第 1 回演習プログラム

新領域創成科学研究科　人間環境学専攻
橋本　学

1．main_ex1.f90

PROGRAM 文，WRITE 文を使用したプログラムです.

```
1:      PROGRAM main_ex1
2:
3:      IMPLICIT NONE
4:
5:      WRITE(6, *) 'Hello, world.'
6:
7:      STOP
8:
9:      END PROGRAM main_ex1
```

２．main_ex2.f90

PROGRAM 文，READ 文，WRITE 文，INTEGER 型変数，INTEGER 型配列変数を使用したプログラムです.

※ INTEGER 型では，4 Byte (32 bit) の記憶域を使用して，最大 19 桁の整数値を定めます.
2,147,483,648～2,147,483,647 の範囲の符号付き整数値を表現できます.

```
1:      PROGRAM main_ex2
2:
3:      IMPLICIT NONE
4:
5:      INTEGER :: a
6:      INTEGER :: aa(2)
7:      INTEGER :: aaa(4)
8:
9:      WRITE(6, *) 'a?'
10:     READ(5, *) a
11:     WRITE(6, *)
12:
13:     WRITE(6, *) 'aa(1) and aa(2)?'
14:     READ(5, *) aa(1), aa(2)
15:     WRITE(6, *)
16:
17:     WRITE(6, *) 'a =', a
18:     WRITE(6, '(A, I6)') 'a = ', a
19:     WRITE(6, '(A, I8)') 'a = ', a
20:     WRITE(6, '(A, 1X, I8)') 'a = ', a
21:     WRITE(6, '(A, 2X, I8)') 'a = ', a
22:     WRITE(6, *)
23:
24:     WRITE(6, *) 'aa(1) = ', aa(1), ', ', 'aa(2) = ', aa(2)
25:     WRITE(6, '( (A, I6, 2A, I6) )') 'aa(1) = ', aa(1), ', ', 'aa(2) = ', aa(2)
26:     WRITE(6, *)
27:
28:     aaa(1) = aa(1)+aa(2)
```

```
29:       aaa(2) = aa(1)-aa(2)
30:       aaa(3) = aa(1)*aa(2)
31:       aaa(4) = aa(1)/aa(2)
32:
33:       WRITE(6, '( 4(A, I8, A) )') 'aaa(1) = ', aaa(1), ', ', 'aaa(2) = ', aaa(2), ', ', &
34:                                   'aaa(3) = ', aaa(3), ', ', 'aaa(4) = ', aaa(4)
35:       WRITE(6, *)
36:
37:       aaa(1) = a*( aa(1)+aa(2) )
38:       aaa(2) = a*aa(1)+aa(2)
39:       aaa(3) = a*( aa(1)*aa(2) )
40:       aaa(4) = a*aa(1)*aa(2)
41:
42:       WRITE(6, '( 4(A, I8, A) )') 'aaa(1) = ', aaa(1), ', ', 'aaa(2) = ', aaa(2), ', ', &
43:                                   'aaa(3) = ', aaa(3), ', ', 'aaa(4) = ', aaa(4)
44:       WRITE(6, *)
45:
46:       STOP
47:
48:       END PROGRAM main_ex2
```

３．main_ex3.f90

PROGRAM 文，READ 文，WRITE 文，REAL(8)型変数，REAL(8)型配列変数を使用したプログラムです.

※ REAL(8)型では，8 Byte (64 bit) の記憶域を使用して，有効数字 16 桁の実数値を定めます．絶対値が 2.2250738585072013D-308～1.7976931348623158D308 の範囲の実数値を表現できます．また，REAL 型では，4 Byte (32 bit) の記憶域を使用して，有効数字 7 桁の実数値を定めます. 絶対値が 1.175494E-38～3.402823E+38 の範囲の実数値を表現できます.

```
 1:      PROGRAM main_ex3
 2:
 3:      IMPLICIT NONE
 4:
 5:      REAL(8) :: b
 6:      REAL(8) :: bb(2)
 7:      REAL(8) :: bbb(4)
 8:
 9:      WRITE(6, *) 'b?'
10:      READ(5, *) b
11:      WRITE(6, *)
12:
13:      WRITE(6, *) 'bb(1) and bb(2)?'
14:      READ(5, *) bb(1), bb(2)
15:      WRITE(6, *)
16:
17:      WRITE(6, *) 'b = ', b
18:      WRITE(6, '(A, F0.6)') 'b = ', b
19:      WRITE(6, '(A, F0.8)') 'b = ', b
20:      WRITE(6, '(A, 1X, F0.8)') 'b = ', b
21:      WRITE(6, '(A, 2X, F0.8)') 'b = ', b
22:      WRITE(6, *)
23:
24:      WRITE(6, *) 'bb(1) = ', bb(1), ', ', 'bb(2) = ', bb(2)
25:      WRITE(6, '(A, F0.8, 2A, F0.8)') 'bb(1) = ', bb(1), ', ', 'bb(2) = ', bb(2)
```

```
26:       WRITE(6, *)
27:
28:       bbb(1) = bb(1)+bb(2)
29:       bbb(2) = bb(1)-bb(2)
30:       bbb(3) = bb(1)*bb(2)
31:       bbb(4) = bb(1)/bb(2)
32:
33:       WRITE(6, '( 4(A, F0.8, A) )') 'bbb(1) = ', bbb(1), ', ', 'bbb(2) = ', bbb(2), ', ', &
34:                                     'bbb(3) = ', bbb(3), ', ', 'bbb(4) = ', bbb(4)
35:       WRITE(6, *)
36:
37:       bbb(1) = b*( bb(1)+bb(2) )
38:       bbb(2) = b*bb(1)+bb(2)
39:       bbb(3) = b*( bb(1)*bb(2) )
40:       bbb(4) = b*bb(1)*bb(2)
41:
42:       WRITE(6, '( 4(A, F0.8, A) )') 'bbb(1) = ', bbb(1), ', ', 'bbb(2) = ', bbb(2), ', ', &
43:                                     'bbb(3) = ', bbb(3), ', ', 'bbb(4) = ', bbb(4)
44:       WRITE(6, *)
45:
46:       STOP
47:
48:       END PROGRAM main_ex3
```

