```
#include <stdio.h>
#include <stdint.h>

////////////////////////////////////////////////////////////
//  Filter Code Definitions
////////////////////////////////////////////////////////////

// maximum number of inputs that can be handled
// in one function call
#define MAX_INPUT_LEN   80
// maximum length of filter than can be handled
#define MAX_FLT_LEN     63
// buffer to hold all of the input samples
#define BUFFER_LEN      (MAX_FLT_LEN - 1 + MAX_INPUT_LEN)

// array to hold input samples
double insamp[ BUFFER_LEN ];

// FIR init
void firFloatInit( void )
{
    memset( insamp, 0, sizeof( insamp ) );
}

// the FIR filter function
void firFloat( double *coeffs, double *input, double *output,
       int length, int filterLength )
{
    double acc;     // accumulator for MACs
    double *coeffp; // pointer to coefficients
    double *inputp; // pointer to input samples
    int n;
    int k;

    // put the new samples at the high end of the buffer
    memcpy( &insamp[filterLength - 1], input,
            length * sizeof(double) );

    // apply the filter to each input sample
    for ( n = 0; n < length; n++ ) {
        // calculate output n
        coeffp = coeffs;
        inputp = &insamp[filterLength - 1 + n];
        acc = 0;
        for ( k = 0; k < filterLength; k++ ) {
            acc += (*coeffp++) * (*inputp--);
        }
        output[n] = acc;
    }
    // shift input samples back in time for next time
    memmove( &insamp[0], &insamp[length],
            (filterLength - 1) * sizeof(double) );

}
```

```
////////////////////////////////////////////////////////////
//  Test program
////////////////////////////////////////////////////////////

// bandpass filter centred around 1000 Hz
// sampling rate = 8000 Hz

#define FILTER_LEN  63
double coeffs[ FILTER_LEN ] =
{
  -0.0448093,  0.0322875,   0.0181163,   0.0087615,   0.0056797,
   0.0086685,  0.0148049,   0.0187190,   0.0151019,   0.0027594,
  -0.0132676, -0.0232561,  -0.0187804,   0.0006382,   0.0250536,
   0.0387214,  0.0299817,   0.0002609,  -0.0345546,  -0.0525282,
  -0.0395620,  0.0000246,   0.0440998,   0.0651867,   0.0479110,
   0.0000135, -0.0508558,  -0.0736313,  -0.0529380,  -0.0000709,
   0.0540186,  0.0766746,   0.0540186,  -0.0000709,  -0.0529380,
  -0.0736313, -0.0508558,   0.0000135,   0.0479110,   0.0651867,
   0.0440998,  0.0000246,  -0.0395620,  -0.0525282,  -0.0345546,
   0.0002609,  0.0299817,   0.0387214,   0.0250536,   0.0006382,
  -0.0187804, -0.0232561,  -0.0132676,   0.0027594,   0.0151019,
   0.0187190,  0.0148049,   0.0086685,   0.0056797,   0.0087615,
   0.0181163,  0.0322875,  -0.0448093
};

void intToFloat( int16_t *input, double *output, int length )
{
    int i;

    for ( i = 0; i < length; i++ ) {
        output[i] = (double)input[i];
    }
}

void floatToInt( double *input, int16_t *output, int length )
{
    int i;

    for ( i = 0; i < length; i++ ) {
        // add rounding constant
        input[i] += 0.5;
        // bound the values to 16 bits
        if ( input[i] > 32767.0 ) {
            input[i] = 32767.0;
        } else if ( input[i] < -32768.0 ) {
            input[i] = -32768.0;
        }
        // convert
        output[i] = (int16_t)input[i];
    }
}
```

```
// number of samples to read per loop
#define SAMPLES   80

int main( void )
{
    int size;
    int16_t input[SAMPLES];
    int16_t output[SAMPLES];
    double floatInput[SAMPLES];
    double floatOutput[SAMPLES];
    FILE   *in_fid;
    FILE   *out_fid;

    // open the input waveform file
    in_fid = fopen( "input.pcm", "rb" );
    if ( in_fid == 0 ) {
        printf("couldn't open input.pcm");
        return;
    }

    // open the output waveform file
    out_fid = fopen( "outputFloat.pcm", "wb" );
    if ( out_fid == 0 ) {
        printf("couldn't open outputFloat.pcm");
        return;
    }

    // initialize the filter
    firFloatInit();

    // process all of the samples
    do {
        // read samples from file
        size = fread( input, sizeof(int16_t), SAMPLES, in_fid );
        // convert to doubles
        intToFloat( input, floatInput, size );
        // perform the filtering
        firFloat( coeffs, floatInput, floatOutput, size,
               FILTER_LEN );
        // convert to ints
        floatToInt( floatOutput, output, size );
        // write samples to file
        fwrite( output, sizeof(int16_t), size, out_fid );
    } while ( size != 0 );

    fclose( in_fid );
    fclose( out_fid );

    return 0;
}
```