```
Module WordPkg
!
!  a collection of routines for manipulating strings
!
!  most are analogous to intrinsic routines in REXX
!
Contains
   FUNCTION WORDS(Line)
   Integer WORDS
!
!  returns number of blank delimited words in Line
!
!--------------------------------------------------------------------
   FUNCTION WordIndex(str, NW)
   Integer                      :: WordIndex
   CHARACTER(LEN=*), Intent(IN) :: str
   Integer,          Intent(IN) :: NW
!
!  Returns the character position of the first character of blank delimited
!  word number NW
!  If NW refers to a non-existing word, 0 is returned.
!--------------------------------------------------------------------
   FUNCTION WordPos(phrase, str, start)
   Integer                       :: WordPos
   CHARACTER(LEN=*), Intent(IN)  :: phrase, str
   Integer, Intent(IN), Optional :: start
!
!  Returns the word number in str which indicates where phrase begins
!  provided that phrase is a subphrase of string.
!  Otherwise 0 is returned.
!  Optional argument start (default 1) is the word in str at which the search starts
!
!--------------------------------------------------------------------
   FUNCTION Word(str, n, SEP)  Result(WN)
   Character(Len=*), Intent(IN)           :: str
   Integer,          Intent(IN)           :: n
   CHARACTER(LEN=*), Intent(IN), Optional :: SEP
   Character(Len=Len(str))      :: WN
!
!  returns word n from str
!     where words are delimited by any character in SEP
!               (default: SEP = ' ')
!--------------------------------------------------------------------
   FUNCTION DelWord(str, start, length)  Result(strnew)
   Character(Len=*),  Intent(IN) :: str
   Integer,           Intent(IN) :: start
   Integer, Optional, Intent(IN) :: length
   Character(Len=Len(str))       :: strnew
!
!  DelWord(string,start[,length])
!   Returns a new string obtained from str by
!      removing length words and all blanks between them, from string,
!      starting at word number start.
!   The default value for length is the rest of the string.
!   All consecutive spaces immediately after the last deleted word, but no
!   spaces before the first deleted word are removed.
!   Nothing is removed if length is zero.
!--------------------------------------------------------------------
   SUBROUTINE WordParse(LINE, NWORDS, WDS, SEP)
      CHARACTER(LEN=*), Intent(IN)           :: Line
      INTEGER,          Intent(INOUT)        :: NWORDS
      CHARACTER(LEN=*), Dimension(:)         :: WDS
      CHARACTER(LEN=*), Intent(IN), Optional :: SEP
!
!     returns NWORDS = number of words in LINE
!         where words are delimited by any character in SEP
!               (default: SEP = ' ')
!         and these words in WDS(1:NWORDS)
!     If there are more words in LINE than is the dimension of WDS,
!         NWORDS is set to the dimension of WDS
!
!--------------------------------------------------------------------
```

```
   FUNCTION Lower(str, start, length)  Result (tr)
      IMPLICIT NONE
      CHARACTER(LEN=*), Intent(IN)           :: str
      Integer,          Intent(IN), Optional :: start, length
      CHARACTER(LEN=Len(str))                :: tr
!
!     translates to lower case the substring of str that starts at start
!     and has length characters
!     length defaults to the rest of the string.  length must be a non-negative integer
!     start defaults to 1.  start must be a postive integer.
!     If  length + start - 1  is greater than the length of str,  the substring is
!     assumed to be to the end of str, i.e., length is assumed to be length + start - 1
!     Examples:
!        Upper('Happy Camper')           returns   'HAPPY CAMPER'
!        Upper('fred', 2)                returns   'fRED'
!        Upper('fred', 1, 1)             returns   'Fred'
!        Upper('fred', 2, 10)            returns   'fRED'
!-----------------------------------------------------------------------
   FUNCTION Upper(str, start, length)  Result (tr)
      IMPLICIT NONE
      CHARACTER(LEN=*), Intent(IN)           :: str
      Integer,          Intent(IN), Optional :: start, length
      CHARACTER(LEN=Len(str))                :: tr
!
!     translates to upper case the substring of str that starts at start
!     and has length characters
!     length defaults to the rest of the string.  length must be a non-negative integer
!     start defaults to 1.  start must be a postive integer.
!     If  length + start - 1  is greater than the length of str,  the substring is
!     assumed to be to the end of str, i.e., length is assumed to be length + start - 1
!-----------------------------------------------------------------------
   FUNCTION Translate(str, tableout, tablein, pad)  Result (tr)
      CHARACTER(LEN=*), Intent(IN)           :: str
      CHARACTER(LEN=*), Intent(IN), Optional :: tableout, tablein
      CHARACTER(LEN=1), Intent(IN), Optional :: pad
      CHARACTER(LEN=Len(str))      :: tr
!
!     translates characters in str according to tables tablein and tableout
!       If a character of str is in tablein, it is changed to the corresponding
!       character in tableout; otherwise it is not changed.
!
!     If tablein and tableout are not specified, the characters of str are translated
!     to upper case
!
!     If Len(tableout) < Len(tablein), tableout is padded with the character pad
!     The default pad character is a blank.
!
!     If tablein is not specified, it is set to the ASCII character sequence
!     given by Achar(0) - Achar(255)
!