４．main_ex4.f90

PROGRAM 文，READ 文，WRITE 文，CHARACTER 型変数を使用したプログラムです.

```
 1:      PROGRAM main_ex4
 2:
 3:      IMPLICIT NONE
 4:
 5:      CHARACTER(10) :: c
 6:      CHARACTER(10) :: cc
 7:      CHARACTER(20) :: ccc
 8:
 9:      WRITE(6, *) 'c?'
10:      READ(5, *) c
11:      WRITE(6, *)
12:
13:      WRITE(6, *) 'cc?'
14:      READ(5, *) cc
15:      WRITE(6, *)
16:
17:      WRITE(6, *) 'c = ', c
18:      WRITE(6, '(2A)') 'c = ', c
19:      WRITE(6, '(A, 1X, A)') 'c = ', c
20:      WRITE(6, '(A, 2X, A)') 'c = ', c
21:      WRITE(6, *)
22:
23:      WRITE(6, *) 'cc = ', cc
24:      WRITE(6, '(2A)') 'cc = ', cc
25:      WRITE(6, *)
26:
27:      ccc = c//cc
28:
29:      WRITE(6, '(2A)') 'ccc = ', ccc
30:      WRITE(6, *)
31:
32:      ccc = TRIM(c)//TRIM(cc)
```

```
33:
34:       WRITE(6, '(2A)') 'ccc = ', ccc
35:       WRITE(6, *)
36:
37:       STOP
38:
39:       END PROGRAM main_ex4
```

５．main_ex5.f90

PROGRAM 文，READ 文，WRITE 文，INTEGER 型変数，REAL(8)型変数，ALLOCATE 文を使用したプログラムです.

```
 1:      PROGRAM main_ex5
 2:
 3:      IMPLICIT NONE
 4:
 5:      INTEGER :: n
 6:      REAL(8), ALLOCATABLE :: d(:)
 7:      REAL(8), ALLOCATABLE :: dd(:, :)
 8:      INTEGER :: size_d
 9:      INTEGER :: size_dd
10:
11:      WRITE(6, *) 'n?'
12:      READ(5, *) n
13:      WRITE(6, *)
14:
15:      ALLOCATE( d(n) )
16:
17:      d = 0.0D0
18:
19:      ALLOCATE( dd(n, n) )
20:
21:      dd = 0.0D0
22:
23:      size_d  = SIZE( d  )
24:      size_dd = SIZE( dd )
25:
26:      WRITE(6, '(A, I8)') 'size_d  = ', size_d
27:      WRITE(6, '(A, I8)') 'size_dd = ', size_dd
28:      WRITE(6, *)
29:
30:      DEALLOCATE( d  )
31:      DEALLOCATE( dd )
```

```
32:
33:       STOP
34:
35:       END PROGRAM main_ex5
```

