

音のプログラミング第1回

音の作成と合成

鹿児島大学工学部

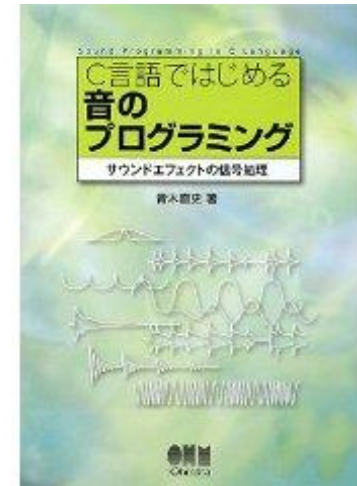
淵田孝康

目的

- 音の仕組みを知る
 - 音のデジタル化
 - 振幅と周波数
 - サンプリング周波数
 - サンプリングビット数
- 音の作成
 - エクセルを用いた音の作成と表示
 - プログラムを用いた音の作成
- 音の合成
 - エクセルを用いた音の合成と表示
 - プログラムを用いた音の合成

参考書

- C言語ではじめる音のプログラミング
 - サウンドエフェクトの信号処理
- 青木直史著
- オーム社
- 2008年初版、2021年第13刷
- 定価2,600円（税別）
- ウェブサイト
 - <http://floor13.sakura.ne.jp/book03/book03.html>
 - プログラムなどダウンロード可能
 - BCC32を使用



コンピュータでの音処理

- AD変換

- マイクで音圧取得：アナログ情報
- LPFで高周波をカット
 - エイリアス歪みを抑制
- 標本化：一定間隔でアナログ情報を計測
- 量子化：デジタル信号へ変換（PCMデータ）

- DA変換

- 逆量子化：数値データ
- LPFで高周波をカット
- スピーカーで音圧再生：アナログ情報

具体例

- AD変換

- マイクから録音
- ボイスレコーダー
 - m4aフォーマット（圧縮データ）
- 仮面舞踏会アプリ
 - wavフォーマット（PCMデータ）
 - 初期のサウンドレコーダーはwavフォーマットで保存できた
が、最近のボイスレコーダーは不可能

- DA変換

- スピーカーから再生
- Windows Media Player
 - さまざまなフォーマットの音を再生可能

課題1

1. 仮面舞踏会アプリをインストールせよ

- <https://www.vector.co.jp/soft/win95/art/se268485.html>
- baile4_9_4.zipを展開してZ:の適当な場所に置くだけ
- フォルダbaile4_9_4の中のbaile.exeがアプリ
- デスクトップにショートカットを作っておくと便利

2. 保存フォルダの変更

- 「設定」⇒「基本設定」から、WAVE保存先を変更できる（あとで）

3. 1～2秒の自分の声を録音せよ

- 録音ボタンを押す
- マイクに向かって話す
- 停止ボタンを押す

4. 録音を確認せよ

- PCMデータは、実行ファイルがあるフォルダの**WAVEフォルダ**にある
- フォーマットは.wav
- ダブルクリックすると再生



C++コンパイラ bcc32

- 元Borland社のC++コンパイラ
 - 現在は買収されてMicro Focus社
- Embarcadero社のHPからダウンロード可能
 - <https://www.embarcadero.com/jp/free-tools/ccompiler>
 - 姓、名、メール等を入力してダウンロード
 - BCC102.zip
 - 展開するとBCC102フォルダができる
 - Z:の適当な場所に移動する
 - その中のbinフォルダにPathを通す
 - Pathの通り方が分からない場合は「Windows10 Path」で検索
 - ついでにカレントフォルダへもPathを通すこと
- 確認
 - コマンドプロンプトを開き、bcc32cと入力
 - エラーが出たらOK

```
SoundProgramming>bcc32c
Embarcadero C++ 7.30 for Win32 Copyright (c) 2012-2017 Embarcadero Technologies, Inc.
bcc32c.exe: error: no input files

SoundProgramming>
```

課題2

- `bcc32`をインストールせよ
 - 前のスライドを参考にすること
- フォルダ作成
 - Z:の適当な場所に**SoundProgramming**フォルダを作成
- `test1`を作成、コンパイル、実行
 - その中に**test1.c**という空のテキストファイルを作成
 - 右図のコードを入力
 - コンパイルして実行を確認（下図）
 - コンパイラは**bcc32c**
 - 黄色のコマンドを入力する

```
#include <stdio.h>

int main(){
    printf("Hello Sound Programming!¥n");
}
```

```
SoundProgramming>bcc32c test1.c
Embarcadero C++ 7.30 for Win32 Copyright (c) 2012-2017 Embarcadero Technologies, Inc.
test1.c:
Turbo Incremental Link 6.90 Copyright (c) 1997-2017 Embarcadero Technologies, Inc.

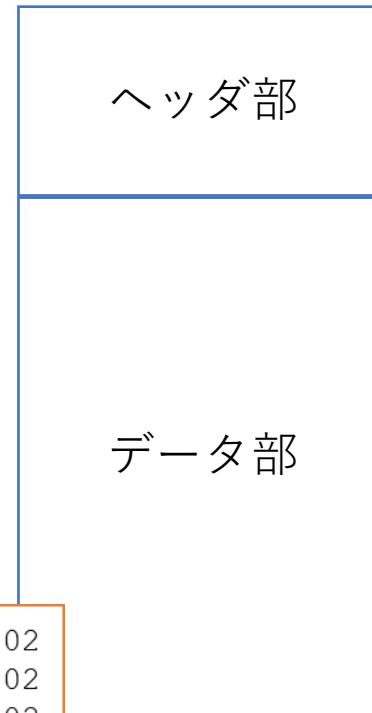
SoundProgramming>test1
Hello Sound Programming!

SoundProgramming>
```

