サウンドプログラミング

[要旨]

音楽データを作成するプログラム。音符や音階を設定すると、CD と同じ形式である wav ファイルを生成する。また、音圧、音色、テンポ、キーの調節を可能とした。このプログラムにより、山崎まさよしさんの楽曲「One more time One more chance」のメロディーと伴奏のみの音楽データを作成した。

[プログラムの概要]

① 音符と音階の設定

メロディー（Vocal）用と伴奏（Base）用の配列を用意し、音符と音階に対応する数値を設定する。

② テンポやキーの設定

それぞれの変数に数値を代入することで、曲のテンポやキー、音圧、音の減衰率を設定する。

1. テンポ（tempo）

テンポ（1 分間に 4 分音符を演奏する回数）を設定する。

2. キー（key）

Vocal と Base に設定した音階に対して、キー（半音）をいくつ上下するか設定する。

3. 音圧（Amp）

曲の基準となる音の大きさを設定する。

4. 減衰率（decay rate）

音が減衰する度合いを設定する。

5. 音の大きさの飽和値（saturation）

減衰した音が飽和するときの音の大きさを設定する。

③ 音色の設定

音色は波形によって決定するため、演奏したい波形の周期関数を実フーリエ級数展開し、その係数（a_0、a_n、b_n）をプログラムに設定する。

④ 音階に対応する周波数を計算する。

「音階」「音階に対応する数値（d）」「音階の周波数（f）」には以下の関係式が成り立つ。

$$d = 69 + 12log_2\left(\frac{f}{440\ Hz}\right) \qquad \therefore \quad f = 2^{(d-69)/12} \times 440\ [Hz]$$

この式を用いて、それぞれに対応する値を求めた結果の一部を表 1 に示した。無音の場合、d は 0 とする。プログラム内で d に対応する周波数 f [Hz]を計算し、配列 f [d] に格納する。

表 1. 「音階」「音階に対応する数値（d)」「音階の周波数（f)」の対応表

音階	ド	ド#	レ	レ#	ミ	ファ	ファ#	ソ	ソ#	ラ	ラ#	シ	ド
d	60	61	62	63	64	65	66	67	68	69	70	71	72
f [Hz]	261.6	277.2	293.7	311.1	329.6	349.2	370.0	392.0	415.3	440.0	466.2	493.9	523.3

⑤ Vocal と Base の音データの値を計算する。

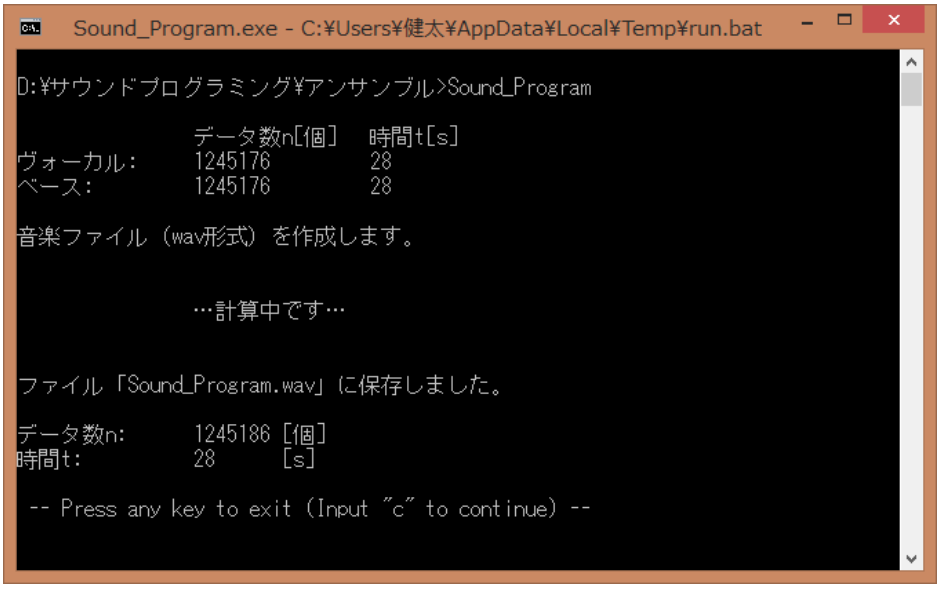
音データは、n = 0, 1, 2, 3, 4, 5 … とその n における値 pcm1.s[n]によって形成される。
ここで、標本化周波数を fs = 44.1 [kHz] としたため、標本化周期は $ts = \frac{1}{44.1k} \fallingdotseq 22.7\ [\mu s]$ である。
よって、22.7 [μs]の間隔で音データを記録し、時間 t =n×ts [s]における音データが pcm1.s[n]となる。