６．main_ex6.f90

PROGRAM 文，READ 文，WRITE 文，INTEGER 型変数，REAL(8)型変数，ALLOCATE 文，DO 文，DO WHILE 文を使用したプログラムです.

```
 1:      PROGRAM main_ex6
 2:!########################################################################
 3:
 4:      IMPLICIT NONE
 5:
 6:!------------------------------------------------------------------------
 7:
 8:      INTEGER :: n
 9:      INTEGER :: i, j
10:      REAL(8), ALLOCATABLE :: e(:)
11:      REAL(8) :: sum_e
12:
13:!------------------------------------------------------------------------
14:
15:      n = 5
16:
17:!------------------------------------------------------------------------
18:
19:      ALLOCATE( e(n) )
20:
21:      e = 0.0D0
22:
23:      DO i = 1, n
24:
25:       e(i) = DFLOAT(i)
26:
27:      END DO
28:
29:!------------------------------------------------------------------------
30:
31:      sum_e = 0.0D0
```

```
32:
33:       DO i = 1, n
34:
35:        sum_e = sum_e+e(i)
36:
37:       END DO
38:
39:       WRITE(6, '(A, F0.8)') 'sum_e = ', sum_e
40:       WRITE(6, *)
41:
42:!-----------------------------------------------------------------------
43:
44:       j = 0
45:       sum_e = 0.0D0
46:
47:       DO WHILE( j .LT. 3 )
48:
49:        j = j+1
50:
51:        sum_e = sum_e+e(j)
52:
53:       END DO
54:
55:       WRITE(6, '(A, F0.8)') 'sum_e = ', sum_e
56:       WRITE(6, *)
57:
58:!-----------------------------------------------------------------------
59:
60:       DEALLOCATE( e )
61:
62:!-----------------------------------------------------------------------
63:
64:       STOP
65:
66:!#######################################################################
67:       END PROGRAM main_ex6
```

7．main_ex7.f90

PROGRAM 文，READ 文，WRITE 文，INTEGER 型変数，REAL(8)型変数，DO 文，IF 文を使用したプログラムです.

```
 1:      PROGRAM main_ex7
 2:!#####################################################################
 3:
 4:      IMPLICIT NONE
 5:
 6:!---------------------------------------------------------------------
 7:
 8:      INTEGER :: i, j
 9:      REAL(8) :: sum_1
10:
11:!---------------------------------------------------------------------
12:
13:      sum_1 = 0.0D0
14:
15:      DO i = 1, 100
16:
17:       sum_1 = sum_1+1.0D0
18:
19:      END DO
20:
21:      WRITE(6, '(A, F0.8)') 'sum_1 = ', sum_1
22:      WRITE(6, *)
23:
24:!---------------------------------------------------------------------
25:
26:      j = 0
27:      sum_1 = 0.0D0
28:
29:      DO i = 1, 1000
30:
31:       j = j+1
```

```
32:
33:       sum_1 = sum_1+1.0D0
34:
35:       IF( j .EQ. 200 ) THEN
36:
37:        EXIT
38:
39:       END IF
40:
41:      END DO
42:
43:      WRITE(6, '(A, F0.8)') 'sum_1 = ', sum_1
44:      WRITE(6, *)
45:
46:!-----------------------------------------------------------------------
47:
48:      j = 0
49:      sum_1 = 0.0D0
50:
51:      DO
52:
53:       j = j+1
54:
55:       sum_1 = sum_1+1.0D0
56:
57:       IF( j .EQ. 300 ) THEN
58:
59:        EXIT
60:
61:       END IF
62:
63:      END DO
64:
65:      WRITE(6, '(A, F0.8)') 'sum_1 = ', sum_1
66:      WRITE(6, *)
67:
```

```
68:!-------------------------------------------------------------------
69:
70:      j = 0
71:      sum_1 = 0.0D0
72:
73:      DO
74:
75:       j = j+1
76:
77:       sum_1 = sum_1+1.0D0
78:
79:       IF( j .EQ. 5 ) THEN
80:
81:        WRITE(6, '(3A, F0.8)') 'j =  5', ', ', 'sum_1 = ', sum_1
82:
83:       ELSE IF( j .EQ. 10 ) THEN
84:
85:        WRITE(6, '(3A, F0.8)') 'j = 10', ', ', 'sum_1 = ', sum_1
86:        WRITE(6, *)
87:
88:        EXIT
89:
90:       ELSE
91:
92:        WRITE(6, '(A, I3, 2A, F0.8)') 'j =', j, ', ', 'sum_1 = ', sum_1
93:
94:       END IF
95:
96:      END DO
97:
98:!-------------------------------------------------------------------
99:
100:      STOP
101:
102:!#####################################################################
103:      END PROGRAM main_ex7
```

８．summation_f.f90

SUBROUTINE 文，INTEGER 型変数，REAL(8)型変数，REAL(8)型配列変数，DO 文を使用したプログラムです.

```
1:      SUBROUTINE summation_f(n, f, sum_f)
2:!######################################################################
3:
4:      IMPLICIT NONE
5:
6:!----------------------------------------------------------------------
7:
8:      INTEGER :: n
9:      REAL(8) :: f(n)
10:     REAL(8) :: sum_f
11:
12:!----------------------------------------------------------------------
13:
14:     INTEGER :: i
15:
16:!----------------------------------------------------------------------
17:
18:     sum_f = 0.0D0
19:
20:     DO i = 1, n
21:
22:      sum_f = sum_f+f(i)
23:
24:     END DO
25:
26:!----------------------------------------------------------------------
27:
28:     RETURN
29:
30:!######################################################################
31:     END SUBROUTINE summation_f
```

