SAS MACRO: BEYOND THE BASICS

ADDITIONAL TOPICS

Agenda

We will discuss:

- Macro Iterative Processing
- Macro Conditional Processing
- Creating Macro Variables at Execution time
- Introduction to Macro Quoting

MACRO ITERATIVE PROCESSING

MACRO ITERATIVE PROCESSING

The following macro program is created with a parameter.

```
%macro claiminfo(year);
   proc print data=health.claims_sample;
     var providerID memberID chgamt lndiscamt eeresp paidamt;
     title "Report for &year";
     where svc_year="&year";
   run;

   proc means data=health.claims_sample;
     title "Summary for &year";
     var chgamt lndiscamt eeresp paidamt;
     where svc_year = "&year";
   run;
%mend;
```

REPETITIVE CALLS

The user supplies values for `year` when the macro is called.

```
%claiminfo(2005)
%claiminfo(2006)
%claiminfo(2007)
```

Suppose that we had 25 values of year that we wish to use in the call to the macro. In such a case, it would be helpful if we could allow the macro facility to supply the values of year for us.

ITERATIVE PROCESSING

We can use iterative processing inside a macro definition in order to repetitively execute macro language statements and generate SAS code.

The general form of the iterative %DO statement:

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

ITERATIVE PROCESSING

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

- %DO is valid inside a macro program
- *macro-variable* names a macro variable or a text expression that generates a macro variable name. Its value functions as an index.
- *Start/stop* specify integers or macro expressions that generate integers to control the number of times the portion of the macro between the iterative %DO and %END statements is processed.

ITERATIVE PROCESSING

```
%macro claiminfo;
   %do year=2005 %to 2007;
   proc print data=health.claims_sample;
      var providerID memberID chgamt lndiscamt eeresp paidamt;
      title "Report for &year";
      where svc_year="&year";
   run;

   proc means data=health.claims_sample;
      title "Summary for &year";
      var chgamt lndiscamt eeresp paidamt;
      where svc_year = "&year";
   run;
   %end;
%mend;
```

ITERATIVE PROCESSING

```
MPRINT(CLAIMINFO):   proc print data=health.claims_sample;
MPRINT(CLAIMINFO):   var providerID memberID chgamt lndiscamt eeresp paidamt;
MPRINT(CLAIMINFO):   title "Report for 2005";
MPRINT(CLAIMINFO):   where svc_year="2005";
MPRINT(CLAIMINFO):   run;

NOTE: There were 337 observations read from the data set HEALTH.CLAIMS_SAMPLE.
      WHERE svc_year='2005';
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.17 seconds
      cpu time            0.17 seconds


MPRINT(CLAIMINFO):   proc means data=health.claims_sample;
MPRINT(CLAIMINFO):   title "Summary for 2005";
MPRINT(CLAIMINFO):   var chgamt lndiscamt eeresp paidamt;
MPRINT(CLAIMINFO):   where svc_year = "2005";
MPRINT(CLAIMINFO):   run;
```

ITERATIVE PROCESSING

```
MPRINT(CLAIMINFO):   proc print data=health.claims_sample;
MPRINT(CLAIMINFO):   var providerID memberID chgamt lndiscamt eeresp paidamt;
MPRINT(CLAIMINFO):   title "Report for 2006";
MPRINT(CLAIMINFO):   where svc_year="2006";
MPRINT(CLAIMINFO):   run;

NOTE: There were 411 observations read from the data set HEALTH.CLAIMS_SAMPLE.
      WHERE svc_year='2006';
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.20 seconds
      cpu time            0.20 seconds


MPRINT(CLAIMINFO):   proc means data=health.claims_sample;
MPRINT(CLAIMINFO):   title "Summary for 2006";
MPRINT(CLAIMINFO):   var chgamt lndiscamt eeresp paidamt;
MPRINT(CLAIMINFO):   where svc_year = "2006";
MPRINT(CLAIMINFO):   run;

NOTE: There were 411 observations read from the data set HEALTH.CLAIMS_SAMPLE.
      WHERE svc_year='2006';
NOTE: PROCEDURE MEANS used (Total process time):
      real time           0.02 seconds
      cpu time            0.03 seconds
```

