

```

                                globals.f90
1  ! . . . . .
2  ! .
3  ! .                               S T A P 9 0
4  ! .
5  ! .      AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90
6  ! .      Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose
7  ! .
8  ! .      Xiong Zhang, (2013)
9  ! .      Computational Dynamics Group, School of Aerospace
10 ! .      Tsinghua Univerity
11 ! .
12 ! . . . . .
13
14 ! . Define global variables
15
16 module GLOBALS
17
18     integer, parameter :: IELMNT=1    ! Unit storing element data
19     integer, parameter :: ILOAD=2     ! Unit storing load vectors
20     integer, parameter :: IIN=5       ! Unit used for input
21     integer, parameter :: IOUT=6      ! Unit used for output
22
23     integer :: NUMNP                  ! Total number of nodal points
24                                     ! = 0 : Program stop
25     integer :: NEQ                    ! Number of equations
26     integer :: NWK                    ! Number of matrix elements
27     integer :: MK                      ! Maximum half bandwidth
28
29     integer :: IND                    ! Solution phase indicator
30                                     ! 1 - Read and generate element information
31                                     ! 2 - Assemble structure stiffness matrix
32                                     ! 3 - Stress calculations
33     integer :: NPAR(10)              ! Element group control data
34                                     ! NPAR(1) - Element type
35                                     ! 1 : Truss element
36                                     ! NPAR(2) - Number of elements
37                                     ! NPAR(3) - Number of different sets of material and
38                                     ! cross-sectional constants
39     integer :: NUMEG                  ! Total number of element groups, > 0
40
41     integer :: MODEX                  ! Solution mode: 0 - data check only; 1 - execution
42
43     real :: TIM(5)                    ! Timing information
44     character*80 :: HED               ! Master heading information for use in labeling the output
45
46     integer :: NFIRST
47     integer :: NLAST
48     integer :: NUMEST
49     integer :: MIDEST
50     integer :: MAXEST
51
52     integer :: NG
53
54 end module GLOBALS

```

```

1  ! . . . . .
2  ! .
3  ! .               S T A P 9 0
4  ! .
5  ! .      AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90
6  ! .      Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose
7  ! .
8  ! .      Xiong Zhang, (2013)
9  ! .      Computational Dynamics Group, School of Aerospace
10 ! .      Tsinghua Univerity
11 ! .
12 ! . . . . .
13
14 PROGRAM STAP90
15
16   USE GLOBALS
17   USE MEMALLOCATE
18
19   IMPLICIT NONE
20   INTEGER :: NLCASE, NEQ1, NLOAD, MM
21   INTEGER :: L, LL, I
22   REAL :: TT
23
24 ! OPEN INPUT DATA FILE, RESULTS OUTPUT FILE AND TEMPORARY FILES
25 CALL OPENFILES() P5: IIN - Input file name;
26                               IOUT - Output file
27 NUMEST=0
28 MAXEST=0
29
30 ! * * * * *
31 ! *               INPUT PHASE
32 ! * * * * *
33
34 WRITE(*, '("Input phase ... ")')
35
36 CALL SECOND (TIM(1))
37
38 ! Read control information
39
40 !   HED      - The master heading informaiton for use in labeling the output
41 !   NUMNP    - Total number of nodal points
42 !             0 : program stop
43 !   NUMEG    - Total number of element group (>0)
44 !   NLCASE   - Number of load case (>0)
45 !   MODEX    - Solution mode
46 !             0 : data check only;
47 !             1 : execution
48
49 READ (IIN, '(A80,/,4I5)') HED, NUMNP, NUMEG, NLCASE, MODEX
50
51 IF (NUMNP.EQ.0) STOP ! Data check mode
52
53 WRITE (IOUT, "(/, ' ', A80, //, &
54   ' C O N T R O L   I N F O R M A T I O N', //, &
55   '      NUMBER OF NODAL POINTS', 10(' . '), (NUMNP) = ', I5, /, &
56   '      NUMBER OF ELEMENT GROUPS', 9(' . '), (NUMEG) = ', I5, /, &
57   '      NUMBER OF LOAD CASES', 11(' . '), (NLCASE) = ', I5, /, &
58   '      SOLUTION MODE ', 14(' . '), (MODEX) = ', I5, /, &
59   '      EQ. 0, DATA CHECK', /, &
60   '      EQ. 1, EXECUTION')") HED, NUMNP, NUMEG, NLCASE, MODEX
61
62 ! Read nodal point data
63
64 ! ALLOCATE STORAGE
65 !   ID(3, NUMNP) : Boundary condition codes (0=free, 1=deleted)
66 !   X(NUMNP)     : X coordinates
67 !   Y(NUMNP)     : Y coordinates
68 !   Z(NUMNP)     : Z coordinates
69
70 CALL MEMALLOC(1, "ID", 3*NUMNP, 1)
71 CALL MEMALLOC(2, "X", NUMNP, ITWO)
72 CALL MEMALLOC(3, "Y", NUMNP, ITWO)
73 CALL MEMALLOC(4, "Z", NUMNP, ITWO)
74

```

shell, plate, Beam?

57848

```

1      CALL INPUT (IA(NP(1)), DA(NP(2)), DA(NP(3)), DA(NP(4)), NUMNP, NEQ)
2
3      NEQ1=NEQ + 1
4
5      ! Calculate and store load vectors
6      !   R(NEQ) : Load vector
7
8      CALL MEMALLOC(5, "R", NEQ, ITWO)
9
10     WRITE (IOUT, "(//, 'LOAD CASE DATA')")
11
12     REWIND ILOAD
13
14     DO L=1, NLCASE
15
16     !   LL - Load case number
17     !   NLOAD - The number of concentrated loads applied in this load case
18
19     READ (IIN, '(2I5)') LL, NLOAD
20
21     WRITE (IOUT, "(/, 'LOAD CASE NUMBER', 7(' '), ' = ', I5, /, &
22             'NUMBER OF CONCENTRATED LOADS . = ', I5)") LL, NLOAD
23
24     IF (LL.NE.L) THEN
25         WRITE (IOUT, "( ' *** ERROR *** LOAD CASES ARE NOT IN ORDER' )")
26         STOP
27     ENDIF
28
29     !   Allocate storage
30     !   NOD(NLOAD) : Node number to which this load is applied (1~NUMNP)
31     !   IDIRN(NLOAD) : Degree of freedom number for this load component
32     !               1 : X-direction;
33     !               2 : Y-direction;
34     !               3 : Z-direction
35     !   FLOAD(NLOAD) : Magnitude of load
36
37     CALL MEMALLOC(6, "NOD", NLOAD, 1)
38     CALL MEMALLOC(7, "IDIRN", NLOAD, 1)
39     CALL MEMALLOC(8, "FLOAD", NLOAD, ITWO)
40
41     CALL LOADS (DA(NP(5)), IA(NP(6)), IA(NP(7)), DA(NP(8)), IA(NP(1)), NLOAD, NEQ)
42
43     END DO
44
45     ! Read, generate and store element data
46
47     ! Clear storage
48     !   MHT(NEQ) - Vector of column heights
49
50     CALL MEMFREEFROM(5)
51     CALL MEMALLOC(5, "MHT", NEQ, 1)
52
53     IND=1 ! Read and generate element information
54     CALL ELCAL
55
56     CALL SECOND (TIM(2))
57
58     ! * * * * *
59     ! *           SOLUTION PHASE           *
60     ! * * * * *
61
62     WRITE(*, '("Solution phase ... ")')
63
64     ! Assemble stiffness matrix
65
66     ! ALLOCATE STORAGE
67     !   MAXA(NEQ+1)
68     CALL MEMFREEFROM(6)
69     CALL MEMFREEFROMTO(2, 4)
70     CALL MEMALLOC(2, "MAXA", NEQ+1, 1)
71
72     CALL ADDRESS (IA(NP(2)), IA(NP(5)))
73
74     ! ALLOCATE STORAGE

