```
MODULE qr

  USE iso_fortran_env, ONLY: WP => REAL64
  IMPLICIT NONE

CONTAINS

  SUBROUTINE gramschmidt(a,r)
    ! ------------------------------------------------------
    ! Determinación de la factorizacion QR de una matriz A
    ! a través del método (modificado) de ortogonalización
    ! de Gram-Schmidt.
    ! ------------------------------------------------------
    ! Argumentos:
    !
    ! REAL(WP), A(:,:): Como argumento de entrada contiene
    !                   la matriz A de mxn a factorizar.
    !                   Como argumento de salida contiene
    !                   la matriz Q ortogonal de mxn de
    !                   la factorización.
    !
    ! REAL(WP), R(:,:): Como argumento de salida contiene
    !                   la matriz R triangular superior
    !                   de nxn de la factorización.
    ! ------------------------------------------------------
    real(wp), intent(inout) :: a(:,:)
    real(wp), intent(out)   :: r(:,:)
    ! ------------------------------------------------------
    integer i,j
    ! ------------------------------------------------------
    r = 0.0_wp
    do j=1,size(a,2)
       do i=1,j-1
          r(i,j) = DOT_PRODUCT(a(:,i),a(:,j))
          a(:,j) = a(:,j)-r(i,j)*a(:,i)
       enddo
       r(j,j) = SQRT(DOT_PRODUCT(a(:,j),a(:,j)))
       a(:,j) = a(:,j)/r(j,j)
    enddo
    ! ------------------------------------------------------
    return
  END SUBROUTINE gramschmidt

END MODULE qr

PROGRAM main
  ! ------------------------------------------------------
  ! Implementación naïve del método QR
  ! ------------------------------------------------------
  USE iso_fortran_env, ONLY: WP => REAL64
  USE qr, ONLY: gramschmidt
  IMPLICIT NONE
  ! ------------------------------------------------------
  real(wp), allocatable :: a(:,:), r(:,:)
  real(wp), allocatable :: d(:),d0(:)
  integer, parameter    :: MAX_ITER = 100
  real(wp), parameter   :: TOL=5.0e-8_wp
  integer               :: n,i,k
  character(80)         :: filename
  ! ------------------------------------------------------
  ! Obtener el nombre del archivo de la línea de comandos
  ! ------------------------------------------------------
```



```
  if ( command_argument_count() /= 1 ) then
     write(*,*) 'Uso: programa archivo_datos'
     stop
  endif
  call get_command_argument(1,filename)
  ! ------------------------------------------------------------
  ! Lectura de la matriz desde un archivo de la forma:
  ! n
  ! a(1,1) ... a(1,n)
  ! ...
  ! a(n,1) ... a(n,n)
  ! ------------------------------------------------------------
  open(8,file=filename)
  read(8,*) n
  allocate(a(n,n),r(n,n),d(n),d0(n))
  do i=1,n
     read(8,*) a(i,:)
  enddo
  close(8)
  ! ------------------------------------------------------------
  ! Proceder
  ! ------------------------------------------------------------
  write(*,'(A)') 'ITER = 0'
  do i=1,n
     write(*,*) a(i,:)
  enddo
  write(*,*)
  d0 = [ ( a(i,i), i=1,n) ]

  do k=1,MAX_ITER
     call gramschmidt(a,r)
     a = matmul(r,a)
     d = [ ( a(i,i), i=1,n) ]
     write(*,'(A,I0)') 'ITER = ', k
     do i=1,n
        write(*,*) a(i,:)
     enddo
     write(*,*)
     if (maxval(abs(d-d0)) < maxval(abs(d))*TOL) exit
     d0 = d
  enddo
  ! ------------------------------------------------------------
END PROGRAM main
```