```fortran
module precision

! Real kinds

integer, parameter :: kr4 = selected_real_kind(6,37)       ! single precision re
al
integer, parameter :: kr8 = selected_real_kind(15,307)     ! double precision re
al

! Integer kinds

integer, parameter :: ki4 = selected_int_kind(9)           ! single precision in
teger
integer, parameter :: ki8 = selected_int_kind(18)          ! double precision in
teger

!Complex kinds

integer, parameter :: kc4 = kr4                            ! single precision co
mplex
integer, parameter :: kc8 = kr8                            ! double precision co
mplex

end module precision

module strings

use precision

private :: value_dr,value_sr,value_di,value_si
private :: write_dr,write_sr,write_di,write_si
private :: writeq_dr,writeq_sr,writeq_di,writeq_si

interface value  ! Generic operator for converting a number string to a
                 ! number. Calling syntax is 'call value(numstring,number,ios)'
                 ! where 'numstring' is a number string and 'number' is a
                 ! real number or an integer (single or double precision).

   module procedure value_dr
   module procedure value_sr
   module procedure value_di
   module procedure value_si
end interface

interface writenum  ! Generic  interface for writing a number to a string. The
                    ! number is left justified in the string. The calling syntax
                    ! is 'call writenum(number,string,format)' where 'number' is
                    ! a real number or an integer, 'string' is a character strin
g
                    ! containing the result, and 'format' is the format desired,

                    ! e.g., 'e15.6' or 'i5'.
   module procedure write_dr
   module procedure write_sr
   module procedure write_di
   module procedure write_si
end interface

interface writeq  ! Generic interface equating a name to a numerical value. The
                  ! calling syntax is 'call writeq(unit,name,value,format)' wher
e
                  ! unit is the integer output unit number, 'name' is the variab
le
                  ! name, 'value' is the real or integer value of the variable,
                  ! and 'format' is the format of the value. The result written
to
                  ! the output unit has the form <name> = <value>.
   module procedure writeq_dr
   module procedure writeq_sr
```

```fortran
   module procedure writeq_di
   module procedure writeq_si
end interface


!**********************************************************************

contains

!**********************************************************************

subroutine parse(str,delims,args,nargs)

! Parses the string 'str' into arguments args(1), ..., args(nargs) based on
! the delimiters contained in the string 'delims'. Preceding a delimiter in
! 'str' by a backslash (\) makes this particular instance not a delimiter.
! The integer output variable nargs contains the number of arguments found.

character(len=*) :: str,delims
character(len=len_trim(str)) :: strsav
character(len=*),dimension(:) :: args

strsav=str
call compact(str)
na=size(args)
do i=1,na
  args(i)=' '
end do
nargs=0
lenstr=len_trim(str)
if(lenstr==0) return
k=0

do
   if(len_trim(str) == 0) exit
   nargs=nargs+1
   call split(str,delims,args(nargs))
   call removebksl(args(nargs))
end do
str=strsav

end subroutine parse

!**********************************************************************

subroutine compact(str)

! Converts multiple spaces and tabs to single spaces; deletes control characters
;
! removes initial spaces.

character(len=*):: str
character(len=1):: ch
character(len=len_trim(str)):: outstr

str=adjustl(str)
lenstr=len_trim(str)
outstr=' '
isp=0
k=0

do i=1,lenstr
  ch=str(i:i)
  ich=iachar(ch)

  select case(ich)

    case(9,32)     ! space or tab character
      if(isp==0) then
```

```fortran
        k=k+1
        outstr(k:k)=' '
      end if
      isp=1

    case(33:)      ! not a space, quote, or control character
      k=k+1
      outstr(k:k)=ch
      isp=0

  end select

end do

str=adjustl(outstr)

end subroutine compact

!**********************************************************************

subroutine removesp(str)

! Removes spaces, tabs, and control characters in string str

character(len=*):: str
character(len=1):: ch
character(len=len_trim(str))::outstr

str=adjustl(str)
lenstr=len_trim(str)
outstr=' '
k=0

do i=1,lenstr
  ch=str(i:i)
  ich=iachar(ch)
  select case(ich)
    case(0:32)  ! space, tab, or control character
         cycle
    case(33:)
      k=k+1
      outstr(k:k)=ch
  end select
end do

str=adjustl(outstr)

end subroutine removesp

!**********************************************************************

subroutine value_dr(str,rnum,ios)

! Converts number string to a double precision real number

character(len=*)::str
real(kr8)::rnum
integer :: ios

ilen=len_trim(str)
ipos=scan(str,'Ee')
if(.not.is_digit(str(ilen:ilen)) .and. ipos/=0) then
   ios=3
   return
end if