実行結果！

WAVEファイル

- 音をサンプリングした生のデータを格納したファイル
- ヘッダ部
 - 標本化周波数 `samplesPerSec`
 - 量子化精度 `bitsPerSample`
 - 音データ長 `dataChunkSize`
 - など
- データ部
 - PCMデータが並んでいる
 - `-32768~+32767`の整数値
 - 16ビット=2バイト
 - 16進数で2桁



```
2a 00 e8 00 6c 00 28 02 40 00 f6 01 9f 00 96 02
89 00 d5 02 fd ff 7e 02 46 00 1b 02 3e 00 6c 02
f8 ff a8 02 d9 ff 88 02 e6 ff 76 02 1c 00 5d 02
...
```

wave.h

- PCMファイルを扱うヘッダ
- 2つの型
 - MONO_PCM：モノラル音声データ
 - STEREO_PCM：ステレオ音声データ
- 4つの関数
 - void mono_wave_read(pcm,name)：モノラル音声読み込み
 - void mono_wave_write(pcm,name)：モノラル音声書き込み
 - void stereo_wave_read(pcm,name)：ステレオ音声読み込み
 - void stereo_wave_write(pcm,name)：ステレオ音声書き込み
 - 引数
 - pcm：MONO_PCM(またはSTEREO_PCM)のポインタ
 - name：ファイル名へのポインタ
- 2ページの参考文献のサポートサイトから入手
 - <http://floor13.sakura.ne.jp/book03/book03.html>
 - chapter01.zipをダウンロードして展開
 - chapter01/chapter01/ex1_1/wave.h
 - ただし修正が必要：144,281,295行目の (short) を (int) に変更する

音の型

MONO_PCM

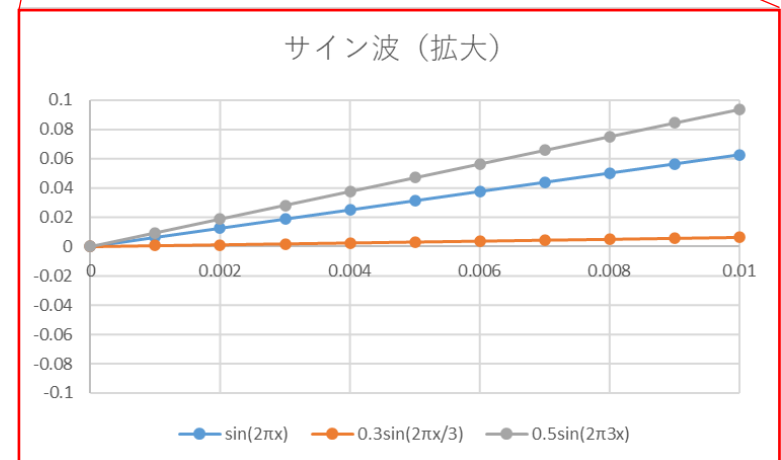
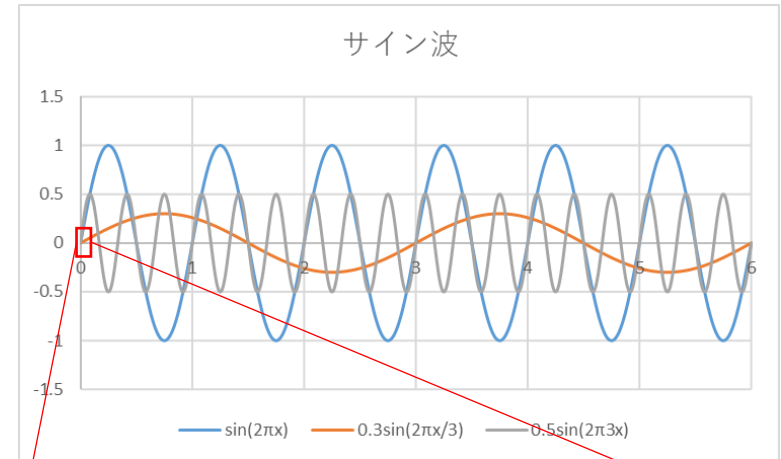
```
typedef struct
{
    int fs; /* 標本化周波数 */
    int bits; /* 量子化精度 */
    int length; /* 音データの長さ */
    double *s; /* 音データ */
} MONO_PCM;
```

STEREO_PCM

```
typedef struct
{
    int fs; /* 標本化周波数 */
    int bits; /* 量子化精度 */
    int length; /* 音データの長さ */
    double *sL; /* 音データ (Lチャンネル) */
    double *sR; /* 音データ (Rチャンネル) */
} STEREO_PCM;
```

