ABSTRACT
Description: This paper reviews the three looping constructs: loop-repeat, do until and do while and offers examples of their use. The purpose of this paper is to provide both pseudo-code and examples so that programmers may understand the difference in logic and make an appropriate choice for their algorithm. The two data step loop processing verbs: continue (return to loop-top), and leave (exit) are illustrated. Macro examples using %goto are shown for continue and leave. The Whitlock subsetting loop — also known as the Do-Whitlock (DOW) loop — and double-DOW are illustrated.

Audience: intermediate users and macro programmers.

Keywords: do until, do while, Do-Whitlock, double DOW, DOW, until last.by-var, loop, repeat, subsetting, until, while, Whitlock do-loop

Information:

<table>
<thead>
<tr>
<th>Loop Type</th>
<th>Pseudo-Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop</td>
<td>loop . . . repeat using goto</td>
</tr>
<tr>
<td>Do Until</td>
<td>do until(...)</td>
</tr>
<tr>
<td>Do While</td>
<td>do while(...)</td>
</tr>
<tr>
<td>Test in Loop</td>
<td>if skip condition then . . . continue</td>
</tr>
<tr>
<td></td>
<td>if exit condition then . . . leave</td>
</tr>
<tr>
<td>Macro</td>
<td>macro %goto</td>
</tr>
<tr>
<td>Do Whitlock</td>
<td>do until(last.var)</td>
</tr>
</tbody>
</table>

Contents

Do Which? 3

Data Step Loops .......................................................... 3
Macro Loops .............................................................. 5

Testing During Loop 6

Using Logic In Conditions 8

Conclusion 10

References ................................................................. 11
Introduction

Overview

SAS® software provides two loop control verbs: until and while. The difference between the two keywords is that while tests its condition at the top of the loop and until tests its condition at the bottom on the loop. This is not obvious because the syntax requires both verbs to come after the keyword do: do while(...) do until(...).

Many questions to the SAS-L listserv are from beginning programmers who do not understand the difference between these two loop control constructs nor the difference in logic needed to implement the same algorithm.

This paper provides a basic pseudo-code algorithm with code examples illustrating the loop repeat, do until, and do while implementations.

The Loop-Repeat Algorithm

Basics: This is the basic pseudo-code of a loop-repeat block. All algorithms implement these eight steps.

```
1 initial: assignment(s)
2 loop :
3      pre-test assignment(s)
4    test : if condition then goto done
5      post-test assignment(s)
6  iterate: assignment
7 repeat : goto loop
8 done :
```

Enhancements: SAS provides extensions within its loop-repeat block.

```
1 initial: assignment(s): e.g.: allocate array(s)
2 assign : Index = value-1
3 loop : *do;
4 test.1 : if Index gt value-last then goto done
5    test.2 : if while-condition then goto done
6      pre-test assignment(s)
7    test.3 : if continue-condition then goto iterate
8    test.4 : if leave-condition then goto done
9      post-test assignment(s)
10    test.5 : if until-condition then goto done
11  iterate: Index = next value
12 repeat : *end; goto loop
13 done :
```

Dorfman [3 sesug2002 MagnificentDo] discusses the details of the iteration process where the loop control variable Index is initialized with the from value and incremented using the by value.
Do Which?

Overview

Examples are provided for these loops:

- loop repeat with goto
- do while: evaluation at top of loop
- do until: evaluation at bottom of loop
- iteration: do I

The difference between while and until is obfuscated by their placement at the top of the loop construct. As shown in loop algorithm enhancements, above, the while-condition is evaluated at the top of the loop, test.2, line 5, whereas the until-condition is evaluated at the bottom of the loop, test.5, line 10.

The following examples show that care must be taken in understanding the logical operators (see Comparison Operators in SAS Language Reference, Concepts) used in the while(...) and until(...) tests. Compare the sets of values in each and note that they are exclusive: lt: less than, ge: greater than or equal.

Data Step Loops

Loop Repeat

Building a loop in the data step is accomplished using labels and goto statements. This illustrates each of the steps in the pseudo-code loop-repeat algorithm shown above.

