

```

                                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 :: MIDEST
49     integer :: MAXEST
50
51     integer :: NG
52
53 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()
26
27     MAXEST=0
28
29 ! * * * * *
30 ! *               INPUT PHASE *
31 ! * * * * *
32
33     WRITE(*, "Input phase ... ")
34
35     CALL SECOND (TIM(1))
36
37 ! Read control information
38
39 !   HED   - The master heading informaiton for use in labeling the output
40 !   NUMNP - Total number of nodal points
41 !         0 : program stop
42 !   NUMEG - Total number of element group (>0)
43 !   NLCASE - Number of load case (>0)
44 !   MODEX - Solution mode
45 !         0 : data check only;
46 !         1 : execution
47
48     READ (IIN, ' (A80,/,4I5)') HED, NUMNP, NUMEG, NLCASE, MODEX
49
50     IF (NUMNP.EQ.0) STOP    ! Data check mode
51
52     WRITE (IOUT, "(/, ' ', A80, //, &
53     ' C O N T R O L   I N F O R M A T I O N', //, &
54     '      NUMBER OF NODAL POINTS', 10(' . '), ' (NUMNP) = ', I5, /, &
55     '      NUMBER OF ELEMENT GROUPS', 9(' . '), ' (NUMEG) = ', I5, /, &
56     '      NUMBER OF LOAD CASES', 11(' . '), ' (NLCASE) = ', I5, /, &
57     '      SOLUTION MODE ', 14(' . '), ' (MODEX) = ', I5, /, &
58     '      EQ. 0, DATA CHECK', /, &
59     '      EQ. 1, EXECUTION')") HED, NUMNP, NUMEG, NLCASE, MODEX
60
61 ! Read nodal point data
62
63 ! ALLOCATE STORAGE
64 !   ID(3, NUMNP) : Boundary condition codes (0=free, 1=deleted)
65 !   X(NUMNP)     : X coordinates
66 !   Y(NUMNP)     : Y coordinates
67 !   Z(NUMNP)     : Z coordinates
68
69     CALL MEMALLOC(1, "ID", 3*NUMNP, 1)
70     CALL MEMALLOC(2, "X", NUMNP, ITWO)
71     CALL MEMALLOC(3, "Y", NUMNP, ITWO)
72     CALL MEMALLOC(4, "Z", NUMNP, ITWO)
73
74     CALL INPUT (IA(NP(1)), DA(NP(2)), DA(NP(3)), DA(NP(4)), NUMNP, NEQ)

```

```

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

```

```

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

```

```

1          ,      stap.f90
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
6  STOP
7
8  END PROGRAM STAP90
9
10 SUBROUTINE SECOND (TIM)
11 ! USE DPORT ! 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 SUBROUTINE WRITED (DISP, ID, NEQ, NUMNP)
24 ! . . . . .
25 ! . . . . .
26 ! . To print displacements . . . . .
27 ! . . . . .
28
29 USE GLOBALS, ONLY : IOUT
30
31 IMPLICIT NONE
32 INTEGER :: NEQ, NUMNP, ID(3, NUMNP)
33 REAL(8) :: DISP(NEQ), D(3)
34 INTEGER :: IC, II, I, KK, IL
35
36 ! Print displacements
37
38 WRITE (IOUT, "(//, ' D I S P L A C E M E N T S', //, ' NODE ', 10X,  &
39           'X-DISPLACEMENT   Y-DISPLACEMENT   Z-DISPLACEMENT' )")
40
41 IC=4
42
43 DO II=1, NUMNP
44   IC=IC + 1
45   IF (IC.GE. 56) THEN
46     WRITE (IOUT, "(//, ' D I S P L A C E M E N T S', //, ' NODE ', 10X,  &
47           'X-DISPLACEMENT   Y-DISPLACEMENT   Z-DISPLACEMENT' )")
48     IC=4
49   END IF
50
51   DO I=1, 3
52     D(I)=0.
53   END DO
54
55   DO I=1, 3
56     KK=ID(I, II)
57     IF (KK.NE. 0) D(I)=DISP(KK)
58   END DO
59
60   WRITE (IOUT, ' (1X, I3, 8X, 3E18.6)' ) II, D
61
62 END DO
63
64 RETURN
65
66 END SUBROUTINE WRITED
67
68
69 SUBROUTINE OPENFILES()
70 ! . . . . .
71 ! . . . . .
72 ! . Open input data file, results output file and temporary files .
73 ! . . . . .
74