音を作る

- 音は空気の振動 = 波
- 音を作る = 波形を作る
- もっとも単純な波形 = サイン波
 - $w(x) = \sin 2\pi x$
 - 振幅1、周期1、周波数1 ($f = 1/T$)
 - $w(x) = 0.5 \sin 2\pi 3x$
 - 振幅0.3、周期1/3、周波数3
 - $w(x) = 0.3 \sin \frac{2\pi x}{3} = 0.3 \sin 2\pi \frac{1}{3} x$
 - 振幅0.3、周期3、周波数1/3
 - $w(x) = A \sin \frac{2\pi x}{T} = A \sin 2\pi f x$
 - 振幅A、周期T、周波数 $f = 1/T$
- 離散的なサイン波
 - 変数 x の刻み幅が離散的
 - 右図では0.001刻み
 - サンプル周波数が1000Hz=1kHz
 - 1秒で1000回サンプリングすること



エクセルで音を作る①

- サイン波を作るために必要なデータ

- 周波数 f
- 振幅 A
- サンプリング周波数 f_s
- 系列長 L
- 時刻 $x = \frac{0}{f_s}, \frac{1}{f_s}, \frac{2}{f_s}, \dots, \frac{L-1}{f_s}$
- 波形 $w(x) = A \sin 2\pi f x$

- エクセルファイルを作成 "wave1.xlsx"

- A1 : 500 (周波数)
- A2 : 0.3 (振幅)
- A3 : 8000 (サンプリング周波数)
- A4 : 16000 (系列長)
- A5, B5, C5 に "n", "x", "sin" と入力
- A6 に 0 を入力

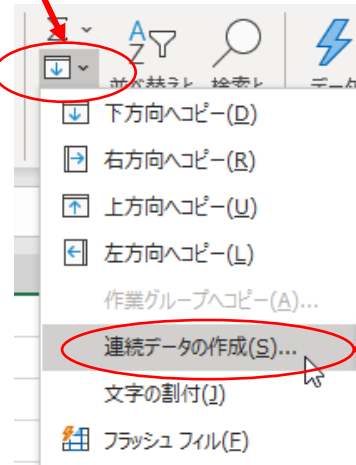
"wave1.xlsx"

	A	B	C
1	500		
2	0.3		
3	8000		
4	16000		
5	n	x	sin
6	0		
7			

エクセルで音を作る②

	A	B	C
1	500		
2	0.3		
3	8000		
4	16000		
5	n	x	sin
6	0		
7			

連続データ生成機能を用いて、
0から15999までの16000個の
系列データを生成する



連続データ

範囲	種類	増加単位
<input type="radio"/> 行(R)	<input checked="" type="radio"/> 加算(L)	<input type="radio"/> 日(A)
<input checked="" type="radio"/> 列(C)	<input type="radio"/> 乗算(G)	<input type="radio"/> 週日(W)
	<input type="radio"/> 日付(D)	<input type="radio"/> 月(M)
	<input type="radio"/> オートフィル(E)	<input type="radio"/> 年(Y)

データ予測(I)

増分値(S): 1 停止値(Q): 15999

OK キャンセル

	A	B	C
1	500		
2	0.3		
3	8000		
4	16000		
5	n	x	sin
6	0		
7	1		
8	2		
9	3		
...			
15999	15993		
16000	15994		
16001	15995		
16002	15996		
16003	15997		
16004	15998		
16005	15999		
16006			
16007			

16000個の連続データができた

エクセルで音を作る③

B列にxを、C列にサイン波を作る

	A	B	C
1	500		
2	0.3		
3	8000		
4	16000		
5	n	x	sin
6	0	=A6/\$A\$3	
7	1		

$$x = \frac{0}{f_s}, \frac{1}{f_s}, \frac{2}{f_s}, \dots, \frac{L-1}{f_s}$$

	A	B	C
1	500		
2	0.3		
3	8000		
4	16000		
5	n	x	sin
6	0	0	0
7	1		
8	2		
9	3		

	A	B	C	D
1	500			
2	0.3			
3	8000			
4	16000			
5	n	x	sin	
6	0		= \$A\$2 * SIN(2 * PI() * \$A\$1 * \$B6)	
7	1			

$$w(x) = A \sin 2\pi f x$$

	A	B	C
1	500		
2	0.3		
3	8000		
4	16000		
5	n	x	sin
6	0	0	0
7	1	0.000125	0.114805
8	2	0.00025	0.212132
9	3	0.000375	0.277164
10	4	0.0005	0.3
11	5	0.000625	0.277164
12	6	0.00075	0.212132
13	7	0.000875	0.114805
14	8	0.001	0
15	9	0.001125	-0.114805
16	10	0.00125	-0.212132
17	11	0.001375	-0.277164
18	12	0.0015	-0.3
19	13	0.001625	-0.277164
20	14	0.00175	-0.212132
21	15	0.001875	-0.114805
22	16	0.002	0
23	17	0.002125	0.114805
24	18	0.00225	0.212132
25	19	0.002375	0.277164
26	20	0.0025	0.3
27	21	0.002625	0.277164
28	22	0.00275	0.212132
29	23	0.002875	0.114805
30	24	0.003	0
31	25	0.003125	-0.114805
32	26	0.00325	-0.212132
33	27	0.003375	-0.277164
34	28	0.0035	-0.3
35	29	0.003625	-0.277164
36	30	0.00375	-0.212132
37	31	0.003875	-0.114805
38	32	0.004	0
39	33	0.004125	0.114805
40	34	0.00425	0.212132
41	35	0.004375	0.277164
42	36	0.0045	0.3
43	37	0.004625	0.277164
44	38	0.00475	0.212132
45	39	0.004875	0.114805
46	40	0.005	0
47	41	0.005125	-0.114805
48	42	0.00525	-0.212132
49	43	0.005375	-0.277164
50	44	0.0055	-0.3
51	45	0.005625	-0.277164
52	46	0.00575	-0.212132
53	47	0.005875	-0.114805
54	48	0.006	0
55	49	0.006125	0.114805
56	50	0.00625	0.212132
57	51	0.006375	0.277164
58	52	0.0065	0.3
59	53	0.006625	0.277164
60	54	0.00675	0.212132
61	55	0.006875	0.114805
62	56	0.007	0
63	57	0.007125	-0.114805
64	58	0.00725	-0.212132
65	59	0.007375	-0.277164
66	60	0.0075	-0.3
67	61	0.007625	-0.277164
68	62	0.00775	-0.212132
69	63	0.007875	-0.114805
70	64	0.008	0
71	65	0.008125	0.114805
72	66	0.00825	0.212132
73	67	0.008375	0.277164
74	68	0.0085	0.3
75	69	0.008625	0.277164
76	70	0.00875	0.212132
77	71	0.008875	0.114805
78	72	0.009	0
79	73	0.009125	-0.114805
80	74	0.00925	-0.212132
81	75	0.009375	-0.277164
82	76	0.0095	-0.3
83	77	0.009625	-0.277164
84	78	0.00975	-0.212132
85	79	0.009875	-0.114805
86	80	0.01	0