read(str,*,iostat=ios) rnum

end subroutine value_dr
```

```fortran
!**********************************************************************

subroutine value_sr(str,rnum,ios)

! Converts number string to a single precision real number

character(len=*)::str
real(kr4) :: rnum
real(kr8) :: rnumd

call value_dr(str,rnumd,ios)
if( abs(rnumd) > huge(rnum) ) then
  ios=15
  return
end if
if( abs(rnumd) < tiny(rnum) ) rnum=0.0_kr4
rnum=rnumd

end subroutine value_sr

!**********************************************************************

subroutine value_di(str,inum,ios)

! Converts number string to a double precision integer value

character(len=*)::str
integer(ki8) :: inum
real(kr8) :: rnum

call value_dr(str,rnum,ios)
if(abs(rnum)>huge(inum)) then
  ios=15
  return
end if
inum=nint(rnum,ki8)

end subroutine value_di

!**********************************************************************

subroutine value_si(str,inum,ios)

! Converts number string to a single precision integer value

character(len=*)::str
integer(ki4) :: inum
real(kr8) :: rnum

call value_dr(str,rnum,ios)
if(abs(rnum)>huge(inum)) then
  ios=15
  return
end if
inum=nint(rnum,ki4)

end subroutine value_si

!**********************************************************************

subroutine shiftstr(str,n)

! Shifts characters in in the string 'str' n positions (positive values
! denote a right shift and negative values denote a left shift). Characters
! that are shifted off the end are lost. Positions opened up by the shift
! are replaced by spaces.

character(len=*):: str
```

```
lenstr=len(str)
nabs=iabs(n)
if(nabs>=lenstr) then
  str=repeat(' ',lenstr)
  return
end if
if(n<0) str=str(nabs+1:)//repeat(' ',nabs)  ! shift left
if(n>0) str=repeat(' ',nabs)//str(:lenstr-nabs)  ! shift right
return

end subroutine shiftstr

!**********************************************************************

subroutine insertstr(str,strins,loc)

! Inserts the string 'strins' into the string 'str' at position 'loc'.
! Characters in 'str' starting at position 'loc' are shifted right to
! make room for the inserted string. Trailing spaces of 'strins' are
! removed prior to insertion

character(len=*):: str,strins
character(len=len(str))::tempstr

lenstrins=len_trim(strins)
tempstr=str(loc:)
call shiftstr(tempstr,lenstrins)
tempstr(1:lenstrins)=strins(1:lenstrins)
str(loc:)=tempstr
return

end subroutine insertstr

!**********************************************************************

subroutine delsubstr(str,substr)

! Deletes first occurrence of substring 'substr' from string 'str' and
! shifts characters left to fill hole. Trailing spaces or blanks are
! not considered part of 'substr'.

character(len=*):: str,substr

lensubstr=len_trim(substr)
ipos=index(str,substr)
if(ipos==0) return
if(ipos == 1) then
   str=str(lensubstr+1:)
else
   str=str(:ipos-1)//str(ipos+lensubstr:)
end if
return

end subroutine delsubstr

!**********************************************************************

subroutine delall(str,substr)

! Deletes all occurrences of substring 'substr' from string 'str' and
! shifts characters left to fill holes.

character(len=*):: str,substr

lensubstr=len_trim(substr)
do
   ipos=index(str,substr)
   if(ipos == 0) exit
   if(ipos == 1) then
```

```
      str=str(lensubstr+1:)
   else
      str=str(:ipos-1)//str(ipos+lensubstr:)
   end if
end do
return

end subroutine delall

!**********************************************************************

function uppercase(str) result(ucstr)

! convert string to upper case

character (len=*):: str
character (len=len_trim(str)):: ucstr

ilen=len_trim(str)
ioffset=iachar('A')-iachar('a')
iquote=0
ucstr=str
do i=1,ilen
  iav=iachar(str(i:i))
  if(iquote==0 .and. (iav==34 .or.iav==39)) then
    iquote=1
    iqc=iav
    cycle
  end if
  if(iquote==1 .and. iav==iqc) then
    iquote=0
    cycle
  end if
  if (iquote==1) cycle
  if(iav >= iachar('a') .and. iav <= iachar('z')) then
    ucstr(i:i)=achar(iav+ioffset)
  else
    ucstr(i:i)=str(i:i)
  end if
end do
return

end function uppercase

!**********************************************************************

function lowercase(str) result(lcstr)

! convert string to lower case

character (len=*):: str
character (len=len_trim(str)):: lcstr

ilen=len_trim(str)
ioffset=iachar('A')-iachar('a')
iquote=0
lcstr=str
do i=1,ilen
  iav=iachar(str(i:i))
  if(iquote==0 .and. (iav==34 .or.iav==39)) then
    iquote=1
    iqc=iav
    cycle
  end if
  if(iquote==1 .and. iav==iqc) then
    iquote=0
    cycle
  end if
  if (iquote==1) cycle
```

```fortran
  if(iav >= iachar('A') .and. iav <= iachar('Z')) then
    lcstr(i:i)=achar(iav-ioffset)
  else
    lcstr(i:i)=str(i:i)
  end if
end do
return

end function lowercase

!**********************************************************************

subroutine readline(nunitr,line,ios)

! Reads line from unit=nunitr, ignoring blank lines
! and deleting comments beginning with an exclamation point(!)

