.

BUILDING MACRO VARIABLE LISTS

There are several ways to build the macro variable list that can be used in the above loop structure:

1. Use a %LET statement to assign values to the list.
2. Assign values in a parameter of the macro.
3. Read the values from a file and dynamically build the list.

The third option provides the most flexibility and allows dynamic execution of the macro. An example of this is the GETCVAR.SAS (figure 2) in the examples below which accesses the SAS Dictionary Views in the SASHELP library to extract the names of all character variables, insert the names into separate local macro variables, and then create a global macro variable named CVARLIST which contains all the names in a space delimited list format.

EXAMPLES

FIGURE 1 - ALLCAPS.SAS

```
..........................................................
* FACILITY: *
* DECISION ANALYTICS *
* SYSTEM NAME:

```
* **CONVON SAS TOOLS LIBRARY**

* **PROGRAM:**
  * ALLCAYS.SAS

* **LANGUAGE/VERSION:**
  * SAS 6.08

* **DESCRIPTION:**
  * THIS MACRO CONVERTS ALL CHARACTER VARIABLES IN THE
    * &list1...&listn DATA SET TO UPPERCASE AND WRITING THE
    * CONVERTED RECORDS TO THE &list1...&listn DATA SET.
    * THE NAMES OF THE CHARACTER VARIABLES ARE OBTAINED
    * BY A CALL TO THE %UPPER MACRO, IF &list1 IS BLANK.
    * IF &list1 IS BLANK, THEY ARE SET TO THEIR
    * RESPECTIVE DATA TABLES, IF THE SAS LIBRARY
    * HAS NOT BEEN ALLOCATED, OR, IF THE SPECIFIED SAS DATA
    * SET IS EMPTY OR DOES NOT EXIST, THEN THE GLOBAL MACRO
    * %EVARS WILL RETURN A VALUE OF 0 (zero).

* **CALL BY:**
* [variable: program]

* **MACROS CALLED:**
  * GITCVAR -
    * RETURNS THE NAMES OF ALL CHARACTER VARIABLES IN A
      * &list1...&listn SAS DATA SET IN A MACROVARIABLE LIST
* **FILES READ:**
  * [none]

* **DATA SETS READ:**
  * &list1...&listn

* **FILES CREATED:**
  * [none]

* **DATA SETS CREATED:**
  * &list1...&listn

* **FILES INCLUDED:**
  * GITCVAR.SAS
    * Refer to definitions in MACROS CALLED section above

* **MACRO VARIABIES:**
  * [gvarname: calling program]

  * &list1 -
    * THE LIBRARY WHERE THE INPUT DATA SET RESIDES

  * &listn -
    * THE SAS DATA SET CONTAINING THE CHARACTER
      * VARIABLES TO BE CONVERTED TO UPPERCASE

  * &list1...&listn -
    * THE LIBRARY WHERE THE READING DATA SET
      * WILL BE WRITTEN TO, IF BLANK THEN
      * THE LIBRARY IN &list1...&listn WILL BE ASSIGNED

  * &list1...&listn -
    * THE OUTPUT DATA SET WHERE THE CONVERTED
      * DATA WILL BE WRITTEN, IF BLANK THEN
      * THE DATA SET SPECIFIED IN &list1...&listn WILL BE ASSIGNED

  * [internal]

  * &i -
    * USED AS AN INDEX FOR LOOKING

  * [glocal: used]

  * %evars -
    * THE NUMBER OF CHARACTER VARIABLES IN THE
      * INPUT DATA SET

  * %evarslist -
    * A LIST OF THE NAMES OF THE CHARACTER
      * VARIABLES

    * IN THE INPUT DATA SET

* **INTERNAL VARS:**

**REVISION HISTORY:**

  * V0.8.01 CLARK ROBERTS 13-FEB-1985
    * INITIAL VERSION

  * ************

  * INCLUDE REQUIRED FILES

  * ************

  * %include 'getcvt.sas';