Vocal と Base の音符と音階を設定した配列 v[i][j]、b[i][j] を参照し、音符に対応する分の n を変化させ、そのときの音階の周波数で音データの値を計算し、pcm1.s[n]に格納する。

[動作の内容]

プログラムの概要で述べた ①音符と音階、および ②テンポやキー などの値を設定して実行すると、音データを計算して wav ファイルを生成する。以下に、実行画面と生成された wav ファイルを再生したときの画面イメージを示す。

プログラムの実行画面



再生画面

[作成した音楽のスコア]

作成した音楽「One more time One more chance」のスコアを以下に示す。

小節	拍子	歌詞	Vocal		Base						
			音階	d	コード名	6弦	5弦	4弦	3弦	2弦	1弦
1	1			0	G	43 ソ	47 シ	50 レ	55 ソ	59 シ	67 ソ
				0							
	2	い	ミ	76							
		つ	ミ	76							
	3	で	レ	74							
		も	ド#	73							
	4	ー	ー	–							
		さ	レ	74							
2	1	ー	ー	–	Gmaj7	43 ソ	47 シ	50 レ	55 ソ	59 シ	66 ファ#
		が	レ	74							
	2	し	ド#	73							
		て	ド#	73							
	3	し	ラ	69							
		ま	ミ	64							
	4	う	ミ	64							
		ー	ー	–							
3	1			0	F	41 ファ	48 ド	53 ファ	57 ラ	60 ド	65 ファ
				–							
	2	ど	ミ	76							
		っ	ミ	76							
	3	か	レ	74							
		に	ド#	73							
	4	ー	ー	–							
		き	レ	74							
4	1	ー	ー	–	E	40 ミ	47 シ	52 ミ	56 ソ#	59 シ	64 ミ
		み	レ	74							
	2	ー	ー	–							
		の	ド#	73							
	3	え	ラ#	70							
		が	シ	71							
	4	お	ド#	73							
		を	シ	71							
5	1			0	Am	0	45 ラ	52 ミ	57 ラ	60 ド	64 ミ
		きゅ	レ	74							
	2	う	レ	74							
		こ	レ	74							
	3	う	ミ	76							
		ま	レ	74							
	4	ち	ド#	73							
		の	レ	74							
6	1			0	AmM7	0	45 ラ	52 ミ	56 ソ#	60 ド	64 ミ
		ふ	レ	74							
	2	み	レ	74							
		き	レ	74							
	3	り	ミ	76							
		あ	レ	74							
	4	た	ド#	73							
		り	レ	74							
7	1			0	Am7	0	45 ラ	52 ミ	55 ソ	60 ド	64 ミ
		こ	シ	71							
	2	ん	シ	71							
		な	シ	71							
	3	と	シ	71							
		こ	シ	71							
	4	に	ド#	73							
		い	レ	74							
8	1	る	ファ#	78	D	0	0	50 レ	57 ラ	62 レ	66 ファ#
		は	ミ	76							
	2	ず	ミ	76							
		も	ド#	73							
	3	な	ミ	76							
		い	ミ	76							
	4	の	ファ#	78							
		に	ミ	76							

[スコアを格納する配列の構造]

スコアを参考に、Vocal、Base の音符と音階を 2 次元配列 v[i][j]、b[i][j]に設定する。

・Vocal の配列 v[i][j]

Vocal の配列は、2 行 j 列から成り立ち、j は作成したい音楽の音符数である。

0 行目に音符、1 行目に音階 d を格納し、それを j 列分設定する。

	い	つ	で	もー	さー	が	し
音符	v[0][0] 4	v[0][1] 8	v[0][2] 8	v[0][3] 8	v[0][4] 4	v[0][5] 4	v[0][6] 8	v[0][7] 8
音階	v[1][0] 0	v[1][1] 76	v[1][2] 76	v[1][3] 74	v[1][4] 73	v[1][5] 74	v[1][6] 74	v[1][7] 73