character (len=*):: line

do
  read(nunitr,'(a)', iostat=ios) line      ! read input line
  if(ios /= 0) return
  line=adjustl(line)
  ipos=index(line,'!')
  if(ipos == 1) cycle
  if(ipos /= 0) line=line(:ipos-1)
  if(len_trim(line) /= 0) exit
end do
return

end subroutine readline

!**********************************************************************

subroutine match(str,ipos,imatch)

! Sets imatch to the position in string of the delimiter matching the delimiter
! in position ipos. Allowable delimiters are (), [], {}, <>.

character(len=*) :: str
character :: delim1,delim2,ch

lenstr=len_trim(str)
delim1=str(ipos:ipos)
select case(delim1)
   case('(')
      idelim2=iachar(delim1)+1
      istart=ipos+1
      iend=lenstr
      inc=1
   case(')')
      idelim2=iachar(delim1)-1
      istart=ipos-1
      iend=1
      inc=-1
   case('[','{','<')
      idelim2=iachar(delim1)+2
      istart=ipos+1
      iend=lenstr
      inc=1
   case(']','}','>')
      idelim2=iachar(delim1)-2
      istart=ipos-1
      iend=1
      inc=-1
   case default
      write(*,*) delim1,' is not a valid delimiter'
      return
end select
```

```fortran
if(istart < 1 .or. istart > lenstr) then
   write(*,*) delim1,' has no matching delimiter'
   return
end if
delim2=achar(idelim2) ! matching delimiter

isum=1
do i=istart,iend,inc
   ch=str(i:i)
   if(ch /= delim1 .and. ch /= delim2) cycle
   if(ch == delim1) isum=isum+1
   if(ch == delim2) isum=isum-1
   if(isum == 0) exit
end do
if(isum /= 0) then
   write(*,*) delim1,' has no matching delimiter'
   return
end if
imatch=i

return

end subroutine match

!**********************************************************************

subroutine write_dr(rnum,str,fmt)

! Writes double precision real number rnum to string str using format fmt

real(kr8) :: rnum
character(len=*) :: str,fmt
character(len=80) :: formt

formt='('//trim(fmt)//')'
write(str,formt) rnum
str=adjustl(str)

end subroutine write_dr

!***********************************************************************

subroutine write_sr(rnum,str,fmt)

! Writes single precision real number rnum to string str using format fmt

real(kr4) :: rnum
character(len=*) :: str,fmt
character(len=80) :: formt

formt='('//trim(fmt)//')'
write(str,formt) rnum
str=adjustl(str)

end subroutine write_sr

!***********************************************************************

subroutine write_di(inum,str,fmt)

! Writes double precision integer inum to string str using format fmt

integer(ki8) :: inum
character(len=*) :: str,fmt
character(len=80) :: formt

formt='('//trim(fmt)//')'
write(str,formt) inum
str=adjustl(str)
```

```
end subroutine write_di

!***********************************************************************

subroutine write_si(inum,str,fmt)

! Writes single precision integer inum to string str using format fmt

integer(ki4) :: inum
character(len=*) :: str,fmt
character(len=80) :: formt

formt='('//trim(fmt)//')'
write(str,formt) inum
str=adjustl(str)

end subroutine write_si

!***********************************************************************

subroutine trimzero(str)

! Deletes nonsignificant trailing zeroes from number string str. If number
! string ends in a decimal point, one trailing zero is added.

