

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define N 4 /* N 次正方行列 */
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 /* 行列の領域確保 */
11 double **dmatrix(int nr1, int nr2, int n11, int n12);
12 /* 行列の領域解放 */
13 void free_dmatrix(double **a, int nr1, int nr2, int n11, int n12);
14 /* ベクトル領域の確保 */
15 double *dvector(int i, int j);
16 /* 領域の解放 */
17 void free_dvector(double *a, int i);
18 /* ガウス消去法 */
19 double *simple_gauss( double **a, double *b );
20
21 int main(void)
22 {
23     FILE *fin, *fout;
24     double **a, *b;
25     int i;
26
27     /* 行列およびベクトルの領域確保 */
28     a = dmatrix(1, N, 1, N); /* 行列 a[1...N][1...N] */
29     b = dvector(1,N); /* b[1...N] */
30
31     /* ファイルのオープン */
32     if ( (fin = fopen( "input.dat", "r")) == NULL )
33     {
34         printf("ファイルが見つかりません : input.dat \n");
35         exit(1);
36     }
37     if( (fout = fopen( "output.dat", "w")) == NULL )
38     {
39         printf("ファイルが作成できません : output.dat \n");
40         exit(1);
41     }
42
43     input_matrix( a, 'A', fin, fout ); /* 行列 A の入出力 */
44     input_vector( b, 'b', fin, fout ); /* ベクトル b の入出力 */
45     b = simple_gauss( a, b ); /* ガウス消去法 */
46
47     /* 結果の出力 */
48     fprintf( fout, "Ax=b の解は次の通りです\n");
49     for( i = 1 ; i <= N ; i++)
50     {
51         fprintf(fout, "%f\n", b[i]);
52     }
53
54     fclose(fin); fclose(fout); /* ファイルのクローズ */
55
56     /* 領域の解放 */
57     free_dmatrix( a, 1, N, 1, N ); free_dvector( b, 1 );
58
59     return 0;
60 }
61
62 /* ガウス消去法 */
63 double *simple_gauss( double **a, double *b )
64 {
65     int i, j, k;
66     double alpha, tmp;
67
68     /* 前進消去 */
69     for( k = 1; k <= N-1; k++)
70     {
71         for( i = k+1; i <= N; i++)
72         {
73             alpha = - a[i][k]/a[k][k];
74             for( j = k+1; j <= N; j++)
75             {
76                 a[i][j] = a[i][j] + alpha * a[k][j];
77             }
78             b[i] = b[i] + alpha * b[k];
79         }
80     }
81
82     /* 後退代入 */
83     b[N] = b[N]/a[N][N];
84     for( k = N-1; k >= 1; k--)
85     {
86         tmp = b[k];
87         for( j = k+1; j <= N; j++)
88         {
89             tmp = tmp - a[k][j] * b[j];
90         }
91         b[k] = tmp/a[k][k];
92     }
93
94     return b;
95 }
96
97 /* a[1...N][1...N] の入力 */
98 void input_matrix(double **a, char c, FILE *fin, FILE *fout)
99 {
100     int i, j;
101
102     fprintf(fout, "行列%c は次の通りです\n", c);
103     for (i = 1; i <= N; i++)
104     {
105         for (j = 1; j <= N; j++)
106         {
107             fscanf(fin, "%lf", &a[i][j]);
108             fprintf(fout, "%5.2f\t", a[i][j]);
109         }
110         fprintf(fout, "\n");
111     }
112 }
113
114 /* b[1...N]の入力 */
115 void input_vector(double *b, char c, FILE *fin, FILE *fout)
116 {
117     int i;
118
119     fprintf(fout, "ベクトル%c は次の通りです\n", c);
120     for(i = 1; i <= N; i++){
121         fscanf(fin, "%lf", &b[i]);
122         fprintf(fout, "%5.2f\t", b[i]);
123         fprintf(fout, "\n");
124     }
125 }
126
127 double **dmatrix(int nr1, int nr2, int n11, int n12)
128 {
129     int i, nrow, ncol;
130     double **a;
131
132     nrow = nr2 - nr1 + 1; /* 行の数 */
133     ncol = n12 - n11 + 1; /* 列の数 */
134
135     /* 行の確保 */
136     if((a = malloc(nrow * sizeof(double *))) == NULL){
137         printf("メモリが確保できません (行列 a)\n");
138         exit(1);
139     }
140     a = a - nr1; /* 行をずらす */
141
142     /* 列の確保 */
143     for(i = nr1; i <= nr2; i++) a[i] = malloc(ncol * sizeof(double));
144     for(i = nr1; i <= nr2; i++) a[i] = a[i] - n11; /* 列をずらす */
145
146     return (a);
147 }
148
149 void free_dmatrix(double **a, int nr1, int nr2, int n11, int n12)
150 {
151     int i;
152
153     /* メモリの解放 */
154     for(i = nr1; i <= nr2; i++) free((void *) (a[i] + n11));
155     free((void *) (a + nr1));
156 }
157
158 double *dvector(int i, int j)
159 {
160     double *a;
161
162     if((a = malloc( ((j - i + 1) * sizeof(double))) ) == NULL)
163     {
164         printf("メモリが確保できません (from dvector) \n");
165         exit(1);
166     }
167
168     return (a - i);
169 }
170
171 void free_dvector(double *a, int i)
172 {
173     free( (void *) (a + i) ); /* (void *) 型へのキャストが必要 */
174 }

```