メッシュ入力：INPUT_GRID（1/2）

```
#include <stdio.h>
#include <stdlib.h>
#include "pfem_util.h"
#include "allocate.h"
void INPUT_GRID()
{
        FILE *fp;
        int i,j,k,ii,kk,nn,icel,iS,iE;
        int NTYPE,IMAT;

        if( (fp=fopen(fname,"r")) == NULL){
                fprintf(stdout,"input file cannot be opened!¥n");
                exit(1);
        }
/**
        NODE
**/
        fscanf(fp,"%d",&N);

        NP=N;
        XYZ=(KREAL**)allocate_matrix(sizeof(KREAL),N,3);

        for(i=0;i<N;i++){
                for(j=0;j<3;j++){
                        XYZ[i][j]=0.0;
                }
        }

        for(i=0;i<N;i++){
                fscanf(fp,"%d %lf %lf %lf",&ii,&XYZ[i][0],&XYZ[i][1],&XYZ[i][2]);
        }
```

XYZ[N][3]

allocate, deallocate関数

```
#include <stdio.h>
#include <stdlib.h>
void* allocate_vector(int size,int m)
{
  void *a;
  if ( ( a=(void * )malloc( m * size ) ) == NULL ) {
     fprintf(stdout,"Error:Memory does not enough! in vector ¥n");
     exit(1);
  }
  return a;
}

void deallocate_vector(void *a)
{
  free( a );
}

void** allocate_matrix(int size,int m,int n)
{
  void **aa;
  int i;
  if ( ( aa=(void ** )malloc( m * sizeof(void*) ) ) == NULL ) {
     fprintf(stdout,"Error:Memory does not enough! aa in matrix ¥n");
     exit(1);
  }
  if ( ( aa[0]=(void * )malloc( m * n * size ) ) == NULL ) {
     fprintf(stdout,"Error:Memory does not enough! in matrix ¥n");
     exit(1);
  }
        for(i=1;i<m;i++) aa[i]=(char*)aa[i-1]+size*n;
  return aa;
}

void deallocate_matrix(void **aa)
{
        free( aa );
}
```

allocateをFORTRAN並みに簡単にやるための関数

cube.0：節点データ，要素数，要素タイプ（NX=NY=NZ=4）

```
     125                                                          =5*5*5                  N
       1   0.000000E+00   0.000000E+00   0.000000E+00                                    i, XYZ[i][0-2]
       2   1.000000E+00   0.000000E+00   0.000000E+00
       3   2.000000E+00   0.000000E+00   0.000000E+00
       4   3.000000E+00   0.000000E+00   0.000000E+00
       5   4.000000E+00   0.000000E+00   0.000000E+00
       6   0.000000E+00   1.000000E+00   0.000000E+00
       7   1.000000E+00   1.000000E+00   0.000000E+00
       8   2.000000E+00   1.000000E+00   0.000000E+00
       9   3.000000E+00   1.000000E+00   0.000000E+00

（途中省略）

     121   0.000000E+00   4.000000E+00   4.000000E+00
     122   1.000000E+00   4.000000E+00   4.000000E+00
     123   2.000000E+00   4.000000E+00   4.000000E+00
     124   3.000000E+00   4.000000E+00   4.000000E+00
     125   4.000000E+00   4.000000E+00   4.000000E+00
      64                                                          =4*4*4                  ICELTOT
     361    361    361    361    361    361    361    361    361    361                   TYPE
     361    361    361    361    361    361    361    361    361    361
     361    361    361    361    361    361    361    361    361    361
     361    361    361    361    361    361    361    361    361    361
     361    361    361    361    361    361    361    361    361    361
     361    361    361    361    361    361    361    361    361    361
     361    361    361    361
```

movie

メッシュ入力：INPUT_GRID（2/2）

```
/**
        ELEMENT
**/
        fscanf(fp,"%d",&ICELTOT);

        ICELNOD=(KINT**)allocate_matrix(sizeof(KINT),ICELTOT,8);
        for(i=0;i<ICELTOT;i++) fscanf(fp,"%d",&NTYPE);


        for(icel=0;icel<ICELTOT;icel++){
                fscanf(fp,"%d %d %d %d %d %d %d %d %d %d",&ii,&IMAT,
                &ICELNOD[icel][0],&ICELNOD[icel][1],&ICELNOD[icel][2],&ICELNOD[icel][3],
                &ICELNOD[icel][4],&ICELNOD[icel][5],&ICELNOD[icel][6],&ICELNOD[icel][7]);
        }
/**
        NODE grp.  info.
**/
        fscanf(fp,"%d",&NODGRPtot);

        NODGRP_INDEX=(KINT*  )allocate_vector(sizeof(KINT),NODGRPtot+1);
        NODGRP_NAME =(CHAR80*)allocate_vector(sizeof(CHAR80),NODGRPtot);
        for(i=0;i<NODGRPtot+1;i++) NODGRP_INDEX[i]=0;

        for(i=0;i<NODGRPtot;i++) fscanf(fp,"%d",&NODGRP_INDEX[i+1]);
        nn=NODGRP_INDEX[NODGRPtot];
        NODGRP_ITEM=(KINT*)allocate_vector(sizeof(KINT),nn);

        for(k=0;k<NODGRPtot;k++){
                iS= NODGRP_INDEX[k];
                iE= NODGRP_INDEX[k+1];
                fscanf(fp,"%s",NODGRP_NAME[k].name);
                nn= iE - iS;
                if( nn != 0 ){
                        for(kk=iS;kk<iE;kk++) fscanf(fp,"%d",&NODGRP_ITEM[kk]);
                }
        }

        fclose(fp);
}
```