下へコピー

サイン波の離散データ完成

音を見る

挿入タブ

範囲選択

グラフ化

右クリック

拡大

サンプリング点

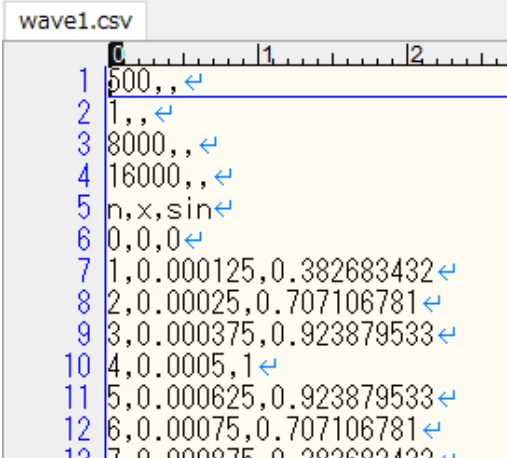
周波数が高すぎ

サイン波が生成されている

	A	B	C
1	500		
2	0.3		
3	8000		
4	16000		
5	n	x	sin
6	0	0	0
7	1	0.000125	0.11
8	2	0.00025	0.212132
9	3	0.000375	0.277164
10	4	0.0005	0.3
11	5	0.000625	0.277164
12	6	0.00075	0.212132
13	7	0.000875	0.11
14	8	0.001	0
15	9	0.001125	-0.11
16	10	0.00125	-0.212132
17	11	0.001375	-0.277164
18	12	0.0015	-0.3
19	13	0.001625	-0.277164
20	14	0.00175	-0.212132
21	15	0.001875	-0.11
22	16	0.002	0
23	17	0.002125	0.11
24	18	0.00225	0.212132
25	19	0.002375	0.277164
26	20	0.0025	0.3
27	21	0.002625	0.277164
28	22	0.00275	0.212132
29	23	0.002875	0.11
30	24	0.003	0
31	25	0.003125	-0.11
32	26	0.00325	-0.212132
33	27	0.003375	-0.277164
34	28	0.0035	-0.3
35	29	0.003625	-0.277164
36	30	0.00375	-0.212132
37	31	0.003875	-0.11
38	32	0.004	0
39	33	0.004125	0.11
40	34	0.00425	0.212132
41	35	0.004375	0.277164
42	36	0.0045	0.3
43	37	0.004625	0.277164
44	38	0.00475	0.212132
45	39	0.004875	0.11
46	40	0.005	0
47	41	0.005125	-0.11
48	42	0.00525	-0.212132
49	43	0.005375	-0.277164
50	44	0.0055	-0.3
51	45	0.005625	-0.277164
52	46	0.00575	-0.212132
53	47	0.005875	-0.11
54	48	0.006	0
55	49	0.006125	0.11
56	50	0.00625	0.212132
57	51	0.006375	0.277164
58	52	0.0065	0.3
59	53	0.006625	0.277164
60	54	0.00675	0.212132
61	55	0.006875	0.11
62	56	0.007	0
63	57	0.007125	-0.11
64	58	0.00725	-0.212132
65	59	0.007375	-0.277164
66	60	0.0075	-0.3
67	61	0.007625	-0.277164
68	62	0.00775	-0.212132
69	63	0.007875	-0.11
70	64	0.008	0
71	65	0.008125	0.11
72	66	0.00825	0.212132
73	67	0.008375	0.277164
74	68	0.0085	0.3
75	69	0.008625	0.277164
76	70	0.00875	0.212132
77	71	0.008875	0.11
78	72	0.009	0
79	73	0.009125	-0.11
80	74	0.00925	-0.212132
81	75	0.009375	-0.277164
82	76	0.0095	-0.3
83	77	0.009625	-0.277164
84	78	0.00975	-0.212132
85	79	0.009875	-0.11
86	80	0.01	0
87	81	0.009875	0.11481
88	82	0.00975	0.212132
89	83	0.009625	0.277164
90	84	0.0095	0.3
91	85	0.009375	0.277164
92	86	0.00925	0.212132
93	87	0.009125	0.11
94	88	0.009	0
95	89	0.008875	-0.11
96	90	0.00875	-0.212132
97	91	0.008625	-0.277164
98	92	0.0085	-0.3
99	93	0.008375	-0.277164
100	94	0.00825	-0.212132
101	95	0.008125	-0.11
102	96	0.008	0
103	97	0.007875	0.11
104	98	0.00775	0.212132
105	99	0.007625	0.277164
106	100	0.0075	0.3
107	101	0.007375	0.277164
108	102	0.00725	0.212132
109	103	0.007125	0.11
110	104	0.007	0
111	105	0.006875	-0.11
112	106	0.00675	-0.212132
113	107	0.006625	-0.277164
114	108	0.0065	-0.3
115	109	0.006375	-0.277164
116	110	0.00625	-0.212132
117	111	0.006125	-0.11
118	112	0.006	0
119	113	0.005875	0.11
120	114	0.00575	0.212132
121	115	0.005625	0.277164
122	116	0.0055	0.3
123	117	0.005375	0.277164
124	118	0.00525	0.212132
125	119	0.005125	0.11
126	120	0.005	0
127	121	0.004875	-0.11
128	122	0.00475	-0.212132
129	123	0.004625	-0.277164
130	124	0.0045	-0.3
131	125	0.004375	-0.277164
132	126	0.00425	-0.212132
133	127	0.004125	-0.11
134	128	0.004	0
135	129	0.003875	0.11
136	130	0.00375	0.212132
137	131	0.003625	0.277164
138	132	0.0035	0.3
139	133	0.003375	0.277164
140	134	0.00325	0.212132
141	135	0.003125	0.11
142	136	0.003	0
143	137	0.002875	-0.11
144	138	0.00275	-0.212132
145	139	0.002625	-0.277164
146	140	0.0025	-0.3
147	141	0.002375	-0.277164
148	142	0.00225	-0.212132
149	143	0.002125	-0.11
150	144	0.002	0
151	145	0.001875	0.11
152	146	0.00175	0.212132
153	147	0.001625	0.277164
154	148	0.0015	0.3
155	149	0.001375	0.277164
156	150	0.00125	0.212132
157	151	0.001125	0.11
158	152	0.001	0
159	153	0.000875	-0.11
160	154	0.00075	-0.212132
16001	15995	1.999375	-0.277164
16002	15996	1.9995	-0.3
16003	15997	1.999625	-0.277164
16004	15998	1.99975	-0.212132
16005	15999	1.999875	-0.11481
16006			
16007			