９．summation_g.f90

SUBROUTINE 文，INTEGER 型変数，REAL(8)型変数，REAL(8)型配列変数，DO 文を使用したプログラムです.

```
1:      SUBROUTINE summation_g(n, g, sum_g)
2:!######################################################################
3:
4:      IMPLICIT NONE
5:
6:!----------------------------------------------------------------------
7:
8:      INTEGER :: n
9:      REAL(8) :: g(n)
10:     REAL(8) :: sum_g
11:
12:!----------------------------------------------------------------------
13:
14:     INTEGER :: i
15:
16:!----------------------------------------------------------------------
17:
18:     sum_g = 0.0D0
19:
20:     DO i = 1, n
21:
22:      sum_g = sum_g+g(i)
23:
24:     END DO
25:
26:!----------------------------------------------------------------------
27:
28:     RETURN
29:
30:!######################################################################
31:     END SUBROUTINE summation_g
```

１０．main_ex8.f90

PROGRAM 文，READ 文，WRITE 文，INTEGER 型変数，REAL(8)型変数，ALLOCATE 文を使用したプログラムです.

```
1:      PROGRAM main_ex8
2:!########################################################################
3:
4:      IMPLICIT NONE
5:
6:!------------------------------------------------------------------------
7:
8:      INTEGER :: n
9:      INTEGER :: i, j
10:     REAL(8), ALLOCATABLE :: f(:)
11:     REAL(8) :: sum_f
12:
13:!------------------------------------------------------------------------
14:
15:     n = 5
16:
17:!------------------------------------------------------------------------
18:
19:     ALLOCATE( f(n) )
20:
21:     f = 0.0D0
22:
23:     f(1) = 1.0D0
24:     f(2) = 1.0D1
25:     f(3) = 1.0D2
26:     f(4) = 1.0D3
27:     f(5) = 1.0D4
28:
29:!------------------------------------------------------------------------
30:
31:     CALL summation_f(n, f, sum_f)
```

```
32:
33:       WRITE(6, '(A, F0.8)') 'sum_f = ', sum_f
34:       WRITE(6, *)
35:
36:!-----------------------------------------------------------------------
37:
38:       CALL summation_g(n, f, sum_f)
39:
40:       WRITE(6, '(A, F0.8)') 'sum_f = ', sum_f
41:       WRITE(6, *)
42:
43:!-----------------------------------------------------------------------
44:
45:       DEALLOCATE( f )
46:
47:!-----------------------------------------------------------------------
48:
49:       STOP
50:
51:!#######################################################################
52:       END PROGRAM main_ex8
```

１１．main_ex9.f90

PROGRAM 文，READ 文，WRITE 文，REAL(8)型変数，CHARACTER 型変数，OPEN 文を使用したプログラムです.

```
 1:      PROGRAM main_ex9
 2:!########################################################################
 3:
 4:      IMPLICIT NONE
 5:
 6:!------------------------------------------------------------------------
 7:
 8:      REAL(8) :: x(4), y(4), z(4)
 9:      REAL(8) :: volume
10:      REAL(8) :: ax, ay, az
11:      REAL(8) :: bx, by, bz
12:      REAL(8) :: cx, cy, cz
13:      REAL(8) :: dx, dy, dz
14:      CHARACTER(1) :: dataname
15:
16:!------------------------------------------------------------------------
17:
18:      OPEN(10, FILE = 'input.dat')
19:
20:      READ(10, *) dataname
21:      READ(10, *) x(1), y(1), z(1)
22:      READ(10, *) dataname
23:      READ(10, *) x(2), y(2), z(2)
24:      READ(10, *) dataname
25:      READ(10, *) x(3), y(3), z(3)
26:      READ(10, *) dataname
27:      READ(10, *) x(4), y(4), z(4)
28:
29:      CLOSE(10)
30:
31:!------------------------------------------------------------------------
```

```
32:
33:       ax = x(2)-x(1)
34:       ay = y(2)-y(1)
35:       az = z(2)-z(1)
36:
37:       bx = x(3)-x(1)
38:       by = y(3)-y(1)
39:       bz = z(3)-z(1)
40:
41:       cx = x(4)-x(1)
42:       cy = y(4)-y(1)
43:       cz = z(4)-z(1)
44:
45:!-----------------------------------------------------------------
46:
47:       dx = ay*bz-az*by
48:       dy = az*bx-ax*bz
49:       dz = ax*by-ay*bx
50:
51:       volume = ( 1.0D0/6.0D0 )*( cx*dx+cy*dy+cz*dz )
52:
53:!-----------------------------------------------------------------
54:
55:       WRITE(6, '(A, F0.8)') 'volume = ', volume
56:       WRITE(6, *)
57:
58:!-----------------------------------------------------------------
59:
60:       OPEN(90, FILE = 'output.dat')
61:
62:       WRITE(90, '(A, F0.8)') 'volume = ', volume
63:
64:       CLOSE(90)
65:
66:!-----------------------------------------------------------------
67:
```

```
68:       STOP
69:
70:!########################################################################
71:       END PROGRAM main_ex9
```

