

```

1 /* Fast Fourier Transform with structured complex type
2    by T. Hirata, 2002, modified by S. Tanaka, 2006 */
3 #include <math.h>
4 #include <stdio.h>
5 #include <time.h>
6
7 #define twoPI  3.1415926535*2.0
8 #define N     512
9
10 typedef struct {
11     double real;
12     double imag;
13 } complex;
14
15 FILE *fpW;
16 FILE *fopen();
17
18 complex cPlus      (complex zz1, complex zz2);
19 complex cMinus     (complex zz1, complex zz2);
20 complex cMultiply  (complex zz1, complex zz2);
21 complex twiddleFactor (int n, int k);
22 void butterfly    (int n, complex x[], complex y[], complex z[]);
23 void shuffle      (int n, complex x[], complex y[], complex z[]);
24 void fft          (int n, complex x[]);
25
26 void main( void )
27 {
28     int t, fn;
29     complex data[N];
30     double power;
31     double start, finish;
32
33     for (t=0; t<N; t++) { /* データの生成 */
34         data[t].real = 0.5 + 1.0 * cos(2*twoPI*t/N)
35                        - 1.0 * cos(3*twoPI*t/N)
36                        + 2.0 * sin(4*twoPI*t/N);
37         data[t].imag = 0.0;
38     }
39
40     start=clock();
41     fft(N, data);
42     finish=clock();
43
44     fpW = fopen("fft-complex.txt", "w");
45     fprintf(fpW, "          高速フーリエ変換\n");
46     fprintf(fpW, "t, fn      Re(F(fn))      Im(F(fn))      振幅\n");
47
48     for (fn=0; fn<N; fn++) {
49         power = data[fn].real*data[fn].real + data[fn].imag*data[fn].imag;
50         fprintf(fpW, "%3d %14.6f %14.6f %14.6f \n",
51             fn, data[fn].real, data[fn].imag, power);
52     }
53     printf("Elapse time: %f seconds!\n", (double)(finish-start)/CLOCKS_PER_SEC);
54     printf("Result written to 'fft-complex.txt.'\n");
55 }
56
57
58 complex cPlus (complex zz1, complex zz2)
59 {
60     complex res; res.real=0.0; res.imag=0.0;
61     res.real = zz1.real + zz2.real;
62     res.imag = zz1.imag + zz2.imag;
63     return res;
64 }
65
66 complex cMinus (complex zz1, complex zz2)

```

```

67 {
68     complex res; res.real=0.0; res.imag=0.0;
69     res.real = zz1.real - zz2.real;
70     res.imag = zz1.imag - zz2.imag;
71     return res;
72 }
73
74 complex cMultiply (complex zz1, complex zz2)
75 {
76     complex res; res.real=0.0; res.imag=0.0;
77     res.real = zz1.real*zz2.real - zz1.imag*zz2.imag;
78     res.imag = zz1.real*zz2.imag + zz1.imag*zz2.real;
79     return res;
80 }
81
82 complex twiddleFactor(int n, int k)
83 {
84     complex w;
85     w.real = cos(twoPI*k/n);
86     w.imag = -sin(twoPI*k/n);
87     return w;
88 }
89
90 void butterfly(int n, complex x[], complex y[], complex z[])
91 {
92     int k; complex tmp;
93     for (k=0; k <= n/2-1; k++) {
94         y[k] = cPlus (x[k], x[k+n/2]);
95         tmp = cMinus (x[k], x[k+n/2]);
96         z[k] = cMultiply (tmp, twiddleFactor(n,k));
97     }
98 }
99
100 void shuffle(int n, complex x[], complex y[], complex z[])
101 {
102     int k;
103     for (k=0; k <= n/2-1; k++) {
104         x[2*k] = y[k];
105         x[2*k+1] = z[k];
106     }
107 }
108
109 void fft(int n, complex x[])
110 {
111     complex y[N], z[N];
112     if (n > 1) {
113         butterfly (n, x, y, z);
114         fft (n/2, y);
115         fft (n/2, z);
116         shuffle (n, x, y, z);
117     }
118 }

```