Note: all statements shown in this section are in file do-loops.sas; the form of iteration statements: I+ +1, indicate that each index variable is retained.

```
1 *initial:; I = 1;
2 loop:  put I=;
3 if I eq 2 then goto done;
4 *iterate:; I+ +1;
5 *repeat :; goto loop;
6 done:  put 'loop-repeat: ' I=;

log:
1 I=1
2 I=2
3 loop-repeat: I=2
```

continued on next page
**Do While**

The while test is evaluated at the top of the loop.

```
1 J = 1;
2 do while(J lt 3);
3     put J=;
4     J+ +1;
5 end;
6 put 'do J: ' J=;
```

*log:* note the logical operator is *lt:* less than.

```
1 J=1
2 J=2
3 do J: J=3
```

---

**Do Until**

The until test is evaluated at the bottom of the loop.

```
1 K = 1;
2 do until(K ge 3);
3     put K=;
4     K+ +1;
5 end;
6 put 'do K: ' K=;
```

*log:* note the logical operator is *ge:* greater than or equal.

```
1 K=1
2 K=2
3 do K: K=3
```

---

**Do Iterate**

Compare the iteration with the do until and do while examples above. The do statement iterates the index variable *L.*

```
1 L = 0;
2 do L = 1 to 2;
3     put L=;
4 end;
5 put 'do L: ' L=;
```

*log:*

```
1 L=1
2 L=2
3 do L: L=3
```
Macro Loops

Macro loops follow the same logic as the data step loops.

Note: all statements shown in this section are in file do-loops.sas.

Loop Repeat:

```
%local I; %let I = 1;
%loop: %put I=&I.;
%*test; %if &I. eq 2 %then %goto done;
%*iterate; %let I = %eval(&I. +1);
%*repeat; %goto loop;
%done: %Put loop-repeat: I=&I.;
```

log:
```
1 I=1
2 I=2
3 loop-repeat: I=2
```

Do While:
```
%local J; %let J = 1;
%do %while(&J lt 3);
%put J=&J.;
%let J = %eval(&J. +1);
%end;
%put do J: J=&J.;
```

log: note the logical operator is \texttt{lt}: less than.
```
1 J=1
2 J=2
3 do J: J=3
```

Do Until:
```
%local K; %let K = 1;
%do %until(&K ge 3);
%put K=&K.;
%let K = %eval(&K. +1);
%end;
%put do K: K=&K.;
```

log: note the logical operator is \texttt{ge}: greater than or equal.
```
1 K=1
2 K=2
3 do K: K=3
```

Do Iterate:
```
%local L;
%do L = 1 %to 2;
%put L=&L;
%end;
%put do L: L=&L.;
```

log:
```
1 L=1
2 L=2
3 do L: L=3
```
Testing During Loop

Overview

Some loop processing algorithms require either a skip pattern or a conditional exit.

- **continue**: skip post-test processing, return to top of loop
- **leave**: conditional exit

Note: all statements shown in this section are in file do-loop-tests.sas.

Continue

Return to top of loop.

```
DATA _Null_
  do I = 1 to 3;
  put I= ‘pre-test’;
  if I le 2 then continue;
  put I= ‘post test’;
  end;
  put ‘done continue: ’ I= ;
log:
1 I=1 pre-test
2 I=2 pre-test
3 I=3 post test
4 done continue: I=4
```

Leave

Conditional exit.

```
do J = 1 to 3;
  put J= ‘pre-test’;
  if J gt 2 then leave;
  put J= ‘post test’;
  end;
  put ‘done leave: ’ J= ;
log:
1 J=1 pre-test
2 J=1 post test
3 J=2 pre-test
4 J=2 post test
5 J=3 pre-test
6 done leave: J=3
```
Macro %goto

There are no comparable %continue nor %leave statements in the macro language. However, as shown in the next examples, they can be implemented using labels: %continue: and %leave:, and %goto.

Note: all statements shown in this section are in file do-loop-tests.sas.

Continue : return to top of loop

```sas
%Macro Do_Tests(i=,j=);
%do I = 1 %to 3;
  %put I=&I. pre-test;
  %if &I le 2 %then %goto continue;
  %put I=&I. post test;
  %continue:
%end;
%put done continue: I=&I. ;
```

log:

```plaintext
1 I=1 pre-test
2 I=2 pre-test
3 I=3 pre-test
4 I=3 post test
5 done continue: I=4
```

Leave : conditional exit

```sas
%do J = 1 %to 3;
  %put J=&J. pre-test;
  %if &J. gt 2 %then %goto leave;
  %put J=&J. post test;
%end;
%leave:
%put done leave: J=&J. ;
```

log:

```plaintext
1 J=1 pre-test
2 J=1 post test
3 J=2 pre-test
4 J=2 post test
5 J=3 pre-test
6 done leave: J=3
```
Using Logic In Conditions

Overview

The following are equivalent:

\[
\text{do until(EndoFile)} \\
\quad \text{do while(not EndoFile)}. \\
\]

This is an important difference to understand: that the same algorithm can be implemented using the two verbs, but the logic is different because of when the condition is evaluated.

Note that boolean values are in (False, not False). The preceding statement means that only zero (0) is false; until and while evaluation treats negative as well as positive values as True.

Data Steps

Note: all statements shown in this section are in file do-boolean.sas.

