```
!*****************************************************************************
! module neighbors_m
!-----------------------------------------------------------------------------
!
! Provides the following public subroutines:
!   neighbors_init : initializes data for neighbors subroutine
!   neighbors      : finds neighbor atoms
!
! Additionally file neighbors_c provides interfaces for calling these routines
! from C programs
!
! This is only an (example of) interface definition. No actual code.
!
!---------- DEPENDENCIES -----------------------------------------------------
!
! External routines used:
!   none
!
! Modules used:
!   precision_m, only: dp                 : double precision real kind
!   alloc_m,     only: re_alloc, de_alloc : (re)allocation utility routines
!
! Libraries used:
!   none
!
!---------- PARALELIZATION ---------------------------------------------------
!
! neighbors_init: serial, not thread safe
! neighbors:      serial and thread safe
!
!---------- C BINDINGS -------------------------------------------------------
!
! void neighbors_init( int *na, double ra[][], double cell[][], double *rcmax,
!                      int *nm, int moved[] )
! void neighbors( double ri[], double *rc, in *maxn, int *nm, int ja[],
!                 double rij[][] )
!
!---------- COMPILATION ------------------------------------------------------
!
! gfortran precision_m.f90 alloc_m.f90 neighbors_m.f90 neighbors_c.f90
! gfortran yourFortranProgram.f90 precision_m.o alloc_m.o neighbors_m.o
!     -o yourProgram.exe
! gcc yourCprogram.c precision_m.o alloc_m.o neighbors_m.o neighbors_c.o
!     -o yourProgram.exe
!
```

```
!---------- ALGORITHMS ------------------------------------------------------
!
! In the first call to neighbors_init, the unit cell (or an orthorrombic cell
! containing all atoms) is divided uniformly into subcells of size ~rcmax/2.
! A list is then stored with the atoms in each nonempty subcell, and it is
! updated when the atoms move. The subcell division is also re-constructed
! if the cell changes, or if rcmax increases.
! neighbors first finds the subcell of the sphere center, looks at the neighbor
! subcells (including periodic conditions, if needed) and checks if the atoms
! within them are indeed inside the sphere.
! Refs: no references available.
!
!---------- AUTHORS ---------------------------------------------------------
!
! Interface written by J.M.Soler, Univ. Autonoma de Madrid. June 2014
! Email: jose.soler@uam.es
!
!*****************************************************************************
!
! subroutine neighbors_init( na, ra, cell, rcmax, nm, moved )
!-----------------------------------------------------------------------------
! Initializes or updates coordinates and periodic cell data, for its use by
! subroutine neighbors
!-------------------------- INPUT ----------------------------------------------
! integer  na        : total number of atoms
! real(dp) ra(3,na)  : atomic cartesian coordinates
! real(dp) cell(3,3) : periodic cell vectors, by columns
! real(dp) rcmax     : max. cutoff radius that will be used to call neighbors
! integer  nm        : number of atoms moved
! integer  moved(nm) : list of moved atoms (since last call)
! ------------------------ UNITS ----------------------------------------------
! Units are arbitrary, but they must be the same for ra, cell, and rcmax
! ------------------------ BEHAVIOUR ------------------------------------------
! - The atomic positions ra need not be within the (first) unit cell
! - If det(cell)==0, assumes that system is not periodic
! - If rcmax<=0, uses the longest cell vector or the max. interatomic distance,
!   if the system is not periodic
! - If nm<=0, assumes that all atoms have moved
!
!*****************************************************************************
!
! subroutine neighbors( ri, rc, maxn, nn, ja, rij )
!-----------------------------------------------------------------------------
! Finds atoms within a sphere of radius rc centered at point ri
!-------------------------- INPUT ----------------------------------------------
! real(dp) ri(3)     : coordinates of sphere center
```

```
! real(dp) rc        : sphere cutoff radius
! integer  maxn      : size of arrays ja and rij
!------------------------- OUTPUT -----------------------------------------
! integer  nn        : number of atoms within sphere
! integer  ja(nn)    : index of atoms within sphere
! real(dp) rij(3,nn) : rj-ri vectors
! ------------------------ UNITS ------------------------------------------
! Units are arbitrary, but they must be the same for ri and rc, and consistent
! with those used to call neighbors_init
! ------------------------ BEHAVIOUR --------------------------------------
! - If neighbors is called without a previous call to neighbors_init, it stops
!   with an error message
! - ri needs not be an atomic position. If it is, that atom will appear first
!   in the ja list, with rij=0
! - ri needs not be within the (first) periodic unit cell
! - The 'same' atom (of different unit cells) can appear repeatedly in the
!   ja list, with different rij vectors
! - Neighbors are ordered by increasing distance to ri
! - If maxn<nn, it prints a warning and fills ja and rij only up to maxn
!
!*******************************************************************************
```

```
! neighbors_c.f90
! C-interoperable interfaces for module neighbors_m

subroutine neighbors_init( na, ra, cell, rcmax, nm, moved ) bind(C)

  use precision_m,   only: dp       ! our double precision real kind
  use iso_c_binding, only: C_INT    ! C-interoperable integer kind
  use iso_c_binding, only: C_DOUBLE ! C-interoperable double precision real kind
  use neighbors_m,   only: neigbors_init_f => neighbors_init

  implicit none
  integer(C_INT),intent(in):: na        ! total number of atoms
  real(C_DOUBLE),intent(in):: ra(3,na)  ! atomic cartesian coordinates
  real(C_DOUBLE),intent(in):: cell(3,3) ! periodic cell vectors, by columns
  real(C_DOUBLE),intent(in):: rcmax     ! max. cutoff radius
  integer(C_INT),intent(in):: nm        ! number of atoms moved
  integer(C_INT),intent(in):: moved(nm) ! list of moved atoms (since last call)

  if (dp/=C_DOUBLE) stop 'neighbors_init ERROR: wrong double precision kind'

  call neighbors_init_f( na, ra, cell, rcmax, nm, moved )

end subroutine neighbors_init


subroutine neighbors( ri, rc, maxn, nn, ja, rij ) bind(C)

  use iso_c_binding, only: C_INT    ! C-interoperable integer kind
  use iso_c_binding, only: C_DOUBLE ! C-interoperable double precision real kind
  use neighbors_m,   only: neigbors_f => neighbors

  implicit none
  real(C_DOUBLE),intent(in) :: ri(3)       ! coordinates of sphere center
  real(C_DOUBLE),intent(in) :: rc          ! sphere cutoff radius
  integer(C_INT),intent(in) :: maxn        ! size of arrays ja and rij
  integer(C_INT),intent(out):: nn          ! number of atoms within sphere
  integer(C_INT),intent(out):: ja(maxn)    ! index of atoms within sphere
  real(C_DOUBLE),intent(out):: rij(3,maxn) ! rj-ri vectors

  call neighbors_f( ri, rc, maxn, nn, ja, rij )

end subroutine neighbors
```