```

```

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

```

```

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, MIDEEST, 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) MIDEEST, NPAR, (AA(I), I=1, MIDEEST)
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 Univerity .
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
58 RETURN
59
60 END SUBROUTINE TRUSS
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)

```

```

                                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         15X,' 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
2      RETURN
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             NG
48             MTYPE=MATP(N)
49
50             XL2=0.
51             DO L=1, 3
52                 D(L) = XYZ(L,N) - XYZ(L+3,N)
53                 XL2=XL2 + D(L)*D(L)
54             END DO
55
56             DO L=1, 3
57                 ST(L)=(D(L)/XL2)*E(MTYPE)
58                 ST(L+3)=-ST(L)
59             END DO
60
61             STR=0.0
62             DO L=1, 3
63                 I=LM(L,N)
64                 IF (I.GT. 0) STR=STR + ST(L)*U(I)
65
66                 J=LM(L+3,N)
67                 IF (J.GT. 0) STR=STR + ST(L+3)*U(J)
68             END DO
69
70             P=STR*AREA(MTYPE)
71
72             WRITE (IOUT, "(1X, I5, 11X, E13.6, 4X, E13.6)") N, P, STR
73         END DO
74

```

```
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, MIDEST, NPAR
21
22  IMPLICIT NONE
23  REAL :: AA(*)
24  INTEGER :: N, I
25
26  REWIND IELMNT
27  DO N=1,NUMEG
28      READ (IELMNT) MIDEST, NPAR, (AA(I), I=1, MIDEST)
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 :: ND, I, J, II, JJ, KK
51
52  DO J=1,ND
53      JJ=LM(J)
54      IF (JJ.GT. 0) THEN
55          DO I=1,J
56              II=LM(I)
57              IF (II.GT. 0) THEN
58                  IF (JJ.GE. II) THEN
59                      KK= MAXA(JJ) + JJ - II
60                  ELSE
61                      KK= MAXA(II) + II - JJ
62                  END IF
63                  A(KK)=A(KK) + S(I, J)
64              END IF
65          END DO
66      END IF
67  END DO
68
69  RETURN
70  END SUBROUTINE ADDBAN
71
72
73  SUBROUTINE COLSOL (A, V, MAXA, NN, NWK, NNM, KKK)
74  ! . . . . .

```



assem. f90

```

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

```

```

1
2      IF (A(KN) .LE. 0) THEN
3          WRITE (IOUT, "(//' STOP - STIFFNESS MATRIX NOT POSITIVE DEFINITE', //, &
4              ' , NONPOSITIVE PIVOT FOR EQUATION ', I8, '//, ' PIVOT = ', E20.12 )")
5      N, A(KN)
6          STOP
7      END IF
8  END DO
9
10     ELSE IF (KKK == 2) THEN
11
12     ! REDUCE RIGHT-HAND-SIDE LOAD VECTOR
13
14         DO N=1, NN
15             KL=MAXA(N) + 1
16             KU=MAXA(N+1) - 1
17             IF (KU-KL .GE. 0) THEN
18                 K=N
19                 C=0.
20                 DO KK=KL, KU
21                     K=K - 1
22                     C=C + A(KK)*V(K)
23                 END DO
24                 V(N)=V(N) - C
25             END IF
26         END DO
27
28     ! BACK-SUBSTITUTE
29
30         DO N=1, NN
31             K=MAXA(N)
32             V(N)=V(N)/A(K)
33         END DO
34
35         IF (NN.EQ.1) RETURN
36
37         N=NN
38         DO L=2, NN
39             KL=MAXA(N) + 1
40             KU=MAXA(N+1) - 1
41             IF (KU-KL .GE. 0) THEN
42                 K=N
43                 DO KK=KL, KU
44                     K=K - 1
45                     V(K)=V(K) - A(KK)*V(N)
46                 END DO
47             END IF
48             N=N - 1
49         END DO
50
51     END IF
52
53 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   real(4) :: A(MTOT)
29   real(8) :: DA(MTOT/ITWO)
30   integer :: IA(MTOT)
31
32   equivalence (A, IA), (A, DA) ! A, DA, and IA share the same storage units
33
34   integer, parameter :: amax = 200 ! Maximum number of arrays allowed
35
36   integer :: np(amax) = 0 ! Pointer to each array
37   integer :: alen(amax) = 0 ! Length of each array
38   integer :: aprec(amax) = 0 ! Precision of each array
39   character*8 :: aname(amax) = ""
40
41   integer :: nplast = 0 ! Pointer to the last allocated element in A
42                       ! nplast is in the unit of single precision
43
44 contains
45
46   subroutine memalloc(num, name, len, prec)
47   ! -----
48   ! - Purpose -
49   ! -   Allocate an array in the storage of A -
50   ! -
51   ! - Input -
52   ! -   num - Number of the array allocated -
53   ! -   name - Name of the array -
54   ! -   len - Length of the array (total number of elements of the array) -
55   ! -   prec - Precision of the array -
56   ! -       1: Single precision -
57   ! -       2 : Double precesion -
58   ! -
59   ! -----
60   implicit none
61   integer :: num, len, prec
62   character*5 name
63
64   integer :: i, npfirst
65
66   if (num < 1 .or. num > amax) then
67     write(*, '("*** Error *** Invalid array number: ", I3)') num
68     stop
69   end if
70
71   if (prec < 1 .or. prec > 2) then
72     write(*, '("*** Error *** Invalid array type: ", I3)') prec
73     stop
74   end if