図 1. Vocal の配列 v[i][j] のイメージ図

・Base の配列 b[i][j]

Base の配列は、7 行 j 列から成り立ち、j は作成したい音楽の音符数である。

0 行目に音符、1~6 行目に音階を設定し、それを j 列分設定する。ここで、1~6 は弦の番号に対応している。(※ Base という名前であるが、楽器はギターである)

	1小節	2小節	3小節	4小節	5小節	6小節	7小節	8小節
音符	b[0][0] 1	b[0][1] 1	b[0][2] 1	b[0][3] 1	b[0][4] 1	b[0][5] 1	b[0][6] 1	b[0][7] 1
1弦	b[1][0] 67	b[1][1] 66	b[1][2] 65	b[1][3] 64	b[1][4] 64	b[1][5] 64	b[1][6] 64	b[1][7] 66
2弦	b[2][0] 59	b[2][1] 59	b[2][2] 60	b[2][3] 59	b[2][4] 60	b[2][5] 60	b[2][6] 60	b[2][7] 62
3弦	b[3][0] 55	b[3][1] 55	b[3][2] 57	b[3][3] 56	b[3][4] 57	b[3][5] 56	b[3][6] 55	b[3][7] 57
4弦	b[4][0] 50	b[4][1] 50	b[4][2] 53	b[4][3] 52	b[4][4] 52	b[4][5] 52	b[4][6] 52	b[4][7] 50
5弦	b[5][0] 47	b[5][1] 47	b[5][2] 48	b[5][3] 47	b[5][4] 45	b[5][5] 45	b[5][6] 45	b[5][7] 0
6弦	b[6][0] 43	b[6][1] 43	b[6][2] 41	b[6][3] 40	b[6][4] 0	b[6][5] 0	b[6][6] 0	b[6][7] 0

図 2. Base の配列 b[i][j] のイメージ図

・音符の設定

設定する音符の値は音符の名前に対応しているため、各配列の 0 行目に 1~16 の値を設定する。

表 2. 音符と設定する値の対応表

音符	全音符	2分音符	4分音符	8分音符	16分音符
設定値	1	2	4	8	16

・音階の設定

表 1 およびスコアより、音階に対応する値「d」、を各配列の 1 行目 (Vocal) および 1~6 行目 (Base) に設定する。

[音データの波形]

音データの値を Excel ファイルへ出力する文をプログラムに追加し、作成した音データの波形を示す。

Vocal のみ、Base のみ、それらを重ね合わせた Mix（作成した音データ）の 3 つに分け、波形の全体像（2 秒間）と拡大像（6.8 ミリ秒間）を比較した。

