```
#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
int16_t insamp[ BUFFER_LEN ];

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

// the FIR filter function
void firFixed( int16_t *coeffs, int16_t *input, int16_t *output,
       int length, int filterLength )
{
    int32_t acc;     // accumulator for MACs
    int16_t *coeffp; // pointer to coefficients
    int16_t *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(int16_t) );

    // apply the filter to each input sample
    for ( n = 0; n < length; n++ ) {
        // calculate output n
        coeffp = coeffs;
        inputp = &insamp[filterLength - 1 + n];
        // load rounding constant
        acc = 1 << 14;
        // perform the multiply-accumulate
        for ( k = 0; k < filterLength; k++ ) {
            acc += (int32_t)(*coeffp++) * (int32_t)(*inputp--);
        }
        // saturate the result
        if ( acc > 0x3fffffff ) {
            acc = 0x3fffffff;
        } else if ( acc < -0x40000000 ) {
            acc = -0x40000000;
        }
        // convert from Q30 to Q15
        output[n] = (int16_t)(acc >> 15);
    }
    // shift input samples back in time for next time
    memmove( &insamp[0], &insamp[length],
            (filterLength - 1) * sizeof(int16_t) );

}

//////////////////////////////////////////////////////////////
//  Test program
//////////////////////////////////////////////////////////////

// bandpass filter centred around 1000 Hz
// sampling rate = 8000 Hz
// gain at 1000 Hz is about 1.13

#define FILTER_LEN  63
int16_t coeffs[ FILTER_LEN ] =
{
 -1468, 1058,   594,   287,    186,  284,   485,   613,
   495,   90,  -435,  -762,   -615,   21,   821,  1269,
   982,    9, -1132, -1721,  -1296,    1,  1445,  2136,
  1570,    0, -1666, -2413,  -1735,   -2,  1770,  2512,
  1770,   -2, -1735, -2413,  -1666,    0,  1570,  2136,
  1445,    1, -1296, -1721,  -1132,    9,   982,  1269,
   821,   21,  -615,  -762,   -435,   90,   495,   613,
   485,  284,   186,   287,    594, 1058, -1468
};

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

int main( void )
{
    int size;
    int16_t input[SAMPLES];
    int16_t output[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( "outputFixed.pcm", "wb" );
    if ( out_fid == 0 ) {
        printf("couldn't open outputFixed.pcm");
        return;
    }

    // initialize the filter
    firFixedInit();
    // process all of the samples
    do {
        // read samples from file
        size = fread( input, sizeof(int16_t), SAMPLES, in_fid );
        // perform the filtering
        firFixed( coeffs, input, output, size, FILTER_LEN );
        // write samples to file
        fwrite( output, sizeof(int16_t), size, out_fid );
    } while ( size != 0 );

    fclose( in_fid );
    fclose( out_fid );

    return 0;
}
```