```
      SUBROUTINE DGEMM(TRANSA,TRANSB,M,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC)
*     .. Scalar Arguments ..
      DOUBLE PRECISION ALPHA,BETA
      INTEGER K,LDA,LDB,LDC,M,N
      CHARACTER TRANSA,TRANSB
*     ..
*     .. Array Arguments ..
      DOUBLE PRECISION A(LDA,*),B(LDB,*),C(LDC,*)
*     ..
*
*  Purpose
*  =======
*
*  DGEMM  performs one of the matrix-matrix operations
*
*     C := alpha*op( A )*op( B ) + beta*C,
*
*  where  op( X ) is one of
*
*     op( X ) = X   or   op( X ) = X',
*
*  alpha and beta are scalars, and A, B and C are matrices, with op( A )
*  an m by k matrix,  op( B )  a  k by n matrix and  C an m by n matrix.
*
*  Arguments
*  ==========
*
*  TRANSA - CHARACTER*1.
*           On entry, TRANSA specifies the form of op( A ) to be used in
*           the matrix multiplication as follows:
*
*              TRANSA = 'N' or 'n',  op( A ) = A.
*
*              TRANSA = 'T' or 't',  op( A ) = A'.
*
*              TRANSA = 'C' or 'c',  op( A ) = A'.
*
*           Unchanged on exit.
*
*  TRANSB - CHARACTER*1.
*           On entry, TRANSB specifies the form of op( B ) to be used in
*           the matrix multiplication as follows:
*
*              TRANSB = 'N' or 'n',  op( B ) = B.
*
*              TRANSB = 'T' or 't',  op( B ) = B'.
*
*              TRANSB = 'C' or 'c',  op( B ) = B'.
*
*           Unchanged on exit.
*
*  M      - INTEGER.
*           On entry,  M  specifies  the number  of rows  of the  matrix
*           op( A )  and of the  matrix  C.  M  must  be at least  zero.
*           Unchanged on exit.
*
*  N      - INTEGER.
*           On entry,  N  specifies the number  of columns of the matrix
*           op( B ) and the number of columns of the matrix C. N must be
*           at least zero.
```

```
*           Unchanged on exit.
*
*  K      - INTEGER.
*           On entry,  K  specifies  the number of columns of the matrix
*           op( A ) and the number of rows of the matrix op( B ). K must
*           be at least  zero.
*           Unchanged on exit.
*
*  ALPHA  - DOUBLE PRECISION.
*           On entry, ALPHA specifies the scalar alpha.
*           Unchanged on exit.
*
*  A      - DOUBLE PRECISION array of DIMENSION ( LDA, ka ), where ka is
*           k  when  TRANSA = 'N' or 'n',  and is  m  otherwise.
*           Before entry with  TRANSA = 'N' or 'n',  the leading  m by k
*           part of the array  A  must contain the matrix  A,  otherwise
*           the leading  k by m  part of the array  A  must contain  the
*           matrix A.
*           Unchanged on exit.
*
*  LDA    - INTEGER.
*           On entry, LDA specifies the first dimension of A as declared
*           in the calling (sub) program. When  TRANSA = 'N' or 'n' then
*           LDA must be at least  max( 1, m ), otherwise  LDA must be at
*           least  max( 1, k ).
*           Unchanged on exit.
*
*  B      - DOUBLE PRECISION array of DIMENSION ( LDB, kb ), where kb is
*           n  when  TRANSB = 'N' or 'n',  and is  k  otherwise.
*           Before entry with  TRANSB = 'N' or 'n',  the leading  k by n
*           part of the array  B  must contain the matrix  B,  otherwise
*           the leading  n by k  part of the array  B  must contain  the
*           matrix B.
*           Unchanged on exit.
*
*  LDB    - INTEGER.
*           On entry, LDB specifies the first dimension of B as declared
*           in the calling (sub) program. When  TRANSB = 'N' or 'n' then
*           LDB must be at least  max( 1, k ), otherwise  LDB must be at
*           least  max( 1, n ).
*           Unchanged on exit.
*
*  BETA   - DOUBLE PRECISION.
*           On entry,  BETA  specifies the scalar  beta.  When  BETA  is
*           supplied as zero then C need not be set on input.
*           Unchanged on exit.
*
*  C      - DOUBLE PRECISION array of DIMENSION ( LDC, n ).
*           Before entry, the leading  m by n  part of the array  C must
*           contain the matrix  C,  except when  beta  is zero, in which
*           case C need not be set on entry.
*           On exit, the array  C  is overwritten by the  m by n  matrix
*           ( alpha*op( A )*op( B ) + beta*C ).
*
*  LDC    - INTEGER.
*           On entry, LDC specifies the first dimension of C as declared
*           in  the  calling  (sub)  program.   LDC  must  be  at  least
*           max( 1, m ).
*           Unchanged on exit.
*
*
```

```
*  Level 3 Blas routine.
*
*  -- Written on 8-February-1989.
*     Jack Dongarra, Argonne National Laboratory.
*     Iain Duff, AERE Harwell.
*     Jeremy Du Croz, Numerical Algorithms Group Ltd.
*     Sven Hammarling, Numerical Algorithms Group Ltd.
*
*
*     .. External Functions ..
      LOGICAL LSAME
      EXTERNAL LSAME
*     ..
*     .. External Subroutines ..
      EXTERNAL XERBLA
*     ..
*     .. Intrinsic Functions ..
      INTRINSIC MAX
*     ..
*     .. Local Scalars ..
      DOUBLE PRECISION TEMP
      INTEGER I,INFO,J,L,NCOLA,NROWA,NROWB
      LOGICAL NOTA,NOTB
*     ..