ITERATIVE PROCESSING

```
MPRINT(CLAIMINFO):   proc print data=health.claims_sample;
MPRINT(CLAIMINFO):   var providerID memberID chgamt lndiscamt eeresp paidamt;
MPRINT(CLAIMINFO):   title "Report for 2007";
MPRINT(CLAIMINFO):   where svc_year="2007";
MPRINT(CLAIMINFO):   run;

NOTE: There were 418 observations read from the data set HEALTH.CLAIMS_SAMPLE.
      WHERE svc_year='2007';
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.20 seconds
      cpu time            0.21 seconds


MPRINT(CLAIMINFO):   proc means data=health.claims_sample;
MPRINT(CLAIMINFO):   title "Summary for 2007";
MPRINT(CLAIMINFO):   var chgamt lndiscamt eeresp paidamt;
MPRINT(CLAIMINFO):   where svc_year = "2007";
MPRINT(CLAIMINFO):   run;

NOTE: There were 418 observations read from the data set HEALTH.CLAIMS_SAMPLE.
      WHERE svc_year='2007';
NOTE: PROCEDURE MEANS used (Total process time):
      real time           0.02 seconds
      cpu time            0.03 seconds
```

This example can be enhanced further by creating parameters that allow the user to supply starting and stopping dates.

```
%macro claiminfo(start, end);
  %do year=&start %to &end;
     proc print data=health.claims_sample;
       var providerID memberID chgamt lndiscamt eeresp paidamt;
       title "Report for &year";
       where svc_year="&year";
     run;
     proc means data=health.claims_sample;
       title "Summary for &year";
       var chgamt lndiscamt eeresp paidamt;
       where svc_year = "&year";
     run;
   %end;
%mend;

%claiminfo(2005,2007)
```

HELPFUL OPTIONS

- **OPTIONS SYMBOLGEN | NOSYMBOLGEN;**
  - Use to display the values of macro variables.
- **OPTIONS MCOMPILENOTE = NONE | ALL;**
  - Beginning in SAS®9, this SAS system option can notify you after a macro completes compilation.
- **OPTIONS MPRINT | NOMPRINT;**
  - displays the SAS statements generated by macro execution.
- **OPTIONS MLOGIC | NOMLOGIC;**
  - prints messages that indicate macro actions that were taken during macro execution.

MACRO CONDITIONAL PROCESSING

CONDITIONAL PROCESSING

Iterative processing executes a section of a macro repetitively based on the value of an index variable.

If you need to conditionally process a portion of a macro, you can use the %IF %THEN macro statement. The general form of this statement is:

```
%IF expression %THEN action;
```

expression is any macro expression that resolves to an integer.

action is either constant text, a text expression, or a macro statement.

CONDITIONAL PROCESSING

Let's change the previous example such that the user can request either a listing report or a summary report based on a parameter.

```
%macro claiminfo(report, year);
   %if %upcase(&report)= LISTING %then %do;
    proc print data=health.claims_sample;
       var providerID memberID chgamt lndiscamt eeresp paidamt;
       title "Report for &year";
       where svc_year="&year";
    run;
   %end;
   %else %do;
    proc means data=health.claims_sample;
       title "Summary for &year";
       var chgamt lndiscamt eeresp paidamt;
       where svc_year = "&year";
    run;
   %end;
%mend;

%claiminfo(listing ,2005)
%claiminfo( ,2005)
```

CONDITIONAL PROCESSING

Note the following:

