

```

1 // 8_1.cpp
2 #include<stdio.h>
3 #include<stdlib.h>
4
5 const int N = 30;           // 読み込むデータの最大値
6 const int Div_Num = 100;    // 分割数を定義
7 const int Data_Num = 101;   // データ数を定義
8
9 typedef struct{ // 構造体を定義
10     double x[N];
11     double y[N];
12 } point;
13
14 // ファイルからデータを読み込む関数
15 int read_data(FILE* ifp, point& p){
16     int num_of_data = 0;    // データ数を格納する変数
17     while(fscanf(ifp, "%lf%lf", &p.x[num_of_data], &p.y[num_of_data]) != EOF){
18         num_of_data++;
19     }
20     printf("Number of data: %d\n\n", num_of_data);
21     return num_of_data;     // データ数を返す
22 }
23
24 // 刻み幅を算出する関数
25 double division(point p, int data, double* max, double* min){
26     *max = p.x[0];
27     *min = p.x[0];
28     double div = 0.0; // 初期化
29
30     for(int i=0; i<data; i++){ // データの中から最大値と最小値を見つける
31         if(*max < p.x[i]) {
32             *max = p.x[i];
33         }
34         if(*min > p.x[i]) {
35             *min = p.x[i];
36         }
37     }
38
39     div = (*max - *min) / (double)Div_Num; // 刻み幅を計算
40     return div; // 刻み幅を返す
41 }
42
43 // ラグランジュ補間
44 double lagrange(point p, int data, double X){
45     double ans = 0.0; // 補間値を格納する変数
46
47     for(int i=0; i<data; i++){
48         double tmp = 1.0; // 計算途中の値を格納する変数
49         for(int j=0; j<data; j++){
50             if(i!=j) tmp *= (X - p.x[j]) / (p.x[i] - p.x[j]);
51         }
52         ans += tmp * p.y[i];
53     }
54     return ans;
55 }
56
57 // ニュートン補間
58 double newton(point p, int data, double X){
59     double ans = 0.0; // 補間値を格納する変数
60     double w[N] = {}; // 作業用配列
61
62     for(int i=0; i<data; i++){
63         double tmp = 1.0; // 計算途中の値を格納する変数
64         w[i] = p.y[i];
65         for(int j=i-1; j>=0; j--){
66             w[j] = (w[j+1] - w[j]) / (p.x[i] - p.x[j]);
67             tmp *= X - p.x[j];
68         }
69         ans += w[0] * tmp;
70     }
71     return ans;
72 }
73
74 // データを表示する関数
75 void print_data(point p, int num_of_data, double x_min, double div){
76     double x[Data_Num] = {}; // xの値

```

```

77 double y_lag[Data_Num] = {}; // ラグランジュ補間により求めたy
78 double y_new[Data_Num] = {}; // ニュートン補間により求めたy
79
80 for(int i=0; i<Data_Num; i++){
81     x[i] = x_min + div*i; // xを計算
82     // ラグランジュ補間
83     y_lag[i] = lagrange(p, num_of_data, x[i]);
84     // ニュートン補間
85     y_new[i] = newton(p, num_of_data, x[i]);
86     // 画面に表示
87     printf("x = %8.4f, y = %10.6f (Lagrange), y = %10.6f (Newton)\n", x[i], y_lag[i], y_new[i]);
88 }
89 }
90
91 // メイン関数
92 int main(int argc, char* argv[]){
93     if(argc!=2){ // コマンドライン引数が2つでなければエラー
94         fprintf(stderr, "usage: %s inputfile\n", argv[0]);
95         exit(1);
96     }
97
98     FILE* ifp = NULL;
99     const char* ifile = argv[1]; // 入力ファイル名
100
101     if((ifp=fopen(ifile,"rt"))==NULL){ // 入力ファイルを開く
102         fprintf(stderr,"Can't open file %s\n", ifile); // 開けなかったらエラー
103         exit(2);
104     }
105
106     point p = {}; // 構造体の初期化
107     double x_max = 0.0; // データの最大値の初期化
108     double x_min = 0.0; // データの最小値の初期化
109     int num_of_data = read_data(ifp, p); // データ点の個数を格納
110     double div = division(p, num_of_data, &x_max, &x_min); // 刻み幅を格納
111
112     print_data(p, num_of_data, x_min, div); // 補間結果を表示
113
114     fclose(ifp);
115
116     return 0;
117 }

```