```

P7

P8

```

                                stap.f90
1  !   A(NWK) - Global structure stiffness matrix K
2  !   R(NEQ) - Load vector R and then displacement solution U
3
4  MM=NWK/NEQ
5
6  CALL MEMALLOC(3,"STFF",NWK,ITWO)
7  CALL MEMALLOC(4,"R",NEQ,ITWO)
8  CALL MEMALLOC(11,"ELEGP",MAXEST,1)
9
10 ! Write total system data
11
12 WRITE (IOUT,"(//,' TOTAL SYSTEM DATA',//, &
13      '   NUMBER OF EQUATIONS',14(' '),', (NEQ) = ',I5,/, &
14      '   NUMBER OF MATRIX ELEMENTS',11(' '),', (NWK) = ',I5,/, &
15      '   MAXIMUM HALF BANDWIDTH ',12(' '),', (MK) = ',I5,/, &
16      '   MEAN HALF BANDWIDTH',14(' '),', (MM) = ',I5)") NEQ, NWK, MK, MM
17
18 ! In data check only mode we skip all further calculations
19
20 IF (MODEX.LE.0) THEN
21   CALL SECOND (TIM(3))
22   CALL SECOND (TIM(4))
23   CALL SECOND (TIM(5))
24 ELSE
25   IND=2 ! Assemble structure stiffness matrix
26   CALL ASSEM (A(NP(11)))
27   ← Element group data
28   CALL SECOND (TIM(3))
29
30 !   Triangularize stiffness matrix
31   CALL COLSOL (DA(NP(3)),DA(NP(4)),IA(NP(2)),NEQ,NWK,NEQ1,1) p16
32   A R MAX
33   CALL SECOND (TIM(4))
34
35   IND=3 ! Stress calculations
36
37   REWIND ILOAD
38   DO L=1,NLCASE
39     CALL LOADV (DA(NP(4)),NEQ) ! Read in the load vector p8
40
41 !   Solve the equilibrium equations to calculate the displacements
42   CALL COLSOL (DA(NP(3)),DA(NP(4)),IA(NP(2)),NEQ,NWK,NEQ1,2)
43
44   WRITE (IOUT,"(//,' LOAD CASE ',I3)") L
45   CALL WRITED (DA(NP(4)),IA(NP(1)),NEQ,NUMNP) ! Print displacements
46
47 !   Calculation of stresses
48   CALL STRESS (A(NP(11))) p10
49   ← Element group data
50   END DO
51
52   CALL SECOND (TIM(5))
53 END IF
54
55 ! Print solution times
56
57 TT=0.
58 DO I=1,4
59   TIM(I)=TIM(I+1) - TIM(I)
60   TT=TT + TIM(I)
61 END DO
62
63 WRITE (IOUT,"(//, &
64      ' S O L U T I O N   T I M E   L O G   I N   S E C',//, &
65      '   TIME FOR INPUT PHASE ',14(' '),', =',F12.2,/, &
66      '   TIME FOR CALCULATION OF STIFFNESS MATRIX . . . =',F12.2,/, &
67      '   TIME FOR FACTORIZATION OF STIFFNESS MATRIX . . . =',F12.2,/, &
68      '   TIME FOR LOAD CASE SOLUTIONS ',10(' '),', =',F12.2,//, &
69      '   T O T A L   S O L U T I O N   T I M E . . . . . =',F12.2)") (TIM(I),I=1,4), TT
70
71
72 WRITE (*,"(//, &
73      ' S O L U T I O N   T I M E   L O G   I N   S E C',//, &
74      '   TIME FOR INPUT PHASE ',14(' '),', =',F12.2,/, &

```

```

                                stap.f90
1      ,    TIME FOR CALCULATION OF STIFFNESS MATRIX . . . . =',F12.2, /,  &
2      ,    TIME FOR FACTORIZATION OF STIFFNESS MATRIX . . . =',F12.2, /,  &
3      ,    TIME FOR LOAD CASE SOLUTIONS ',10(' .'),' =',F12.2, /,  &
4      ,    T O T A L   S O L U T I O N   T I M E . . . . =',F12.2)~) (TIM(I), I=1, 4), TT
5  STOP
6
7  END PROGRAM STAP90
8
9
10 SUBROUTINE SECOND (TIM)
11 ! USE DFPORT ! Only for Compaq Fortran
12 IMPLICIT NONE
13 REAL :: TIM
14
15 ! This is a Fortran 95 intrinsic subroutine
16 ! Returns the processor time in seconds
17
18 CALL CPU_TIME(TIM)
19
20 RETURN
21 END SUBROUTINE SECOND
22
23
24 SUBROUTINE WRITED (DISP, ID, NEQ, NUMNP)
25 ! . . . . .
26 ! .
27 ! . To print displacements
28 ! . . . . .
29
30 USE GLOBALS, ONLY : IOUT
31
32 IMPLICIT NONE
33 INTEGER :: NEQ, NUMNP, ID(3, NUMNP)
34 REAL(8) :: DISP(NEQ), D(3)
35 INTEGER :: IC, II, I, KK, IL
36
37 ! Print displacements
38
39 WRITE (IOUT, "(//, ' D I S P L A C E M E N T S', //, ' NODE ', 10X,  &
40           ' X-DISPLACEMENT   Y-DISPLACEMENT   Z-DISPLACEMENT' )")
41
42 IC=4
43
44 DO II=1, NUMNP
45   IC=IC + 1
46   IF (IC.GE.56) THEN
47     WRITE (IOUT, "(//, ' D I S P L A C E M E N T S', //, ' NODE ', 10X,  &
48           ' X-DISPLACEMENT   Y-DISPLACEMENT   Z-DISPLACEMENT' )")
49     IC=4
50   END IF
51
52   DO I=1, 3
53     D(I)=0.
54   END DO
55
56   DO I=1, 3
57     KK=ID(I, II)
58     IL=1 I
59     IF (KK.NE.0) D(I)=DISP(KK)
60   END DO
61
62   WRITE (IOUT, ' (1X, I3, 8X, 3E18.6)' ) II, D
63
64 END DO
65
66 RETURN
67
68 END SUBROUTINE WRITED
69
70
71 SUBROUTINE OPENFILES()
72 ! . . . . .
73 ! .
74 ! . Open input data file, results output file and temporary files .