- The absence of quotes on the %IF statement
- %DO groups are required

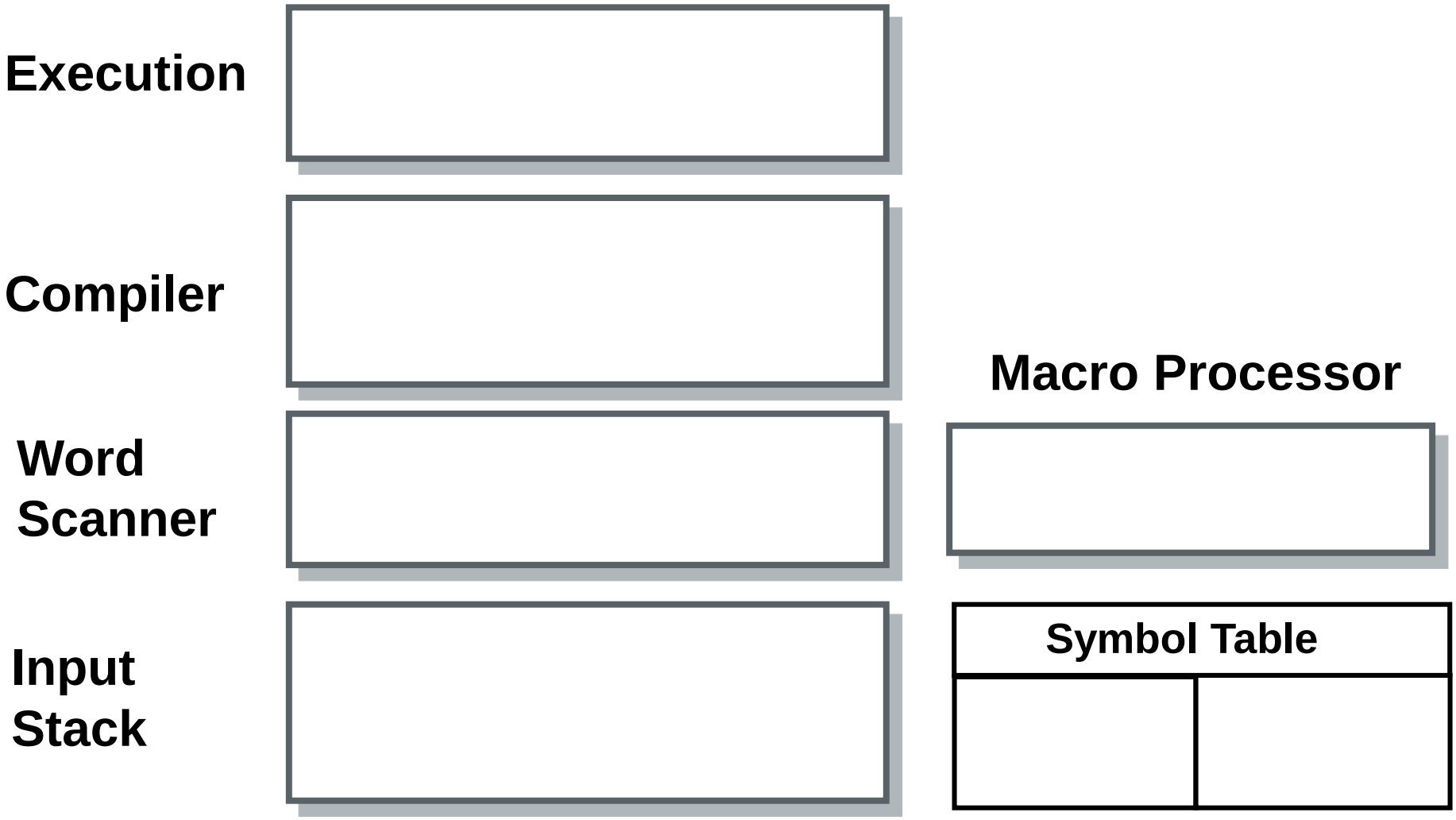
```
%macro claiminfo(report, year);
   %if %upcase(&report)= LISTING %then %do;
         proc print data=health.claims_sample;
            var providerID memberID chgamt lndiscamt eeresp paidamt;
            title "Report for &year";
            where svc_year="&year";
         run;
   %end;
   %else %do;
         proc means data=health.claims_sample;
            title "Summary for &year";
            var chgamt lndiscamt eeresp paidamt;
            where svc_year = "&year";
         run;
   %end;
%mend;

%claiminfo(listing ,2005)
%claiminfo( ,2005)
```

DATA STEP VS. MACRO

The purpose of %IF %THEN is to decide which text is placed onto the input stack. The DATA step IF THEN is processed at execution of the DATA step and determines how to process data.