*     .. Parameters ..
      DOUBLE PRECISION ONE,ZERO
      PARAMETER (ONE=1.0D+0,ZERO=0.0D+0)
*     ..
*
*     Set  NOTA  and  NOTB  as  true if  A  and  B  respectively are not
*     transposed and set  NROWA, NCOLA and  NROWB  as the number of rows
*     and  columns of  A  and the  number of  rows  of  B  respectively.
*
      NOTA = LSAME(TRANSA,'N')
      NOTB = LSAME(TRANSB,'N')
      IF (NOTA) THEN
          NROWA = M
          NCOLA = K
      ELSE
          NROWA = K
          NCOLA = M
      END IF
      IF (NOTB) THEN
          NROWB = K
      ELSE
          NROWB = N
      END IF
*
*     Test the input parameters.
*
      INFO = 0
      IF ((.NOT.NOTA) .AND. (.NOT.LSAME(TRANSA,'C')) .AND.
     +    (.NOT.LSAME(TRANSA,'T'))) THEN
          INFO = 1
      ELSE IF ((.NOT.NOTB) .AND. (.NOT.LSAME(TRANSB,'C')) .AND.
     +         (.NOT.LSAME(TRANSB,'T'))) THEN
          INFO = 2
      ELSE IF (M.LT.0) THEN
          INFO = 3
      ELSE IF (N.LT.0) THEN
          INFO = 4
```

```
      ELSE IF (K.LT.0) THEN
          INFO = 5
      ELSE IF (LDA.LT.MAX(1,NROWA)) THEN
          INFO = 8
      ELSE IF (LDB.LT.MAX(1,NROWB)) THEN
          INFO = 10
      ELSE IF (LDC.LT.MAX(1,M)) THEN
          INFO = 13
      END IF
      IF (INFO.NE.0) THEN
          CALL XERBLA('DGEMM ',INFO)
          RETURN
      END IF
*
*     Quick return if possible.
*
      IF ((M.EQ.0) .OR. (N.EQ.0) .OR.
     +    (((ALPHA.EQ.ZERO).OR. (K.EQ.0)).AND. (BETA.EQ.ONE))) RETURN
*
*     And if  alpha.eq.zero.
*
      IF (ALPHA.EQ.ZERO) THEN
          IF (BETA.EQ.ZERO) THEN
              DO 20 J = 1,N
                  DO 10 I = 1,M
                      C(I,J) = ZERO
   10             CONTINUE
   20         CONTINUE
          ELSE
              DO 40 J = 1,N
                  DO 30 I = 1,M
                      C(I,J) = BETA*C(I,J)
   30             CONTINUE
   40         CONTINUE
          END IF
          RETURN
      END IF
*
*     Start the operations.
*
      IF (NOTB) THEN
          IF (NOTA) THEN
*
*           Form  C := alpha*A*B + beta*C.
*
              DO 90 J = 1,N
                  IF (BETA.EQ.ZERO) THEN
                      DO 50 I = 1,M
                          C(I,J) = ZERO
   50                 CONTINUE
                  ELSE IF (BETA.NE.ONE) THEN
                      DO 60 I = 1,M
                          C(I,J) = BETA*C(I,J)
   60                 CONTINUE
                  END IF
                  DO 80 L = 1,K
                      IF (B(L,J).NE.ZERO) THEN
                          TEMP = ALPHA*B(L,J)
                          DO 70 I = 1,M
                              C(I,J) = C(I,J) + TEMP*A(I,L)
   70                     CONTINUE
```

```
                    END IF
   80           CONTINUE
   90         CONTINUE
         ELSE
*
*           Form  C := alpha*A'*B + beta*C
*
             DO 120 J = 1,N
                 DO 110 I = 1,M
                     TEMP = ZERO
                     DO 100 L = 1,K
                         TEMP = TEMP + A(L,I)*B(L,J)
  100                CONTINUE
                     IF (BETA.EQ.ZERO) THEN
                         C(I,J) = ALPHA*TEMP
                     ELSE
                         C(I,J) = ALPHA*TEMP + BETA*C(I,J)
                     END IF
  110            CONTINUE
  120         CONTINUE
         END IF
      ELSE
         IF (NOTA) THEN
*
*           Form  C := alpha*A*B' + beta*C
*
             DO 170 J = 1,N
                 IF (BETA.EQ.ZERO) THEN
                     DO 130 I = 1,M
                         C(I,J) = ZERO
  130                CONTINUE
                 ELSE IF (BETA.NE.ONE) THEN
                     DO 140 I = 1,M
                         C(I,J) = BETA*C(I,J)
  140                CONTINUE
                 END IF
                 DO 160 L = 1,K
                     IF (B(J,L).NE.ZERO) THEN
                         TEMP = ALPHA*B(J,L)
                         DO 150 I = 1,M
                             C(I,J) = C(I,J) + TEMP*A(I,L)
  150                    CONTINUE
                     END IF
  160            CONTINUE
  170         CONTINUE
         ELSE
*
*           Form  C := alpha*A'*B' + beta*C
*
             DO 200 J = 1,N
                 DO 190 I = 1,M
                     TEMP = ZERO
                     DO 180 L = 1,K
                         TEMP = TEMP + A(L,I)*B(J,L)
  180                CONTINUE
                     IF (BETA.EQ.ZERO) THEN
                         C(I,J) = ALPHA*TEMP
                     ELSE
                         C(I,J) = ALPHA*TEMP + BETA*C(I,J)
                     END IF
  190            CONTINUE
```

```
   200          CONTINUE
            END IF
         END IF
*
         RETURN
*
*        End of DGEMM .
*
         END
```