```

```

1  ! . . . . .
2
3  USE GLOBALS
4  ! use DFLIB ! for NARGS() ! Only for Compaq Fortran
5
6  IMPLICIT NONE
7  LOGICAL :: EX
8  CHARACTER*80 FileInp
9
10 ! Only for Compaq Fortran
11 ! if(NARGS().ne.2) then
12 !   stop 'Usage: mpm3d InputFileName'
13 ! else
14 !   call GETARG(1,FileInp)
15 ! end if
16
17 if(COMMAND_ARGUMENT_COUNT().ne.1) then
18   stop 'Usage: STAP90 InputFileName'
19 else
20   call GET_COMMAND_ARGUMENT(1,FileInp)
21 end if
22
23 INQUIRE(FILE = FileInp, EXIST = EX)
24 IF (.NOT. EX) THEN
25   PRINT *, "*** STOP *** FILE STAP90.IN DOES NOT EXIST !"
26   STOP
27 END IF
28
29 OPEN(IIN , FILE = FileInp, STATUS = "OLD")
30 OPEN(IOUT , FILE = "STAP90.OUT", STATUS = "REPLACE")
31
32 OPEN(IELMNT, FILE = "ELMNT.TMP", FORM = "UNFORMATTED")
33 OPEN(ILOAD , FILE = "LOAD.TMP", FORM = "UNFORMATTED")
34 END SUBROUTINE OPENFILES
35
36
37 SUBROUTINE CLOSEFILES()
38 ! . . . . .
39 ! .
40 !   Close all data files
41 ! . . . . .
42
43 USE GLOBALS
44 IMPLICIT NONE
45 CLOSE(IIN)
46 CLOSE(IOUT)
47 CLOSE(IELMNT)
48 CLOSE(ILOAD)
49 END SUBROUTINE CLOSEFILES

```

```

1  ! . . . . .
2  ! .
3  ! .               S T A P 9 0
4  ! .
5  ! .      AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90
6  ! .      Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose
7  ! .
8  ! .      Xiong Zhang, (2013)
9  ! .      Computational Dynamics Group, School of Aerospace
10 ! .      Tsinghua University
11 ! .
12 ! . . . . .
13
14 SUBROUTINE INPUT (ID,X,Y,Z,NUMNP,NEQ)
15 ! . . . . .
16 ! .
17 ! .      To read, generate, and print nodal point input data
18 ! .      To calculate equation numbers and store them in id array
19 ! .
20 ! .      N = Element number
21 ! .      ID = Boundary condition codes (0=free,1=deleted)
22 ! .      X,Y,Z = Coordinates
23 ! .      KN = Generation code
24 ! .      i.e. increment on nodal point number
25 ! .
26 ! . . . . .
27
28 USE GLOBALS, ONLY : IIN, IOUT
29
30 IMPLICIT NONE
31 INTEGER :: NUMNP,NEQ, ID(3,NUMNP)
32 REAL(8) :: X(NUMNP),Y(NUMNP),Z(NUMNP)
33 INTEGER :: I, J, N
34
35 ! Read and generate nodal point data
36
37 N = 0
38 DO WHILE (N.NE.NUMNP)
39   READ (IIN,"(4I5,3F10.0,I5)") N, (ID(I,N),I=1,3),X(N),Y(N),Z(N)
40 END DO
41
42 ! Write complete nodal data
43
44 WRITE (IOUT,"(//,' N O D A L   P O I N T   D A T A',/)" )
45
46 WRITE (IOUT,"('  NODE',10X,' BOUNDARY',25X,' NODAL POINT',/, &
47   ' NUMBER          CONDITION CODES',21X,' COORDINATES', /,15X, &
48   ' X      Y      Z',15X,' X',12X,' Y',12X,' Z' )" )
49
50 DO N=1,NUMNP
51   WRITE (IOUT,"(I5,6X,3I5,6X,3F13.3)" ) N, (ID(I,N),I=1,3),X(N),Y(N),Z(N)
52 END DO
53
54 ! Number unknowns
55
56 NEQ=0
57 DO N=1,NUMNP
58   DO I=1,3
59     IF (ID(I,N) .EQ. 0) THEN
60       NEQ=NEQ + 1
61       ID(I,N)=NEQ
62     ELSE
63       ID(I,N)=0
64     END IF
65   END DO
66 END DO
67
68 ! Write equation numbers
69 WRITE (IOUT,"(//,' EQUATION NUMBERS',//,'  NODE',9X, &
70   ' DEGREES OF FREEDOM',/, ' NUMBER',/, &
71   ' N',13X,' X      Y      Z',/, (1X,I5,9X,3I5))" ) (N, (ID(I,N),I=1,3),N=1,NUMNP)
72
73 RETURN
74

```

```

1  END SUBROUTINE INPUT
2
3
4  SUBROUTINE LOADS (R,NOD, IDIRN, FLOAD, ID, NLOAD, NEQ)
5  ! . . . . .
6  ! .
7  ! .   To read nodal load data .
8  ! .   To calculate the load vector r for each load case and .
9  ! .   write onto unit ILOAD .
10 ! .
11 ! . . . . .
12 USE GLOBALS, ONLY : IIN, IOUT, ILOAD, MODEX
13
14 IMPLICIT NONE
15 INTEGER :: NLOAD, NEQ, ID(3,*), NOD(NLOAD), IDIRN(NLOAD)
16 REAL(8) :: R(NEQ), FLOAD(NLOAD)
17 INTEGER :: I, L, LI, LN, II
18
19 WRITE (IOUT, "(/, '   NODE          DIRECTION          LOAD', /, '   NUMBER', 19X, 'MAGNITUDE' ) ")
20
21 READ (IIN, "(2I5, F10.0)") (NOD(I), IDIRN(I), FLOAD(I), I=1, NLOAD)
22
23 WRITE (IOUT, "( ' ', I6, 9X, I4, 7X, E12.5)") (NOD(I), IDIRN(I), FLOAD(I), I=1, NLOAD)
24
25 IF (MODEX.EQ.0) RETURN
26
27 DO I=1, NEQ
28   R(I)=0.
29 END DO
30
31 DO L=1, NLOAD
32   LN=NOD(L)
33   LI=IDIRN(L)
34   II=ID(LI, LN)
35   IF (II > 0) R(II)=R(II) + FLOAD(L)
36 END DO
37
38 WRITE (ILOAD) R
39
40 RETURN
41
42 END SUBROUTINE LOADS
43
44
45 SUBROUTINE LOADV (R, NEQ)
46 ! . . . . .
47 ! .
48 ! .   To obtain the load vector .
49 ! . . . . .
50 !
51 USE GLOBALS, ONLY : ILOAD
52
53 IMPLICIT NONE
54 INTEGER :: NEQ
55 REAL(8) :: R(NEQ)
56
57 READ (ILOAD) R
58
59 RETURN
60 END SUBROUTINE LOADV

