```
c===========================================================
c     Computes and reports estimate of machine epsilon.
c
c     Recall: machine epsilon is smallest positive 'eps'
c     such that
c
c             (1.0d0 + eps ) .ne. (1.0d0)
c
c     Program accepts optional argument which specifies
c     division factor: values close to 1.0 will result
c     in more accurate estimate of machine epsilon.
c===========================================================
      program         meps

      implicit        none

c-----------------------------------------------------------
c     Note use of 'r8arg', available in 'libp410f.a' which
c     works exactly like 'i4arg' except that it returns
c     a real*8 value parsed from the specified command-line
c     argument
c-----------------------------------------------------------
      real*8          r8arg

      real*8          default_fac
      parameter     ( default_fac = 2.0d0 )

      real*8          eps,          neweps,        fac

      fac = r8arg(1,default_fac)
      write(0,*) 'meps: using division factor: ', fac
```

```
eps    = 1.0d0
neweps = 1.0d0
do while( .true. )
   if( 1.0d0 .eq. (1.0d0 + neweps) ) then
      write(*,*) eps
      stop
   else
      eps    = neweps
      neweps = neweps / fac
   end if
end do

stop

end
```

```
##############################################################
# Output from 'meps' on SGIs (IEEE 64-bit floating point).
##############################################################

Script started on Wed Sep 20 20:15:05 2000

sgi1 1> make meps
f77 -g -64 -c meps.f
f77 -g -64 -L/usr/local/lib meps.o -lp410f -o meps

sgi1 2> meps
 meps: using division factor:    2.000000000000000
   2.2204460492503131E-16

sgi1 3> meps 1.01
 meps: using division factor:    1.010000000000000
   1.1104218387155329E-16

sgi1 4> meps 1.0001
 meps: using division factor:    1.000100000000000
   1.1102645224601785E-16
```

```
##############################################################
# Output from 'meps' on PC Linux machine (80 bit floating pt)
##############################################################

Script started on Wed Sep 20 20:18:42 2000

[phys410@vnfe1 ex6]$ make meps
f77 -g -c meps.f
f77 -g -L/usr/local/lib meps.o -lp410f -o meps

[phys410@vnfe1 ex6]$ meps
 meps: using division factor:   2.
  1.08420217E-19

[phys410@vnfe1 ex6]$ meps 1.01
 meps: using division factor:   1.01
  5.43645349E-20

[phys410@vnfe1 ex6]$ meps 1.0001
 meps: using division factor:   1.0001
  5.42121463E-20
```

```
c===========================================================
c     Program illustrating "catastrophic" loss of precision
c     resulting from the subtraction of two nearly equal
c     floating point values.
c===========================================================
      program         catprec

      implicit        none

      real*8          x
      parameter     ( x = 0.2d0 )

      integer         i
      real*8          h,     dsinx

      write(*,*) '      h      d(sin) approx   '//
     &           'd(sin) exact    d(sin) err'
      write(*,*)

      h = 0.5d0
      do i = 1 , 16
c-----------------------------------------------------------
c        Algebraically, in the limit h -> 0, dsinx should
c        approach cos(x), but sin(x+h) -> sin(x) so
c        catastrophic loss of precision occurs.
c-----------------------------------------------------------
         dsinx = (sin(x+h) - sin(x)) / h
         write(*,1000) h,  dsinx,  cos(x),   dsinx - cos(x)
1000     format(1P,E12.3,2E16.8,E12.3)
         h = 0.125d0 * h
      end do

      stop
      end
```

```
##############################################################
# Output from 'catprec' illustrating catastrophic precision
# loss due to subtraction of nearly-equal floating point
# values.
##############################################################

Script started on Wed Sep 20 20:16:12 2000

sgi1 1> make catprec
f77 -g -64 -c catprec.f
f77 -g -64 -L/usr/local/lib catprec.o -o catprec

sgi1 2> catprec

       h      d(sin) approx   d(sin) exact    d(sin) err

   5.000E-01  8.91096713E-01  9.80066578E-01  -8.897E-02
   6.250E-02  9.73222242E-01  9.80066578E-01  -6.844E-03
   7.813E-03  9.79280560E-01  9.80066578E-01  -7.860E-04
   9.766E-04  9.79969416E-01  9.80066578E-01  -9.716E-05
   1.221E-04  9.80054450E-01  9.80066578E-01  -1.213E-05
   1.526E-05  9.80065062E-01  9.80066578E-01  -1.516E-06
   1.907E-06  9.80066388E-01  9.80066578E-01  -1.895E-07
   2.384E-07  9.80066554E-01  9.80066578E-01  -2.369E-08
   2.980E-08  9.80066575E-01  9.80066578E-01  -2.731E-09
   3.725E-09  9.80066583E-01  9.80066578E-01   4.719E-09
   4.657E-10  9.80066597E-01  9.80066578E-01   1.962E-08
   5.821E-11  9.80066776E-01  9.80066578E-01   1.984E-07
   7.276E-12  9.80068207E-01  9.80066578E-01   1.629E-06
   9.095E-13  9.80072021E-01  9.80066578E-01   5.444E-06
   1.137E-13  9.80224609E-01  9.80066578E-01   1.580E-04
   1.421E-14  9.80468750E-01  9.80066578E-01   4.022E-04
```

```
c===========================================================
c     Implements matrix-matrix multiply
c
c     c = a b
c
c     where a, b and c are n x n (square) real*8 matrices.
