Introduction

Macro statements start a percent sign (%).

Macro variables must be defined and given values before being used.

Macro variables are assigned values via a %let statement, use of a parameter in a macro definition and invocation, or via a call to a special routine named SYMPUT.

When assigning a value via SYMPUT, provide the variable name and value.

Macro variables start with an ampersand (&) when referenced.

The %STR function is used to add quotes to a value.

The %EVAL function is used to perform arithmetic operations.

There are many other special SAS macro functions.

When a macro variable is found, the value currently assigned to the variable is substituted.

If you wish to assign the value of a macro variable to a regular variable, the macro variable must be enclosed in double quotes ('') rather than single quotes/apostrophes (').

Macros must be defined before being used. This is done starting with a %macro *macroname*; statement and ended with a %mend *macroname* statement.

Macros can have no parameters, positional parameters, or keyword parameters.

Will examine seven different scenarios in which macro variables and macros have been found useful in reducing programming effort and errors. This will cover only the basics. The SAS macro language has many other features and functions.

Writing Re-Usable Programs

Problem: A program must be run each term by different users each of whom has a unique user name and password.

Solution: Use macro variables to provide values for parameters such as term, user name, and password.

Variable Output Based on Parameters

Problem: A program is run at the end of each fiscal year and displays five years of data. Depending on the number of occurrences of the fiscal year to be changed, this can be a time-consuming and error prone process.

Solution: Define the "base" fiscal year as a variable and calculate the "out" years via SAS macros.

Variable Output Based on Derived Values

Problem: With migration to a new student records system a change in terminology occurred and what was previously called the winter semester (WS) is now called the spring semester (SP). Enrollment data is presented on a term-by-term basis. The term names needed to be kept consistent. This could be changed manually. However this is time-consuming and error-prone.

Solution: Define a SAS macro to examine the year and term and provide the appropriate display value.

Variable-Size Arrays

Problem: Need to load an unknown number of observations into one or more arrays and use the arrays in subsequent data steps.

Solution: Input data and determine number of observations to be loaded from an external file; define arrays, load data, and save to a file; and use arrays in subsequent data steps.

Determine Length of Incoming Character Variables

Problem: An external organization requires using the descriptions they provide in data to be returned. One possible way to deal with this is to check and manually update the descriptions each time the data is requested.

Solution: Use SAS functions and macro variables to determine maximum length of the incoming variables and use the resulting information to define the variables in subsequent steps.

Using Program Data to Produce Additional SAS Code - %Include

Problem: Need to assign subjects and course numbers as labels to variables.

Solution: Use extracted data to generate SAS statements, save to an external file, and use in a subsequent data step.

Using Program Data to Produce Additional SAS Code - Macro Variables

Problem: Need to construct portions of an Oracle select clause based on a list of subjects and course numbers imported from a spreadsheet. The %include statement does not work with the SAS/Oracle pass-through interface.

Solution: Use macro variables to produce the statements and a macro to insert them in the Oracle select statement.

Writing Re-Usable Programs

```
/* Define macro variables */

%let username=myid;          /* Oracle id */
options nosource;            /* Suppress printing to log */
%let password=mypass;        /* Oracle password */
options source;              /* Resume printing to log */
%let term=FS2012;            /* Term value */
%let qterm=%str(%'&term%');  /* Add single quotes to the term */

/* List resolved values of macro variables */

data a;
  username="&username"; term="&term"; qterm="&qterm";
run;

proc print data=a;
  title 'Resolved Macro Variables';
run;

/* Retrieve data from Oracle */

proc sql;
connect to oracle as ora1 (user=&username orapw=&password
                  path="@tns:datawarehouse.myschool.edu");
create view termdataV as
select * from connection to ora1 (
  stu.data.id,
  stu.data.name
  from stu.data
  where stu.data.hours>0 and
             stu.data.term = &qterm
);
disconnect from ora1;
quit;

data termdata;
  set termdataV;
run;
```

