バイナリファイルの扱いと WAVE ファイルの操作
プログラミング言語演習 IV（後期）

Copyright(C)27July2002
25Aug2003

coskx

１．バイナリファイルの扱い（第１週）

１．１テキストファイルとバイナリファイル
テキストファイル：テキストエディタで書いたり読んだり出来るファイル。
バイナリファイル：テキストファイル以外のファイル。実行形式ファイルやデータファイルなど。

１．２ファイルのバイナリオープン
ファイルのオープンにはテキストオープンとバイナリオープンがある。テキストファイルのみをオープンする場合はテキストオープンするが，そうでない場合はバイナリオープンする。バイナリファイルはバイナリオープンしないと正しく読んだり書いたり出来ない。

ファイルのオープンの際に，テキストファイルの場合は
```
fp=fopen(argv[1],"r");
fp=fopen(argv[1],"w");
```
と書くがバイナリファイルの場合は
```
fp=fopen(argv[1],"rb");
fp=fopen(argv[1],"wb");
```
と書く。

例　ファイルをバイナリオープンして，先頭の 100 文字をそのまま表示するプログラム typeFile100.c を次に示す。

```
/* binary open test */
#include <stdio.h>
#include <stdlib.h>

void main(int argc, char *argv[])
{
    FILE *fp;
    int i;
    int chr;
    if (argc!=2) {
        printf("** error ** no filename !! ¥n");
        exit(1);
    }
    fp=fopen(argv[1],"rb");
    if (fp==NULL) {
        printf("** error ** can't open the file¥n");
        exit(1);
    }
    printf("input file name: <<%s>>¥n",argv[1]);
    for (i=0;i<200;i++) {
        chr=fgetc(fp);
        putchar(chr);
    }
    putchar('¥n');
    fclose(fp);
}
```

```
――― 実行結果 ―――
input file name: <<binfile.c>>
/* binary open test */
#include <stdio.h>
#include <stdlib.h>

void main(int argc, char *argv[])
{
    FILE *fp;
    int i;
    int chr;
    if (argc!=2) {
        printf("** error ** no fil
```

課題　1　ファイルの先頭 200 文字分をテキスト表示とバイナリ表示（16 進数表示）の両方を表示するプログラム typeFile.c を作りなさい。処理対象のファイル名はコマンドラインから与えることとする。ただし，テキスト表示する場合は 0x21 以上 0x7e 以下のみ通常表示し，それ以外は小数点を表示することとする。プログラム動作テストとして，ソースファイルと実行形式ファイルを表示しなさい。提出は tnct20 の各自の public_html にファイル名「typeFile.txt」で公開することとする。
例としてソースファイルを対象にした時の表示例を次に示す。（MSwindows の時の例）
ヒント：関数 fgetc()，fread()について検討しなさい。

```
#include.<stdio.h>..
#include.<stdlib.h>.
...void.main(int.arg
c,.char.*argv[])..{.
.....FILE.*fp;......
int.i;.....int.chr;
......if.(argc!=2).{
..........printf("**
.error.**.no.filenam
e.!!.¥n");..........
-----------------------------------
23 69 6e 63 6c 75 64 65 20 3c 73 74 64 69 6f 2e 68 3e 0d 0a
23 69 6e 63 6c 75 64 65 20 3c 73 74 64 6c 69 62 2e 68 3e 0d
0a 0d 0a 76 6f 69 64 20 6d 61 69 6e 28 69 6e 74 20 61 72 67
63 2c 20 63 68 61 72 20 2a 61 72 67 76 5b 5d 29 0d 0a 7b 0d
0a 20 20 20 20 46 49 4c 45 20 2a 66 70 3b 0d 0a 20 20 20 20
69 6e 74 20 69 3b 0d 0a 20 20 20 20 69 6e 74 20 63 68 72 3b
0d 0a 20 20 20 20 69 66 20 28 61 72 67 63 21 3d 32 29 20 7b
0d 0a 20 20 20 20 20 20 20 20 70 72 69 6e 74 66 28 22 2a 2a
20 65 72 72 6f 72 20 2a 2a 20 6e 6f 20 66 69 6c 65 6e 61 6d
65 20 21 21 20 5c 6e 22 29 3b 0d 0a 20 20 20 20 20 20 20 20
```