!-----------------------------------------------------------------------
   FUNCTION IntegerWord(W)
    Logical IntegerWord
    CHARACTER(LEN=*), Intent(IN) :: W
!
!     returns  .TRUE.  if W is compatible with being an integer
!-----------------------------------------------------------------------
   FUNCTION NaturalWord(W)
    Logical NaturalWord
    CHARACTER(LEN=*), Intent(IN) :: W
!
!     returns  .TRUE.  if W is compatible with being a natural number,
!        i.e., unsigned  integer
!     N.B. 0 is considered a natural number, contrary to some conventions
!-----------------------------------------------------------------------
   FUNCTION RealWord(W)
    Logical RealWord
    CHARACTER(LEN=*), Intent(IN) :: W
!
!     returns  .TRUE.  if W is compatible with being a real
!
!-----------------------------------------------------------------------
```

```
   FUNCTION Space(str, length, padchar)  Result (trs)
      CHARACTER(LEN=*), Intent(IN)           :: str
      Integer,          Intent(IN), Optional :: length
      CHARACTER(LEN=1), Intent(IN), Optional :: padchar
      CHARACTER(LEN=Len(str)*10)             :: trs
!
!     string is returned, stripped of any trailing or leading blanks,
!     and any consecutive blanks inside string replaced by length(default 1)
!     occurences of character padchar (default blank)
!
!     Note that trs can be longer than str if length > 1
!     Hopefully declaring tr 10 times longer than str is sufficient in practice
!
!---------------------------------------------------------------------
   FUNCTION Strip(str, ltb, char)  Result (trs)   rs)
      CHARACTER(LEN=*), Intent(IN)           :: str
      Character(LEN=*), Intent(IN), Optional :: ltb
      CHARACTER(LEN=1), Intent(IN), Optional :: char
      CHARACTER(LEN=Len(str))                :: trs
!
!     string is returned, stripped of any trailing and/or leading occurences
!     of char (default blank)
!     if ltb(1:1) is L only leading  occurences are stripped
!     if ltb(1:1) is T only trailing occurences are stripped
!     inter-word occurences of chara are not affected
!
!---------------------------------------------------------------------
   FUNCTION Copies(str, n)  Result (tr)
      CHARACTER(LEN=*), Intent(IN)   :: str
      Integer,          Intent(IN)   :: n
      CHARACTER(LEN=Len(str)*n)      :: tr
!
!     Returns a string with n concatenated copies of str.
!     Note: there is no space inserted between the copies
!           n = 0   or  n > 0
!
!---------------------------------------------------------------------
   FUNCTION Center(str, n, pad)  Result(tr)
      IMPLICIT NONE
      CHARACTER(LEN=*), Intent(IN)           :: str
      Integer,          Intent(IN)           :: n
      CHARACTER(LEN=1), Intent(IN), Optional :: pad
      CHARACTER(LEN=n)                       :: tr
!
!     Returns a string of length n with the input string str centered in it
!     If n > length(str), str is padded with the character pad (default space)
!     If n < length(str), characters are removed from each end
!     If the total number of characters to be added or removed is odd,
!        one character more is added or removed from the right end than from the left.
!
!---------------------------------------------------------------------
   FUNCTION WordFormat(x)  Result(WF)
      Character(Len=30) :: WF
!
!     returns string with character representation of x
!     where x may be logical, integer, real, or double precision
!
!---------------------------------------------------------------------
   FUNCTION InsertWord(strin, strold, start)  Result(strnew)
   Character(Len=*),  Intent(IN) :: strin, strold
   Integer, Optional, Intent(IN) :: start
   Character(Len=Len(strin)+Len(strold)+2) :: strnew
!
!    Returns a new string obtained from strold by
!       inserting strin, preceded and followed by a blank, in strold
!       after word number start.
!    The default value for start is 0, i.e., insert before the first word
!    If start is greater than the number of words in strold,
!       strin is added at the end
!
!---------------------------------------------------------------------
```

```
   FUNCTION Right(string, N, padchar)  Result(WN)
   Character(Len=*), Intent(IN) :: string
   Integer,          Intent(IN) :: N
   Character(Len=1), Optional   :: padchar
   Character(Len=Abs(N))        :: WN
!
!  returns right-most N characters of Trim(string) padded on left with padchar
!  default for padchar is a blank
!
!  In principle, N cannot be negative, but here use sign of N:
!     If N < 0, use the entire string rather than Trim(string)
!     e.g.,  Right('abcd  ',  3)  returns  'bcd'
!     e.g.,  Right('abcd  ', -3)  returns  'd  '
!
!--------------------------------------------------------------------
   FUNCTION Left(string, N, padchar)  Result(WN)
   Character(Len=*), Intent(IN) :: string
   Integer,          Intent(IN) :: N
   Character(Len=1), Optional   :: padchar
   Character(Len=Abs(N))        :: WN
!
!  returns left-most N characters of Trim(string) padded on right with padchar
!  default for padchar is a blank
!
!  In principle, N cannot be negative, but here use sign of N:
!     If N < 0, use the entire string rather than Trim(string)
!     e.g.,  Left('a ',  3, '%')  returns  'a%%'
!     e.g.,  Left('a ', -3, '%')  returns  'd %'
!
```