CONCEPTS AND FEATURES OF MACRO PROGRAMMING

CONTENTS

- What is SAS Macro
- Advantages of Macro
- Macro variable-Global and Local
- Automatic macro variable and User-define macro variable
- Main features Of Macro variable
- How to create macro variable
- Combining macro variable with text
- Macro program
- Positional macro Parameters and Keyword macro Parameters
- Displaying Messages about Macro Program Processing in the SAS Log
- Constructing macro expressions
- Macro Evaluation functions
- Conditional Processing of SAS Steps in macro language
- Iterative Processing of SAS Steps in macro language

The SAS Macro & Advantages

You associate a macro reference with text.

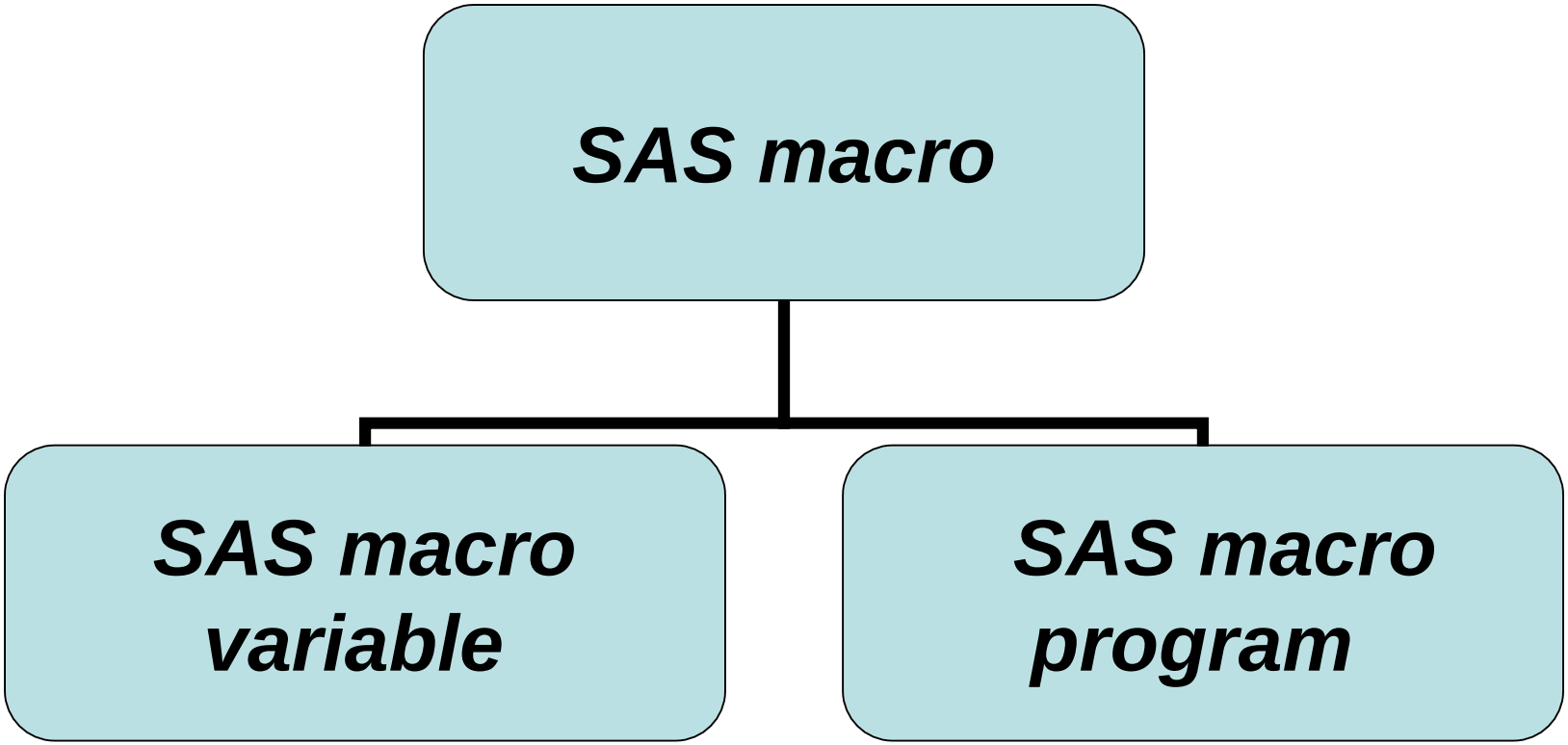
- The SAS macro is a tool for text substitution.
- When the macro processor encounters that reference, it replaces

  the reference with the associated text.
- Programs can become reusable, shorter, and easier to follow.
- Repetitive tasks can be done quickly and efficiently .
- A macro program can be reused many times in same program or in different program.