CONDITIONAL PROCESSING



DATA STEP VS. MACRO

```
%macro settax(taxrate);
   %let taxrate = %upcase(&taxrate);
   %if &taxrate = CHANGE %then %do;
      data thisyear;
         set lastyear;
         if sale > 100 then tax = .05;
         else tax = .08;
      run;
   %end;
   %else %if &taxrate = SAME %then %do;
      data thisyear;
         set lastyear;
         tax = .03;
      run;
   %end;
%mend settax;
```

DATA STEP VS. MACRO

When the value of the macro variable TAXRATE is CHANGE, then the macro generates the following DATA step:

```
DATA THISYEAR;
  SET LASTYEAR;
  IF SALE > 100 THEN TAX = .05;
      ELSE TAX = .08;
RUN;
```

When the value of the macro variable TAXRATE is SAME, then the macro generates the following DATA step:

```
DATA THISYEAR;
  SET LASTYEAR;
  TAX = .03;
RUN;
```

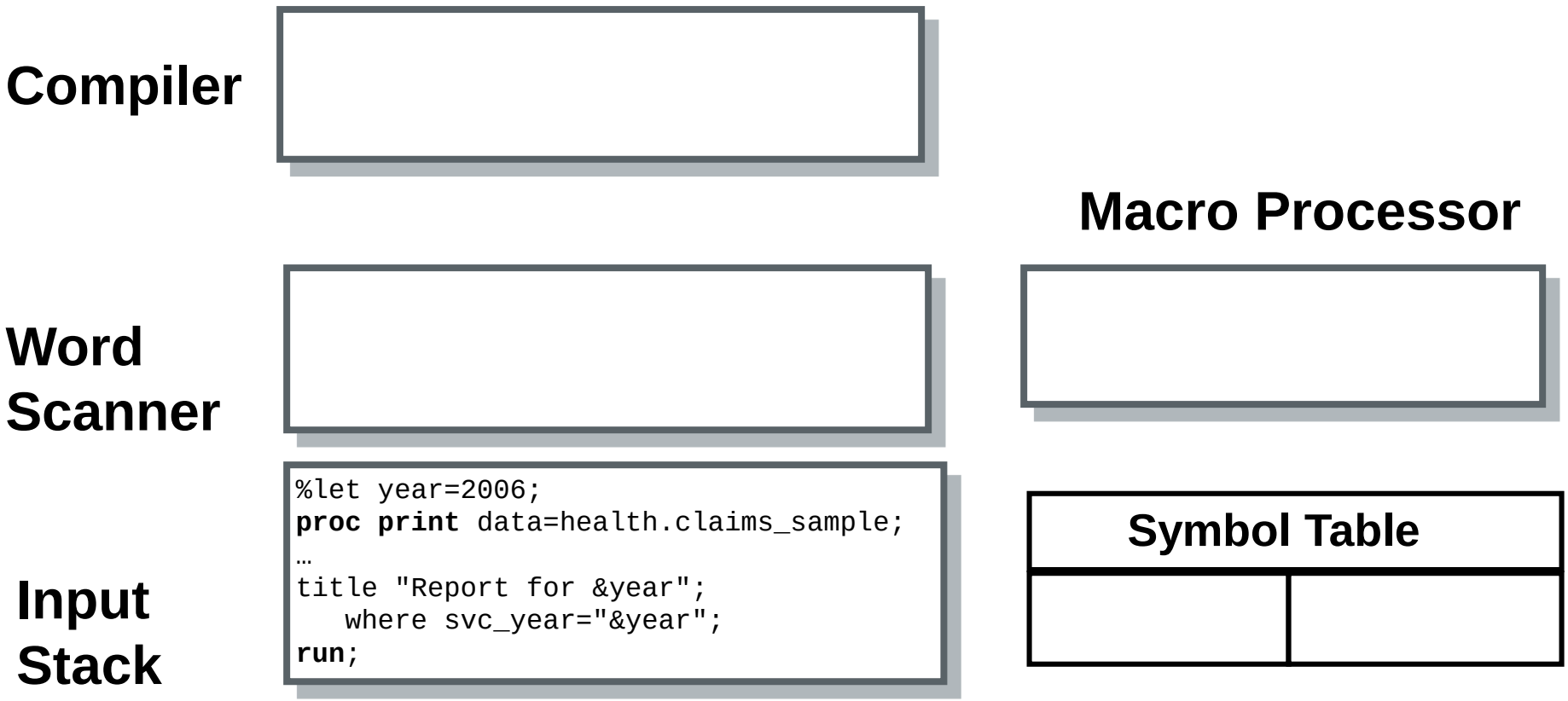
CREATING MACRO VARIABLES AT EXECUTION TIME