提出するテキストファイル書式は次の通りとします。（本講義での提出テキストファイルはすべてこの形式にします。）

課題１　ファイルの先頭 100 文字分をテキスト表示とバイナリ表示の両方を表示するプログラム

１．提出者名　4Jxx　高専太郎

２．課題概要

３．製作したプログラムソース typeFile.c
　#include .....
　：

４．実行結果
　この課題では実行結果は２つ
　（１）ソースファイルに対してこのプログラムを実行した時
　（２）実行形式ファイルに対してこのプログラムを実行した時

５．まとめと感想
　（１）苦労したポイント，理解してよかったポイント
　（２）課題の難易度について
　（３）提言
　（４）その他

２．ファイルのダンプ（第２週）

ファイルダンププログラムはファイルの内容をバイナリ表示（16 進表示）とテキスト表示を対応がわかるように表示するプログラムである。以下に表示例を示す。なお，テキスト表示では 0x21 以上 0x7e 以下のみ通常表示し，それ以外は小数点を表示することにします。次に示すのはプログラムソースファイル「typeFile100.c」をダンプしたところである。

```
typeFile100.c
          +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f  0123456789abcdef
0000000 : 2f 2a 20 62 69 6e 61 72 79 20 6f 70 65 6e 20 74  /*.binary.open.t
0000010 : 65 73 74 20 2a 2f 0d 0a 23 69 6e 63 6c 75 64 65  est.*/..#include
0000020 : 20 3c 73 74 64 69 6f 2e 68 3e 0d 0a 23 69 6e 63  .<stdio.h>..#inc
0000030 : 6c 75 64 65 20 3c 73 74 64 6c 69 62 2e 68 3e 0d  lude.<stdlib.h>.
0000040 : 0a 0d 0a 76 6f 69 64 20 6d 61 69 6e 28 69 6e 74  ...void.main(int
0000050 : 20 61 72 67 63 2c 20 63 68 61 72 20 2a 61 72 67  .argc,.char.*arg
0000060 : 76 5b 5d 29 0d 0a 7b 0d 0a 09 46 49 4c 45 20 2a  v[])..{...FILE.*
0000070 : 66 70 3b 0d 0a 09 69 6e 74 20 69 3b 0d 0a 09 69  fp;...int.i;...i
0000080 : 6e 74 20 63 68 72 3b 0d 0a 09 69 66 20 28 61 72  nt.chr;...if.(ar
0000090 : 67 63 21 3d 32 29 20 7b 0d 0a 09 09 70 72 69 6e  gc!=2).{....prin
00000a0 : 74 66 28 22 2a 2a 20 65 72 72 6f 72 20 2a 2a 20  tf("**.error.**.
00000b0 : 6e 6f 20 66 69 6c 65 6e 61 6d 65 20 21 21 20 5c  no.filename.!!.¥
00000c0 : 6e 22 29 3b 0d 0a 09 09 65 78 69 74 28 31 29 3b  n");....exit(1);
00000d0 : 0d 0a 09 7d 0d 0a 09 66 70 3d 66 6f 70 65 6e 28  ...}...fp=fopen(
00000e0 : 61 72 67 76 5b 31 5d 2c 22 72 62 22 29 3b 0d 0a  argv[1],"rb");..
00000f0 : 09 69 66 20 28 66 70 3d 3d 4e 55 4c 4c 29 20 7b  .if.(fp==NULL).{
0000100 : 0d 0a 09 09 70 72 69 6e 74 66 28 22 2a 2a 20 65  ....printf("**.e
0000110 : 72 72 6f 72 20 2a 2a 20 63 61 6e 27 74 20 6f 70  rror.**.can't.op
0000120 : 65 6e 20 74 68 65 20 66 69 6c 65 5c 6e 22 29 3b  en.the.file¥n");
0000130 : 0d 0a 09 09 65 78 69 74 28 31 29 3b 0d 0a 09 7d  ....exit(1);...}
0000140 : 0d 0a 09 70 72 69 6e 74 66 28 22 69 6e 70 75 74  ...printf("input
0000150 : 20 66 69 6c 65 20 6e 61 6d 65 3a 20 3c 3c 25 73  .file.name:.<<%s
0000160 : 3e 3e 5c 6e 22 2c 61 72 67 76 5b 31 5d 29 3b 0d  >>¥n",argv[1]);.
0000170 : 0a 09 66 6f 72 20 28 69 3d 30 3b 69 3c 31 30 30  ..for.(i=0;i<100
0000180 : 3b 69 2b 2b 29 20 7b 0d 0a 09 09 63 68 72 3d 66  ;i++).{....chr=f
0000190 : 67 65 74 63 28 66 70 29 3b 0d 0a 09 09 70 75 74  getc(fp);....put
00001a0 : 63 68 61 72 28 63 68 72 29 3b 0d 0a 09 7d 0d 0a  char(chr);...}..
00001b0 : 09 70 75 74 63 68 61 72 28 27 5c 6e 27 29 3b 0d  .putchar('¥n');.
00001c0 : 0a 09 66 63 6c 6f 73 65 28 66 70 29 3b 0d 0a 7d  ..fclose(fp);..}
00001d0 : 0d 0a                                            ..
```