Variable Output Based on Parameters

```
/* Define macro variables */

%let basefy=2005;           /* Assign base fiscal year value */
%let fy2=%eval(&basefy+1);  /* Calculate second FY value */
%let fy3=%eval(&basefy+2);  /* Calculate third FY value */
%let fy4=%eval(&basefy+3);  /* Calculate fourth FY value */
%let fy5=%eval(&basefy+4);  /* Calculate fifth FY value */

/* List resolved values of macro variables */

data a;
  basefy="&basefy"; fy2="&fy2"; fy3="&fy3"; fy4="&fy4";
  fy5="&fy5";
run;

proc print data=a;
  title 'Resolved Macro Variables';
run;

filename out1 dde 'Excel|C:\My Documents\[Book1]Sheet1!R1C1:R900C100' notab
lrecl=1000;

data _null_;
  file out1;
  x='09'x;
  if _n_=1 then put 'College' x 'Department' x "FY&basefy" x "FY&fy2" x
"FY&fy3" x "FY&fy4" x "FY&fy5" /;
run;
```

Variable Output Based on Derived Values

```
/* Define macro variables */

%let sfy=2006;               /* Degree starting fiscal year */
%let sub=%eval(&sfy-1);      /* Year for baseline data */
%let fy2=%eval(&sfy+1);      /* Second fiscal year */
%let fy3=%eval(&sfy+2);      /* Third fiscal year */
%let fy4=%eval(&sfy+3);      /* Fourth fiscal year */
%let fy5=%eval(&sfy+4);      /* Fifth fiscal year */
%let fy6=%eval(&sfy+5);      /* Sixth fiscal year */

options mprint;              /* Print results of macro invocation */

/* Define macro */

%macro wssp (n);             /* n is a positional parameter */
  %if &n>=2009 %then "SP&n"; /* For value of 2009, resolves to SP2009 */
  %else "WS&n";              /* For value of 2008, resolves to WS2008 */
%mend wssp;

filename out1 dde 'Excel|C:\My Documents\[Book1]Sheet2!R1C1:R900C100' notab
lrecl=1000;

data _null_;
  file out1;
  x='09'x;
  if _n_=1 then do;
    put  "Enrollment by Fiscal Year, Semester and Course Level - FY&fy2-&fy6"
/ 'Description' / ' ' x ' ' x ' ' x 'Baseline' /
      'Academic' x ' ' x ' ' x
      "FY&sfy" x " " x " " x " " x "FY&fy2" x " " x " " x " " x
      "FY&fy3" x " " x " " x " " x "FY&fy4" x " " x " " x " " x
      "FY&fy5" x " " x " " x " " x "FY&fy6" /
      "Group" x "Department" x "Course Level" x

/* For each reference to the macro, %wssp, the value of the parameter will be
   evaluated and the resulting value assigned. */

      "SS&sub" x "FS&sub" x %wssp(&sfy) x "Total" x
      "SS&sfy" x "FS&sfy" x %wssp(&fy2) x "Total" x
      "SS&fy2" x "FS&fy2" x %wssp(&fy3) x "Total" x
      "SS&fy3" x "FS&fy3" x %wssp(&fy4) x "Total" x
      "SS&fy4" x "FS&fy4" x %wssp(&fy5) x "Total" x
      "SS&fy5" x "FS&fy5" x %wssp(&fy6) x "Total";
  end;
run;
```

Variable-Size Arrays

```
/* Input look up data from an external file */
proc import file='c:\users\westermeyerl\documents\presentations\midair
2012\cohort 13 form.xlsx' dbms=excel2007 out=sample;
sheet='Certificates';
run;

/* Determine the number of observations */
data _null_;
  set sample end=lst;
/* The SAS variable _n_ counts the number of observations/rows/records.  The
macro variable nrecs will be the number of array elements. */
  if lst=1 then call symput('nrecs',trim(left(put(_n_,8.0))));
run;

/* Define arrays, load data, and save to a file */
data arrays (keep=certcode1-certcode&nrecs area1-area&nrecs);
  set sample end=lst;
  array certcode(&nrecs);                        /* Define array */
  array area(&nrecs) $44.0;                      /* Define array */
  certcode(_n_)=Certification_Subject_Area_Code; /* Assign value to array */
  area(_n_)=Certification_Subject_Area;          /* Assign value to array */
  if lst=1 then output;                          /* Save completed array */
  retain certcode1-certcode&nrecs area1-area&nrecs;
run;

/* Input data with code to be located from an external file */
proc import file='c:\users\westermeyerl\documents\presentations\midair
2012\sample data.xlsx' dbms=excel2007 out=data;
run;

/* Use the data in a subsequent data step */
data trans (drop=certcode1-certcode&nrecs area1-area&nrecs);
  set data;
  array certcode(&nrecs);
  array area(&nrecs) $44.0;
  if _n_=1 then set arrays;
  do i=1 to &nrecs;
    if icode=certcode(i) then do;
      areaect=area(i);
      go to done;
    end;
  end;
  put 'Code not found: ' _n_ icode;
 done:
  retain certcode1-certcode&nrecs area1-area&nrecs;
run;
```