```



```

1  ! . . . . .
2  ! .
3  ! .           S T A P 9 0           .
4  ! .
5  ! .   AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90
6  ! .   Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose
7  ! .
8  ! .   Xiong Zhang, (2013)
9  ! .   Computational Dynamics Group, School of Aerospace
10 ! .   Tsinghua Univerity
11 ! .
12 ! . . . . .
13
14 SUBROUTINE ELCAL
15 ! . . . . .
16 ! .
17 ! .   To loop over all element groups for reading,
18 ! .   generating and storing the element data
19 ! .
20 ! . . . . .
21   USE GLOBALS
22   USE MEMALLOCATE
23
24   IMPLICIT NONE
25   INTEGER :: N, I
26
27   REWIND IELMNT
28   WRITE (IOUT,"(//,' E L E M E N T   G R O U P   D A T A',//)")
29
30 ! Loop over all element groups
31
32   DO N=1,NUMEG
33     IF (N.NE.1) WRITE (IOUT,'(1X)')
34
35     READ (IIN,'(10I5)') NPAR
36
37     CALL ELEMNT
38
39     IF (MIDEST.GT.MAXEST) MAXEST=MIDEST
40
41     WRITE (IELMNT) MIDEST,NPAR,(A(I),I=NFIRST,NLAST)
42
43   END DO
44
45   RETURN
46
47 END SUBROUTINE ELCAL
48
49
50 SUBROUTINE ELEMNT
51 ! . . . . .
52 ! .
53 ! .   To call the appropriate element subroutine
54 ! .
55 ! . . . . .
56
57   USE GLOBALS
58
59   IMPLICIT NONE
60   INTEGER :: NPAR1
61
62   NPAR1=NPAR(1)
63
64   IF (NPAR1 == 1) THEN
65     CALL TRUSS
66   ELSE
67     ! Other element types would be called here, identifying each
68     ! element type by a different NPAR(1) parameter
69   END IF
70
71   RETURN
72 END SUBROUTINE ELEMNT
73
74

```

for 90

```

1  SUBROUTINE STRESS (AA)
2  ! .....
3  ! .
4  ! .   To call the element subroutine for the calculation of stresses .
5  ! .
6  ! .....
7
8  USE GLOBALS, ONLY : IELMNT, NG, NUMEST, NPAR, NUMEG
9
10 IMPLICIT NONE
11 REAL :: AA(*)
12 INTEGER N, I
13
14 ! Loop over all element groups
15
16 REWIND IELMNT
17
18 DO N=1, NUMEG
19   NG=N
20
21   READ (IELMNT) NUMEST, NPAR, (AA(I), I=1, NUMEST)
22
23   CALL ELEMNT
24 END DO
25
26 RETURN
27 END subroutine STRESS

```

```

truss.f90
1  ! . . . . .
2  ! .
3  ! .          S T A P 9 0          .
4  ! .
5  ! .      AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90
6  ! .      Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose
7  ! .
8  ! .      Xiong Zhang, (2013)
9  ! .      Computational Dynamics Group, School of Aerospace
10 ! .      Tsinghua University
11 ! .
12 ! . . . . .
13
14 SUBROUTINE TRUSS
15 ! . . . . .
16 ! .
17 ! .      To set up storage and call the truss element subroutine
18 ! .
19 ! . . . . .
20
21 USE GLOBALS
22 USE MEMALLOCATE
23
24 IMPLICIT NONE
25 INTEGER :: NUME, NUMMAT, MM, N101, N102, N103, N104, N105, N106
26
27 NUME = NPAR(2)
28 NUMMAT = NPAR(3)
29
30 ! Allocate storage for element group data
31 IF (IND == 1) THEN
32     MM = 2*NUMMAT*ITWO + 7*NUME + 6*NUME*ITWO
33     CALL MEMALLOC(11, "ELEGP", MM, 1)
34 END IF
35
36 NFIRST=NP(11)    ! Pointer to the first entry in the element group data array
37                  ! in the unit of single precision (corresponding to A)
38
39 ! Calculate the pointer to the arrays in the element group data
40 ! N101: E(NUMMAT)
41 ! N102: AREA(NUMMAT)
42 ! N103: LM(6, NUME)
43 ! N104: XYZ(6, NUME)
44 ! N105: MTAP(NUME)
45 N101=NFIRST
46 N102=N101+NUMMAT*ITWO
47 N103=N102+NUMMAT*ITWO
48 N104=N103+6*NUME
49 N105=N104+6*NUME*ITWO
50 N106=N105+NUME
51 NLAST=N106
52
53 MIDEST=NLAST - NFIRST
54
55 CALL RUSS (IA(NP(1)), DA(NP(2)), DA(NP(3)), DA(NP(4)), DA(NP(4)), IA(NP(5)), &
56           A(N101), A(N102), A(N103), A(N104), A(N105))
57 RETURN
58
59 END SUBROUTINE TRUSS
60
61
62
63 SUBROUTINE RUSS (ID, X, Y, Z, U, MHT, E, AREA, LM, XYZ, MATP)
64 ! . . . . .
65 ! .
66 ! .      TRUSS element subroutine
67 ! .
68 ! . . . . .
69
70 USE GLOBALS
71 USE MEMALLOCATE
72
73 IMPLICIT NONE
74 INTEGER :: ID(3, NUMNP), LM(6, NPAR(2)), MATP(NPAR(2)), MHT(NEQ)