Do Until(EndoFile) : evaluate EndoFile at bottom of loop

```sas
DATA do_until_endofile;
  do until(EndoFile);
    set SAShelp.Class end = EndoFile;
    output;
  end;
  stop;

log:
  NOTE: There were 17 observations read from the data set SASHELP.CLASS.
  NOTE: The data set WORK.DO_UNTIL_ENDOFILE has 17 observations and 5 variables.
```

Do While(Not EndoFile) : evaluate EndoFile at top of loop

```sas
DATA do_while_not_endofile;
  do while(not EndoFile);
    set SAShelp.Class end = EndoFile;
    output;
  end;
  stop;

log:
  NOTE: There were 17 observations read from the data set SASHELP.CLASS.
  NOTE: The data set WORK.DO_WHILE_NOT_ENDOFILE has 17 observations and 5 variables.
```
Combining Iteration With Loop Control

An iteration loop may be combined with an until condition. As noted above care should be taken to ensure that the variable tested in the until clause has boolean values, — in (0,1) — only.

Note: all statements shown in this section are in file do-l-eq-until.sas.

```
DATA do_I_eq_until;
*initialize; retain Done 0;
do I = 1 to 3 until(Done);
   put I=;
output;
   Done = (I ge 2);*boolean;
end;
put 'done: ' I=;
stop;
run;
```

log: done: I=2 shows the evaluation of the until is before the iteration.

```
1 I=1
2 I=2
3 done: I=2
```

Whitlock subsetting: the DOW loop

Ian Whitlock Whitlock [3] posted a solution to SAS-L with a do until(last.id) which has come to be known as the Do-Whitlock (DOW) loop.

```
do until(last.byvar) :
 data t ( keep = id v1 - v6 ) ;
 array v (2,3) ;
 array var ( 3) ;
do until { last.id } ;
set w ;
by id ;
do i = 1 to dim ( var ) ;
v ( method , i ) = var ( i ) ;
end ;
end ;
run ;
```

```
do until(first.by-var) :
 The DOW can be used with first.by-var as well.
```

```
DATA do_Whitlock_first;
do until(first.Sex);
set Class;
by Sex;
end;
output;
put Sex= Name=;
run;
```

log:

```
1 Sex=F Name=Alice
2 Sex=M Name=Alfred
3 NOTE: There were 17 observations read from the data set WORK.CLASS.
4 NOTE: The data set WORK.DO_WHITLOCK_FIRST
5 has 2 observations and 5 variables.
```
Double-DOW

Paul Dorfman has posted several examples to SAS-L using the DOW algorithm and showing expanded usages.

Note: The statements in this example are in file do-double-dow.sas.

```
DATA do_double_Class;
  do until(first.Sex);
    set Class end = EndoFile;
    by Sex;
    end;
  put Sex= Name=;
  output;
  do until(last.Sex);
    set Class end = EndoFile;
    by Sex;
    end;
  put Sex= Name=;
  output;
run;
```

log:

```
1  Sex=F Name=Alice
2  Sex=F Name=Mary
3  Sex=M Name=Alfred
4  Sex=M Name=Thomas
5  NOTE: There were 11 observations read from the data set WORK.CLASS.
6  NOTE: There were 17 observations read from the data set WORK.CLASS.
7  NOTE: The data set WORK.DO_DOUBLE_CLASS has 4 observations and 5 variables.
```

Conclusion

Summary

The two do-loop verbs until and while are distinguished by the execution of their loop-exit tests. To implement the same algorithm requires using different test conditions. The DOW and double-DOW are an interesting use of the do until loops.

Suggested Readings

Cassell [1] sugi29.046 shows a use of the DOW with the prx (Perl) functions.


Dunn and Chung [4] pharmasug2005.tu09, examples 9-10, show how to calculate sum of variables using double-DOW.

A current set of references for do until(last.var) is maintained on the sas community wiki: et al [5] sas-wiki.do-until-last-var

Other versions of this paper are:

- Fehd [8] pharma2008.cc01

To get the code examples in this paper search Fehd [6] sas-wiki.do-which for Do-Which-Loop-Until-or-While.zip.
References


Author Information

Author: Ronald J. Fehd
Centers for Disease Control
4770 Buford Hwy NE
Atlanta GA 30341-3724
mailto:RJF2@cdc.gov

about the author:

education:
B.S. Computer Science, U/Hawaii, 1986
SAS User Group conference attendee since 1989
SAS-L reader since 1994

experience:
programmer: 20+ years
data manager at CDC, using SAS: 17+ years
author: 30+ SUG papers

SAS-L:
author: 5,000+ messages to SAS-L since 1997

SAS Institute Information

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.