CSVで保存

- まずエクセルブックを保存
 - 保存ボタンを押すだけ
- CSV形式
 - 単純なテキストであり、いろいろなプログラムで利用可能
 - テキストデータならば形式が簡単で扱いやすい
 - エクセルでもそのまま扱える
 - ただし、図や書式は保存されないので注意
 - バイナリ形式と比較するとサイズが大きい
- データをCSV形式で保存
 - 「ファイル」⇒「名前を付けて保存」
 - 形式を「CSV(コンマ区切り)(*.csv)」に変更
 - "wave1.csv"で保存
 - 警告が出るのでOKを押して単一シートのみを保存
 - 保存したCSVファイルをテキストエディタで見ると、図のように各列がカンマで区切られて並んでいる



```
wave1.csv
0
1
2
3
4
5
6
7
8
9
10
11
12
13
```

n	x	sin
500	,	,
1	,	,
8000	,	,
16000	,	,
1	0.000125	0.382683432
2	0.00025	0.707106781
3	0.000375	0.923879533
4	0.0005	1
5	0.000625	0.923879533
6	0.00075	0.707106781
7	0.000875	0.382683432

音への変換

- CSVファイルからWAVEファイルへ変換する
 - `csv2wav csv wav`
- CSVファイルはエクセルから保存したもの
 - 1行目：周期（読み飛ばす）
 - 2行目：振幅（読み飛ばす）
 - 3行目：サンプリング周波数`fs`
 - 4行目：系列長`length`
 - 5行目：ヘッダ（読み飛ばす）
 - 6行目以降
 - 3つの数値が並んでいる
 - 3つ目がデータ
 - 1つ目と2つ目を読み飛ばすためにダミーの変数`n,x`を使っている
- WAVEファイルへの保存
 - `mono_wave_save()`関数
- コンパイル
 - > `bcc32c csv2wav.c`
- 実行
 - > `csv2wav wave1.csv wave1.wav`

csv2wav.c

```
#include <stdio.h>
#include <stdlib.h>
#include "wave.h"

int main(int argc, char **argv) {
    int N=300;
    MONO_PCM pcm;

    // 引数チェック
    if (argc != 3) {
        fprintf(stderr, "Usage: fromCSV csv wave¥n");
        exit(1);
    }

    // CSVデータの読み込み
    char buf[N];
    FILE *fp = fopen(argv[1], "rt");
    fgets(buf, N, fp); // 周波数読み飛ばし
    fgets(buf, N, fp); // 振幅読み飛ばし
    pcm.fs = atoi(fgets(buf, N, fp)); // サンプリング周波数
    pcm.bits = 16; // 量子化ビット数 (16に固定)
    pcm.length = atoi(fgets(buf, N, fp)); // 系列長
    fgets(buf, N, fp); // ヘッダの読み飛ばし
    pcm.s = malloc(pcm.length * sizeof(double));
    int n; // n用
    double x; // x用
    for (int i = 0; i < pcm.length; i++) {
        fgets(buf, N, fp);
        sscanf(buf, "%d, %lf, %lf", &n, &x, &pcm.s[i]);
    }
    fclose(fp);

    // WAVEデータの保存
    mono_wave_write(&pcm, argv[2]);

    free(pcm.s);
}
```