```

ID X Y Z U MHT
E AREA LM XYZ MTAP
2VF. Fortran | Diagnostics | check
Routine interfaces | No

```

truss.f90
1  REAL (8) :: X(NUMNP), Y(NUMNP), Z(NUMNP), E(NPAR(3)), AREA(NPAR(3)), &
2      XYZ(6, NPAR(2)), U(NEQ)
3  REAL (8) :: S(6,6), ST(6), D(3)
4
5  INTEGER :: NPAR1, NUME, NUMMAT, ND, I, J, L, N
6  INTEGER :: MTYPE, IPRINT
7  REAL (8) :: XL2, XL, SQRT, XX, YY, STR, P
8
9  NPAR1 = NPAR(1)
10 NUME = NPAR(2)
11 NUMMAT = NPAR(3)
12
13 ND=6
14
15 ! Read and generate element information
16 IF (IND.EQ. 1) THEN
17
18     WRITE (IOUT, "( ' E L E M E N T   D E F I N I T I O N' , //, &
19         '   ELEMENT TYPE' , 13(' . '), ' ( NPAR(1) ) . . =', I5, /, &
20         '   EQ. 1, TRUSS ELEMENTS' , /, &
21         '   EQ. 2, ELEMENTS CURRENTLY' , /, &
22         '   EQ. 3, NOT AVAILABLE' , //, &
23         '   NUMBER OF ELEMENTS.' , 10(' . '), ' ( NPAR(2) ) . . =', I5, /) ") NPAR1, NUME
24
25     IF (NUMMAT.EQ. 0) NUMMAT=1
26
27     WRITE (IOUT, "( ' M A T E R I A L   D E F I N I T I O N' , //, &
28         '   NUMBER OF DIFFERENT SETS OF MATERIAL' , /, &
29         '   AND CROSS-SECTIONAL CONSTANTS' , &
30         4(' . '), ' ( NPAR(3) ) . . =', I5, /) ") NUMMAT
31
32     WRITE (IOUT, "( '   SET           YOUNG' 'S      CROSS-SECTIONAL' , /, &
33         '   NUMBER           MODULUS' , 10X, ' AREA' , /, &
34         15 X, ' E' , 14X, ' A' ) ")
35
36     DO I=1, NUMMAT
37         READ (IIN, '(I5, 2F10.0)') N, E(N), AREA(N) ! Read material information
38         WRITE (IOUT, "(I5, 4X, E12.5, 2X, E14.6) ") N, E(N), AREA(N)
39     END DO
40
41     WRITE (IOUT, "( //, ' E L E M E N T   I N F O R M A T I O N' , //, &
42         '   ELEMENT           NODE           NODE           MATERIAL' , /, &
43         '   NUMBER-N           I           J           SET NUMBER' ) ")
44
45     N=0
46     DO WHILE (N.NE. NUME)
47         READ (IIN, '(5I5)') N, I, J, MTYPE ! Read in element information
48
49         ! Save element information
50         XYZ(1,N)=X(I) ! Coordinates of the element's left node
51         XYZ(2,N)=Y(I)
52         XYZ(3,N)=Z(I)
53
54         XYZ(4,N)=X(J) ! Coordinates of the element's right node
55         XYZ(5,N)=Y(J)
56         XYZ(6,N)=Z(J)
57
58         MATP(N)=MTYPE ! Material type
59
60         DO L=1, 6
61             LM(L,N)=0
62         END DO
63
64         DO L=1, 3
65             LM(L,N)=ID(L, I) ! Connectivity matrix
66             LM(L+3, N)=ID(L, J)
67         END DO
68
69         ! Update column heights and bandwidth
70         CALL COLHT (MHT, ND, LM(1, N))
71
72         WRITE (IOUT, "(I5, 6X, I5, 4X, I5, 7X, I5) ") N, I, J, MTYPE
73
74     END DO

```

```

1      RETURN
2
3
4      ! Assemble structure stiffness matrix
5      ELSE IF (IND.EQ. 2) THEN
6
7          DO N=1, NUME
8              MTYPE=MATP(N)
9
10             XL2=0.
11             DO L=1, 3
12                 D(L)=XYZ(L,N) - XYZ(L+3,N)
13                 XL2=XL2 + D(L)*D(L)
14             END DO
15             XL=SQRT(XL2)      ! Length of element N
16
17             XX=E(MTYPE)*AREA(MTYPE)*XL      ! E*A*1
18
19             DO L=1, 3
20                 ST(L)=D(L)/XL2
21                 ST(L+3)=-ST(L)
22             END DO
23
24             DO J=1, ND
25                 YY=ST(J)*XX
26                 DO I=1, J
27                     S(I,J)=ST(I)*YY
28                 END DO
29             END DO
30
31             CALL ADDBAN (DA(NP(3)), IA(NP(2)), S, LM(1,N), ND)
32
33             END DO
34
35             RETURN
36
37             ! Stress calculations
38             ELSE IF (IND.EQ. 3) THEN
39
40                 IPRINT=0
41                 DO N=1, NUME
42                     IPRINT=IPRINT + 1
43                     IF (IPRINT.GT.50) IPRINT=1
44                     IF (IPRINT.EQ.1) WRITE (IOUT, "(//, 'STRESS CALCULATIONS FOR ', &
45                                     'ELEMENT GROUP', I4, '//, &
46                                     'ELEMENT', 13X, 'FORCE', 12X, 'STRESS', '/', 'NUMBER')")
47
48                     NG
49
50                     MTYPE=MATP(N)
51
52                     XL2=0.
53                     DO L=1, 3
54                         D(L) = XYZ(L,N) - XYZ(L+3,N)
55                         XL2=XL2 + D(L)*D(L)
56                     END DO
57
58                     DO L=1, 3
59                         ST(L)=(D(L)/XL2)*E(MTYPE)
60                         ST(L+3)=-ST(L)
61                     END DO
62
63                     STR=0.0
64                     DO L=1, 3
65                         I=LM(L,N)
66                         IF (I.GT.0) STR=STR + ST(L)*U(I)
67
68                         J=LM(L+3,N)
69                         IF (J.GT.0) STR=STR + ST(L+3)*U(J)
70                     END DO
71
72                     P=STR*AREA(MTYPE)
73
74                     WRITE (IOUT, "(1X, I5, 11X, E13.6, 4X, E13.6)") N, P, STR
75                 END DO

```

$$K^e = A^e E^e l^e S^e S^e$$

$$S^e = \frac{1}{(l^e)^2} \begin{bmatrix} -x_1^e & -y_1^e & -z_1^e & x_1^e & y_1^e & z_1^e \end{bmatrix}$$

$$D \rightarrow D$$

$$S^T \leftrightarrow S^e$$

$$L = \text{全部}$$

$$A \text{ MAXA}$$

$$\sigma^e = \frac{E^e}{(l^e)^2} \begin{bmatrix} -x_1^e & -y_1^e & -z_1^e & x_1^e & y_1^e & z_1^e \end{bmatrix} d^e$$

