

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4
5 #define N 4 /* N 次正方行列 */
6
7 /* 行列の入力 */
8 void input_matrix( double **a, char c, FILE *fin, FILE *fout);
9 /* ベクトルの入力 */
10 void input_vector( double *b, char c, FILE *fin, FILE *fout);
11 /* 行列の領域確保 */
12 double **dmatrix(int nr1, int nr2, int nl1, int nl2);
13 /* 行列の領域解放 */
14 void free_dmatrix(double **a, int nr1, int nr2, int nl1, int nl2);
15 /* ベクトル領域の確保 */
16 double *dvector(int i, int j);
17 /* 領域の解放 */
18 void free_dvector(double *a, int i);
19 /* 部分ピボット選択付きガウス消去法 */
20 double *gauss( double **a, double *b );
21
22 int main(void)
23 {
24     FILE *fin, *fout;
25     double **a, *b;
26     int i;
27
28     /* 行列およびベクトルの領域確保 */
29     a = dmatrix(1, N, 1, N); /* 行列 a[1...N][1...N] */
30     b = dvector(1,N); /* b[1...N] */
31
32     /* ファイルのオープン */
33     if ( (fin = fopen( "input.dat", "r")) == NULL )
34     {
35         printf("ファイルが見つかりません : input.dat \n");
36         exit(1);
37     }
38     if( (fout = fopen( "output.dat", "w")) == NULL )
39     {
40         printf("ファイルが作成できません : output.dat \n");
41         exit(1);
42     }
43
44     input_matrix( a, 'A', fin, fout ); /* 行列 A の入出力 */
45     input_vector( b, 'b', fin, fout ); /* ベクトル b の入出力 */
46     b = gauss( a, b ); /* ガウス消去法 */
47
48     /* 結果の出力 */
49     fprintf( fout, "Ax=b の解は次の通りです\n");
50     for( i = 1 ; i <= N ; i++)
51     {
52         fprintf(fout, "%f\n", b[i]);
53     }
54
55     fclose(fin); fclose(fout); /* ファイルのクローズ */
56
57     /* 領域の解放 */
58     free_dmatrix( a, 1, N, 1, N ); free_dvector( b, 1 );
59
60     return 0;
61 }
62
63 /* 部分ピボット選択付きガウス消去法 */
64 double *gauss( double **a, double *b )
65 {
66     int i, j, k, ip;
67     double alpha, tmp;
68     double amax, eps=pow(2.0, -50.0); /* eps = 2^(-50)とする */
69
70     for( k = 1; k <= N-1; k++)
71     {
72         /* ピボットの選択 */
73         amax = fabs(a[k][k]); ip = k;
74         for( i = k+1; i <= N; i++)
75         {
76             if ( fabs(a[i][k]) > amax )
77             {
78                 amax = fabs(a[i][k]); ip = i;
79             }
80         }
81         /* 正則性の判定 */
82         if ( amax < eps ) printf("入力した行列は正則ではない!!!\n");
83         /* 行交換 */
84         if ( ip != k )
85         {
86             for( j = k; j <= N; j++)
87             {
88                 tmp = a[k][j]; a[k][j]=a[ip][j]; a[ip][j]=tmp;
89             }
90             tmp = b[k] ; b[k]=b[ip]; b[ip]=tmp;
91         }
92         /* 前進消去 */
93         for( i = k+1; i <= N; i++)
94         {
95             alpha = - a[i][k]/a[k][k];
96             for( j = k+1; j <= N; j++)
97             {
98                 a[i][j] = a[i][j] + alpha * a[k][j];
99             }
100             b[i] = b[i] + alpha * b[k];
101         }
102     }
103
104     /* 後退代入 */
105     b[N] = b[N]/a[N][N];
106     for( k = N-1; k >= 1; k--)
107     {
108         tmp = b[k];
109         for( j = k+1; j <= N; j++)
110         {
111             tmp = tmp - a[k][j] * b[j];
112         }
113         b[k] = tmp/a[k][k];
114     }
115
116     return b;
117 }
118
119 /* a[1...N][1...N] の入力 */
120 void input_matrix(double **a, char c, FILE *fin, FILE *fout)
121 {
122     int i, j;
123
124     fprintf(fout, "行列%c は次の通りです\n", c);
125     for (i = 1; i <= N; i++)
126     {
127         for (j = 1; j <= N; j++)
128         {
129             fscanf(fin, "%lf", &a[i][j]);
130             fprintf(fout, "%5.2f\t", a[i][j]);
131         }
132         fprintf(fout, "\n");
133     }
134 }
135
136 /* b[1...N]の入力 */
137 void input_vector(double *b, char c, FILE *fin, FILE *fout)
138 {
139     int i;
140
141     fprintf(fout, "ベクトル%c は次の通りです\n", c);
142     for(i = 1; i <= N; i++){
143         fscanf(fin, "%lf", &b[i]);
144         fprintf(fout, "%5.2f\t", b[i]);
145         fprintf(fout, "\n");
146     }
147 }
148
149 double **dmatrix(int nr1, int nr2, int nl1, int nl2)
150 {
151     int i, nrow, ncol;
152     double **a;
153
154     nrow = nr2 - nr1 + 1; /* 行の数 */
155     ncol = nl2 - nl1 + 1; /* 列の数 */
156
157     /* 行の確保 */
158     if((a = malloc(nrow * sizeof(double *))) == NULL){
159         printf("メモリが確保できません (行列 a)\n");
160         exit(1);
161     }
162     a = a - nr1; /* ずれをずらす */
163
164     /* 列の確保 */
165     for(i = nr1; i <= nr2; i++) a[i] = malloc(ncol * sizeof(double));
166     for(i = nr1; i <= nr2; i++) a[i] = a[i] - nl1; /* 列をずらす */
167
168     return (a);
169 }
170
171 void free_dmatrix(double **a, int nr1, int nr2, int nl1, int nl2)
172 {
173     int i;
174
175     /* メモリの解放 */
176     for(i = nr1; i <= nr2; i++) free((void *) (a[i] + nl1));
177     free((void *) (a + nr1));
178 }
179
180 double *dvector(int i, int j)
181 {
182     double *a;
183
184     if((a = malloc( ((j - i + 1) * sizeof(double))) ) == NULL)
185     {
186         printf("メモリが確保できません (from dvector) \n");
187         exit(1);
188     }
189
190     return (a - i);
191 }
192
193 void free_dvector(double *a, int i)
194 {
195     free( (void *) (a + i) ); /* (void *) 型へのキャストが必要 */
196 }

```