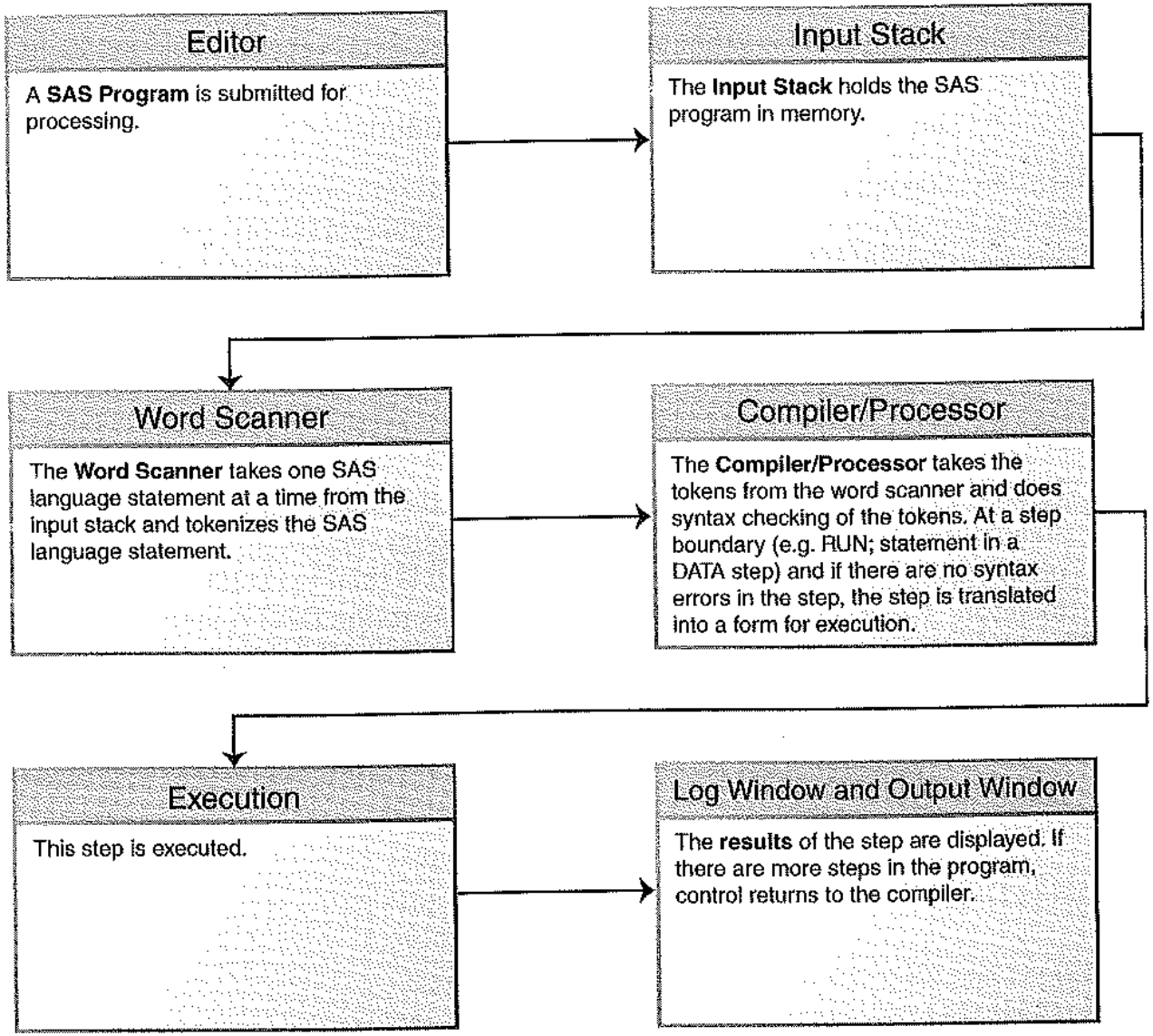
  & refers to a macro variable.

  – **%** refers to a macro .

The two main components of the SAS macro



Basic processing of a SAS program



SAS macro variables

- Macro variables can be stored in either the global symbol table or in a local symbol table.
- So accordingly there are two types of macro variables:
    1. global macro variables and
    2. local macro variables
- Global macro variables are created in open code.
- The value assign to a global macro variable remains same throughout the SAS session unless you change it.
- Macro variable created in a macro program are defined as local macro variable.
- The value that you assign to a local macro variable remains the same only throughout **execution of the macro program.**

Macro variables can be created by SAS and by your programs:

- **Automatic macro variables :**
  - SAS automatically defines a set of macro variables when a SAS session starts.
  - These variables are maintained with their values in the macro symbol table.
  - E.g &systime , &sysday , &sysdate
- **User defined macro variables:**
  - User-defined macro variables are defined by you for your own applications.