$$P = \text{力}$$

```
truss.f90
1      ELSE
2          STOP "*** ERROR *** Invalid IND value."
3      END IF
4
5  END SUBROUTINE RUSS
```

```

1  ! . . . . .
2  ! .
3  ! .           S T A P 9 0
4  ! .
5  ! .   AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90
6  ! .   Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose
7  ! .
8  ! .   Xiong Zhang, (2013)
9  ! .   Computational Dynamics Group, School of Aerospace
10 ! .   Tsinghua Univerity
11 ! .
12 ! . . . . .
13
14 SUBROUTINE COLHT (MHT,ND,LM)
15 ! . . . . .
16 ! .
17 ! .   To calculate column heights
18 ! .
19 ! . . . . .
20
21 USE GLOBALS, ONLY : NEQ
22 IMPLICIT NONE
23 INTEGER :: ND, LM(ND),MHT(NEQ)
24 INTEGER :: I, LS, II, ME
25
26 LS=HUGE(1)  ! The largest integer number
27
28 DO I=1,ND
29   IF (LM(I) .NE. 0) THEN
30     IF (LM(I)-LS .LT. 0) LS=LM(I)
31   END IF
32 END DO
33
34 DO I=1,ND
35   II=LM(I)
36   IF (II.NE.0) THEN
37     ME=II - LS
38     IF (ME.GT.MHT(II)) MHT(II)=ME
39   END IF
40 END DO
41
42 RETURN
43 END SUBROUTINE COLHT
44
45
46 SUBROUTINE ADDRES (MAXA,MHT)
47 ! . . . . .
48 ! .
49 ! .   To calculate addresses of diagonal elements in banded
50 ! .   matrix whose column heights are known
51 ! .
52 ! .   MHT = Active column heights
53 ! .   MAXA = Addresses of diagonal elements
54 ! .
55 ! . . . . .
56
57 USE GLOBALS, ONLY : NEQ, MK, NWK
58
59 IMPLICIT NONE
60 INTEGER :: MAXA(NEQ+1),MHT(NEQ)
61 INTEGER :: NN, I
62
63 ! Clear array maxa
64
65 NN=NEQ + 1
66 DO I=1,NN
67   MAXA(I)=0.0
68 END DO
69
70 MAXA(1)=1
71 MAXA(2)=2
72 MK=0
73 IF (NEQ.GT.1) THEN
74   DO I=2,NEQ

```

```

1      IF (MHT(I).GT.MK) MK=MHT(I)
2      MAXA(I+1)=MAXA(I) + MHT(I) + 1
3      END DO
4      END IF
5      MK=MK + 1
6      NWK=MAXA(NEQ+1) - MAXA(1)
7
8      RETURN
9  END SUBROUTINE ADDRES
10
11
12  SUBROUTINE ASSEM (AA)
13  ! . . . . .
14  ! .
15  ! . To call element subroutines for assemblage of the .
16  ! . structure stiffness matrix .
17  ! .
18  ! . . . . .
19
20  USE GLOBALS, ONLY : IELMNT, NUMEG, NUMEST, NPAR
21
22  IMPLICIT NONE
23  REAL :: AA(*)
24  INTEGER :: N, I
25
26  REWIND IELMNT
27  DO N=1, NUMEG
28      READ (IELMNT) NUMEST, NPAR, (AA(I), I=1, NUMEST)
29      CALL ELEMNT
30  END DO
31
32  RETURN
33  END SUBROUTINE ASSEM
34
35
36  SUBROUTINE ADDBAN (A, MAXA, S, LM, ND)
37  ! . . . . .
38  ! .
39  ! . To assemble element stiffness into compacted global stiffness .
40  ! .
41  ! . A = GLOBAL STIFFNESS (1D skyline storage) .
42  ! . S = ELEMENT STIFFNESS .
43  ! . ND = DEGREES OF FREEDOM IN ELEMENT STIFFNESS .
44  ! .
45  ! . . . . .
46  USE GLOBALS, ONLY : NWK, NEQ
47  IMPLICIT NONE
48  REAL (8) :: A(NWK), S(ND, ND)
49  INTEGER :: MAXA(NEQ+1), LM(ND)
50  INTEGER :: I, ND, II, MJ, J, JJ, IJ, KK
51
52  KK=0
53  DO J=1, ND
54      JJ=LM(J)
55      IF (JJ.GT. 0) THEN
56          MJ=MAXA(JJ)
57          DO I=1, J
58              II=LM(I)
59              IF (II.GT. 0) THEN
60                  IJ=JJ-II
61                  IF (IJ.GE. 0) THEN
62                      KK=MJ + IJ
63                      A(KK)=A(KK) + S(I, J)
64                  END IF
65              END IF
66          END DO
67      END IF
68  END DO
69
70  RETURN
71  END SUBROUTINE ADDBAN
72
73
74  SUBROUTINE COLSOL (A, V, MAXA, NN, NWK, NNM, KKK)

```

```

DO J=1, ND
  JJ=LM(J)
  IF (JJ.GT. 0) THEN
    DO I=1, J
      II=LM(I)
      IF (II.GT. 0) THEN
        IF (JJ.GE. II) THEN
          KK= MAXA(JJ) + JJ - II
        ELSE
          KK= MAXA(II) + II - JJ
        END IF
        A(KK)=A(KK) + S(I, J)
      END IF
    END DO
  END IF
END DO

```



```

1  ! .....
2  !
3  !   To solve finite element static equilibrium equations in
4  !   core, using compacted storage and column reduction scheme
5  !
6  !   -- Input variables --
7  !   A(NWK)   = Stiffness matrix stored in compacted form
8  !   V(NN)    = Right-hand-side load vector
9  !   MAXA(NNM) = Vector containing addresses of diagonal
10 !               elements of stiffness matrix in a
11 !   NN        = Number of equations
12 !   NWK       = Number of elements below skyline of matrix
13 !   NNM       = NN + 1
14 !   KKK       = Input flag
15 !   EQ. 1     = Triangularization of stiffness matrix
16 !   EQ. 2     = Reduction and back-substitution of load vector
17 !   IOUT      = UNIT used for output
18 !
19 !   -- OUTPUT --
20 !   A(NWK)    = D and L - Factors of stiffness matrix
21 !   V(NN)     = Displacement vector
22 !
23 ! .....
24
25   USE GLOBALS, ONLY : IOUT
26
27   IMPLICIT NONE
28   INTEGER :: MAXA(NNM), NN, NWK, NNM, KKK
29   REAL(8) :: A(NWK), V(NN), C, B
30   INTEGER :: N, K, KN, KL, KU, KH, IC, KLT, KI, J, ND, KK, L
31   INTEGER :: MINO
32
33   ! Perform L*D*L(T) factorization of stiffness matrix
34
35   IF (KKK == 1) THEN
36     DO N=1, NN
37       KN=MAXA(N)
38       KL=KN + 1
39       KU=MAXA(N+1) - 1
40       KH=KU - KL
41       KH=KH - 1
42
43       IF (KH > 0) THEN
44         K=N - KH
45         IC=0
46         KLT=KU
47         DO J=1, KH
48           IC=IC + 1
49           KLT=KLT - 1
50           KI=MAXA(K)
51           ND=MAXA(K+1) - KI - 1
52           IF (ND .GT. 0) THEN
53             KK=MINO(IC, ND)
54             C=0.
55             DO L=1, KK
56               C=C + A(KI+L)*A(KLT+L)
57             END DO
58             A(KLT)=A(KLT) - C
59           END IF
60           K=K + 1
61         END DO
62       ELSE IF (KH == 0) THEN
63         K=N
64         B=0.
65         DO KK=KL, KU
66           K=K - 1
67           KI=MAXA(K)
68           C=A(KK)/A(KI)
69           B=B + C*A(KK)
70           A(KK)=C
71         END DO
72         A(KN)=A(KN) - B
73       ELSE IF (KH < 0) THEN
74         IF (A(KN) .LE. 0) THEN