c===========================================================
      subroutine dmmmult(a,b,c,n)

         implicit         none

         integer          n
         real*8           a(n,n),    b(n,n),    c(n,n)

         integer          i,     j,     k

         do j = 1 , n
            do i = 1 , n
               c(i,j) = 0.0d0
               do k = 1 , n
                  c(i,j) = c(i,j) + a(i,k) * b(k,j)
               end do
            end do
         end do

         return

      end
```

```
c===========================================================
c     Writes a double precision matrix (two dimensional
c     array) to file 'fname'.  If 'fname' is the
c     string '-', the matrix is written to standard input.
c
c     This routine is modelled on 'dvto' previously
c     discussed in class: see ~phys410/f77/ex3/dvto.f
c===========================================================
      subroutine dmto(fname,a,d1,d2)
c-----------------------------------------------------------
c     Arguments:
c
c        fname:  (I)    File name
c        a:      (I)    Input matrix
c        d1:     (I)    First dimension of a
c        d2:     (I)    Second dimension of a
c-----------------------------------------------------------
         implicit         none
         integer          indlnb,      getu

         character*(*)    fname
         integer          d1,          d2
         real*8           a(d1,d2)

         integer          ustdout
         parameter      ( ustdout = 6 )

         integer          uto,         rc

c-----------------------------------------------------------
c        Parse fname: either "attach" 'uto' to stdout or
c        get a unit number using 'getu', and open the
c        file 'fname' for formatted I/O via 'uto'
c-----------------------------------------------------------
```

```
         if( fname .eq. '-' ) then
            uto = ustdout
         else
            uto = getu()
            open(uto,file=fname(1:indlnb(fname)),
     &           form='formatted',iostat=rc)
            if( rc .ne. 0 ) then
               write(0,*) 'dmto: Error opening ',
     &                    fname(1:indlnb(fname))
               return
            end if
         end if
c------------------------------------------------------------
c        Write dimensions, then array elements
c------------------------------------------------------------
         write(uto,*,iostat=rc) d1, d2
         if( rc .ne. 0 ) then
            write(0,*) 'dmto: Error writing dimensions'
         go to 500
         end if

         write(uto,*,iostat=rc) a
         if( rc .ne. 0 ) then
            write(0,*) 'dmto: Error reading matrix'
         end if
c------------------------------------------------------------
c        Exit: Close file and return
c------------------------------------------------------------
 500     continue
         close(uto)

         return
         end
```

```
c===========================================================
c     Returns a double precision matrix (two dimensional
c     array) read from file 'fname'.  If 'fname' is the
c     string '-', the matrix is read from standard input.
c
c     The dimensions of the matrix must precede the matrix
c     elements themselves in the file.  Specifically, the
c     file should have been created using the following
c     list-directed, formatted READ statement
c     (or equivalent):
c
c        integer      d1,    d2
c        real*8       a(d1,d2)
c        integer      uout
c        write(uout,*) d1, d2
c        write(uout,*) a
c
c     This routine is modelled on 'dvfrom' previously
c     discussed in class: see ~phys410/f77/ex3/dvfrom.f
c
c     Note the use of helper routine 'dmfrom1' which
c     reads actual array values once bounds have been
c     extracted from file.
c===========================================================
      subroutine dmfrom(fname,a,d1,d2,asize)
c-----------------------------------------------------------
c     Arguments:
c
c        fname:  (I)    File name
c        a:      (O)    Return matrix
c        d1:     (O)    First dimension of a
c        d2:     (O)    Second dimension of a
c        asize:  (I)    Maximum size (d1 * d2) of a
c-----------------------------------------------------------
```

```
      implicit          none

      integer           indlnb,      getu

      character*(*)     fname
      integer           d1,          d2,         asize
      real*8            a(d1,d2)

      integer           ustdin
      parameter       ( ustdin = 5 )

      integer           ufrom,       rc

c-------------------------------------------------------------
c       Parse fname: either "attach" 'ufrom' to stdin or
c       get a unit number using 'getu', and open the
c       file 'fname' for formatted I/O via 'ufrom'
c-------------------------------------------------------------
      if( fname .eq. '-' ) then
         ufrom = ustdin
      else
         ufrom = getu()
         open(ufrom,file=fname(1:indlnb(fname)),
     &           form='formatted',iostat=rc,status='old')
         if( rc .ne. 0 ) then
            write(0,*) 'dmfrom: Error opening ',
     &                 fname(1:indlnb(fname))
            return
         end if
      end if
```

```
c-----------------------------------------------------------
c        Read dimensions and abort if there is insufficient
c        storage for the entire matrix.  Note the 'go to'
c        to the 'exit block' since we've opened a file now
c        and should close it, even if there's an error.