COMPILATION AND EXECUTION

The %LET statement can be used to create a macro variable and assign it a value.

```
%let year=2006;
```

TIMING OF COMPILATION AND EXECUTION



TIMING OF COMPILATION AND EXECUTION

Compiler

Word Scanner

```
%let
```

Input Stack

```
year=2006;
proc print data=health.claims_sample;
…
title "Report for &year";
   where svc_year="&year";
run;
```

Macro Processor

Symbol Table

TIMING OF COMPILATION AND EXECUTION

Compiler

Word Scanner

Input Stack

```
proc print data=health.claims_sample;
…
title "Report for &year";
   where svc_year="&year";
run;
```

Macro Processor

```
%let year=2006;
```

Symbol Table	
year	2006

TIMING OF COMPILATION AND EXECUTION

Compiler

```
proc print data=health.claims_sample;
…
title "Report for
```

Word Scanner

```
&year
```

Input Stack

```
";
   where svc_year="&year";
run;
```

Macro Processor

Symbol Table	
year	2006

TIMING OF COMPILATION AND EXECUTION

Compiler

```
proc print data=health.claims_sample;
…
title "Report for
```

Word Scanner

Input Stack

```
";
   where svc_year="&year";
run;
```

Macro Processor

```
&year
```

Symbol Table	
year	2006

TIMING OF COMPILATION AND EXECUTION

Compiler

```
proc print data=health.claims_sample;
…
title "Report for
```

Word Scanner

Input Stack

```
2006";
   where svc_year="&year";
run;
```

Macro Processor

Symbol Table	
year	2006

TIMING OF COMPILATION AND EXECUTION

Compiler

```
proc print data=health.claims_sample;
…
title "Report for 2006";
where svc_year=“2006";
run;
```