課題3

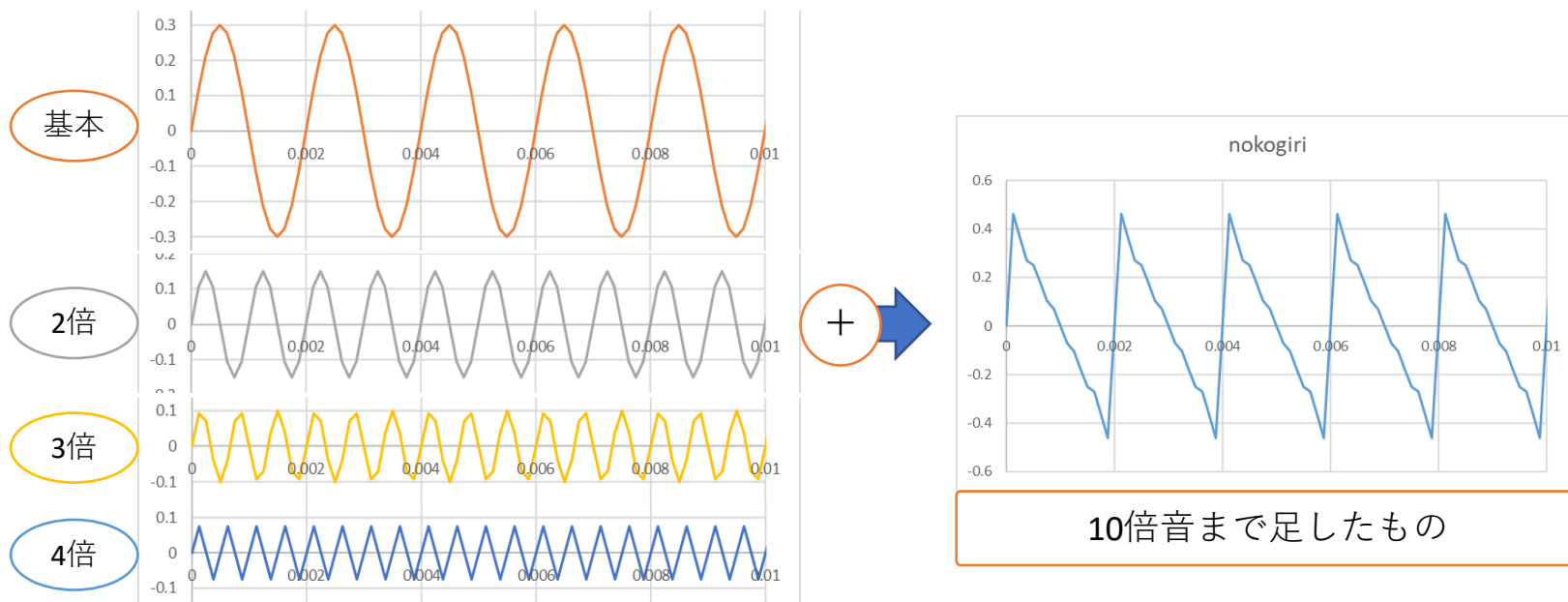
1. 作成したサイン波を音として再生せよ
 - 保存したWAVEファイルをダブルクリック
 - Windows Media Playerで再生する
 - どんな音が再生されたか述べよ
2. 楽器の音階を再現せよ。
 - 各音階の周波数とファイル名は右表のとおり
 - 8つのファイルに音データを保存し、Media Playerで連続的に再生せよ
 - wave1.xlsxの周波数を変更しつつCSVファイルに保存し、それをWAVファイルに変換すること
3. 各音の長さを1秒にするにはエクセルファイルのどこを変えればよいか説明せよ。



音階	周波数	ファイル名
ド	523Hz	wave1-1.wav
レ	587Hz	wave1-2.wav
ミ	659Hz	wave1-3.wav
ファ	698Hz	wave1-4.wav
ソ	783Hz	wave1-5.wav
ラ	880Hz	wave1-6.wav
シ	988Hz	wave1-7.wav
ド	1047Hz	wave1-8.wav

音の合成

- サイン波の周波数と振幅を変化させつつ波を足し合わせる
 $S(f, A)$: 周波数 f 、振幅 A のサイン波
- ノコギリ波
 - 周波数2倍、3倍・・・、振幅1/2倍、1/3倍・・・の波を足し合わせる
 - $$N = S(f, A) + S\left(2f, \frac{A}{2}\right) + S\left(3f, \frac{A}{3}\right) + S\left(4f, \frac{A}{4}\right) + \dots$$



エクセルによる音の合成①

- wave1.xlsxをコピーしてwave2.xlsxを作る
 - グラフは削除しておく
- C列を挿入し、ノコギリ波を作る場所を空ける
- D列からM列までに基本波、2倍波、3倍波・・・10倍波を作ることにする
- C6にD6からM6までの和を計算する

右クリック

The image illustrates the steps to insert a column and calculate a sum in Excel. It consists of three sequential screenshots of an Excel spreadsheet.

