```
MODULE Tools
  IMPLICIT NONE
  ! -------------------------------------
  TYPE Cell2D
     INTEGER                :: ik, jk
     TYPE(Cell2D), POINTER :: Next=>NULL()
  END TYPE Cell2D

  TYPE Cell
     INTEGER              :: Val
     TYPE(Cell), POINTER :: Next=>NULL()
  END TYPE Cell
  ! -------------------------------------
  ! -------------------------------------
CONTAINS
  ! -------------------------------------
  ! ------------------------------------------------------------------------------
  ! ------------------------------------------------------------------------------
  SUBROUTINE InsertList2D(Head, iNew, jNew)
    TYPE(Cell2D),INTENT(InOUT)        :: Head
    INTEGER, INTENT(IN)               :: iNew, jNew
    TYPE(Cell2D), POINTER             :: TheCell=>NULL(), NewCell, PrevCell=>NULL
()
    ! -------------------------------------


    TheCell  => Head%next

    IF( .NOT. ASSOCIATED(TheCell) )  THEN
       ALLOCATE(NewCell); NewCell%ik = iNew ; NewCell%jk = jNew
       Head%Next => NewCell
    ELSE
       IF(TheCell%ik > inew ) THEN
          ALLOCATE(NewCell); NewCell%ik = iNew ; NewCell%jk = jNew
          NewCell%Next  => TheCell
          Head%Next     => NewCell
       ELSE IF( TheCell%ik == inew  .AND. TheCell%jk > jnew ) THEN
          ALLOCATE(NewCell); NewCell%ik = iNew ; NewCell%jk = jNew
          NewCell%Next  => TheCell
          Head%Next     => NewCell
       ELSE

          PrevCell => Head%next
          TheCell  => PrevCell%next
          DO WHILE( ASSOCIATED(TheCell) )
             IF(TheCell%ik > inew  ) THEN
                EXIT
             ELSE IF( TheCell%ik == inew  .AND. TheCell%jk > jnew  ) THEN
                EXIT
             END IF
             PrevCell   => PrevCell%Next
             TheCell    => TheCell%Next
          END DO

          !          WRITE(6,*) PrevCell%ik,  PrevCell%jk

          IF( PrevCell%ik == inew .AND.  PrevCell%jk == jnew )  THEN
          ELSE

             ALLOCATE(NewCell); NewCell%ik = iNew ; NewCell%jk = jNew
             NewCell%Next  => TheCell
             PrevCell%Next => NewCell
          END IF
       END IF
    END IF
  END SUBROUTINE InsertList2D
  ! ------------------------------------------------------------------------------
  ! ------------------------------------------------------------------------------
  SUBROUTINE InsertList(Head, NewVal)
```



```
    TYPE(Cell),INTENT(InOUT)        :: Head
    INTEGER, INTENT(IN)             :: NewVal
    TYPE(Cell), POINTER             :: TheCell=>NULL(), NewCell, PrevCell=>NULL()
    ! -------------------------------------


    TheCell  => Head%next

    IF( .NOT. ASSOCIATED(TheCell) )  THEN
       ALLOCATE(NewCell); NewCell%Val=NewVal
       Head%Next => NewCell
    ELSE
       IF(TheCell%Val > NewVal ) THEN
          ALLOCATE(NewCell); NewCell%Val=NewVal
          NewCell%Next  => TheCell
          Head%Next => NewCell
       ELSE

          PrevCell => Head%next
          TheCell  => PrevCell%next

          DO WHILE( ASSOCIATED(TheCell) )
             IF(TheCell%Val > NewVal ) THEN
                EXIT
             END IF
             PrevCell   => PrevCell%Next
             TheCell    => TheCell%Next
          END DO

          IF( PrevCell%Val == NewVal  )  THEN
          ELSE
             ALLOCATE(NewCell); NewCell%Val=NewVal
             NewCell%Next  => TheCell
             PrevCell%Next => NewCell
          END IF
       END IF
    END IF
  END SUBROUTINE InsertList
  ! ------------------------------------------------------------------------------
  ! ------------------------------------------------------------------------------
  SUBROUTINE ContainOfList2D(Head)
    TYPE(Cell2D),INTENT(In)          :: Head
    TYPE(Cell2D), POINTER            :: TheCell=>NULL()
    INTEGER                          :: i
    ! -------------------------------------
    i=0
    TheCell => Head%Next
    WRITE(6,*) " ------ - list contained - ------------------ "
    DO WHILE( ASSOCIATED(TheCell) )
       i = i +1
       WRITE(6,'(A9,I6,A12,I6,I6)') " number ", i, " >> ValueS = ", TheCell%ik, TheCell%jk
       TheCell       => TheCell%Next
    END DO
    WRITE(6,*) " ------ - End of list - ------------------ "
  END SUBROUTINE ContainOfList2D