Determine Length of Incoming Character Variables

```
/* Read file from Excel */

proc import file='c:\users\westermeyerl\documents\presentations\midair
2012\cohort 13 form.xlsx' dbms=excel2007 out=sample;
sheet='Certificates';
run;

data _null_;
  set sample end=lst;                    /* Variable lst=1 at end of file */
  if _n_=1 then mxlen=0;                    /* Set maximum length to zero */
  csalen=length(certification_subject_area); /* Find length of input data */
  if csalen>mxlen then mxlen=csalen;         /* If current>maximum, change */
  if lst=1 then do;
    call symput('ncsa',trim(left(put(_n_,8.0)))); /* Set number of records */
    csalen=mxlen;
    call symput('csalen',trim(left(put(csalen,8.0)))); /* Set max length */
  end;
  retain mxlen;                 /* Keep maximum value between observations */
run;

/* List resolved values of macro variables */

data a;
  ncsa="&ncsa"; csalen="&csalen";
run;

proc print data=a;
  title 'Resolved Macro Variables';
run;

proc sort data=ca; by certification_subject_area_code; run;

data aca (keep=acsac1-acsac&ncsa acsa1-acsa&ncsa);
  set sample end=lst;
  array acsac(&ncsa);             /* Define array - &ncsa - # of elements */
  array acsa(&ncsa) $&csalen..0;  /* Define array - &csalen - length */
  acsac(_n_)=certification_subject_area_code;
  acsa(_n_)=certification_subject_area;
  if lst=1 then output;
  retain acsac1-acsac&ncsa acsa1-acsa&ncsa;
run;
```

Using Program Data to Produce Additional SAS Code - %Include File

```
/* Input course data */

proc import file='c:\users\westermeyerl\documents\presentations\midair
2012\course data.xlsx' dbms=excel2007 out=courses;
run;

/* Determine the number of courses */

data _null_;
  set courses end=lst;
  if lst=1 then call symput('mxccrses',trim(left(put(_n_,8.0))));
run;

/* Place course subject, number, and title in array */

data acourses (keep=asubj1-asubj&mxccrses acn1-acn&mxccrses
                    act1-act&mxccrses);
  set courses end=lst;
  array asubj(&mxccrses) $8.0;
  array acn(&mxccrses) $10.0;
  array act(&mxccrses) $30.0;
  asubj(_n_)=csubject;
  acn(_n_)=crse;
  act(_n_)=ctitle;
  if lst=1 then output;
  retain asubj1-asubj&mxccrses acn1-acn&mxccrses act1-act&mxccrses;
run;

/* Produce spreadsheet with labels and SAS file for later use */

filename out1 dde 'Excel|C:\My Documents\[Book1]Sheet3!R1C1:R900C100' notab
lrecl=1000;

filename out2 'c:\users\westermeyerl\documents\presentations\midair
2012\labels.sas' recfm=v lrecl=1000;

data _null_;
  set acourses;
  file out1;                  /* Set output destination to Excel */
  array asubj(&mxccrses) $8.0;
  array acn(&mxccrses) $10.0;
  array act(&mxccrses) $30.0;
  attrib outrec format=$varying1000.0;
  x='09'x;
  if _n_=1 then put 'Course List Crosswalk' //
    'Subscript' x 'Subject' x 'Course Number' x 'Course Title' /;
  do i=1 to &mxccrses;
    put i x asubj(i) x acn(i) x act(i);
  end;
  file out2;                  /* Set output destination to the include file */
  do i=1 to &mxccrses;        /* Construct label for variable grdNNN */
```

```
    outrec='grd'||trim(left(i))||'='''||trim(asubj(i))||trim(acn(i))||'-
Grade''';
    put outrec;
  end;
  do i=1 to &mxccrses;       /* Construct label for variable crNNN */
    outrec='cr'|| trim(left(i))||'='''||trim(asubj(i))||trim(acn(i))||'-
Credit''';
    put outrec;
  end;
  do i=1 to &mxccrses;       /* Construct label for variable trmNNN */
    outrec='trm'|| trim(left(i))||'='''||trim(asubj(i))||trim(acn(i))||'-
Term''';
    put outrec;
  end;
run;

/* Input student data */

proc import file='c:\users\westermeyerl\documents\presentations\midair
2012\student data.xlsx' dbms=excel2007 out=student;
run;

options source2; /* To display program code from the %include statement */

/* Place the course data for each student into an array */

data asc (keep=sid grd1-grd&mxccrses cr1-cr&mxccrses
              trm1-trm&mxccrses);
  set student;
  by sid;
  array grd(&mxccrses) $3.0;
  array cr(&mxccrses);
  array trm(&mxccrses) $10.0;
  array asubj(&mxccrses) $8.0;
  array acn(&mxccrses) $10.0;
  array act(&mxccrses) $30.0;
  label %include out2; ;
  if _n_=1 then set acourses;
  if first.sid=1 then do i=1 to &mxccrses;
    grd(i)=' ';
    cr(i)=.;
    trm(i)=' ';
  end;
  do i=1 to &mxccrses;
    if csubject=asubj(i) and crse=acn(i) then do;
      grd(i)=cgrade;
      cr(i)=cunits;
      trm(i)=cstrm_ext;
      go to done;
    end;
  end;
 done:
  if last.sid=1 then output;
  retain asubj1-asubj&mxccrses acn1-acn&mxccrses act1-act&mxccrses
         grd1-grd&mxccrses cr1-cr&mxccrses trm1-trm&mxccrses;
run;
```