Main features of Macro variables:

1. A macro variable can be referenced anywhere in a SAS program other than in data lines.
2. Macro variables can be used in open code as well as in macro programs.
3. Macro variable values are text values.
4. The case of the character value assigned to a macro variable is preserved.
5. The maximum length of text that can be assigned to a macro variable value is 65,534characters and the minimum length is zero characters.
6. The name assigned to a macro variable must be a valid SAS name.
7. The macro variable is referred by preceding the macro variable name with an ampersand(&)

Something more……..

- No arithmetic calculations are done.

  e.g: %let value1=9;

  %let value2=8;

  %let result=&value1 + &value2;

  So the value of macro variable result is 9 + 8 and

  NOT 17

- Leading blanks are removed from the value of a macro variable.

  e.g: %let City= I stay in Mumbai.;

  The value of macro variable CITY is I stay in Mumbai.

- Blanks and special characters (!@#$%^&*) are valid macro variable values.

Note: A macro variable has only one value while a data set variable can have multiple values, one for each observation in a data set.

Create Macro Variables

- Macro variables can be created using following techniques and statements:
  1. %LET statement
  2. macro parameters (named and positional)
  3. iterative %DO statement
  4. using the INTO in PROC SQL
  5. using the CALL SYMPUT routine

%let statement

- Example: %let x= 5;

  Here a macro variable “X” is created with value as 5.

%put statement

- Writes text or macro variable information to the SAS log.

%put <text | _ALL_ | _AUTOMATIC_ | _GLOBAL_ | _LOCAL_ | _USER_ >;

Create Macro Variables cont…

- **Text** : text or a text expression that is written to the SAS log
- **_ALL_**: Lists the values of all user-defined and automatic macro variables.
- **_AUTOMATIC_** : Lists the values of automatic macro variables.
- **_GLOBAL_** : Lists user-defined global macro variables.
- **_LOCAL_**:Lists user-defined local macro variables.
- **_USER_** : Lists user-defined global and local macro variables.

Combining Macro Variables with Text

```
%let prefix=QUESTION;
    proc freq data=books.survey;
            tables &prefix.1 &prefix.2 &prefix.3
                    &prefix.4 &prefix.5;
    run;
```

- When text is followed by a macro variable reference, place a period at the end of the macro variable reference to terminate the reference.

```
proc freq data=books.survey;
        tables QUESTION1 QUESTION2 QUESTION3
                QUESTION4 QUESTION5;
 Run;
```

Combining Macro Variables with Text cont…

```
%let year=4;
%let level=25;

Data city&year&level;
      Set city;
       If level>50 then over&level= “Yes”;
Run;
```

After the resolution :

```
Data city425;
        Set city;
        If level>50 then over25= “Yes”;
Run;
```

- When placing text before a macro variable reference or when combining macro variable references, do **not** put the period between macro reference and text.

Referencing Permanent SAS Data Set Names and Macro Variables

```
libname abc  '/home/inamdarv';
data abc.one;
      input apple orange;
      cards;
      12 34
     61 34
        ;
run;

%let path=abc;
```

The required proc means is as follows:

```
 proc means data=abc.one   n sum;
 run;
proc means data=&path..one   n sum;
run;
```

Referencing Permanent SAS Data Set Names and Macro Variables cont…

```
%let section1=Certification and Training;
%let section2=Internet;
%let section3=Networking and Communication;
%let section4=Operating Systems;
%let section5=Programming and Applications;
%let section6=Web Design;
%let n=4;

proc means data=books.ytdsales;
        title "Sales for Section: &&section&n";
        where section="&&section&n";
        var saleprice;
run;
```

After macro variable resolution, the preceding program becomes:

Macro Programs

A macro program is defined with the following statements:

%MACRO program <(parameter-list)>;

<text>

%MEND <program>;

%MACRO: Marks the beginning of the macro program definition.

Program : Name assigned to the macro program.

The macro program name must be a valid SAS name.

<parameterlist>: Names one or more local macro variables whose values you specify when you invoke the macro program. Defining parameters is optional.

Positional Parameters:

%macro program(positional-1, positional-2, ...,positional-n);

Positional macro program parameters define a one-to-one correspondence between the list of parameters on the %MACRO statement and the values of the parameters on the macro program call.

