

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 /* ベクトルの入力 */
5 void input_vector3( double *b, int m, int n, FILE *fin );
6 /* ベクトル領域の確保 */
7 double *dvector(int i, int j);
8 /* ベクトル領域の解放 */
9 void free_dvector(double *a, int i);
10 /* ラグランジュ補間 */
11 double lagrange( double *x, double *y, int m, int n, double xi );
12
13 int main(void)
14 {
15     FILE *fin, *fout;
16     double *x, *y, xi; /* x[i]は補間点 */
17     int n;
18
19     printf("データの個数を入力してください--->");
20     scanf("%d", &n);
21     n -= 1; /* データ数が n なので, n <- n-1として添字を0,1,...,nとする */
22
23     printf("補間点を入力してください--->");
24     scanf("%lf", &xi);
25
26     /* ベクトルの領域確保 */
27     x = dvector(0,n); /* x[0...n] */
28     y = dvector(0,n); /* y[0...n] */
29
30     /* ファイルのオープン */
31     if ( (fin = fopen( "input_lag.dat", "r")) == NULL )
32     {
33         printf("ファイルが見つかりません : input_lag.dat \n");
34         exit(1);
35     }
36     if( (fout = fopen( "output_lag.dat", "w")) == NULL )
37     {
38         printf("ファイルが作成できません : output_lag.dat \n");
39         exit(1);
40     }
41
42     input_vector3( x, 0, n, fin ); /* ベクトル x の入出力 */
43     input_vector3( y, 0, n, fin ); /* ベクトル y の入出力 */
44
45     printf("補間の結果は, P(%f)=%f\n", xi, lagrange(x,y,0,n,xi) );
46
47     /* グラフを描くために結果をファイルに出力 */
48     for( xi = x[0]; xi <= x[n]; xi += 0.01 )
49     {
50         fprintf(fout, "%f \t %f\n", xi, lagrange(x,y,0,n,xi) );
51     }
52     fclose(fin); fclose(fout); /* ファイルのクローズ */
53
54     /* 領域の解放 */
55     free_dvector( x, 0 ); free_dvector( y, 0 );
56
57     return 0;
58 }
59
60 /* ラグランジュ補間 */
61 double lagrange( double *x, double *y, int m, int n, double xi )
62 {
63     int i, k;
64     double pn = 0.0, li;
65
66     /* P_n(x) の計算 */
67     for ( i = m; i <= n ; i++ )
68     {
69         li = 1.0;
70         /* l_i(x) の計算 */
71         for( k = m; k <= n; k++ )
72         {
73             if( k != i ) li *= (xi - x[k]) / (x[i] - x[k]);
74         }
75         pn += li * y[i];
76     }
77
78     return pn;
79 }
80
81 /* b[m...n] の入力 */
82 void input_vector3( double *b, int m, int n, FILE *fin )
83 {
84     int i;
85     for( i = m ; i <= n ; i++)
86     {
87         fscanf(fin, "%lf", &b[i]);
88     }
89 }
90
91 double **dmatrix(int nr1, int nr2, int n11, int n12)
92 {
93     int i, nrow, ncol;
94     double **a;
95
96     nrow = nr2 - nr1 + 1; /* 行の数 */
97     ncol = n12 - n11 + 1; /* 列の数 */
98
99     /* 行の確保 */
100     if((a = malloc(nrow * sizeof(double *))) == NULL){
101         printf("メモリが確保できません (行列 a)\n");
102         exit(1);
103     }
104     a = a - nr1; /* 行をずらす */
105
106     /* 列の確保 */
107     for(i = nr1; i <= nr2; i++) a[i] = malloc(ncol * sizeof(double));
108     for(i = nr1; i <= nr2; i++) a[i] = a[i] - n11; /* 列をずらす */
109
110     return (a);
111 }
112
113 void free_dmatrix(double **a, int nr1, int nr2, int n11, int n12)
114 {
115     int i;
116
117     /* メモリの解放 */
118     for(i = nr1; i <= nr2; i++) free((void *)a[i] + n11);
119     free((void *)a + nr1);
120 }
121
122 double *dvector(int i, int j) /* a[i]~a[j] の領域を確保 */
123 {
124     double *a;
125
126     if ((a = malloc(((j - i + 1) * sizeof(double)))) == NULL)
127     {
128         printf("メモリが確保できません (from dvector) \n");
129         exit(1);
130     }
131
132     return (a - i);
133 }
134
135 void free_dvector(double *a, int i)
136 {
137     free((void *)a + i); /* (void *) 型へのキャストが必要 */
138 }
139
140 /* ベクトル a[m...n] と b[m...n] の内積を計算する */
141 double inner_product(int m, int n, double *a, double *b)
142 {
143     int i;
144     double s = 0.0;
145
146     for (i = m; i <= n; i++)
147         s += a[i] * b[i];
148
149     return s;
150 }

```