また、音色の設定において、Vocal は三角波、Base はノコギリ波のフーリエ級数展開した各係数をプログラムに設定した。

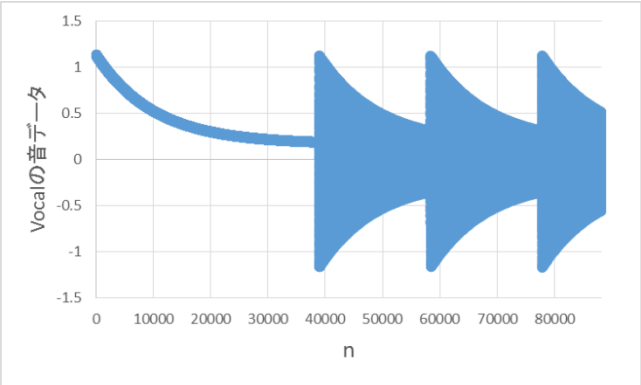


図 3. Vocal：2 秒間（0 ~ n ~ 88200）

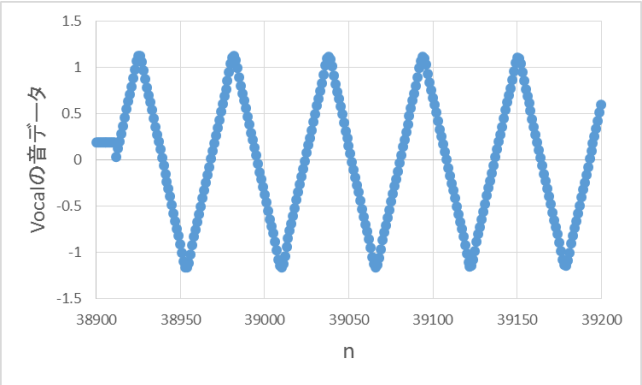


図 4. Vocal：6.8 ミリ秒間（38900 ~ n ~ 39200）

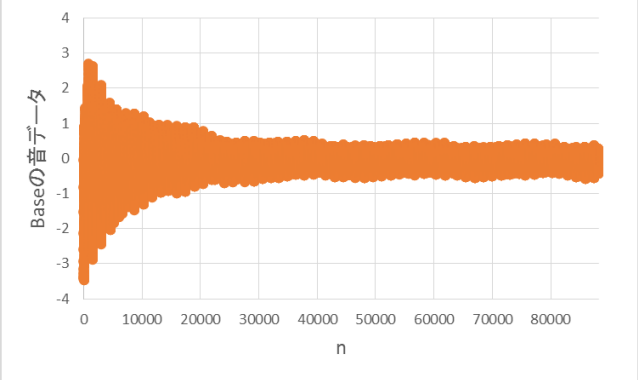


図 5. Base：2 秒間（0 ~ n ~ 88200）

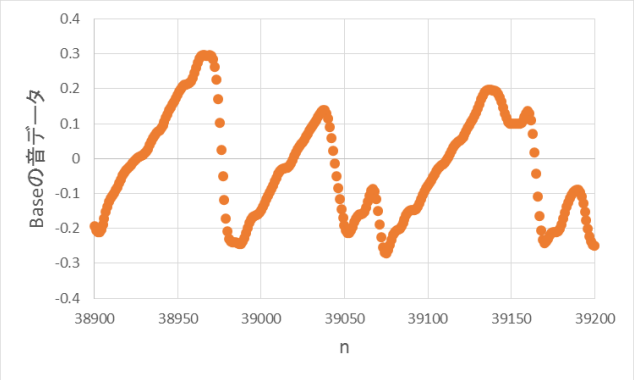


図 6. Base：6.8 ミリ秒間（38900 ~ n ~ 39200）

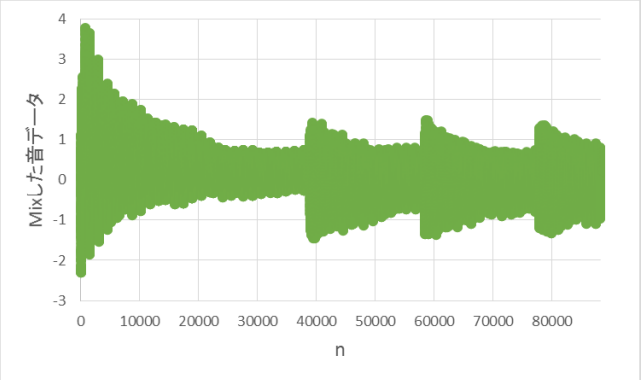


図 7. Mix：2 秒間（0 ~ n ~ 88200）

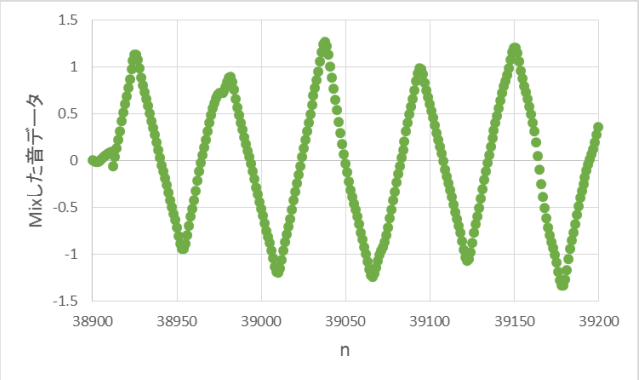


図 8. Mix：6.8 ミリ秒間（38900 ~ n ~ 39200）

[問題点・工夫点]

sin 波を鳴らすところから始まり、音楽データを作成してファイル出力をするプログラムに至るまで、様々な問題に直面した。その例を以下に示した。この中から 1 つをピックアップし、解決方法を示す。

1. 音符と音階をプログラムに設定する方法の問題
2. 音データの長さの計算方法とその誤差の問題
3. 同じ音階が続いた場合、音の区切りがわからない問題
4. Vocal と Base の重ね合わせ方の問題
5. PC のメモリ不足より、n を扱う配列を増やせない問題
6. 音色の設定方法の問題