１２．main_ex10.f90

PROGRAM 文，READ 文，WRITE 文，構造体変数，REAL(8)型変数，CHARACTER 型変数，OPEN 文を使用したプログラムです.

```
 1:      PROGRAM main_ex10
 2:!#########################################################################
 3:
 4:      IMPLICIT NONE
 5:
 6:!-------------------------------------------------------------------------
 7:
 8:      TYPE :: struct_vertices
 9:
10:       !-----------------------------------------------------
11:       !
12:       ! n
13:       ! The total number of vertices
14:       !
15:       ! x(:), y(:), z(:)
16:       ! Cartesian coordinates of a vertex
17:       !
18:       !-----------------------------------------------------
19:
20:       INTEGER :: n
21:
22:       REAL(8), ALLOCATABLE :: x(:), y(:), z(:)
23:
24:       !-----------------------------------------------------
25:
26:      END TYPE struct_vertices
27:
28:!-------------------------------------------------------------------------
29:
30:      TYPE :: struct_tetrahedron
31:
```

```
32:       !-------------------------------------------------------
33:       !
34:       ! volume
35:       ! Volume of a tetrahedron
36:       !
37:       !-------------------------------------------------------
38:
39:       REAL(8) :: volume
40:
41:       !-------------------------------------------------------
42:
43:      END TYPE struct_tetrahedron
44:
45:!---------------------------------------------------------------------
46:
47:      TYPE(struct_vertices) :: vs
48:      TYPE(struct_tetrahedron) :: t
49:      REAL(8) :: ax, ay, az
50:      REAL(8) :: bx, by, bz
51:      REAL(8) :: cx, cy, cz
52:      REAL(8) :: dx, dy, dz
53:      CHARACTER(1) :: dataname
54:
55:!---------------------------------------------------------------------
56:
57:      vs%n = 4
58:
59:      ALLOCATE( vs%x(4) )
60:      ALLOCATE( vs%y(4) )
61:      ALLOCATE( vs%z(4) )
62:
63:      vs%x = 0.0D0
64:      vs%y = 0.0D0
65:      vs%z = 0.0D0
66:
67:      t%volume = 0.0D0
```

```
68:
69:!----------------------------------------------------------------------
70:
71:      OPEN(10, FILE = 'input.dat')
72:
73:      READ(10, *) dataname
74:      READ(10, *) vs%x(1), vs%y(1), vs%z(1)
75:      READ(10, *) dataname
76:      READ(10, *) vs%x(2), vs%y(2), vs%z(2)
77:      READ(10, *) dataname
78:      READ(10, *) vs%x(3), vs%y(3), vs%z(3)
79:      READ(10, *) dataname
80:      READ(10, *) vs%x(4), vs%y(4), vs%z(4)
81:
82:      CLOSE(10)
83:
84:!----------------------------------------------------------------------
85:
86:      ax = vs%x(2)-vs%x(1)
87:      ay = vs%y(2)-vs%y(1)
88:      az = vs%z(2)-vs%z(1)
89:
90:      bx = vs%x(3)-vs%x(1)
91:      by = vs%y(3)-vs%y(1)
92:      bz = vs%z(3)-vs%z(1)
93:
94:      cx = vs%x(4)-vs%x(1)
95:      cy = vs%y(4)-vs%y(1)
96:      cz = vs%z(4)-vs%z(1)
97:
98:!----------------------------------------------------------------------
99:
100:      dx = ay*bz-az*by
101:      dy = az*bx-ax*bz
102:      dz = ax*by-ay*bx
103:
```

```
104:       t%volume = ( 1.0D0/6.0D0 )*( cx*dx+cy*dy+cz*dz )
105:
106:!-----------------------------------------------------------------
107:
108:       WRITE(6, '(A, F0.8)') 'volume = ', t%volume
109:       WRITE(6, *)
110:
111:!-----------------------------------------------------------------
112:
113:       OPEN(90, FILE = 'output.dat')
114:
115:       WRITE(90, '(A, F0.8)') 'volume = ', t%volume
116:
117:       CLOSE(90)
118:
119:!-----------------------------------------------------------------
120:
121:       STOP
122:
123:!#################################################################
124:       END PROGRAM main_ex10
```

