

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define N 4
5
6 /* 行列の入力 */
7 void input_matrix(double **a, char c, FILE *fin, FILE *fout);
8 /* ベクトルの入力 */
9 void input_vector(double *b, char c, FILE *fin, FILE *fout);
10
11 /* 行列の領域確保 */
12 double **dmatrix(int nr1, int nr2, int n11, int n12);
13 /* 行列の領域解放 */
14 void free_dmatrix(double **a, int nr1, int nr2, int n1, int n12);
15 double *dvector(int i, int j); /* ベクトル領域の確保 */
16 void free_dvector(double *a, int i); /* 領域の解放 */
17
18 int main(void)
19 {
20     FILE *fin, *fout;
21     double **a, *b;
22
23     /* 行列およびベクトルの領域確保 */
24     a = dmatrix(1, N, 1, N); /* 行列 a[1...N][1...N] */
25     b = dvector(1, N); /* b[1...N] */
26
27     /* ファイルのオープン */
28     if( (fin = fopen("input.dat", "r")) == NULL )
29     {
30         printf("ファイルが見つかりません : input.dat \n");
31         exit(1);
32     }
33
34     if( (fout = fopen("output.dat", "w")) == NULL )
35     {
36         printf("ファイルが作成できません : output.dat \n");
37         exit(1);
38     }
39
40     input_matrix(a, 'A', fin, fout); /* 行列 A の入出力 */
41     input_vector(b, 'b', fin, fout); /* ベクトル b の入出力 */
42
43     fclose(fin); fclose(fout);
44
45     /* 領域の解放 */
46     free_dmatrix(a, 1, N, 1, N);
47     free_dvector(b, 1);
48
49     return 0;
50 }
51
52 /* a[1...N][1...N] の入力 */
53 void input_matrix(double **a, char c, FILE *fin, FILE *fout)
54 {
55     int i, j;
56
57     fprintf(fout, "行列%c は次の通りです\n", c);
58     for(i = 1; i <= N; i++){
59         for(j = 1; j <= N; j++){
60             fscanf(fin, "%lf", &a[i][j]);
61             fprintf(fout, "%5.2f\t", a[i][j]);
62         }
63         fprintf(fout, "\n");
64     }
65 }
66
67 /* b[1...N]の入力 */
68 void input_vector(double *b, char c, FILE *fin, FILE *fout)
69 {
70     int i;
71
72     fprintf(fout, "ベクトル%c は次の通りです\n", c);
73     for(i = 1; i <= N; i++){
74         fscanf(fin, "%lf", &b[i]);
75         fprintf(fout, "%5.2f\t", b[i]);
76         fprintf(fout, "\n");
77     }
78 }
79
80 double **dmatrix(int nr1, int nr2, int n11, int n12)
81 {
82     int i, nrow, ncol;
83     double **a;
84
85     nrow = nr2 - nr1 + 1; /* 行の数 */
86     ncol = n12 - n11 + 1; /* 列の数 */
87
88     /* 行の確保 */
89     if((a = malloc(nrow * sizeof(double))) == NULL){
90         printf("メモリが確保できません (行列 a)\n");
91         exit(1);
92     }
93     a = a - nr1; /* 行をずらす */
94
95     /* 列の確保 */
96     for(i = nr1; i <= nr2; i++) a[i] = malloc(ncol * sizeof(double));
97     for(i = nr1; i <= nr2; i++) a[i] = a[i] - n11; /* 列をずらす */
98
99     return (a);
100 }
101
102 void free_dmatrix(double **a, int nr1, int nr2, int n11, int n12)
103 {
104     int i;
105
106     /* メモリの解放 */
107     for(i = nr1; i <= nr2; i++) free((void *) (a[i] + n11));
108     free((void *) (a + nr1));
109 }
110
111 double *dvector(int i, int j)
112 {
113     double *a;
114
115     if((a = malloc( ((j - i + 1) * sizeof(double))) ) == NULL)
116     {
117         printf("メモリが確保できません (from dvector) \n");
118         exit(1);
119     }
120
121     return (a - i);
122 }
123
124 void free_dvector(double *a, int i)
125 {
126     free( (void *) (a + i) ); /* (void *) 型へのキャストが必要 */
127 }

```