3 の問題について

「ミ」「ミ」のように同じ音階の音が続いた場合、音の大きさが同じため一定の音に聞こえ、区切りがわからない問題が生じた。そのため、音符の意味がなくなり、音楽的に不自然となってしまった。

この問題を解決するため、音を減衰させることを考えた。

その方法として、図 9 の関数 $f(n) = a^{-n}$ を音データの値にかけることにより、n の増加に従って音の大きさを減衰させることを考えた。これより、前の音と次の音の大きさに差が生じ、区別が可能になった。しかし、音符によって音が聞こえなくなるまで減衰してしまう問題が新たに生じた。

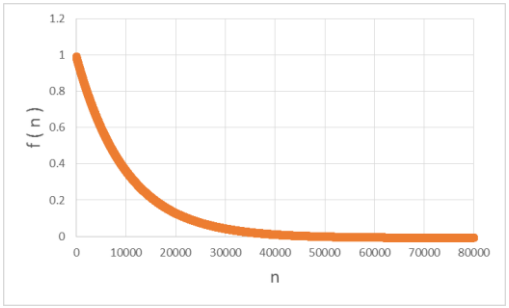


図 9. 関数 $f(n) = a^{-n}$ のグラフ

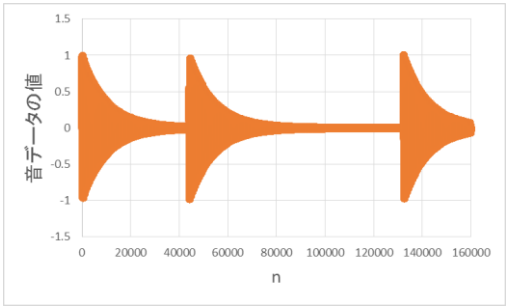


図 10. 左のf(n)をかけた後の音データ

そのため、図 11 のような関数 $f(n) = a^{-n} + b$ として漸近線を設けることにより、全ての音符に対して、音の大きさを保つことができ、目標の音を演奏することが可能になった。

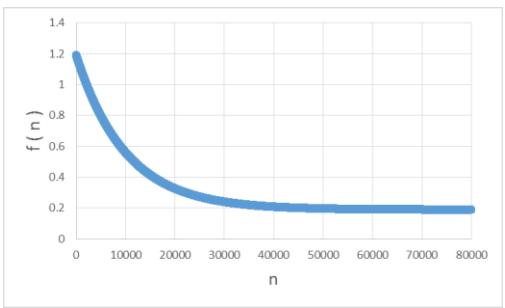


図 11. 関数 $f(n) = a^{-n} + b$ のグラフ

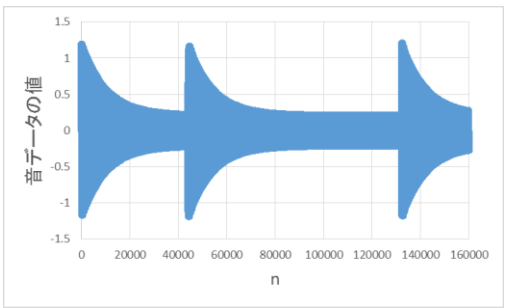


図 12. 左のf(n)をかけた後の音データ

[プログラム]

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include "wave.h"
#define PI M_PI
```

③ 音色の設定

```
#define ao_v( A )     ( 0.0                                    ) // Vocal のフーリエ級数 a0 を代入
#define an_v( A,k ) ( 4.0*A / ( k*k*PI*PI )*( 1.0-cos(k*PI) )  ) // Vocal のフーリエ級数 an を代入
#define bn_v( A,k ) ( 0.0                                      ) // Vocal のフーリエ級数 bn を代入

#define ao_b( A )     ( 0.0                                    ) // Base のフーリエ級数 a0 を代入
#define an_b( A,k ) ( 0.0                                      ) // Base のフーリエ級数 an を代入
#define bn_b( A,k ) ( -A/( k*PI )                              ) // Base のフーリエ級数 bn を代入

int main(void)
{
    MONO_PCM pcm1 ;
    int    count_v, n_v, v[2][72] ;
    int    count_b, n_b, b[7][27] ;
    int    key, tempo, note, d, sum, n=0, n_j, j, p, c, k ;
    double    tv=0.0, tb=0.0, y_b[7] ;
    double    Amp, decayrate, saturation, A, ts, fs, fo, f[100] ;
    double    y, Ao_v, Ao_b, An=0.0, Bn=0.0 ;
```

① 音符と音階の設定（Vocal）

