```fortran
      PROGRAM SAMPLE_DQ
      USE FMZM
      USE FM_DOUBLE_INT
      USE FM_QUAD_INT
      USE FM_QUAD_REAL

      IMPLICIT NONE

!  This program illustrates the use of the three modules for using FM, IM, or ZM variables
!  with double and quadruple length integer and real variables supplied by the compiler.

!  These would be needed for cases where the user's program uses all three precisions,
!  single, double, quad.  Then using a compiler switch to automatically make 64-bits the
!  default would not work.

!  The variables used to select the kind of integers and floating-point real and complex
!  numbers to be used, DOUBLE_INT, QUAD_INT, and QUAD_FP, are defined in the three modules.


      INTEGER ::                J, J1, J2
      INTEGER (DOUBLE_INT) ::  K, K1, K2
      INTEGER (QUAD_INT) ::    L, L1, L2

      REAL ::                   A1, A2
      REAL (KIND(1.0D0)) ::    B1, B2
      REAL (QUAD_FP) ::        C1, C2

      COMPLEX ::                R1, R2
      COMPLEX (KIND(1.0D0)) :: S1, S2
      COMPLEX (QUAD_FP) ::     T1, T2

      TYPE(FM) ::               F1, REL_ERROR
      TYPE(ZM) ::               Z1, Z2

!             For a simple example calculation, sum several pieces of the harmonic series,
!             1/n + 1/(n+1) + ... + 1/(n+10**5).

!             n = 10**2  for single precision,
!             n = 10**13 for double precision,
!             n = 10**29 for quad precision.

!             In each case, use type(fm) to get a more accurate sum and compute the relative
!             error for the other sum.

      J1 = 10**2
      J2 = J1 + 10**5
      A1 = 0
      F1 = 0
      DO J = J1, J2
         A2 = J
         A1 = A1 + 1/A2

         F1 = F1 + 1/TO_FM(J)
      ENDDO
      REL_ERROR = ABS( (F1-A1)/F1 )
```

```
      WRITE (*,"(/A,ES15.7/)") ' Relative error for the single precision real sum = ',  &
                               TO_QUAD(REL_ERROR)

      K1 = 10_DOUBLE_INT**13
      K2 = K1 + 10**5
      B1 = 0
      F1 = 0
      DO K = K1, K2
         B2 = K
         B1 = B1 + 1/B2

         F1 = F1 + 1/TO_FM(K)
      ENDDO
      REL_ERROR = ABS( (F1-B1)/F1 )
      WRITE (*,"(/A,ES15.7/)") ' Relative error for the double precision real sum = ',  &
                               TO_QUAD(REL_ERROR)

      L1 = 10_QUAD_INT**29
      L2 = L1 + 10**5
      C1 = 0
      F1 = 0
      DO L = L1, L2
         C2 = L
         C1 = C1 + 1/C2

         F1 = F1 + 1/TO_FM(L)
      ENDDO
      REL_ERROR = ABS( (F1-C1)/F1 )
      WRITE (*,"(/A,ES15.7/)") ' Relative error for the quad precision real sum   = ',  &
                               TO_QUAD(REL_ERROR)

!             Do a similar sum using complex numbers.
!             1/sqrt(n+i) + 1/sqrt(n+1+i) + ... + 1/sqrt(n+10**4+i).

!             n = 10**2  for single precision,
!             n = 10**13 for double precision,
!             n = 10**29 for quad precision.

!             In each case, use type(zm) to get a more accurate sum and compute the relative
!             error for the other sum.

      J1 = 10**2
      J2 = J1 + 10**4
      R1 = 0
      Z1 = 0
      Z2 = TO_ZM(' i ')
      DO J = J1, J2
         R2 = J + CMPLX(0.0, 1.0)
         R1 = R1 + 1/SQRT(R2)

         Z1 = Z1 + 1/SQRT(J+Z2)
      ENDDO
      REL_ERROR = ABS( (Z1-R1)/Z1 )
      WRITE (*,"(/A,ES15.7/)") ' Relative error for the single precision complex sum = ',  &
                               TO_QUAD(REL_ERROR)
```

```
K1 = 10_DOUBLE_INT**13
K2 = K1 + 10**4
S1 = 0
Z1 = 0
Z2 = TO_ZM(' i ')
DO K = K1, K2
   S2 = K + CMPLX(0.0D0, 1.0D0, KIND(1.0D0))
   S1 = S1 + 1/SQRT(S2)

   Z1 = Z1 + 1/SQRT(K+Z2)
ENDDO
REL_ERROR = ABS( (Z1-S1)/Z1 )
WRITE (*,"(/A,ES15.7/)") ' Relative error for the double precision complex sum = ',  &
                         TO_QUAD(REL_ERROR)

L1 = 10_QUAD_INT**29
L2 = L1 + 10**4
T1 = 0
Z1 = 0
Z2 = TO_ZM(' i ')
DO L = L1, L2
   T2 = L + CMPLX(0.0_QUAD_FP, 1.0_QUAD_FP, QUAD_FP)
   T1 = T1 + 1/SQRT(T2)

   Z1 = Z1 + 1/SQRT(L+Z2)
ENDDO
REL_ERROR = ABS( (Z1-T1)/Z1 )
WRITE (*,"(/A,ES15.7/)") ' Relative error for the quad precision complex sum   = ',  &
                         TO_QUAD(REL_ERROR)

END PROGRAM SAMPLE_DQ
```