Macro Processor

Word Scanner

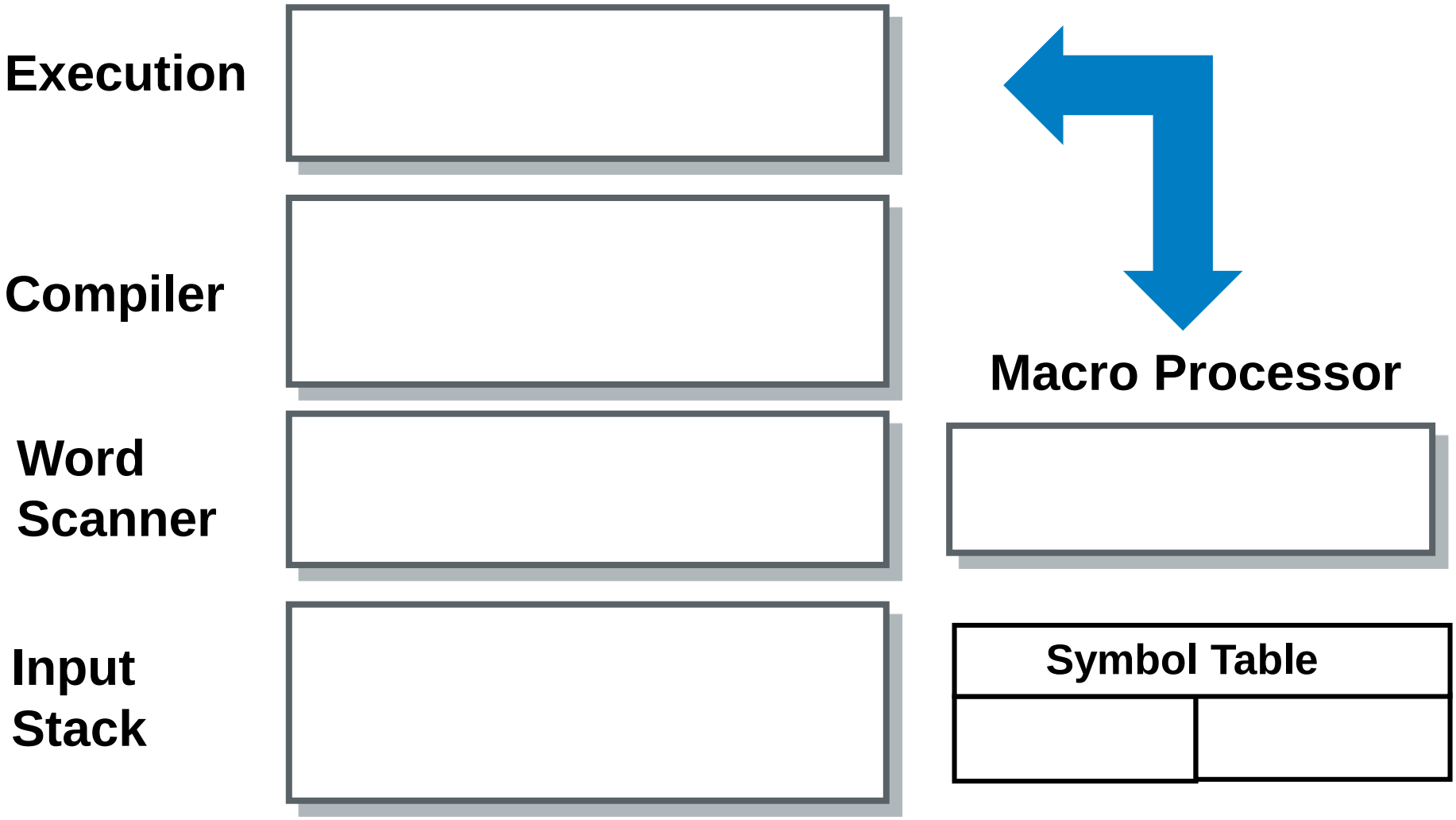
Input Stack

Symbol Table	
year	2006

SUMMARY

- When creating macro variables with %LET, all macro processing is completed **PRIOR** to compile time.
- Since data in a SAS data set is accessed at execution time, this means that we do not have SAS data available to create a macro variable when using the %LET statement.
- We sometimes need a way to interact with the macro facility during DATA step execution.