```
    v[0][0]=4;    v[1][0]=0;      v[0][7]=8;    v[1][7]=73;      v[0][48]=8;   v[1][48]=78;
    v[0][1]=8;    v[1][1]=76;     v[0][8]=8;    v[1][8]=73;      v[0][49]=8;   v[1][49]=76;
    v[0][2]=8;    v[1][2]=76;     v[0][9]=8;    v[1][9]=69;      v[0][50]=8;   v[1][50]=76;
    v[0][3]=8;    v[1][3]=74;     v[0][10]=8;   v[1][10]=64;     v[0][51]=8;   v[1][51]=73;
    v[0][4]=4;    v[1][4]=73;     v[0][11]=4;   v[1][11]=64;     v[0][52]=8;   v[1][52]=76;
    v[0][5]=4;    v[1][5]=74;                                    v[0][53]=8;   v[1][53]=76;
    v[0][5]=4;    v[1][5]=74;          (  …省略…  )              v[0][54]=8;   v[1][54]=78;
    v[0][6]=8;    v[1][6]=74;                                    v[0][55]=8;   v[1][55]=76;
```

① 音符と音階の設定（Base）

```
b[0][0]=1;  b[1][0]=67;  b[2][0]=59;  b[3][0]=55;  b[4][0]=50;  b[5][0]=47;  b[6][0]=43;
b[0][1]=1;  b[1][1]=66;  b[2][1]=59;  b[3][1]=55;  b[4][1]=50;  b[5][1]=47;  b[6][1]=43;
b[0][2]=1;  b[1][2]=65;  b[2][2]=60;  b[3][2]=57;  b[4][2]=53;  b[5][2]=48;  b[6][2]=41;
b[0][3]=1;  b[1][3]=64;  b[2][3]=59;  b[3][3]=56;  b[4][3]=52;  b[5][3]=47;  b[6][3]=40;

b[0][4]=1;  b[1][4]=64;  b[2][4]=60;  b[3][4]=57;  b[4][4]=52;  b[5][4]=45;  b[6][4]=0;
b[0][5]=1;  b[1][5]=64;  b[2][5]=60;  b[3][5]=56;  b[4][5]=52;  b[5][5]=45;  b[6][5]=0;
b[0][6]=1;  b[1][6]=64;  b[2][6]=60;  b[3][6]=55;  b[4][6]=52;  b[5][6]=45;  b[6][6]=0;
b[0][7]=1;  b[1][7]=66;  b[2][7]=62;  b[3][7]=57;  b[4][7]=50;  b[5][7]=0;   b[6][7]=0;
```

② テンポやキーの設定

```
tempo = 68;                                  // テンポの設定
key = 3;                                     // キーの調節
Amp = 1.0;                                   // 音の大きさを設定
decayrate = 1.0001;                          // 音の減衰率を設定
saturation = 0.2;                            // 減衰した音の飽和値を設定

count_v = 55;                                // v[ ][ j ]の最後の j を設定
count_b = 7;                                 // b[ ][ j ]の最後の j を設定


for( j= 0; j<= count_v ; j++ ){ tv = tv+1.0/( v[0][j] ); }   // Vocal の音符を足し合わせ
n_v = 4.0*44100.0*60.0 / (double)tempo*(double)tv;           // Vocal の全データ数 n_v を計算

for( j= 0; j<= count_b ; j++ ){ tb = tb+1.0/( b[0][j] ); }   // Base の音符を足し合わせ
n_b = 4.0*44100.0*60.0 / (double)tempo*(double)tb;           // Base の全データ数 n_b を計算

if( n_v > n_b )   sum = n_v;                 // Vocal と Base のデータ数を比較し、
if( n_b >= n_v ) sum = n_b;                  // 多い方のデータ数を sum に設定

printf( "¥n¥t¥t データ数 n[個]¥t 時間 t[s]¥n" );
printf( "ヴォーカル:¥t%d¥t¥t%d¥n",n_v,n_v/44100 );
printf( "ベース:¥t¥t%d¥t¥t%d¥n",n_b,n_b/44100 );
printf( "¥n 音楽ファイル（wav 形式）を作成します。¥n" );
getch( );
printf( "¥n¥n¥t¥t…計算中です…¥n" );
```