課題　２　ここに挙げたような表示形式のファイルのダンププログラム dumpFile.c を作成しなさい。ただし対象ファイル名はコマンドラインから入力することとします。提出は tnct20 の各自の public_html にファイル名「dumpFile.txt」で公開することにします。書式は課題１と同じにします。

３．Windows Sound file (wav)の解析（第３週，第４週，第５週）

Windows のサウンドファイル形式の１つに wav 形式があります。この形式のファイルはヘッダ部とデータ部とで構成されている。ヘッダ部には wav 形式であることやサンプリングに関する情報が書かれている。またデータ部はサンプリングされたデータ列が保存されています。ヘッダ部は表３．１のような構成となっている。(ここに示す wav 形式に関する記述は情報工学科西村亮先生の資料から引用し，小坂が加筆しています。)

表３．１　wav ファイルの形式（情報工学科西村亮先生のまとめ）

先頭 らのバイ 数		項目番号	サイズ	内　　容	数　値　例 計　算　式 備　考
(10進)	(16		[byte]		
0	0	1	4	'RIFF'	
4	4	2	4	ファイルサイズ–8 [byte]	
8	8	3	4	'WAVE'	
12	C	4	4	'fmt '	tの後は空白（' '）
16	10	5	4	項目6～11の部分の合計サイズ	下記(3)を参照
20	14	6	2	フォーマットID	リニアPCM：1
22	16	7	2	チャネル数	monaural：1,stereo：2
24	18	8	4	サンプリング周波数[Hz]	
28	1C	9	4	平均データ速度[byte/s]	項目8×項目10
32	20	10	2	ブロックサイズ[byte/sample]	項目7×(項目11÷8)
34	22	11	2	1サンプル当たりのビット数[bit]	量 子 化 ビ ッ ト 数
＊	＊	＊1	2	項目＊2のサイズ	
＊	＊	＊2		ヘッダ拡張部（サイズ不定,*1参	
＊	＊	＊3	4	'fact'	
＊	＊	＊4	4	項目＊5のサイズ	
＊	＊	＊5		wav情報（サイズ不定,*4参照）	
36	24	12	4	'data'	
40	28	13	4	項目14のサイズ[byte]	
44	2C	14		データ #0	#0(先頭)から#1,#2…の（時間順）に量子化し振幅値を書き込む
・	・			・	
・	・			・	
・	・			・	

補足説明

(1) 項目 1，3，4，12 は ' ' に囲まれた文字列がそのまま書かれる．
(2) 項目 1，3，4，12，14，＊2，＊3 以外の項目は，それぞれのサイズの整数であり，いずれもバイナリで書かれる．2 バイト以上に渡る整数値は下位バイトから順に並んでいる。
(3) 項目 5 は，項目＊1 及び＊2 がある場合には，項目 6～11 及び＊1，＊2 の部分の合計サイズとなる．
(4) 項目 6 が 1（リニア PCM）の場合には，項目＊1 から＊5 までの項目は不要（ある場合もある）．これらの項目がある場合，その分だけ，項目 12 から 14 については先頭からのバイト数が変化する．例えば，項目＊1，＊2 がなく，項目＊3 から＊5 の各項目がある場合，項目 12，13，14 の先頭からのバイト数はそれぞれ，48，52，56（10 進）または 30，34，38（16 進）となる．
(5) 項目 11 は，項目 6 が 1 の場合には，8 または 16 となる．
(6) 項目 14 は，項目 11 が 8 の場合は 1[byte]の符号なし整数，16 の場合は 2[byte]の符号付き整数（リトルエンディアン）であり，いずれの場合もバイナリで書かれる．図 3.1 は 16bit の場合のサンプルデータの例であり，また図 3.2 は 8bit の場合のサンプルデータ例である。
(7) 項目 14 は，項目 7 が 2（ステレオ）の場合には，サンプル毎に左チャネル，右チャネルの順で記録される．
(8) 振幅値が 0（無音）の場合の量子化値は，量子化ビット数が 8[bit]の場合は 128（0x80），16[bit]の場合は 0（0x0000）である．

