A Macro to Word Wrap Long Text Strings into a SAS® Array
H. Ian Whitlock, Westat Inc.

Introduction

When reading data and associated comment fields, it may be necessary to handle character strings longer than 200 bytes. It is a simple task to store the string in 200 byte chunks, but it is harder to break up the chunks into an array of print sized lines which are space filled at the end so that no word is split between two lines.

A new SAS Institute book [1] on report writing gives such a macro, but it is three pages long, has 5 calls to a one page embedded macro, and uses GOTO's. Hence the algorithm is obscured, and too much SAS code is generated.

This paper presents a one page macro using only SAS code, macro variable references and macro invocation to develop a simple direct algorithm illustrating how to use pointers to an arbitrarily long substring stored as part of a character array. Two one line "function" macros, %PX (PTR) and %PP (PTR) are used in a key role to convert an absolute string pointer to the index of the corresponding array element and a relative pointer to the position within that element.

Test and Illustrated Use

/* read in 1 long string, apply %wordwrap, then write resulting print ready array */
data _null_
  length s1 - 5 $ 40 ;
  input (s1-5) ($char40. ) ;
  array in (5) s1 - s5 ;
  array out (15) $ 20 _ temporary_;
  do i = 1 to dim (in) ; /* show input */
      put in (i) = ;
  end;

  %wordwrap
    (p=in , q=out , plen=40 , qlen=20 )
  do i = 1 to dim (out) ; /* show output */
      put out (i) = ;
  end;

The Macro

%macro wordwrap ( 

  p= , /* input array name */ 
  plen=200, /* len vars in input array */ 
  q= , /* output array name */ 
  qlen = /* len vars in output array*/ 
); 

  /* ------------------------------------------------------------------*/ 
  Purpose: Move text stored in an array from a long character string (usually over 200 bytes) where array elements need not mark word boundaries to an array with space filling at the end so that no word crosses an array element.

  Usage Notes: Parameters P, Q, and QLEN are required. If a word in &p is longer than &qlen, abort. Drop variables beginning with a double underscore.

  Working variables:
  _px , _qx - index to resp arrays
  _pb , _qpb - rel beg ptrs within element
  _pe - rel end ptr within element
  _pb , _pe - abs (full string) begin and end ptrs to string of &p
  _pc - one byte char val from &p
  _sublen - length of substring

  Basic algorithm: initialize _pe to 0
  loop over index of &q until finished
      set _pe to _pe + 1
      set _pe to maximum that will fit in &q

  cards ;
  aaaaaaaaaaaa10 aaaaaaaaaaaa11 aaaaaaaaaaa8 aaa
  aaaa bbbbbbbbbbb10 bbbbbbb8 bbbbbbbbbbb10
  cccccccccccc11 cccccccccccc12 ccccccccc8
dddddddeeed11 dddddddeed8 dddddddeed
  10 eeeeeee eeeeeee eeeeeee eeeeeee e
run ;
when needed back up to a space
move substring to &q (using two or
more moves when the substring begins
and ends on different elements)
end loop

•.
drop_;;
_pe = 0;
do _qx = lbound(&q) to hbound(&q)
   until (_pe >= &plen * dim(&p)) ;
   _pb = _pe + 1;
if &p (%px(_pb)) = "" then
do;
   _qx = _qx - 1;
   leave;
end;
else
do;
   _pe = min (_pe, &plen * dim(&p),
               _pe + &plen - _rpb + 1);
/* skip backup when _pe is at end */
if _pe = &plen * dim(&p) then;
else
   if substr(&p(%px(_pe)), 1) ^= "" then
do;
   /* back up to first blank */
do_pe = _pe to _pb by -1
      until (_pc ^= "" or _pe < _pb);
   _pc = substr ( &p(%px(_pe)),
                  %pp(%px(_pe)), 1) ;
end;
if _pe < _pb then
do; /* token too long */
   _px = %px(_pb);
   put 'WORDRAP: token too long"
      "- will abort"
      "&p(_px + (-1)) ='
      &p(_px) $char&plen..;
   _px = _px + 1;
   put '
      _px + (-1) ='
      &p(_px) $char&plen..;
   abort 99 ;
end;
else
   /* move substring to q array */
   _rqb = 1;
do while (%px(_pb) ^= %px(_pe)) ;
   _rpb = %pp(_pb);
   _sublen = &plen - _rpb + 1;
   substr (&q (_qx), _rpb ) = substr( &p(%px(_pb)),
                                      _rpb,
                                      _sublen) ;
   _rqb = _sublen + 1;
   _pb = (_px) * &plen + 1;
end;
/* move part on the end element */
substr (&q (_qx), _rqb ) = substr ( &p(%px(_pe)),
                                     %pp(_pb), _pe - _pb + 1) ;
end;
end;
do _qx = _qx + 1 to dim (&q) ;
   _q(_qx) = ";
end;
%mend wordwrap ;

%macro pp ( ptr);
   /* rel ptr to p array subst */
   mod (( &ptr - 1 ), &plen) + 1
%mend pp ;
%macro px ( ptr)
   /* p array index of pointer */
   int (( &ptr - 1 ) / &plen) + 1
%mend px ;

The author can be contacted by mail or e-mail

H. Ian Whitlock
Westat
1650 Research Boulevard
Rockville, MD 20850-3129

Whitloi@westat.com


SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.