

```

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 /* 行列の領域確保 */
7 double **dmatrix(int nr1, int nr2, int nl1, int nl2);
8 /* 行列の領域解放 */
9 void free_dmatrix(double **a, int nr1, int nr2, int nl1, int nl2);
10 /* ベクトル領域の確保 */
11 double *dvector(int i, int j);
12 /* ベクトル領域の解放 */
13 void free_dvector(double *a, int i);
14 /* ニュートン補間 */
15 double newton_ip( double *x, double *y, int n, double xi );
16
17 int main(void)
18 {
19     FILE *fin, *fout;
20     double *x, *y, xi; /* xi は補間点 */
21     int n;
22
23     printf("データの個数を入力してください--->");
24     scanf("%d", &n);
25     n -= 1; /* データ数がnなので, n <- n-1 として添え字を 0,1,...,n とする */
26
27     printf("補間点を入力してください--->");
28     scanf("%lf", &xi);
29
30     /* ベクトル領域の確保 */
31     x = dvector(0,n); /* x[0...n] */
32     y = dvector(0,n); /* y[0...n] */
33
34     /* ファイルのオープン */
35     if ( (fin = fopen( "input_lag.dat", "r")) == NULL )
36     {
37         printf("ファイルが見つかりません : input_lag.dat \n");
38         exit(1);
39     }
40     if( (fout = fopen( "output_lag.dat", "w")) == NULL )
41     {
42         printf("ファイルが作成できません : output_lag.dat \n");
43         exit(1);
44     }
45
46     input_vector3( x, 0, n, fin ); /* ベクトル x の入出力 */
47     input_vector3( y, 0, n, fin ); /* ベクトル y の入出力 */
48
49     printf("補間の結果は, P(%f)=%f\n", xi, newton_ip(x,y,n,xi) );
50
51     /* グラフを描くために結果をファイルに出力 */
52     for( xi = x[0]; xi <= x[n]; xi += 0.01 )
53     {
54         fprintf(fout, "%f \t %f\n", xi, newton_ip(x,y,n,xi) );
55     }
56     fclose(fin); fclose(fout); /* ファイルのクローズ */
57
58     /* 領域の解放 */
59     free_dvector( x, 0 ); free_dvector( y, 0 );
60
61     return 0;
62 }
63
64 /* ニュートン補間 */
65 /* 添字は 0,1,...,n と仮定 */
66 double newton_ip( double *x, double *y, int n, double xi )
67 {
68     int i, j;
69     double pn = 0.0, li, **a;
70
71     a = dmatrix(0, n, 0, n);
72
73     for( i = 0; i <= n; i++) a[i][0] = y[i] ;
74
75     /* 差商の計算 */
76     for( j = 1; j <= n; j++)
77     {
78         for( i = 0; i <= n-j; i++ )
79         {
80             a[i][j] = ( a[i+1][j-1] - a[i][j-1] ) / ( x[i+j]-x[i] );
81         }
82     }
83
84     /* 補間の計算 */
85     pn = y[0]; li = 1.0;
86     for ( j = 1; j <= n; j++ )
87     {
88         li *= ( xi - x[j-1] );
89         pn += a[0][j] * li ;
90     }
91
92     free_dmatrix( a, 0, n, 0, n );
93
94     return pn;
95 }
96
97 /* b[m...n] の入力 */
98 void input_vector3(double *b, int m, int n, FILE *fin)
99 {
100     int i;
101     for (i = m; i <= n; i++)
102     {
103         fscanf(fin, "%lf", &b[i]);
104     }
105 }
106
107 double **dmatrix(int nr1, int nr2, int nl1, int nl2)
108 {
109     int i, nrow, ncol;
110     double **a;
111
112     nrow = nr2 - nr1 + 1; /* 行の数 */
113     ncol = nl2 - nl1 + 1; /* 列の数 */
114
115     /* 行の確保 */
116     if ((a = malloc(nrow * sizeof(double))) == NULL)
117     {
118         printf("メモリが確保できません (行列 a)\n");
119         exit(1);
120     }
121     a = a - nr1; /* 行をずらす */
122
123     /* 列の確保 */
124     for (i = nr1; i <= nr2; i++)
125         a[i] = malloc(ncol * sizeof(double));
126     for (i = nr1; i <= nr2; i++)
127         a[i] = a[i] - nl1; /* 列をずらす */
128
129     return (a);
130 }
131
132 void free_dmatrix(double **a, int nr1, int nr2, int nl1, int nl2)
133 {
134     int i;
135
136     /* メモリの解放 */
137     for (i = nr1; i <= nr2; i++)
138         free((void *) (a[i] + nl1));
139     free((void *) (a + nr1));
140 }
141
142 double *dvector(int i, int j) /* a[i]~a[j] の領域を確保 */
143 {
144     double *a;
145
146     if ((a = malloc(((j - i + 1) * sizeof(double)))) == NULL)
147     {
148         printf("メモリが確保できません (from dvector) \n");
149         exit(1);
150     }
151
152     return (a - i);
153 }
154
155 void free_dvector(double *a, int i)
156 {
157     free((void *) (a + i)); /* (void *) 型へのキャストが必要 */
158 }
159
160 /* ベクトル a[m...n] と b[m...n] の内積を計算する */
161 double inner_product(int m, int n, double *a, double *b)
162 {
163     int i;
164     double s = 0.0;
165
166     for (i = m; i <= n; i++)
167         s += a[i] * b[i];
168
169     return s;
170 }

```