  ! ------------------------------------------------------------------------------
  ! ------------------------------------------------------------------------------
  FUNCTION  CountEdges2D(Head) RESULT(Nedge)
    TYPE(Cell2D),INTENT(In)          :: Head
    TYPE(Cell2D), POINTER            :: TheCell=>NULL()
    INTEGER                          :: Nedge
    ! -------------------------------------
    Nedge=0
    TheCell => Head%Next
    DO WHILE( ASSOCIATED(TheCell) )
       Nedge        = Nedge +1
       TheCell       => TheCell%Next
    END DO
  END FUNCTION CountEdges2D
```

```
  ! ------------------------------------------------------------------------------
  ! ------------------------------------------------------------------------------
  FUNCTION  CountEdges(HashTable) RESULT(Nedge)
    TYPE(Cell), DIMENSION(:)     ::  HashTable
    TYPE(Cell), POINTER          :: TheCell=>NULL()
    INTEGER                      :: Nedge, is
    ! ------------------------------------
    Nedge=0
    DO is = 1, SIZE(HashTable)
       TheCell => HashTable(is)%Next
       DO WHILE( ASSOCIATED(TheCell) )
          Nedge       = Nedge +1
          TheCell      => TheCell%Next
       END DO
    END DO

  END FUNCTION CountEdges
  ! ------------------------------------------------------------------------------
  ! ------------------------------------------------------------------------------
  SUBROUTINE ContainOfList(Head)
    TYPE(Cell),INTENT(In)          :: Head
    TYPE(Cell), POINTER            :: TheCell=>NULL()
    INTEGER                        :: i
    ! ------------------------------------
    i=0
    TheCell => Head%Next
    WRITE(6,*)  " ------ - list contained - ------------------- "
    DO WHILE( ASSOCIATED(TheCell) )
       i = i +1
       WRITE(6,'(A9,I4,A12,I4)')  " number ", i,  " >> Value = ", TheCell%Val
       TheCell       => TheCell%Next
    END DO
    WRITE(6,*)  " ------ -   End of list - ------------------- "
  END SUBROUTINE ContainOfList
  ! ------------------------------------------------------------------------------
  ! ------------------------------------------------------------------------------
  ! ------------------------------------------------------------------------------
  ! ------------------------------------------------------------------------------
END MODULE Tools
! ------------------------------------------------------------------------------
! ------------------------------------------------------------------------------
! ------------------------------------------------------------------------------
! ------------------------------------------------------------------------------
PROGRAM EdgeOfMesh
  USE Tools
  IMPLICIT NONE

  INTEGER         :: NVrtx, Ntria, Nquad
  INTEGER         :: is, ie, is1, is2, is3, is4, imin, imax
  INTEGER         :: Nargs
  REAL            :: TimeTop, TimeEnd
  TYPE(Cell2D)    :: Head2D
  TYPE(Cell)      :: Head
  TYPE(Cell2D)    :: Edges2D

  INTEGER    , DIMENSION(:,:), ALLOCATABLE :: TriaVrtx, QuadVrtx
  REAL       , DIMENSION(:,:), ALLOCATABLE :: Coor
  TYPE(Cell), DIMENSION(:)   , POINTER     ::  HashTable

  CHARACTER(LEN=20) :: KeyWd, Method, MeshFile, Exe
  ! ------------------------------------------------------------------------------
  ! ------------------------------------------------------------------------------

  MeshFile = "MeshFile"
  Method   = "AllH"

  CALL GetArg(0,Exe)
  CALL GetArg(1,Method)
```



```
  IF( Nargs < 2 ) THEN
     WRITE(6,*)  " usage :: ", TRIM(Exe),  " General   Mesh0 "
     WRITE(6,*)  " -------- ", TRIM(Exe),  " GeneralH  Mesh0 "
  END IF
  CALL GetArg(2,MeshFile)

  CALL CPU_TIME(TimeTop)
  CALL ReadMesh()
  CALL CPU_TIME(TimeEnd)
  WRITE(6,*)  " CPU time for mesh reading      (s) =", TimeEnd-TimeTop
  WRITE(6,'(50("-"))')
  WRITE(6,*)  " >> Number of vertices   = ", NVrtx
  WRITE(6,*)  " >> Number of triangles  = ", Ntria
  WRITE(6,*)  " >> Number of quadrangles = ", Nquad


  SELECT CASE(TRIM(Method))

  CASE("General")
     CALL CPU_TIME(TimeTop)
     DO ie = 1, Ntria
        is1 = TriaVrtx(1,ie)
        is2 = TriaVrtx(2,ie)
        is3 = TriaVrtx(3,ie)
        CALL InsertList2D(Edges2D, is1, is2)
        CALL InsertList2D(Edges2D, is1, is3)