```

$$K = L U$$

$$K_{ij} = \sum_{r=1}^{i-1} L_{ir} U_{rj} + U_{ij}$$

$$U_{ij} = K_{ij} - \sum_{r=0}^{i-1} L_{ir} U_{rj}$$

max(m_i, m_j)

⑧ DO N=1, NN
 KN=MAXA(N) 第N列最后一个非对角元(K_{N-1,N})的地址
 KL=KN + 1
 KU=MAXA(N+1) - 1 第N列第一个非零元(K_{m_N,N})的地址
 KH=KU - KL ← N - m_N - 1

① IF (KH > 0) THEN
 K=N - KH
 IC=0
 KLT=KU
 DO J=1, KH
 IC=IC + 1
 KLT=KLT - 1 ← K_j的地址, i = m_j + 1
 KI=MAXA(K) ← 第j列第一个非零元的地址
 ND=MAXA(K+1) - KI - 1 第j列非零元个数(非零元个数) i - m_i
 IF (ND .GT. 0) THEN
 KK=MINO(IC, ND) ← i - m_i
 C=0. ← i - m_j
 DO L=1, KK
 C=C + A(KI+L)*A(KLT+L)
 END DO
 A(KLT)=A(KLT) - C
 END IF
 K=K + 1 ← i = m_j + 1 : j - 1
 END DO
 ELSE IF (KH == 0) THEN 第N列只有对角元一个非零元。
 K=N
 B=0.
 DO KK=KL, KU
 K=K - 1
 KI=MAXA(K) ← j - 1 非零元的地址
 C=A(KK)/A(KI) ← L_{ri} = U_{ri} / d_{rr}
 B=B + C*A(KK)
 A(KK)=C
 END DO
 A(KN)=A(KN) - B
 ELSE IF (KH < 0) THEN 第N列只有对角元一个非零元。
 IF (A(KN) .LE. 0) THEN

② m_j + 1
 IC = J
 i = j - m_j - 1
 = i - m_j
 U<sub>ij} = K<sub>ij} - \sum_{r=1}^{i-1} L_{ir} U_{rj}
 i - m_m 22. 27
 min(i - m_i, i - m_j)
 i = m_j + 1 : j - 1
 m_m = max(m_i, m_j)
 L<sub>ij} = U<sub>ij} / d_{ii}
 d_{ij} = K_{ij} - \sum_{r=m_j}^{i-1} L_{ir} K_{rj}}}</sub></sub></sub></sub>

```

1                                assem.f90
2                                WRITE (IOUT,"(//' STOP - STIFFNESS MATRIX NOT POSITIVE DEFINITE',//, &
3                                N, A (KN)                                ' , NONPOSITIVE PIVOT FOR EQUATION ',I8, '//,' PIVOT = ',E20.12 )")
4                                STOP
5                                END IF
6                                END IF
7                                END DO
8
9                                ELSE IF (KKK == 2) THEN
10
11                                ! REDUCE RIGHT-HAND-SIDE LOAD VECTOR
12
13                                DO N=1, NN
14                                    KL=MAXA(N) + 1
15                                    KU=MAXA(N+1) - 1
16                                    IF (KU-KL .GE. 0) THEN
17                                        K=N
18                                        C=0.
19                                        DO KK=KL, KU
20                                            K=K - 1
21                                            C=C + A (KK)*V (K)
22                                        END DO
23                                        V (N)=V (N) - C
24                                    END IF
25                                END DO
26
27                                ! BACK-SUBSTITUTE
28
29                                DO N=1, NN
30                                    K=MAXA (N)
31                                    V (N)=V (N) /A (K)
32                                END DO
33
34                                IF (NN.EQ. 1) RETURN
35
36                                N=NN
37                                DO L=2, NN
38                                    KL=MAXA(N) + 1
39                                    KU=MAXA(N+1) - 1
40                                    IF (KU-KL .GE. 0) THEN
41                                        K=N
42                                        DO KK=KL, KU
43                                            K=K - 1
44                                            V (K)=V (K) - A (KK)*V (N)
45                                        END DO
46                                    END IF
47                                    N=N - 1
48                                END DO
49
50                                END IF
51
52                                END SUBROUTINE COLSOL

```

```

1  ! -----
2  ! -
3  ! - MEMALLOCATE : A storage manage package for finite element code
4  ! -
5  ! -   Xiong Zhang, (2013)
6  ! -   Computational Dynamics Group, School of Aerospace
7  ! -   Tsinghua Univerity
8  ! -
9  ! - List of subroutine
10 ! -
11 ! -   memalloca - allocate an array in the shared storage
12 ! -   memfree   - deallocate the specified array
13 ! -   memfreefrom - deallocate all arrays from the specified array
14 ! -   memfreefromto - deallocate all arrays between the specified arrays
15 ! -   memprint   - print the contents of the specified array
16 ! -   memprintptr - print a subset of the storage in given format
17 ! -   meminfo    - list all allocated arrays
18 ! -
19 ! -----
20
21
22 module memAllocate
23
24   integer, parameter :: MTOT = 10000 ! Speed storage available for execution
25   integer, parameter :: ITWO = 2     ! Double precision indicator
26                                     ! 1 - Single precision arithmetic
27                                     ! 2 - Double precision arithmetic
28
29   real(4) :: A(MTOT)
30   real(8) :: DA(MTOT/ITWO)
31   integer :: IA(MTOT)
32
33   equivalence (A, IA), (A, DA) ! A, DA, and IA share the same storage units
34
35   integer, parameter :: amax = 200 ! Maximum number of arrays allowed
36
37   integer :: np(amax) = 0 ! Pointer to each array
38   integer :: alen(amax) = 0 ! Length of each array
39   integer :: aprec(amax) = 0 ! Precision of each array
40   character*8 :: aname(amax) = ""
41
42   integer :: nplast = 0 ! Pointer to the last allocated element in A
43                       ! nplast is in the unit of single precision
44
45 contains
46
47   subroutine memalloc(num, name, len, prec)
48   ! -----
49   ! - Purpose
50   ! -   Allocate an array in the storage of A
51   ! -
52   ! - Input
53   ! -   num - Number of the array allocated
54   ! -   name - Name of the array
55   ! -   len - Length of the array (total number of elements of the array)
56   ! -   prec - Precision of the array
57   ! -           1: Single precision
58   ! -           2: Double precesion
59   ! -----
60
61   implicit none
62   integer :: num, len, prec
63   character*5 name
64
65   if (num < 1 .or. num > amax) then
66     write(*, '("*** Error *** Invalid array number: ", I3)') num
67     stop
68   end if
69
70   if (prec < 1 .or. prec > 2) then
71     write(*, '("*** Error *** Invalid array type: ", I3)') prec
72     stop
73   end if
74
75   if (np(num) > 0) call memfree(num) ! array num exists