SYMPUTX ROUTINE

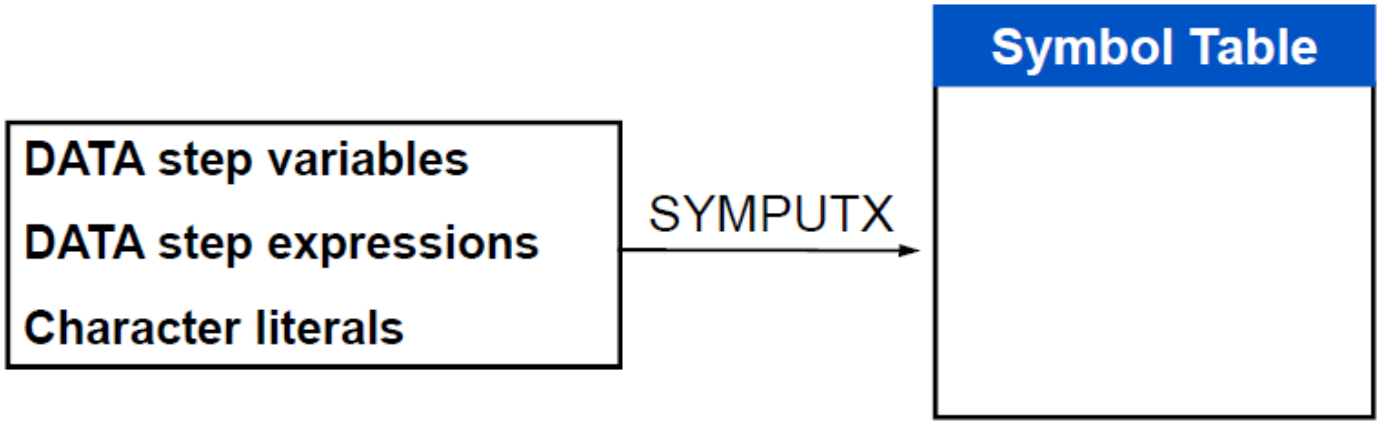


SYMPUTX Routine

The SYMPUTX routine assigns to a macro variable any value available to the DATA step during execution time.

It can create macro variables with the following:

- static values
- dynamic (data dependent) values
- dynamic (data dependent) names



SYMPUTX ROUTINE

The SYMPUTX routine assigns to a macro variable any value available to the DATA step during execution.

Syntax

```
CALL SYMPUTX(macro-variable, value <,symbol-table> );
```

Note: SYMPUTX is a DATA step routine that assigns a value to a macro variable, and removes both leading and trailing blanks.

DEMONSTRATION

- Create macro variables using CALL SYMPUTX.
- Create a series of macro variables using CALL SYMPUTX.

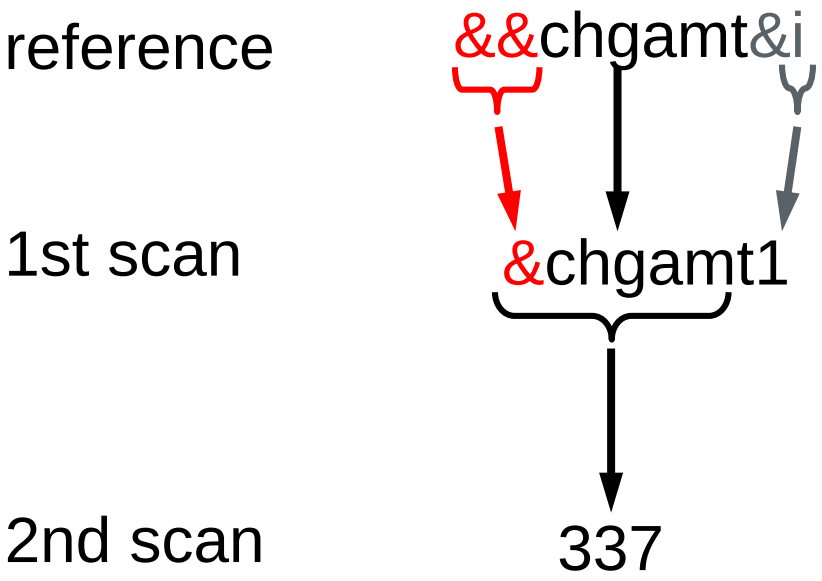
INDIRECT REFERENCES TO MACRO VARIABLES

The Forward Rescan Rule

- Multiple ampersands preceding a name token denote an indirect reference.
- Two ampersands (&&) resolve to one ampersand (&).
- The macro processor will rescan an indirect reference, left to right, from the point where multiple ampersands begin.
- Scanning continues until no more references can be resolved.

INDIRECT REFERENCES TO MACRO VARIABLES

The indirect reference causes a second scan.



MACRO QUOTING

MACRO FUNCTIONS

Macro functions manipulate arguments within the context of the macro language.