```

```

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

```

```

                                memalloc.f90
1  ! -      num - Number of the array to be deallocated from      -
2  ! -      -      -      -      -      -      -      -      -      -
3  ! -----
4      implicit none
5      integer :: i, num
6
7      do i=amax,num,-1
8          call memfree(i)
9      end do
10
11  end subroutine memfreefrom
12
13
14  subroutine memfreefromto(n1,n2)
15  ! -----
16  ! - Purpose      -
17  ! -   Free all arrays from n1 to n2      -
18  ! -      -      -
19  ! - Input      -
20  ! -   n1 - Number of the array to be deallocated from      -
21  ! -   n2 - Number of the array to be deallocated to      -
22  ! -      -      -
23  ! -----
24      implicit none
25      integer :: i, n1, n2
26
27      do i=n2,n1,-1
28          call memfree(i)
29      end do
30
31  end subroutine memfreefromto
32
33
34  subroutine memprint(num)
35  ! -----
36  ! - Purpose      -
37  ! -   Print the contents of the array num      -
38  ! -      -      -
39  ! - Input      -
40  ! -   num - Number of the array to be printed      -
41  ! -      -      -
42  ! -----
43      implicit none
44      integer :: num,i
45
46      if (np(num) <= 0) then
47          write(*,'(*** Error *** Array ", I3, " has not been allocated.")') num
48          return
49      end if
50
51      write(*,'(Contents of Array ", A5, ":")') aname(num)
52      if (aprec(num) == 1) then
53          write(*,'(8I10)') (IA(i), i=np(num),np(num)+alen(num)-1)
54      else
55          write(*,'(8E10.2)') (DA(i), i=np(num),np(num)+alen(num)-1)
56      end if
57
58  end subroutine memprint
59
60
61  subroutine memprintptr(ptr, len, atype)
62  ! -----
63  ! - Purpose      -
64  ! -   Print the contents of the stroage starting from ptr      -
65  ! -      -      -
66  ! - Input      -
67  ! -   ptr - Pointer to the first entry (in single precision unit)      -
68  ! -   len - Total number of entries to be printed      -
69  ! -   atype - Type of the entries (0 - integer; 1 - float; 2 - double)      -
70  ! -      -      -
71  ! -----
72      implicit none
73      integer :: i, ptr, len, atype
74      character*8 dtype(3)

```

```

1      memalloc.f90
2      data dtype/"integer","real","double"/
3      write(*,('Contents of storage starting from ', I5, ' in ', A8, ":",')) ptr, dtype(atype+1)
4      if (atype == 0) then
5          write(*,('8I10')) (IA(i), i=ptr,ptr+len-1)
6      else if (atype == 1) then
7          write(*,('8E10.2')) (A(i), i=ptr,ptr+len-1)
8      else if (atype == 2) then
9          write(*,('8E10.2')) (DA(i), i=(ptr-1)/ITWO+1, (ptr-1)/ITWO+len)
10     end if
11
12 end subroutine memprintptr
13
14
15 subroutine meminfo
16 ! -----
17 ! - Purpose -
18 ! - Print the information of the storage -
19 ! - -
20 ! -----
21     implicit none
22     integer :: i
23
24     write(*,('List of all arrays:'))
25     write(*,('  Number   Name   Length   Pointer   Precision'))
26     do i=1,amax
27         if (np(i) == 0) cycle
28         write(*,('I7, 4X, A5, I9, I10, I12')) i, aname(i), alen(i), np(i), aprec(i)
29     end do
30 end subroutine meminfo
31
32 end module memAllocate

```

函数索引

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