character(len=*) :: str
character :: ch
character(len=10) :: exp

ipos=scan(str,'eE')
if(ipos>0) then
   exp=str(ipos:)
   str=str(1:ipos-1)
endif
lstr=len_trim(str)
do i=lstr,1,-1
   ch=str(i:i)
   if(ch=='0') cycle
   if(ch=='.') then
      str=str(1:i)//'0'
      if(ipos>0) str=trim(str)//trim(exp)
      exit
   endif
   str=str(1:i)
   exit
end do
if(ipos>0) str=trim(str)//trim(exp)

end subroutine trimzero

!***********************************************************************

subroutine writeq_dr(unit,namestr,value,fmt)

! Writes a string of the form <name> = value to unit

real(kr8) :: value
integer :: unit
character(len=*) :: namestr,fmt
character(len=32) :: tempstr

call writenum(value,tempstr,fmt)
call trimzero(tempstr)
write(unit,*) trim(namestr)//' = '//trim(tempstr)

end subroutine writeq_dr

!***********************************************************************
```

```
subroutine writeq_sr(unit,namestr,value,fmt)

! Writes a string of the form <name> = value to unit

real(kr4) :: value
integer :: unit
character(len=*) :: namestr,fmt
character(len=32) :: tempstr

call writenum(value,tempstr,fmt)
call trimzero(tempstr)
write(unit,*) trim(namestr)//' = '//trim(tempstr)

end subroutine writeq_sr

!***********************************************************************

subroutine writeq_di(unit,namestr,ivalue,fmt)

! Writes a string of the form <name> = ivalue to unit

integer(ki8) :: ivalue
integer :: unit
character(len=*) :: namestr,fmt
character(len=32) :: tempstr
call writenum(ivalue,tempstr,fmt)
call trimzero(tempstr)
write(unit,*) trim(namestr)//' = '//trim(tempstr)

end subroutine writeq_di

!***********************************************************************

subroutine writeq_si(unit,namestr,ivalue,fmt)

! Writes a string of the form <name> = ivalue to unit

integer(ki4) :: ivalue
integer :: unit
character(len=*) :: namestr,fmt
character(len=32) :: tempstr
call writenum(ivalue,tempstr,fmt)
call trimzero(tempstr)
write(unit,*) trim(namestr)//' = '//trim(tempstr)

end subroutine writeq_si

!***********************************************************************

function is_letter(ch) result(res)

! Returns .true. if ch is a letter and .false. otherwise

character :: ch
logical :: res

select case(ch)
case('A':'Z','a':'z')
  res=.true.
case default
  res=.false.
end select
return

end function is_letter

!***********************************************************************
```

```fortran
function is_digit(ch) result(res)

! Returns .true. if ch is a digit (0,1,...,9) and .false. otherwise

character :: ch
logical :: res

select case(ch)
case('0':'9')
  res=.true.
case default
  res=.false.
end select
return

end function is_digit

!**********************************************************************

subroutine split(str,delims,before,sep)

! Routine finds the first instance of a character from 'delims' in the
! the string 'str'. The characters before the found delimiter are
! output in 'before'. The characters after the found delimiter are
! output in 'str'. The optional output character 'sep' contains the
! found delimiter. A delimiter in 'str' is treated like an ordinary
! character if it is preceded by a backslash (\). If the backslash
! character is desired in 'str', then precede it with another backslash.

character(len=*) :: str,delims,before
character,optional :: sep
logical :: pres
character :: ch,cha

pres=present(sep)
str=adjustl(str)
call compact(str)
lenstr=len_trim(str)
if(lenstr == 0) return        ! string str is empty
k=0
ibsl=0                        ! backslash initially inactive
before=' '
do i=1,lenstr
   ch=str(i:i)
   if(ibsl == 1) then          ! backslash active
      k=k+1
      before(k:k)=ch
      ibsl=0
      cycle
   end if
   if(ch == '\') then          ! backslash with backslash inactive
      k=k+1
      before(k:k)=ch
      ibsl=1
      cycle
   end if
   ipos=index(delims,ch)
   if(ipos == 0) then          ! character is not a delimiter
      k=k+1
      before(k:k)=ch
      cycle
   end if
   if(ch /= ' ') then          ! character is a delimiter that is not a space
      str=str(i+1:)
      if(pres) sep=ch
      exit
   end if
   cha=str(i+1:i+1)            ! character is a space delimiter
   iposa=index(delims,cha)
```

```fortran
   if(iposa > 0) then          ! next character is a delimiter
      str=str(i+2:)
      if(pres) sep=cha
      exit
   else
      str=str(i+1:)
      if(pres) sep=ch
      exit
   end if
end do
if(i >= lenstr) str=''
str=adjustl(str)              ! remove initial spaces
return

end subroutine split

!**********************************************************************

subroutine removebksl(str)

! Removes backslash (\) characters. Double backslashes (\\) are replaced
! by a single backslash.

character(len=*):: str
character(len=1):: ch
character(len=len_trim(str))::outstr

str=adjustl(str)
lenstr=len_trim(str)
outstr=' '
k=0
ibsl=0                        ! backslash initially inactive

do i=1,lenstr
  ch=str(i:i)
  if(ibsl == 1) then          ! backslash active
   k=k+1
   outstr(k:k)=ch
   ibsl=0
   cycle
  end if
  if(ch == '\') then          ! backslash with backslash inactive
   ibsl=1
   cycle
  end if
  k=k+1
  outstr(k:k)=ch              ! non-backslash with backslash inactive
end do

str=adjustl(outstr)

end subroutine removebksl

!**********************************************************************

end module strings
```