Using Program Data to Produce Additional SAS Code - Macro Variables

```
data _null_;              /* Produce list of ids to be used in SQL */
  set ids end=lst;
  if _n_=1 then nout=0;  /* Set number produced to zero */
  attrib outrec format=$varying1000.0;
  nout=nout+1;           /* Increment number produced */
  if lst=0 then outrec=''''||trim(id)||''',';  /*  Add trailing comma if not
                                                   last id */
  else outrec=''''||trim(id)||'''';
  call symput('idn'||left(nout),outrec);  /* Add to list */
  if lst=1 then call symput('nout',left(trim(put(nout,8.0))));  /* Number of
                                                                   ids produced */
  retain nout;          /* Keep count of ids produced across observations */
run;

options mprint;         /* Display statements generated by macro execution */

%macro mids;            /* Define macro and give it a name */
%do i=1 %to &nout;      /* Iterate &nout times */
  &&idn&I               /* The double ampersand resolves to &idn&I in code */
%end;
%mend mids;             /* End of macro definition */

proc sql;
connect to oracle as ora1 (user=myid orapw=mypass
                  path="@tns:datawarehouse.myschool.edu");
create view crsesV as
select * from connection to ora1 (
  SELECT "STU"."ENRL"."EMPLID",
         "STU"."CLASS"."SUBJECT",
         "STU"."CLASS"."CATALOG_NBR",
         "STU"."ENRL"."CRSE_GRADE_OFF",
         "STU"."ENRL"."GRD_PTS_PER_UNIT",
         "STU"."ENRL"."UNT_TAKEN",
         "STU"."CLASS"."CRSE_ID"
    FROM "STU"."ENRL",
         "STU"."CLASS"
   WHERE ( "STU"."ENRL"."STRM" = "STU"."CLASS"."STRM" ) and
         ( "STU"."ENRL"."CLASS_NBR" = "STU"."CLASS"."CLASS_NBR" ) and
         ( ( STU."ENRL"."EMPLID" in ( %mids ) ) AND
         ( STU."ENRL"."STDNT_ENRL_STATUS" = 'E' ) AND
         ( STU."ENRL"."ENRL_STATUS_REASON" in ( 'ENRL','EWAT' ) ) ) );
disconnect from ora1;
quit;
```