        CALL InsertList2D(Edges2D, is2, is1)
        CALL InsertList2D(Edges2D, is2, is3)
        CALL InsertList2D(Edges2D, is3, is1)
        CALL InsertList2D(Edges2D, is3, is2)
     END DO
     CALL CPU_TIME(TimeEnd)

      WRITE(6,*)  " << Number of Edges in the Mesh  = ", CountEdges2D(Edges2D)

  CASE("GeneralH")
     ALLOCATE( HashTable(NVrtx))
     CALL CPU_TIME(TimeTop)
     DO ie = 1, Ntria
        is1 = TriaVrtx(1,ie)
        is2 = TriaVrtx(2,ie)
        is3 = TriaVrtx(3,ie)
        CALL InsertList(HashTable(is1), is2 )
        CALL InsertList(HashTable(is1), is3 )
        CALL InsertList(HashTable(is2), is1 )
        CALL InsertList(HashTable(is2), is3 )
        CALL InsertList(HashTable(is3), is1 )
        CALL InsertList(HashTable(is3), is2 )
     END DO


     CALL CPU_TIME(TimeEnd)
     WRITE(6,*)  " << Number of Edges in the Mesh  = ", CountEdges(HashTable)

   CASE("EFP1H")
     ALLOCATE( HashTable(NVrtx))
     CALL CPU_TIME(TimeTop)
     DO ie = 1, Ntria
        is1 = TriaVrtx(1,ie)
        is2 = TriaVrtx(2,ie)
        is3 = TriaVrtx(3,ie)

        CALL InsertList(HashTable(is1), is2 )
        CALL InsertList(HashTable(is1), is3 )
        CALL InsertList(HashTable(is2), is1 )
        CALL InsertList(HashTable(is2), is3 )
        CALL InsertList(HashTable(is3), is1 )
        CALL InsertList(HashTable(is3), is2 )
```

```
      DO is = 1, NVrtx
         CALL InsertList(HashTable(is), is )
      END DO
      CALL CPU_TIME(TimeEnd)
      WRITE(6,*)  " << Number of Edges in the Mesh  = ", CountEdges(HashTable)

   CASE("Symmetric")
      CALL CPU_TIME(TimeTop)
      DO ie = 1, Ntria
         is1 = TriaVrtx(1,ie)
         is2 = TriaVrtx(2,ie)
         is3 = TriaVrtx(3,ie)
         CALL InsertList2D(Edges2D, MIN(is1, is2), MAX(is1, is2)  )
         CALL InsertList2D(Edges2D, MIN(is1, is3), MAX(is1, is3)  )
         CALL InsertList2D(Edges2D, MIN(is2, is3), MAX(is2, is3)  )
      END DO
      CALL CPU_TIME(TimeEnd)
       WRITE(6,*)  " << Number of Edges in the Mesh  = ", CountEdges2D(Edges2D)

   CASE("SymmetricH")
      ALLOCATE( HashTable(NVrtx))
      CALL CPU_TIME(TimeTop)
      DO ie = 1, Ntria
         is1 = TriaVrtx(1,ie)
         is2 = TriaVrtx(2,ie)
         is3 = TriaVrtx(3,ie)

         imin = MIN(is1, is2) ;   imax = MAX(is1, is2)
         CALL InsertList(HashTable(imin), imax )

         imin = MIN(is1, is3) ;   imax = MAX(is1, is3)
         CALL InsertList(HashTable(imin), imax )

         imin = MIN(is3, is2) ;   imax = MAX(is3, is2)
         CALL InsertList(HashTable(imin), imax )

      END DO
      CALL CPU_TIME(TimeEnd)
      WRITE(6,*)  " << Number of Edges in the Mesh  = ", CountEdges(HashTable)
   END SELECT

   WRITE(6,*)  " >> Mesh File used          = ", TRIM(MeshFile)//".mesh"
   WRITE(6,'(50("-"))')

   WRITE(6,*)  " CPU time for Edges construction (s) =", TimeEnd-TimeTop



   ! ------------------------------------------------------------------------
CONTAINS
   ! ------------------------------------------------------------------------

   ! ------------------------------------------------------------------------
   SUBROUTINE ReadMesh()
     INTEGER :: Ndim

     OPEN( 12, FILE=TRIM(MeshFile)//".mesh")
     DO
        READ(12,*) KeyWd
        !      WRITE(6,*) TRIM(KeyWd)
        SELECT CASE(TRIM(KeyWd))
        CASE("Dimension")
           READ(12,*) Ndim
        CASE("Edges")

           KeyWd = " "
! NetCDF and HDF5
```



```
        CASE("Vertices")
           READ(12,*) NVrtx
           ALLOCATE( COOR(Ndim,NVrtx) )
           DO is = 1, NVrtx
              READ(12,*) COOR(:,is)
           END DO
           KeyWd = " "
        CASE("Triangles")
           READ(12,*) Ntria
           ALLOCATE( TriaVrtx(3,Ntria) )
           DO ie = 1, Ntria
              READ(12,*) TriaVrtx(:, ie)
           END DO
           KeyWd = " "

        CASE("Quadrilaterals")
           READ(12,*) Nquad
           ALLOCATE( QuadVrtx(4,Nquad) )
           DO ie = 1, Nquad
              READ(12,*) QuadVrtx(:, ie)
           END DO
           KeyWd = " "

        CASE ("End", "END")
           EXIT
        END SELECT
     END DO


   END SUBROUTINE ReadMesh
   ! ------------------------------------------------------------------------
END PROGRAM EdgeOfMesh
```