```c
/* Fast Fourier Transform with structured complex type
   by T. Hirata, 2002, modified by S. Tanaka, 2006 */
#include <math.h>
#include <stdio.h>
#include <time.h>

#define twoPI   3.1415926535*2.0
#define N   512

typedef struct {
    double real;
    double imag;
} complex;

FILE *fpW;
FILE *fopen();

complex cPlus         (complex zz1, complex zz2);
complex cMinus        (complex zz1, complex zz2);
complex cMultiply     (complex zz1, complex zz2);
complex twiddleFactor (int n, int k);
void    butterfly     (int n, complex x[], complex y[], complex z[]) ;
void    shuffle       (int n, complex x[], complex y[], complex z[]);
void    fft           (int n, complex x[]) ;

void main( void )
{
    int    t, fn;
    complex data[N];
    double  power;
    double  start, finish;

    for (t=0; t<N; t++) { /* データの生成 */
        data[t].real = 0.5  + 1.0 * cos(2*twoPI*t/N)
                            - 1.0 * cos(3*twoPI*t/N)
                            + 2.0 * sin(4*twoPI*t/N);
        data[t].imag = 0.0;
    }

    start=clock();
    fft(N, data);
    finish=clock();

    fpW = fopen("fft-complex.txt", "w");
    fprintf(fpW, "              高速フーリエ変換¥n");
    fprintf(fpW, "t,fn       Re(F(fn))       Im(F(fn))       振幅¥n");

    for (fn=0; fn<N; fn++) {
        power = data[fn].real*data[fn].real + data[fn].imag*data[fn].imag;
        fprintf(fpW," %3d %14.6f %14.6f %14.6f ¥n",
            fn, data[fn].real, data[fn].imag, power);
    }
    printf("Elapse time: %f seconds!¥n", (double)(finish-start)/CLOCKS_PER_SEC);
    printf("Result written to 'fft-complex.txt.'¥n");
}


complex cPlus (complex zz1, complex zz2)
{
   complex res; res.real=0.0; res.imag=0.0;
   res.real = zz1.real + zz2.real;
   res.imag = zz1.imag + zz2.imag;
   return res;
}

complex cMinus (complex zz1, complex zz2)
```

```c
{
   complex res; res.real=0.0; res.imag=0.0;
   res.real = zz1.real - zz2.real;
   res.imag = zz1.imag - zz2.imag;
   return res;
}

complex cMultiply (complex zz1, complex zz2)
{
   complex res; res.real=0.0; res.imag=0.0;
   res.real = zz1.real*zz2.real - zz1.imag*zz2.imag;
   res.imag = zz1.real*zz2.imag + zz1.imag*zz2.real;
   return res;
}

complex twiddleFactor(int n, int k)
{
   complex w;
   w.real =  cos(twoPI*k/n);
   w.imag = -sin(twoPI*k/n);
   return w;
}

void butterfly(int n, complex x[], complex y[], complex z[])
{
   int k; complex tmp;
   for (k=0; k <= n/2-1; k++) {
    y[k] = cPlus  (x[k], x[k+n/2]);
    tmp  = cMinus (x[k], x[k+n/2]);
    z[k] = cMultiply (tmp, twiddleFactor(n,k));
   }
}

void shuffle(int n, complex x[], complex y[], complex z[])
{
   int k;
   for (k=0; k <= n/2-1; k++) {
    x[2*k]   = y[k];
    x[2*k+1] = z[k];
   }
}

void fft(int n, complex x[])
{
   complex y[N], z[N];
   if (n > 1) {
    butterfly (n,x,y,z);
    fft (n/2,y);
    fft (n/2,z);
    shuffle   (n,x,y,z);
   }
}
```