Wav ファイルの先頭部の例を次に示します。(情報工学科西村亮先生のまとめ)

例 1. 録音条件その 1

サンプリング周波数 f_s=22050[Hz]　　量子化ビット数=8[bit]　　モノラル
データ数=24938[sample]　　ファイルサイズ=24994[byte]

(1) ファイル先頭部分のダンプリスト

```
          00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F
00000000  52 49 46 46 9A 61 00 00 - 57 41 56 45 66 6D 74 20  RIFF.a..WAVEfmt.
00000010  10 00 00 00 01 00 01 00 - 22 56 00 00 22 56 00 00  ........"V.."V..
00000020  01 00 08 00 66 61 63 74 - 04 00 00 00 6A 61 00 00  ....fact....ja..
00000030  64 61 74 61 6A 61 00 00 - 80 80 80 80 80 80 80 80  dataja..........
    .                 .                       .           .
    .                 .                       .           .
    .                 .                       .           .
```

(2) ヘッダ部の内容

始点	終点	ダンプ内容	内容の読み	値	対応項目
0	3	52 49 46 46	左に同じ	'RIFF'	1
4	7	9A 61 00 00	00 00 61 9A	24986	2
8	0B	57 41 56 45	左に同じ	'WAVE'	3
0C	0F	66 6D 74 20	左に同じ	'fmt '	4
10	13	10 00 00 00	00 00 00 10	16	5
14	15	01 00	00 01	1	6
16	17	01 00	00 01	1	7
18	1B	22 56 00 00	00 00 56 22	22050	8
1C	1F	22 56 00 00	00 00 56 22	22050	9
20	21	01 00	00 01	1	10
22	23	08 00	00 08	8	11
24	27	66 61 63 74	左に同じ	'fact'	*3
28	2B	04 00 00 00	00 00 00 04	4	*4
2C	2F	6A 61 00 00	00 00 61 6A	24938	*5
30	33	64 61 74 61	64 61 74 61	'data'	12
34	37	6A 61 00 00	00 00 61 6A	24938	13
38		80	左に同じ	128	14:#0
39		80	〃	128	14:#1
3A		80	〃	128	14:#2
3B		80	〃	128	14:#3
3C		80	〃	128	14:#4
3D		80	〃	128	14:#5
3E		80	〃	128	14:#6
3F		80	〃	128	14:#7
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

例 2. 録音条件その２

サンプリング周波数 f_s=10000[Hz]　　量子化ビット数=16[bit]　　モノラル

データ数=6100[sample]　　ファイルサイズ=12244[byte]

(1) ファイル先頭部分のダンプリスト

```
          00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F
00000000  52 49 46 46 CC 2F 00 00 - 57 41 56 45 66 6D 74 20  RIFF./..WAVEfmt.
00000010  10 00 00 00 01 00 01 00 - 10 27 00 00 20 4E 00 00  .........'... N..
00000020  02 00 10 00 64 61 74 61 - A8 2F 00 00 04 00 06 00  ....data./......
00000030  08 00 0D 00 09 00 04 00 - FD FF F9 FF FD FF FC FF  ................
    .                 .                       .           .
    .                 .                       .           .
    .                 .                       .           .
```