Screenshot 1: Shows a spreadsheet with columns A through F. A right-click context menu is open over column C. The menu options include: 切り取り(I), コピー(C), 貼り付けのオプション, 形式を選択して貼り付け(S)..., **挿入(I)** (highlighted with a red circle), 削除(D), 数式と値のクリア(N), セルの書式設定(E)..., 列の幅(W)..., 非表示(H), and 再表示(U). A red arrow points from the '挿入(I)' option to the next screenshot.

Screenshot 2: Shows the same spreadsheet after column C has been inserted. Column C is highlighted in green. A red box highlights the new empty cell in row 6 of column C. A red arrow points from this box to the next screenshot.

Screenshot 3: Shows the spreadsheet with the formula `=SUM(D6:M6)` entered in cell C6. The formula bar shows the formula. A red box highlights the formula in the cell. The spreadsheet data is as follows:

	A	B	C	D
1	500			
2	0.3			
3	8000			
4	16000			
5	n	x	sin	
6	0	0		0
7	1	0.000125	0.114805	
8	2	0.00025	0.212132	
9	3	0.000375	0.277164	

エクセルによる音の合成②

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	500												
2	0.3												
3	8000												
4	16000			1									
5	n	x	nokogiri	sin1									
6	0	0	0	0									
7	1	0.000125		0.114805									
8	2	0.00025		0.212132									

	A	B	C	D	E	F	G	H	I	J	K	L	M
4	16000			1									
5	n	x	nokogiri	sin1									
6	0	0	0	$=\$A\$2/\$D\$4*\text{SIN}(2*\text{PI}()*\$D\$4*\$A\$1*\$B6)$									
7	1	0.000125		0.114805									

	A	B	C	D	E	F	G	H	I	J	K	L	M
4	16000			1									
5	n	x	nokogiri	sin1									
6	0	0	0	0									
7	1	0.000125		0.114805									

	A	B	C	D	E	F	G	H	I	J	K	L	M
4	16000			1	2	3	4	5	6	7	8	9	10
5	n	x	nokogiri	sin1	sin2	sin3	sin4	sin5	sin6	sin7	sin8	sin9	sin10
6	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1	0.000125		0.114805									

文字を入力

式を修正

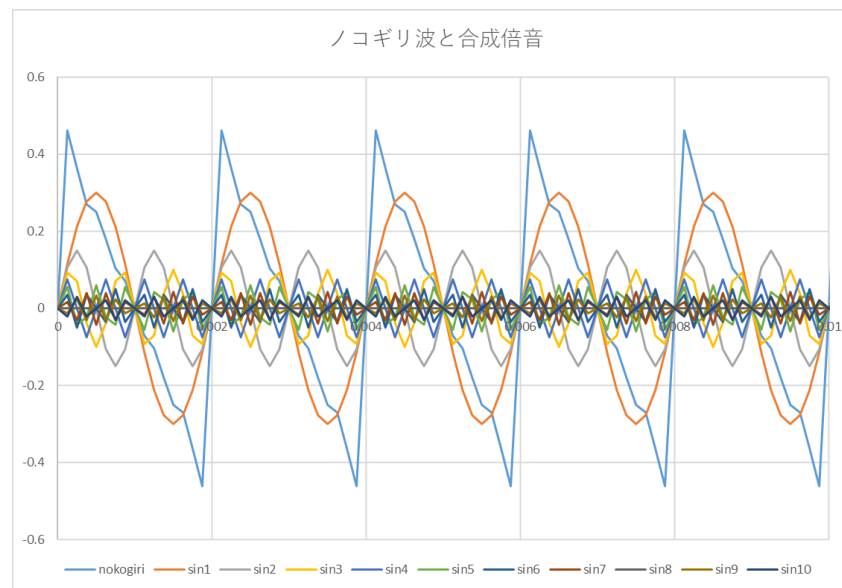
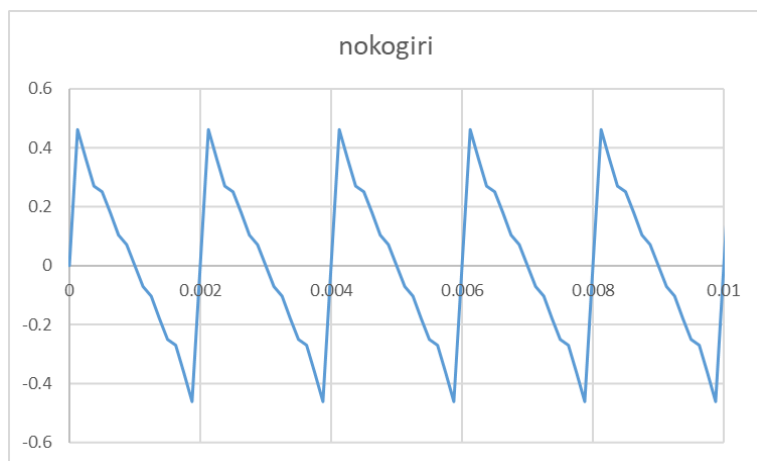
フィルハンドル

右へコピー

- D列に基本波を作成する
- このとき、D4の数字を倍数とするように式を修正する
- D2:D4を、フィルハンドルを引っ張って右へコピーする
- E列からM列に倍音波が作成される