%program(value-1, value-2, ..., value-n)

Macro Programs

- Positional parameters are enclosed in parentheses and are separated with commas.
- There is no limit to the number of positional parameters that can be defined.
- When you call a macro program that uses positional parameters, you must specify the same number of values in the macro program call as the number of parameters listed on the %MACRO statement.
- Valid values include null values and text.

Keyword Parameters :

%macro program(keyword1=value, keyword2=value, ...,keywordn=value);

A call to a macro program that has been defined with keyword parameters contains both the parameter names and the initial parameter values.

Advantage of using keyword parameters:

1. Keyword parameters can be specified in any order.
2. A one-to-one correspondence between the parameters on the %MACRO statement and the values on the call to the macro program is not required as it is with positional parameters.

Note:

Positional parameters and keyword parameters can both be defined for the same macro program. Positional parameters must be placed ahead of keyword parameters in the definition and in the call to the macro program.

Macro program syntax

%MACRO program <(parameter-list)>;
    <text>

%MEND <program>;

<text>:Text is any combination of

text strings

macro variables, macro functions ,macro language statements

SAS programming statements

%MEND: Marks the end of the macro program.

Displaying Messages about Macro Program Processing in the SAS Log

- **MPRINT** You can see the SAS code in Log file.
- **MLOGIC:** The MLOGIC option traces the execution of macro programs , that includes the beginning and ending of the macro program and the results of arithmetic and logical macro language operations.
- **SYMBOLGEN** :displays information about macro variables that are created in open code or inside macro programs.

Difference between %put and SYMBOLGEN

- SYMBOLGEN displays the values of all macro variables you reference in your program,
- %PUT lets you selectively display macro variable values.

- The %PUT statement gives you control of where and when a macro variable value is displayed.
- Option SYMBOLGEN displays macro variable values only when they are referenced.

Macro Expressions and **Macro Evaluation Functions**

Types of macro expressions: text, logical, and arithmetic

A text expression is any combination of text, macro variables, macro functions, or macro calls.

Logical expressions and arithmetic expressions are sequences of operators and operands forming sets of instructions that are evaluated to produce a result.

Arithmetic Expressions
- 1 + 2
- 4 * 3

Logical expressions
- &DAY = FRIDAY
- A < a

Note: MIN and MAX operators are not available in the macro language.

%EVAL and **%SYSEVALF**, evaluate arithmetic expressions and logical expressions.

→The arguments to one of these macro evaluation functions are temporarily converted to numbers

→The macro evaluation function converts the result that it returns to text.

Macro Expressions and **Macro Evaluation Functions cont..**

- The %EVAL function supports only integer arithmetic
- The %SYSEVALF function also supports floating point arithmetic.

Syntax of the %EVAL function :

%EVAL(arithmetic expression|logical expression)

Syntax of the %SYSEVALF function :

%SYSEVALF(arithmetic expression|logical expression,<conversion-type>)

Conversion Type Result that is returned by %SYSEVALF

- BOOLEAN: Returns 0 if the result of the expression is 0 or null, Returns 1 if the result is any other value
- CEIL : Returns smallest integer >= result of the expression
- FLOOR : Returns largest integer <= result of the expression
- INTEGER : Returns integer portion of the expression's result

Conditional Processing of SAS Steps

Syntax of the %IF statement:

%IF *expression* %THEN *action*;
<%ELSE *elseaction*;>

- The expression is usually a logical expression.
- The macro processor invokes the %EVAL function around the expression and resolves the expression to true or false.
- When the evaluation of the expression is true, *action* is executed.
- When the evaluation of the expression is false and a %ELSE statement is specified, *elseaction* is executed.

Note: Conditional and iterative macro statements do not work in an open code

Examples of %IF Statement

1)
```
%let a=1;
%macro test;
    %if &a=1 %then %put a=1;
    %else %put a ne 1;
%mend test;
%test;
```

2)
```
%let a=10;
%let b=2.1;
%macro test1;
    %if &a > &b %then %put a>b;
  %if %sysevalf( &a > &b) %then %put a>b;
%mend test1;
 %test1;
```

3)
```
% let a=1;
%let b=1;
%macro test2;
    %if &a = 1 %then %do;
        data a;
        set sashelp.buy;
            %if &b=1 %then %do;
                if amount>=48000;
            %end;
        run;
    %end;
%mend test2;
%test2;
```

Conditional Iteration with %DO %WHILE

Syntax of %DO %WHILE :

%DO %WHILE (*expression*);
macro language statements and/or text
%END;

- With %DO %WHILE, a section of code is executed *while* the condition on the %DO %WHILE statement is true.
- The expression is a macro expression that resolves to a true-false value.
- The macro processor evaluates the expression at the top of the loop.
- A %DO %WHILE loop does not execute even ones when the condition starts out as false.