```

```

1
2   if (nplast+len*prec > MTOT) then
3       write(*, '("*** Error *** No adequate storage available in A",/, &
4           "      Required :", I10, /, &
5           "      Available :", I10)') len*prec, MTOT - nplast
6       stop
7   end if
8
9   np(num) = nplast/prec + 1 ! In the unit of allocated array
10  aname(num) = name
11  alen(num) = len
12  aprec(num) = prec
13
14  nplast = nplast + len*prec
15  nplast = ceiling(nplast/2.0)*2 ! Make nplast an even number
16
17  end subroutine memalloc
18
19
20  subroutine memfree(num)
21  ! -----
22  ! - Purpose -
23  ! - Free the array num and compact the storage if necessary -
24  ! -
25  ! - Input -
26  ! - num - Number of the array to be deallocated -
27  ! -
28  ! -----
29  implicit none
30  integer :: i, num, npbase, nplen
31
32  if (np(num) <= 0) return ! The array has not been allocated
33
34  ! Base address of the array num in the single precision unit
35  npbase = (np(num)-1)*aprec(num)
36
37  ! Length of the array num in the single precision unit
38  nplen = ceiling(alen(num)*aprec(num)/2.0)*2 ! Make nplen an even number
39
40  ! Compact the storage if necessary
41  if (npbase+nplen < nplast) then
42  ! Move arrays behind the array num forward to reuse its storage
43      do i = npbase+nplen+1, nplast
44          A(i-nplen) = A(i)
45      end do
46
47  ! Update the pointer of arrays behind the array num
48      do i = 1, amax
49          if ((np(i)-1)*aprec(i) > npbase) np(i) = np(i) - nplen/aprec(i)
50      end do
51  end if
52
53  np(num) = 0
54  aname(num) = ""
55  alen(num) = 0
56  aprec(num) = 0
57
58  nplast = nplast - nplen
59  end subroutine memfree
60
61
62  subroutine memfreefrom(num)
63  ! -----
64  ! - Purpose -
65  ! - Free all arrays from num to the end -
66  ! -
67  ! - Input -
68  ! - num - Number of the array to be deallocated from -
69  ! -
70  ! -----
71  implicit none
72  integer :: i, num
73
74  do i=amax, num, -1

```

```

1      call memfree(i)
2      end do
3
4  end subroutine memfreefrom
5
6
7  subroutine memfreefromto(n1,n2)
8  ! -----
9  ! - Purpose -
10 ! -   Free all arrays from n1 to n2 -
11 ! - -
12 ! - Input -
13 ! -   n1 - Number of the array to be deallocated from -
14 ! -   n2 - Number of the array to be deallocated to -
15 ! - -
16 ! -----
17      implicit none
18      integer :: i, n1, n2
19
20      do i=n2,n1,-1
21          call memfree(i)
22      end do
23
24  end subroutine memfreefromto
25
26
27  subroutine memprint(num)
28  ! -----
29  ! - Purpose -
30  ! -   Print the contents of the array num -
31  ! - -
32  ! - Input -
33  ! -   num - Number of the array to be printed -
34  ! - -
35  ! -----
36      implicit none
37      integer :: num, i
38
39      if (np(num) <= 0) then
40          write(*, '("*** Error *** Array ", I3, " has not been allocated.")') num
41          return
42      end if
43
44      write(*, '("Contents of Array ", A5, ":")') aname(num)
45      if (aprec(num) == 1) then
46          write(*, '(8I10)') (IA(i), i=np(num), np(num)+alen(num)-1)
47      else
48          write(*, '(8E10.2)') (DA(i), i=np(num), np(num)+alen(num)-1)
49      end if
50
51  end subroutine memprint
52
53
54  subroutine memprintptr(ptr, len, atype)
55  ! -----
56  ! - Purpose -
57  ! -   Print the contents of the stroage starting from ptr -
58  ! - -
59  ! - Input -
60  ! -   ptr - Pointer to the first entry (in single precision unit) -
61  ! -   len - Total number of entries to be printed -
62  ! -   atype - Type of the entries (0 - integer; 1 - float; 2 - double) -
63  ! - -
64  ! -----
65      implicit none
66      integer :: i, ptr, len, atype
67      character*8 dtype(3)
68      data dtype/"integer","real","double"/
69
70      write(*, '("Contents of storage starting from ", I5, " in ", A8, ":")') ptr, dtype(atype+1)
71      if (atype == 0) then
72          write(*, '(8I10)') (IA(i), i=ptr, ptr+len-1)
73      else if (atype == 1) then
74          write(*, '(8E10.2)') (A(i), i=ptr, ptr+len-1)

```

```

memalloc.f90
1      else if (atype == 2) then
2          write(*,'(8E10.2)') (DA(i), i=(ptr-1)/ITW0+1, (ptr-1)/ITW0+len)
3      end if
4
5  end subroutine memprintptr
6
7
8  subroutine meminfo
9  ! -----
10 ! - Purpose -
11 ! -   Print the information of the storage -
12 ! - -
13 ! -----
14      implicit none
15      integer :: i
16
17      write(*,'("List of all arrays:")')
18      write(*,'("  Number   Name   Length   Pointer   Precision")')
19      do i=1,amax
20          if (np(i) == 0) cycle
21          write(*,'(I7, 4X, A5, I9, I10, I12)') i, aname(i), alen(i), np(i), aprec(i)
22      end do
23  end subroutine meminfo
24
25 end module memAllocate

```

ADDBAN, 16
ADDRES, 15
ASSEM, 16
CLOSEFILES, 6
COLHT, 15
COLSOL, 16
ELCAL, 9
ELEMNT, 9
GLOBALS, 1
INPUT, 7
LOADS, 8
LOADV, 8
memalloc, 19
memAllocate, 19
memfree, 20
memfreefrom, 20
memfreefromto, 21
meminfo, 22
memprint, 21
memprintptr, 21
OPENFILES, 5
RUSS, 11
SECOND, 5
STAP90, 2
STRESS, 10
TRUSS, 11
WRITED, 5