c        Also, we set the dimensions to 0 for all error
c        conditions as a way of communicating failure to
c        the calling routine.
c-----------------------------------------------------------
         read(ufrom,*,iostat=rc) d1, d2
         if( rc .ne. 0 ) then
            write(0,*) 'dmfrom: Error reading dimensions'
            d1 = 0
            d2 = 0
         go to 500
         end if
         if( (d1 * d2) .gt. asize ) then
            write(0,*) 'dmfrom: Insufficient storage'
            d1 = 0
            d2 = 0
         go to 500
         end if
c-----------------------------------------------------------
c        Now that dimensions have been determined call
c        helper routine to read values
c-----------------------------------------------------------
         call dmfrom1(ufrom,a,d1,d2,rc)
         if( rc .ne. 0 ) then
            write(0,*) 'dmfrom: Error reading matrix'
            d1 = 0
            d2 = 0
         end if
```

```
c--------------------------------------------------------------
c        Exit: Close file and return
c--------------------------------------------------------------
 500     continue
         close(ufrom)

         return
      end

c==============================================================
c     Helper routine for dmfrom: Reads array values, returns
c     I/O status to calling routine via 'rc'
c==============================================================
      subroutine dmfrom1(ufrom,a,d1,d2,rc)

         implicit      none

         integer       d1,       d2,     ufrom,     rc
         real*8        a(d1,d2)

         read(ufrom,*,iostat=rc) a

         return

      end
```

```
c===========================================================
c     Test program for subroutine 'dmfrom', 'dmto' and
c     'dmmmult' (see 'dmroutines.f')
c
c     Program expects one argument, the name of a file which
c     contains a real*8 square matrix written as descibed
c     in the documentation for 'dmfrom' in 'dmroutines.f'
c     Use '-' to read from stdin.  Program then computes
c     square of matrix and outputs result to stdout.
c===========================================================

      program        tdm

      implicit       none

      integer        iargc

      character*256  fname

c-----------------------------------------------------------
c     Maximum size for input and output arrays (matrices).
c-----------------------------------------------------------
      integer        maxsize
      parameter    ( maxsize = 100 000 )
      real*8         a(maxsize),   asq(maxsize)
      integer        d1a,          d2a

      if( iargc() .ne. 1 ) go to 900
      call getarg(1,fname)

c-----------------------------------------------------------
c     Read matrix ...
c-----------------------------------------------------------
      call dmfrom(fname,a,d1a,d2a,maxsize)
```

```
      if( d1a .gt. 0  .and.  d2a .gt. 0 ) then
         if( d1a .eq. d2a ) then
c------------------------------------------------------------
c           Compute square ...
c------------------------------------------------------------
            call dmmmult(a,a,asq,d1a,d1a)
c------------------------------------------------------------
c           ... and output.
c------------------------------------------------------------
            call dmto('-',asq,d1a,d1a)
         else
            write(0,*) 'tdm: Input array not square'
         end if
      else
         write(0,*) 'tdm: dmfrom() failed'
      end if

      stop

 900  continue
         write(0,*) 'usage: tdm <file name>'
         write(0,*)
         write(0,*) '       Use ''tdm -'' to read ',
     &              'from standard input'

      stop

      end
```

```
############################################################
# Building 'tdm' and sample output
#############################################################

Script started on Wed Sep 20 20:31:45 2000
sgi1 1> pwd; ls
/usr/people/phys410/f77/ex6
Makefile       catprec.f      dmroutines.f   meps.f         tdm.f

sgi1 2> make tdm
f77 -g -64 -c tdm.f
f77 -g -64 -c dmroutines.f
f77 -g -64 -L/usr/local/lib tdm.o dmroutines.o -lp410f -o tdm

sgi1 3> tdm
 usage: tdm <file name>

        Use 'tdm -' to read from standard input

sgi1 4> tdm -
2 2
1 2 3 4
            2            2
    7.000000000000000       10.00000000000000       15.00000000000000
    22.00000000000000

sgi1 5> tdm -
2 3
1 2 3 4 5 6
 tdm: Input array not square
```

```
.IGNORE:

F77_COMPILE  = $(F77) $(F77FLAGS) $(F77CFLAGS)
F77_LOAD     = $(F77) $(F77FLAGS) $(F77LFLAGS)

.f.o:
   $(F77_COMPILE) $*.f

EXECUTABLES = meps catprec tdm

all: $(EXECUTABLES)

meps: meps.o
   $(F77_LOAD) meps.o -lp410f -o meps

catprec: catprec.o
   $(F77_LOAD) catprec.o -o catprec

tdm: tdm.o dmroutines.o
   $(F77_LOAD) tdm.o dmroutines.o -lp410f -o tdm

clean:
   rm *.o
   rm $(EXECUTABLES)
   rm core
```