```
pcm1.fs = 44100;                                    /* 標本化周波数     */
pcm1.bits = 16;                                     /* 量子化精度       */
pcm1.length = sum+10;                               /* 音データの長さ   */
pcm1.s = calloc( pcm1.length, sizeof(double) );     /* 音データ         */

ts = 1.0 / pcm1.fs;
Ao_v = ao_v(A);
Ao_b = ao_b(A);
```

④ 音階に対応する周波数の計算

```
f[0] = 0.0;
for( d= 20 ;d<= 90 ;d++ ){
        f[d] = pow( 2.0,((double)d-69.0+key ) / 12.0 )*440.0;
}
```

⑤ Vocal の音データを計算

```
for( j= 0 ;j<= count_v ;j++ ){                  // Vocal に設定した音符の数（55 個）までループ

        note = v[0][j];                         // Vocal の j 個目における音符（note）を参照
        d = v[1][j];                            // Vocal の j 個目における音階を参照
        fo = f[d];                              // 参照した音階における周波数を fo に代入
        n_j = 4.0*pcm1.fs / (double)note*60.0 / (double)tempo;
                                                // Vocal の j 個目におけるデータ数を計算
        for( p= 0 ;p<= n_j ;p++ ){              // Vocal の j 個目におけるデータ数だけループ

                A = Amp*pow( decayrate,-(double)p )+saturation;   // 音の大きさを減衰

                for( k= 1 ;k<= 10 ;k++ ){       // フーリエ級数を展開する項数 k を設定

                        An += an_v( A,k )*cos( (double)k*2.0*PI*fo*n*ts );   // ∑_{k=1}^{k=10} a_k を計算
                        Bn += bn_v( A,k )*sin( (double)k*2.0*PI*fo*n*ts );   // ∑_{k=1}^{k=10} b_k を計算
                }
                y = Ao_v+An+Bn;
                pcm1.s[n] = y;                  // Vocal の n における音データを格納
                n = n+1;
                An = 0.0;
                Bn = 0.0;
        }
}
n = 0;
```

⑤ Base の音データを計算

```
    for( j= 0 ;j<= count_b ;j++ ){                  // Base に設定した音符の数（7 個）までループ

        note = b[0][j];                             // Base の j 個目における音符（note）を参照
        n_j = 4.0*pcm1.fs / (double)note*60.0 / (double)tempo;
                                                    // Base の j 個目におけるデータ数を計算
        for( p= 0 ;p<= n_j ;p++ ){                  // Base の j 個目におけるデータ数を計算

            A = Amp*pow( decayrate,-(double)p )+saturation;     // 音の大きさを減衰

            for( c= 1 ;c<= 6 ;c++ ){                // 第 1 弦～第 6 弦までループ

                d = b[c][j];                        // Base の第 c 弦における音階を参照
                fo = f[d];                          // 参照した音階における周波数を fo に代入

                for( k= 1 ;k<= 10 ;k++ ){           // フーリエ級数を展開する項数 k を設定

                    An += an_b( A,k )*cos( (double)k*2.0*PI*fo*n*ts );    // ∑_{k=1}^{k=10} a_k を計算
                    Bn += bn_b( A,k )*sin( (double)k*2.0*PI*fo*n*ts );    // ∑_{k=1}^{k=10} b_k を計算
                }
                y_b[c] = Ao_b+An+Bn;
                An = 0.0;
                Bn = 0.0;
            }                                       // 弦の音を重ね合わせ、n における
            y = y_b[1]+y_b[2]+y_b[3]+y_b[4]+y_b[5]+y_b[6];    // Base の音データを計算
            pcm1.s[n] = pcm1.s[n]+y;                // Vocal と Base の音を合わせて音データを格納
            n = n+1;
        }
    }

    wave_write_16bit_mono( &pcm1, "Sound_Program.wav" );    // wav ファイルへ書き込み
    free( pcm1.s );

    printf( "¥n¥n ファイル「Sound_Program.wav」に保存しました。¥n" );
    printf( "¥nデータ数 n:¥t%d¥t[個]¥n 時間 t:¥t%d¥t[s]¥n¥n",pcm1.length,pcm1.length/pcm1.fs );

    return( 0 );
}
```