```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <sf.h>
 4: #include <sfmisc.h>
 5:
 6: #include "fm.h"
 7: #include "cat.h"
 8:
 9: int main(void)
10: {
11:     int nchan = 1,          // number of channels (1)
12:         sr = 44100 ;        // samples per second
13:
14:     int i, nTones = 5 ;     // counter, total number of tones
15:
16:     double dB = 90 ;        // amplitude of each tone
17:
18:     double Carrier[] = {100, 200, 300, 400, 500} ;  // carrier freqs
19:     double Modulator[] = {200, 200, 200, 200, 200} ;// mod freqs
20:     double PD[] = {10, 20, 30, 40, 50} ;            // peak deviation
21:
22:     double dur = 5,         // duration of each tone
23:            startTime,       // start time for each tone
24:            totalDur ;       // total duration of the tones
25:
26:     short *tone, *output ;  // arrays for each tone, and the output
27:
28:     totalDur = 0 ;          // compute the total duration of the tones
29:     for ( i = 0 ; i < nTones ; i++ )
30:         totalDur += dur ;
31:
32:     // allocate memory for each tone, and for the output
33:     tone = (short *)Malloc(dur * sr * sizeof(short)) ;
34:     output = (short *)Malloc(totalDur * sr * sizeof(short)) ;
35:
36:     // create the tones
37:     startTime = 0 ;
38:     for ( i = 0 ; i < nTones ; i++ ) {
39:         // create one FM tone
40:         fm(tone, dur, sr, dB, Carrier[i], Modulator[i], PD[i]) ;
41:         // concatenate the sequence
42:         concatenate(output, tone, startTime, dur, sr) ;
43:         // increment the start time
44:         startTime = startTime + dur ;
45:     }
46:
47:     // save the set of tones
48:     sfsave("fmsounds.wav", output, totalDur, sr, nchan) ;
49:
50:     Free(tone) ;    // release memory
51:     Free(output) ;
52:
53:     exit(EXIT_SUCCESS) ;
54: }
55:
56:
57:
58:
59:
60:
61:
62:
63:
64:
```

```
 65:
 66: #include <stdio.h>
 67: #include <stdlib.h>
 68: #include <math.h>
 69: #include <sf.h>
 70: #include <sfmisc.h>
 71:
 72: #include "env.h"
 73:
 74: void fm(short sound[], double dur, int sr, double dB,
 75:         double carrier,
 76:         double modulator,
 77:         double peakDeviation)
 78: {
 79:     double cPhase, mPhase,      // carrier and modulator phase
 80:         cInc, mInc,             // increments for the two
 81:         amp ;                   // linear amplitude
 82:     int t, samples ;            // counter for time, and samples
 83:
 84:     static int counter ;        // function counter
 85:
 86:     // arrays for the envelope: T = time, A = amplitude
 87:     double T[] = {0.0, 0.2, 0.4, 0.6, 0.8, 1.0} ;
 88:     double A[] = {0.0, 1.0, 0.75, 0.5, 1.0, 0.0} ;
 89:     int nPoints = 6 ;   // number of points in the envelope
 90:     double *env ;       // array for the envelope
 91:
 92:     // print out information to the screen
 93:     fprintf(stderr,
 94:         "\tFM(%d): C = %g  M = %g  pd = %g  dur = %g  dB = %g\n",
 95:             counter++, carrier, modulator, peakDeviation,
 96:             dur, dB) ;
 97:
 98:     samples = dur * sr ;            // translate seconds into samples
 99:     amp = pow(10.0, dB/20.0) / 2 ;  // translate dB into linear
100:
101:     cPhase = 0 ;        // initialize the phase for both
102:     mPhase = 0 ;        // carrier and modulator
103:
104:     cInc = ( 2.0 * M_PI * carrier ) / sr ;  // increments for C & M
105:     mInc = ( 2.0 * M_PI * modulator ) / sr ;
106:
107:     env = linearEnv(dur, sr, T, A, nPoints) ;   // make envelope
108:
109:     for ( t = 0 ; t < samples ; t++ ) {     // FM loop
110:         sound[t] = amp * sin(cPhase
111:                 + (peakDeviation * env[t]) * sin(mPhase)) ;
112:         cPhase += cInc ;        // increment cPhase & mPhase
113:         mPhase += mInc ;
114:     }
115:
116:     Free(env) ;     // release memory
117:
118:     return ;
119: }
120:
```