ICELNOD[i][j]の中身としては「1」から始まる通し節点番号がそのまま読み込まれている。
要素番号は「0」から番号付け。

ICELNOD[ICELTOT][8]

cube.0：要素データ

1	1	1	2	7	6	26	27	32	31	icel, imat, ICELNOD[icel][0-7]
2	1	2	3	8	7	27	28	33	32	
3	1	3	4	9	8	28	29	34	33	
4	1	4	5	10	9	29	30	35	34	
5	1	6	7	12	11	31	32	37	36	
6	1	7	8	13	12	32	33	38	37	
7	1	8	9	14	13	33	34	39	38	
8	1	9	10	15	14	34	35	40	39	
9	1	11	12	17	16	36	37	42	41	
10	1	12	13	18	17	37	38	43	42	
11	1	13	14	19	18	38	39	44	43	
12	1	14	15	20	19	39	40	45	44	
13	1	16	17	22	21	41	42	47	46	
（途中省略）										
42	1	62	63	68	67	87	88	93	92	
43	1	63	64	69	68	88	89	94	93	
44	1	64	65	70	69	89	90	95	94	
45	1	66	67	72	71	91	92	97	96	
46	1	67	68	73	72	92	93	98	97	
47	1	68	69	74	73	93	94	99	98	
48	1	69	70	75	74	94	95	100	99	
49	1	76	77	82	81	101	102	107	106	
50	1	77	78	83	82	102	103	108	107	
51	1	78	79	84	83	103	104	109	108	
52	1	79	80	85	84	104	105	110	109	
53	1	81	82	87	86	106	107	112	111	
54	1	82	83	88	87	107	108	113	112	
55	1	83	84	89	88	108	109	114	113	
56	1	84	85	90	89	109	110	115	114	
57	1	86	87	92	91	111	112	117	116	
58	1	87	88	93	92	112	113	118	117	
59	1	88	89	94	93	113	114	119	118	
60	1	89	90	95	94	114	115	120	119	
61	1	91	92	97	96	116	117	122	121	
62	1	92	93	98	97	117	118	123	122	
63	1	93	94	99	98	118	119	124	123	
64	1	94	95	100	99	119	120	125	124	

メッシュ入力：INPUT_GRID（2/2）

```
/**
        ELEMENT
**/
        fscanf(fp,"%d",&ICELTOT);

        ICELNOD=(KINT**)allocate_matrix(sizeof(KINT),ICELTOT,8);
        for(i=0;i<ICELTOT;i++) fscanf(fp,"%d",&NTYPE);


        for(icel=0;icel<ICELTOT;icel++){
                fscanf(fp,"%d %d %d %d
                &ICELNOD[icel][0],&ICEL
                &ICELNOD[icel][4],&ICEL
        }
/**
        NODE grp.  info.
**/
        fscanf(fp,"%d",&NODGRPtot);

        NODGRP_INDEX=(KINT*  )allocate_vector(sizeof(KINT),NODGRPtot+1);
        NODGRP_NAME =(CHAR80*)allocate_vector(sizeof(CHAR80),NODGRPtot);
        for(i=0;i<NODGRPtot+1;i++) NODGRP_INDEX[i]=0;

        for(i=0;i<NODGRPtot;i++) fscanf(fp,"%d",&NODGRP_INDEX[i+1]);
        nn=NODGRP_INDEX[NODGRPtot];
        NODGRP_ITEM=(KINT*)allocate_vector(sizeof(KINT),nn);

        for(k=0;k<NODGRPtot;k++){
                iS= NODGRP_INDEX[k];
                iE= NODGRP_INDEX[k+1];
                fscanf(fp,"%s",NODGRP_NAME[k].name);
                nn= iE - iS;
                if( nn != 0 ){
                        for(kk=iS;kk<iE;kk++) fscanf(fp,"%d",&NODGRP_ITEM[kk]);
                }
        }

        fclose(fp);
}
```

NODGRP_INDEX[NODGRPtot+1]
NODGRP_NAME [NODGRPtot]
NODGRP_ITEM [nn]
** nn= NODGRP_INDEX[NODGRPtot]**

節点グループの中身も「1」から始まる通し節点番号がそのまま読み込まれている。

cube.0：節点グループデータ

```
    4                                                                                  NODGRPtot
   25         50         75        100                                                 NODGRP_INDEX[1-4]
Xmin                                                                                   NODGRP_NAME[0]
    1          6         11         16         21         26         31         36         41         46   NODGRP_ITEM[0-24]
   51         56         61         66         71         76         81         86         91         96
  101        106        111        116        121
Ymin                                                                                   NODGRP_NAME[1]
    1          2          3          4          5         26         27         28         29         30   NODGRP_ITEM[25-49]
   51         52         53         54         55         76         77         78         79         80
  101        102        103        104        105
Zmin                                                                                   NODGRP_NAME[2]
    1          2          3          4          5          6          7          8          9         10   NODGRP_ITEM[50-74]
   11         12         13         14         15         16         17         18         19         20
   21         22         23         24         25
Zmax                                                                                   NODGRP_NAME[3]
  101        102        103        104        105        106        107        108        109        110   NODGRP_ITEM[75-99]
  111        112        113        114        115        116        117        118        119        120
  121        122        123        124        125
```