１３．mod_vertices.f90

MODULE 文，構造体変数，INTEGER 型変数，REAL(8)型配列変数，ALLOCATE 文，SUBTOUTINE 文を使用したプログラムです.

```
 1:      MODULE mod_vertices
 2:!#########################################################################
 3:
 4:      IMPLICIT NONE
 5:
 6:!-------------------------------------------------------------------------
 7:
 8:      TYPE :: struct_vertices
 9:
10:       !-----------------------------------------------------
11:
12:       PRIVATE
13:
14:       !-----------------------------------------------------
15:       !
16:       ! n
17:       ! The total number of vertices
18:       !
19:       ! x(:, :)
20:       ! Cartesian coordinates of a vertex
21:       ! (x, y, z)
22:       !
23:       !-----------------------------------------------------
24:
25:       INTEGER :: n
26:
27:       REAL(8), ALLOCATABLE :: x(:, :)
28:
29:       !-----------------------------------------------------
30:
31:     END TYPE struct_vertices
```

```
32:
33:!---------------------------------------------------------------------
34:
35:      CONTAINS
36:
37:
38:      ! Set the total number of vertices
39:!#####################################################################
40:      SUBROUTINE set_vertices_n(vs, n)
41:!#####################################################################
42:
43:      TYPE(struct_vertices), POINTER, INTENT(OUT) :: vs
44:
45:      INTEGER, INTENT(IN) :: n
46:
47:!---------------------------------------------------------------------
48:
49:      vs%n = n
50:
51:!---------------------------------------------------------------------
52:
53:      RETURN
54:
55:!#####################################################################
56:      END SUBROUTINE set_vertices_n
57:!#####################################################################
58:
59:
60:      ! Get the total number of vertices
61:!#####################################################################
62:      SUBROUTINE get_vertices_n(vs, n)
63:!#####################################################################
64:
65:      TYPE(struct_vertices), POINTER, INTENT(IN) :: vs
66:
67:      INTEGER, INTENT(OUT) :: n
```

```
68:
69:!---------------------------------------------------------------------
70:
71:      n = vs%n
72:
73:!---------------------------------------------------------------------
74:
75:      RETURN
76:
77:!#####################################################################
78:      END SUBROUTINE get_vertices_n
79:!#####################################################################
80:
81:
82:      ! Set Cartesian coordinates of a vertex
83:!#####################################################################
84:      SUBROUTINE set_vertices_x(vs, x)
85:!#####################################################################
86:
87:      TYPE(struct_vertices), POINTER, INTENT(INOUT) :: vs
88:
89:      REAL(8), INTENT(IN) :: x(3, vs%n)
90:
91:!---------------------------------------------------------------------
92:
93:      INTEGER :: i
94:      INTEGER :: id
95:
96:!---------------------------------------------------------------------
97:
98:      DO id = 1, vs%n
99:
100:       DO i = 1, 3
101:
102:        vs%x(i, id) = x(i, id)
103:
```

```
104:       END DO
105:
106:      END DO
107:
108:!---------------------------------------------------------------------
109:
110:      RETURN
111:
112:!#####################################################################
113:      END SUBROUTINE set_vertices_x
114:!#####################################################################
115:
116:
117:      ! Get Cartesian coordinates of a vertex
118:!#####################################################################
119:      SUBROUTINE get_vertices_x(vs, x)
120:!#####################################################################
121:
122:      TYPE(struct_vertices), POINTER, INTENT(IN) :: vs
123:
124:      REAL(8), INTENT(OUT) :: x(3, vs%n)
125:
126:!---------------------------------------------------------------------
127:
128:      INTEGER :: i
129:      INTEGER :: id
130:
131:!---------------------------------------------------------------------
132:
133:      DO id = 1, vs%n
134:
135:       DO i = 1, 3
136:
137:        x(i, id) = vs%x(i, id)
138:
139:       END DO
```

```
140:
141:      END DO
142:
143:!-----------------------------------------------------------------
144:
145:      RETURN
146:
147:!#################################################################
148:      END SUBROUTINE get_vertices_x
149:!#################################################################
150:
151:
152:      ! Initialize vertices
153:!#################################################################
154:      SUBROUTINE init_vertices(vs, n)
155:!#################################################################
156:
157:      TYPE(struct_vertices), POINTER, INTENT(INOUT) :: vs
158:
159:      INTEGER, INTENT(IN) :: n
160:
161:!-----------------------------------------------------------------
162:
163:      vs%n = n
164:
165:      !---------------------------------------------------------
166:
167:      ALLOCATE( vs%x(3, n) )
168:
169:      vs%x = 0.0D0
170:
171:!-----------------------------------------------------------------
172:
173:      RETURN
174:
175:!#################################################################
```

```
176:      END SUBROUTINE init_vertices
177:!########################################################################
178:
179:
180:      ! Delete vertices
181:!########################################################################
182:      SUBROUTINE del_vertices(vs)
183:!########################################################################
184:
185:      TYPE(struct_vertices), POINTER, INTENT(INOUT) :: vs
186:
187:!------------------------------------------------------------------------
188:
189:      IF( vs%n .EQ. 0 ) THEN
190:
191:       RETURN
192:
193:      END IF
194:
195:!------------------------------------------------------------------------
196:
197:      vs%n = 0
198:
199:      !-------------------------------------------------------------
200:
201:      DEALLOCATE( vs%x )
202:
203:!------------------------------------------------------------------------
204:
205:      RETURN
206:
207:!########################################################################
208:      END SUBROUTINE del_vertices
209:!########################################################################
210:
211:
```

```
212:!########################################################################
213:      END MODULE mod_vertices
```