Conditional Iteration with %DO %UNTIL

Syntax of %DO %UNTIL :

%DO %UNTIL (*expression*);
macro language statements and/or text
%END;

- With %DO %UNTIL, a section of code is executed *until* the condition on the %DO %UNTIL statement is true.
- The expression is a macro expression that resolves to a true-false value.
- The macro processor evaluates the expression at the bottom of each iteration.
- Therefore, a %DO %UNTIL loop always executes at least once.

Iterative %DO Loops in the Macro Language

Syntax of an iterative %DO loop :

%DO *macro-variable=start* %TO *stop* <%BY *increment*>;
macro language statements and/or text
%END;

- Instructs the macro processor to execute a section of code repeatedly.
- The number of times the section executes is based on the value of an index variable which is a macro variable.
- The start value and stop value of this index variable need to be defined.
- The increment of the steps between the start value and the stop value can be controlled.

CALL SYMPUT

- CALL SYMPUT is a SAS language routine that assigns a value produced in a DATA step to a macro variable. It is one of the DATA step interface tools that provides a dynamic link for communication between the SAS language and the macro facility.

SYMPUT is a routine that you CALL inside a data step to produce a global symbolic variable. The variable can not be referenced inside the creating data step. The data step must end with a RUN**;** statement before you reference the variable in your code. The value of the macro variable is assigned during the execution of the data step. The syntax is:

```
DATA dataset(can be _NULL_);
   CALL SYMPUT(name, value);
RUN;
```

```
DATA p_values;
INPUT test : $8. label : $25.;
CALL SYMPUT (‘dm’, label);
CARDS;
CMH Cochran-Mantel-Haenzel
;
RUN;
```

Write the syntax to see the macro variable values in the log

CALL SYMPUT cont…

```
DATA not_available;
     test='23rd Ocotber 2002';
    CALL SYMPUT ('NotAvail',test);
    retrieve="&NotAvail";
  RUN;
```

Run the code and see the log for any error or warning.

Creating Macro variable from proc sql

“INTO:” host-variable in PROC SQL is a powerful tool. It simplifies programming code while minimizing the risk of typographical errors. SQL INTO: creates one or more macro variables, based on the results of a SELECT statement. This macro variable(s) reflects a list of values that can then be used to manipulate data in both DATA and PROC steps.

```
PROC SQL NOPRINT;
  SELECT  EMPID, DIAG
  INTO :EMP_LIST, : DIAGLIST
  FROM MASTER;
 QUIT;
 %PUT &EMP_LIST &DIAGLIST;
```

Creating Macro variable from proc sql cont…

```
data test;
  input age id salary resigned @@;
  datalines ;
  12 01 20005 0 12 15 2336 1 12 45 56663 1 12 45 5666 1
  ;
quit;
proc contents data=test out=c;
run;
proc sql;
  select count(*)  into :g from c;
Quit;
```

Creating Macro variable from proc sql cont..

```
proc sql;
  select name into :b1 -:b&a from c   ;
quit;
%macro test();
Proc means data =  test;
var
%do Small = 1 %to &a;
&&b&Small
%end;
run;
%mend;
%test();
```

Example

```
%macro mylogit1(all_deps);
  %let k=1;
  %let dep = %scan(&all_deps, &k);
  %do %while("&dep" NE "");
    title "dependent variable is &dep";
    proc logistic data=xxx des;
      model &dep = ind1 ind2;
    run;
    %let k = %eval(&k + 1);
    %let dep = %scan(&all_deps, &k);
  %end;
%mend;
```

What is Symbol Table

It's a Table which keeps track of Symbols (Variables) generated in a Code.
It is created in the Semantic Analysis Phase of Complier

Macro Symbol Table

Global macro symbol table

Accessible to all Macros

- Create at the start of SAS Session (Mostly Automatic variable)
- Open Code
- Defined with Global
- Created in Data step with call symput or call symputx

Local macro symbol table

Accessible to only defined code

- Variables defined in Macro
- Deleted when program associated with it ends.

Macro Processor Keeps track of the domain of each macro variable that is defined in SAS program

Revision Questions

Q1. How can you create a macro variable with in data step?

Q2. What is the difference between %LOCAL and %GLOBAL?

Q3. What is the difference between %PUT and SYMBOLGEN?

Q4. Describe the ways in which you can create macro variables?

Q5. How would you code a macro statement to produce information on the SAS log?

Thank You ☺