  * &listn = work;
FIGURE 2 - GETCVAR.SAS

******************************************************************************
* FACILITY:                  DECISION ANALYTICS                        
* SYSTEM NAME:               COMMON SAS TOOLS LIBRARY                    
* PROGRAM:                   GETCVAR.SAS                                
* LANGUAGE/VERSION:          SAS 6.08                                  
* DESCRIPTION:               
  * THIS MACRO USES THE VOCUSLUM DATA DICTIONARY VIEW
  * TO EXTRACT THE NAMES OF ALL THE
  * GLOBAL VARIABLES IN THE SAS DATA SET GIVEN BY THE
  * E111 and Eds PARAMETERS PASSED TO GETCVAR. A GLOBAL
  * MACRO VARIABLE IS CREATED AS OUTPUT FROM THE PROCESS.
  * A SPACE DELIMITED LIST CALLED %overlist is populated
  * WITH THE NAMES OF THE CHARACTER VARIABLES.
  * CALLED BY:
  *  [varlist program]          
  * MACROS CALLED:
  *  [macro]                   
  * FILES READ:
  *  [files]                   
  * DATA SETS (SAS VIEWS) READ:
  *  VOCUSLUM: VOCUSLUM
  *  A SAS VIEW THAT CONTAINS INFORMATION DICTED TO THE
  *  VARIABLE LEVEL FOR EACH DATA SET IN EVERY SAS DATA LIBRARY
  *  THAT IS CURRENTLY BINDED WITH A LIBNAME
  * FILES CREATED:
  *  [files]                   
  * DATA SETS CREATED:
  *  [files]                   
  * FILES INCLUDED:
  *  [files]                   
  * MACRO VARIABLES:
  *  [varlist from calling program]
  * E111: THE NAME OF THE DATA LIBRARY THAT CONTAINS
  *       THE SAS DATA SET TO BE QUERIED
  * Eds: THE NAME OF THE SAS DATA SET WHERE THE NAMES
  *      OF THE CHARACTER VARIABLES WILL BE EXTRACTED FROM
  *  [list varlist]
  * &1: USED AS A LOOPING INDEX
  * &length: NUMBER OF CHARACTER VARIABLES IN THE DATA SET
  *  [globals created]
  * %overlist: A SPACE DELIMITED LIST CONTAINING THE NAMES
  *            OF THE CHARACTER VARIABLES IN THE DATA SET
  * REVISION HISTORY:
  *  06.00 CLARK ROBERTS
  *  INITIAL VERSION
******************************************************************************

%global overlist;
%local 1

%let overlist = ;
%let overlist = 

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% EXTRACT THE NAMES OF THE CHARACTER VARIABLES IN THE
%%% &Eds DATA SET FROM THE VOCUSLUM DATA DICTIONARY
%%% VIEW IN THE SASHelp LIBRARY.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

: data _all_;
  set sashelp.voccupm(where=(library = 'VOCUSLUM(Eds)' and
    type = 'DATA' and
    name(type) = 'C')
   )
  name = name;
  call spelect('overlist')(left(set,4.0),name);
  if nf then call spelect('overlist',set(0.0));
run;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% FOLULATE THE GLOBAL MACRO VARIABLE %overlist WITH THE
%%% NAMES OF THE CHARACTER VARIABLES EXTRACTED IN THE
%%% PREVIOUS STEP. DELIMIT THE ENTRIES WITH ONE BLANK SPACE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

: %do %while: %let overlist = %overlist %do |&overlist;&
  %end:

%end getover;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% END OF PROGRAM: GETCVAR.SAS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

;

FIGURE 3 - ALLCAPS2.SAS

%macro allcaps2(1111) = work ,
    id = ,
    still = ,
    outid =
    |
%if 1111 = work |
   %let overlist = %overlist | &work |
%else |
   %let overlist = %overlist | &id |
%end if;

%if 1111 = still |
   %let overlist = %overlist | &still |
%else |
   %let overlist = %overlist | &outid |
%end if;

%global overlist;

%let overlist = work
%
%do %while: %let overlist = %overlist %do |&overlist;&
  %end:
%end:

%end allcaps2;