１４．mod_tetrahedron.f90

MODULE 文, 構造体変数, REAL(8)型変数, SUBTOUTINE 文を使用したプログラムです.

```
 1:      MODULE mod_tetrahedron
 2:!#########################################################################
 3:
 4:      USE mod_vertices
 5:
 6:!-------------------------------------------------------------------------
 7:
 8:      IMPLICIT NONE
 9:
10:!-------------------------------------------------------------------------
11:
12:      TYPE :: struct_tetrahedron
13:
14:       !-----------------------------------------------------
15:
16:       PRIVATE
17:
18:       !-----------------------------------------------------
19:
20:       TYPE(struct_vertices), POINTER :: vs => NULL()
21:
22:       !-----------------------------------------------------
23:       !
24:       ! volume
25:       ! Volume of a tetrahedron
26:       !
27:       !-----------------------------------------------------
28:
29:       REAL(8) :: volume
30:
31:       !-----------------------------------------------------
32:
```

```
33:       END TYPE struct_tetrahedron
34:
35:!---------------------------------------------------------------------
36:
37:       CONTAINS
38:
39:
40:       ! Set volume
41:!#####################################################################
42:       SUBROUTINE set_tetrahedron_volume(t, volume)
43:!#####################################################################
44:
45:       TYPE(struct_tetrahedron), POINTER, INTENT(OUT) :: t
46:
47:       REAL(8), INTENT(IN) :: volume
48:
49:!---------------------------------------------------------------------
50:
51:       t%volume = volume
52:
53:!---------------------------------------------------------------------
54:
55:       RETURN
56:
57:!#####################################################################
58:       END SUBROUTINE set_tetrahedron_volume
59:!#####################################################################
60:
61:
62:       ! Get volume
63:!#####################################################################
64:       SUBROUTINE get_tetrahedron_volume(t, volume)
65:!#####################################################################
66:
67:       TYPE(struct_tetrahedron), POINTER, INTENT(IN) :: t
68:
```

```
69:       REAL(8), INTENT(OUT) :: volume
70:
71:!------------------------------------------------------------------------
72:
73:       volume = t%volume
74:
75:!------------------------------------------------------------------------
76:
77:       RETURN
78:
79:!########################################################################
80:       END SUBROUTINE get_tetrahedron_volume
81:!########################################################################
82:
83:
84:       ! Initialize tetrahedron
85:!########################################################################
86:       SUBROUTINE init_tetrahedron(t, vs)
87:!########################################################################
88:
89:       TYPE(struct_tetrahedron), POINTER, INTENT(INOUT) :: t
90:
91:       TYPE(struct_vertices), POINTER, INTENT(IN) :: vs
92:
93:!------------------------------------------------------------------------
94:
95:       t%vs => vs
96:
97:!------------------------------------------------------------------------
98:
99:       t%volume = 0.0D0
100:
101:!------------------------------------------------------------------------
102:
103:       RETURN
104:
```

```
105:!########################################################################
106:      END SUBROUTINE init_tetrahedron
107:!########################################################################
108:
109:
110:      ! Calculate tetrahedron
111:!########################################################################
112:      SUBROUTINE cal_tetrahedron(t)
113:!########################################################################
114:
115:      TYPE(struct_tetrahedron), POINTER, INTENT(INOUT) :: t
116:
117:!------------------------------------------------------------------------
118:
119:      REAL(8) :: vs_x(3, 4)
120:      REAL(8) :: ax, ay, az
121:      REAL(8) :: bx, by, bz
122:      REAL(8) :: cx, cy, cz
123:      REAL(8) :: dx, dy, dz
124:
125:!------------------------------------------------------------------------
126:
127:      CALL get_vertices_x(t%vs, vs_x)
128:
129:!------------------------------------------------------------------------
130:
131:      ax = vs_x(1, 2)-vs_x(1, 1)
132:      ay = vs_x(2, 2)-vs_x(2, 1)
133:      az = vs_x(3, 2)-vs_x(3, 1)
134:
135:      bx = vs_x(1, 3)-vs_x(1, 1)
136:      by = vs_x(2, 3)-vs_x(2, 1)
137:      bz = vs_x(3, 3)-vs_x(3, 1)
138:
139:      cx = vs_x(1, 4)-vs_x(1, 1)
140:      cy = vs_x(2, 4)-vs_x(2, 1)
```

```
141:       cz = vs_x(3, 4)-vs_x(3, 1)
142:
143:!------------------------------------------------------------------
144:
145:       dx = ay*bz-az*by
146:       dy = az*bx-ax*bz
147:       dz = ax*by-ay*bx
148:
149:       t%volume = ( 1.0D0/6.0D0 )*( cx*dx+cy*dy+cz*dz )
150:
151:!------------------------------------------------------------------
152:
153:       RETURN
154:
155:!######################################################################
156:       END SUBROUTINE cal_tetrahedron
157:!######################################################################
158:
159:
160:       ! Delete tetrahedron
161:!######################################################################
162:       SUBROUTINE del_tetrahedron(t)
163:!######################################################################
164:
165:       TYPE(struct_tetrahedron), POINTER, INTENT(INOUT) :: t
166:
167:!------------------------------------------------------------------
168:
169:       NULLIFY( t%vs )
170:
171:!------------------------------------------------------------------
172:
173:       t%volume = 0.0D0
174:
175:!------------------------------------------------------------------
176:
```

```
177:      RETURN
178:
179:!######################################################################
180:      END SUBROUTINE del_tetrahedron
181:!######################################################################
182:
183:
184:!######################################################################
185:      END MODULE mod_tetrahedron
```