The following are true for macro functions:

- manipulate macro variables and expressions
- may mimic the functionality of DATA step functions
- are executed by the macro processor

QUOTING FUNCTIONS

Macro quoting functions tell the macro processor to interpret special characters and mnemonics as text rather than as part of the macro language. The following functions are most commonly used:

- %STR and %NRSTR
- %BQUOTE and %NRBQUOTE
- %SUPERQ

QUOTING FUNCTIONS

Macro quoting functions tell the macro processor to interpret special characters and mnemonics as text rather than as part of the macro language. The following functions are most commonly used:

- %STR and %NRSTR
- %BQUOTE and %NRBQUOTE
- %SUPERQ

Hint: Think of NR as ‘Not Resolved’.

Special Characters Masked by the %STR and %NRSTR Functions

blank	)	=	NE
;	(	|	LE
¬	+	#	LT
^	—	AND	GE
~	*	OR	GT
, (comma)	/	NOT	
'	<	IN	
“	>	EQ	

In addition to these special characters and mnemonics, %NRSTR masks & and %.

%STR

The following code attempts to create a macro variable named `printit` that contains a PROC PRINT step.

```
%let printit=proc print; run;;
%put value of printit is: &printit;
```

Partial Log:

```
623  %put value of printit is: &printit;
value of printit is: proc print
```

%STR

The use of %STR can mask the normal syntactical meaning of the semicolon.

```
%let printit=%str(proc print; run;);
%put value of printit is: &printit;
```

Partial Log:

```
625  %put value of printit is: &printit;
value of printit is: proc print; run;
```

%STR

The literal OR is interpreted as a logical operator that requires an expression on each side of the OR operator.

```
%macro operator(state);
   %if &state = OR %then %put State is Oregon;
      %else %put State is &state;
%mend operator;

%operator(PA)
```

The %IF statement resolves to
%if PA= OR %then,
which is not allowed.

%STR

%STR can be used to tell the macro facility to treat the value OR as text instead of an operator.

```
%macro operator(state);
   %if &state = %STR(OR) %then %put State is Oregon;
      %else %put State is &state;
%mend operator;

%operator(PA)
```

%STR AND COMPILE TIME

The %STR function is considered a compile time function. Values quoted by %STR are most often values that we type and can be seen by the programmer.

%BQUOTE

%BQUOTE is an execution time function. %BQUOTE and %NRBQUOTE mask values during execution of a macro or a macro language statement in open code.

If a function is an execution time function, you cannot see the special character. This means that the macro facility supplied the value at execution time.

%BQUOTE

The following code produces an error message.

```
%macro operator(state);
   %if &state = NC %then %put State is North Carolina;
       %else %put State is &state;
%mend operator;

%operator(OR)
```

%BQUOTE

The %IF statement resolves as follows:

```
%if OR = NC %then %put State is North Carolina;
```

%BQUOTE

%BQUOTE can be used to quote the value of state at execution time.

```
%macro operator(state);
   %if %bquote(&state) = NC %then %put State is North Carolina;
      %else %put State is &state;
%mend operator;

%operator(OR)
```

%SUPERQ

The %SUPERQ function locates the macro variable named in its argument and quotes the value of that macro variable without permitting any resolution to occur.

%SUPERQ(*argument* **)**

Note the absence of an & in the argument.

%SUPERQ

```
%let corpname=%str(Smith&Jones);
%put corpname is: &corpname;

%let testvar=%superq(corpname);
%put value of testvar is: &testvar;
```

%SUPERQ allows resolution to the value Smith&Jones without trying to resolve &Jones.

- SAS® Macro Language Reference

http://support.sas.com/documentation/onlinedoc/base/index.html

- Papers & SAS Notes

http://www2.sas.com/proceedings/sugi30/130-30.pdf
http://www2.sas.com/proceedings/sugi30/237-30.pdf

- SAS Training

https://support.sas.com/edu/schedules.html?id=246&

- RSS & Blogs

http://support.sas.com/community/rss/index.html

http://blogs.sas.com

- Discussion Forums

http://communities.sas.com/index.jspa

It's where you'll find the latest information on:

- New training courses and services
- Special offers and discounts
- The latest course schedules
- New training locations
- Events and conferences
- SAS certification news
- And, much more.

Everything you need – in one place.
Visit and bookmark it today.

THANK YOU FOR ATTENDING!

sas.com