波形の確認

- 作成したノコギリ波をグラフ化する
 - "x"と"nokogiri"の列を選択
 - "x"と"nokogiri"の2つを選択し、シフトキーを押しながら下線をダブルクリック
 - 「挿入」⇒「グラフ」⇒「散布図（直線）」
 - 横軸を右クリックし最大値を0.01に
- 合成した波形全部のグラフ化も同様に可能



課題4

1. エクセルで以下のノコギリ波を生成し再生せよ
 - 周波数880Hz
 - 振幅0.5
 - 長さ2秒
 - ファイル名：wave2.wav
2. 矩形波とはサイン波の奇数倍音だけの和として以下のように計算される
 - $R = S(f, A) + S\left(3f, \frac{A}{3}\right) + S\left(5f, \frac{A}{5}\right) + \dots$
 - エクセルでwave3.xlsxを作り、矩形波を以下の矩形波を生成し再生せよ
 - 周波数880Hz
 - 振幅0.5
 - 長さ2秒
 - ファイル名：wave3.wav
3. それぞれの音を聞き、自分が感じた特徴を述べよ

プログラムによる音の生成

- C言語のプログラムで音を生成する
- サイン波を作るために必要なデータ
 - 振幅 A
 - 周波数 f
 - サンプルング周波数 f_s
 - 系列長 L
 - 時刻 $x = \frac{0}{f_s}, \frac{1}{f_s}, \frac{2}{f_s}, \dots, \frac{L-1}{f_s}$
 - 波形 $w(x) = A \sin 2\pi f x$
- 波形 $w(x)$ の値を作ればよい
 - 振幅0.3
 - 周波数500Hz
 - サンプルング周波数8000
 - 量子化ビット数16
 - 音の長さ16000 (= 2秒)
 - MONO_PCM型の配列 s に入れる
- 右図のプログラム"makeSinWave.c"
 - > `bcc32c makeSinWave.c`
 - > `makeSinWave test1.wav`

```
typedef struct
{
    int fs; /* 標本化周波数 */
    int bits; /* 量子化精度 */
    int length; /* 音データの長さ */
    double *s; /* 音データ */
} MONO_PCM;
```

makeSinWave.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "wave.h"

int main(int argc, char **argv) {
    if (argc != 2) {
        fprintf(stderr, "Usage: makeSinWave wav¥n");
        exit(1);
    }
    MONO_PCM pcm; // 音
    double A=0.3; // 振幅
    double f=500; // 周波数
    pcm.fs=8000; // サンプルング周波数
    pcm.bits=16; // サンプルングビット数
    pcm.length=16000; // 長さ
    pcm.s=(double*)malloc(pcm.length*sizeof(double));
    for (int i=0; i<pcm.length; i++) {
        double x=(double)i/pcm.fs; // x
        pcm.s[i]=A*sin(2*M_PI*f*x); // A*sin(2πfx)
    }
    mono_wave_write(&pcm, argv[1]); // 保存
    free(pcm.s);
}
```

音階の生成

- プログラムで音階を生成する
 - ドからドまでの8音階
 - それぞれの音を1秒

音階	周波数
ド	523Hz
レ	587Hz
ミ	659Hz
ファ	698Hz
ソ	783Hz
ラ	880Hz
シ	988Hz
ド	1047Hz

makeScale.c

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "wave.h"

int main(int argc, char **argv){
    if(argc!=2){
        fprintf(stderr, "Usage: makeScale wav¥n");
        exit(1);
    }
    MONO_PCM pcm; // 音
    pcm.fs=8000; // サンプリング周波数
    pcm.bits=16; // サンプリングビット数
    pcm.length=64000; // 長さ (8音階で8秒)
    pcm.s=(double*)malloc(pcm.length*sizeof(double));
    double scale_f[]={ 523,587,659,698,783,880,988,1047 }; // 音階の周波数
    int i=0;
    double A=0.3; // 振幅
    for(int j=0;j<8;j++){ // 8音階回す
        double f=scale_f[j]; // 音階jの周波数
        for(int k=0;k<8000;k++){ // 1つの音階で8000回 (1秒)
            double x=(double)i/pcm.fs; // x
            pcm.s[i++]=A*sin(2*M_PI*f*x); // A*sin(2πfx)
        }
    }
    mono_wave_write(&pcm,argv[1]); // 保存
    free(pcm.s);
}

```

課題5

1. ノコギリ波を生成するプログラムmakeSawWave.cを作成せよ
 - ノコギリ波： $N = S(f, A) + S\left(2f, \frac{A}{2}\right) + S\left(3f, \frac{A}{3}\right) + S\left(4f, \frac{A}{4}\right) + \dots$
 - パラメータ
 - 周波数500Hz
 - 振幅0.3
 - サンプリング周波数8000Hz
 - 系列長16000（2秒）
 - 倍音は10倍まで
 - ヒント
 - 2重forループ構造になる
 - 内側のループで倍音の足し合わせを行う
 - pcm.s[i]の初期化を忘れないこと
2. ノコギリ波を再生し、エクセルで作成した音と比較せよ
3. サンプリング周波数を800Hz、8000Hz、44100Hzに変えて2秒の音を作成し、それらを聞き比べてみよ
 - 音がどのように変化するか？
 - 同じ振幅と周波数なのに音が違う理由を考察せよ

第1回 終了