１５．mod_appli.f90

MODULE 文，READ 文，WRITE 文，CHARACTER 型変数，OPEN 文を使用したプログラムです.

```
 1:      MODULE mod_appli
 2:!######################################################################
 3:
 4:      USE mod_vertices
 5:      USE mod_tetrahedron
 6:
 7:!----------------------------------------------------------------------
 8:
 9:      IMPLICIT NONE
10:
11:!----------------------------------------------------------------------
12:
13:      TYPE(struct_vertices), POINTER    :: vs
14:      TYPE(struct_tetrahedron), POINTER :: t
15:
16:!----------------------------------------------------------------------
17:
18:      CONTAINS
19:
20:
21:      ! Start appli
22:!######################################################################
23:      SUBROUTINE start_appli()
24:!######################################################################
25:
26:      REAL(8) :: vs_x(3, 4)
27:
28:      CHARACTER(1) :: dataname
29:
30:!----------------------------------------------------------------------
31:
```

```
32:      ALLOCATE( vs )
33:      ALLOCATE( t )
34:
35:!------------------------------------------------------------------
36:
37:      CALL init_vertices(vs, 4)
38:      CALL init_tetrahedron(t, vs)
39:
40:!------------------------------------------------------------------
41:
42:      OPEN(10, FILE = 'input.dat')
43:
44:      READ(10, *) dataname
45:      READ(10, *) vs_x(1, 1), vs_x(2, 1), vs_x(3, 1)
46:      READ(10, *) dataname
47:      READ(10, *) vs_x(1, 2), vs_x(2, 2), vs_x(3, 2)
48:      READ(10, *) dataname
49:      READ(10, *) vs_x(1, 3), vs_x(2, 3), vs_x(3, 3)
50:      READ(10, *) dataname
51:      READ(10, *) vs_x(1, 4), vs_x(2, 4), vs_x(3, 4)
52:
53:      CLOSE(10)
54:
55:      CALL set_vertices_x(vs, vs_x)
56:
57:!------------------------------------------------------------------
58:
59:      RETURN
60:
61:!######################################################################
62:      END SUBROUTINE start_appli
63:!######################################################################
64:
65:
66:      ! Run appli
67:!######################################################################
```

```
68:       SUBROUTINE run_appli()
69:!########################################################################
70:
71:       REAL(8) :: t_volume
72:
73:!------------------------------------------------------------------------
74:
75:       CALL cal_tetrahedron(t)
76:
77:       CALL get_tetrahedron_volume(t, t_volume)
78:
79:!------------------------------------------------------------------------
80:
81:       WRITE(6, '(A, F0.8)') 'volume = ', t_volume
82:       WRITE(6, *)
83:
84:!------------------------------------------------------------------------
85:
86:       OPEN(90, FILE = 'output.dat')
87:
88:       WRITE(90, '(A, F0.8)') 'volume = ', t_volume
89:
90:       CLOSE(90)
91:
92:!------------------------------------------------------------------------
93:
94:       RETURN
95:
96:!########################################################################
97:       END SUBROUTINE run_appli
98:!########################################################################
99:
100:
101:      ! Finish appli
102:!########################################################################
103:      SUBROUTINE finish_appli()
```

```
104:!####################################################################
105:
106:      CALL del_vertices(vs)
107:      CALL del_tetrahedron(t)
108:
109:!--------------------------------------------------------------------
110:
111:      DEALLOCATE( vs )
112:      DEALLOCATE( t )
113:
114:!--------------------------------------------------------------------
115:
116:      RETURN
117:
118:!####################################################################
119:      END SUBROUTINE finish_appli
120:!####################################################################
121:
122:
123:!####################################################################
124:      END MODULE mod_appli
```

１６．main_appli.f90

PROGRAM 文を使用したプログラムです.

```
 1:      PROGRAM main_appli
 2:!####################################################################
 3:
 4:      USE mod_appli
 5:
 6:!--------------------------------------------------------------------
 7:
 8:      IMPLICIT NONE
 9:
10:!--------------------------------------------------------------------
11:
12:      ! Start appli
13:      CALL start_appli()
14:
15:!--------------------------------------------------------------------
16:
17:      ! Run appli
18:      CALL run_appli()
19:
20:!--------------------------------------------------------------------
21:
22:      ! Finish appli
23:      CALL finish_appli()
24:
25:!--------------------------------------------------------------------
26:
27:      STOP
28:
29:!####################################################################
30:      END PROGRAM main_appli
```