(2) ヘッダ部の内容

始点	終点	ダンプ内容	内容の読み	値	対応項目
0	3	52 49 46 46	左に同じ	'RIFF'	1
4	7	CC 2F 00 00	00 00 2F CC	12236	2
8	0B	57 41 56 45	左に同じ	'WAVE'	3
0C	0F	66 6D 74 20	左に同じ	'fmt '	4
10	13	10 00 00 00	00 00 00 10	16	5
14	15	01 00	00 01	1	6
16	17	01 00	00 01	1	7
18	1B	10 27 00 00	00 00 27 10	10000	8
1C	1F	20 4E 00 00	00 00 4E 20	20000	9
20	21	02 00	00 02	2	10
22	23	01 00	00 01	16	11
24	27	64 61 74 61	64 61 74 61	'data'	12
28	2B	A8 2F 00 00	00 00 2F A8	12200	13
2C	2D	04 00	00 04	4	14:#0
2E	2F	06 00	00 06	6	14:#1
30	31	08 00	00 08	8	14:#2
32	33	0D 00	00 0D	13	14:#3
34	35	09 00	00 09	9	14:#4
36	37	04 00	00 04	4	14:#5
38	39	FD FF	FF FD	-3	14:#6
3A	3B	F9 FF	FF F9	-7	14:#7
3C	3D	FD FF	FF FD	-3	14:#8
3E	3F	FC FF	FF FC	-4	14:#9
・	・	・	・	・	・
・	・	・	・	・	・
・	・	・	・	・	・

1 サンプルデータが 16bit の場合と 8bit の場合の例を図３．１，３．２に示す。

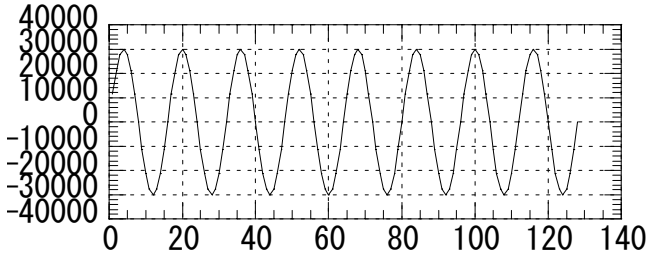

図 3.1　1 サンプルデータが 16bit で出来ている場合のデータ列の例
値域は-32768～32767 であり，無信号時の値は 0 である。

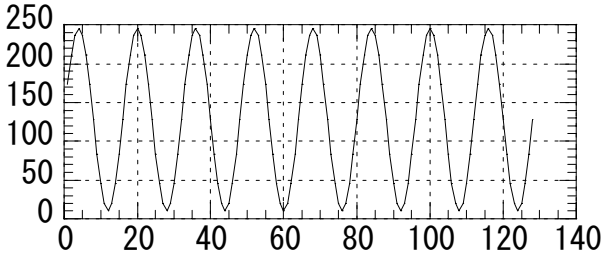

図 3.2　1 サンプルデータが 8bit で出来ている場合のデータ列の例
値域は 0～255 であり，無信号時の値は 128 である。

課題　3

（1）wav ファイルを課題 2 で作成した dumpFile でダンプし，各項目内容を調べて表にまとめなさい。

（2）wav ファイルのヘッダ部分を読み出し，コンソール画面に表示するプログラム extractInfo.c を作成しなさい。ただし表 3.1 の＊1～＊4 は表示しない事にします。

プログラムおよび結果をファイル名「extractInfo.txt」で公開しなさい。書式は課題 1 と同じにします。また dumpFile.c を用いたダンプ結果は「まとめと感想」に書きなさい。

この課題では次の 2 つの wav ファイルを対象として用いてよい。他の wav ファイルを対象としてもよい。

hello.wav　readytoprint.wav

ヒント１：

先頭の 4 バイトの文字列'RIFF'，ファイルサイズ-8，文字列'WAVE'，　文字列'fmt 'を確認する。「項目 6～11，＊1＊2」のサイズを覚えておき，項目 6～11 を読み取る。項目 6～11 までは本来 16 バイトであるがそれを超えた場合は＊1＊2 があるという意味なのでこの部分は読み飛ばす。次に 4 文字読んで文字列'data'でなく，文字列'fact'だったら，＊4 を読んで，＊5 のバイト数を知り，＊5 を読み飛ばす。次に 4 文字読んで文字列'data'だったら，次の項目 13 でサウンドデータの総バイト数を知ることが出来る。

ヒント２：

次の３つの関数の役割を考えなさい。

```
void getString4(char *buff)
{
    fread(buff,1,4,fpi);
}

static short int getShortInt(void)
{
    unsigned char buff[2];
    short int x=0;
    fread(buff,1,2,fpi);
    x=(short int)buff[0]+(((short int)buff[1])<<8);
    return x;
}

static unsigned long int getLongInt(void)
{
    unsigned char buff[4];
    unsigned long int x=0;
    fread(buff,1,4,fpi);
    x=(unsigned long int)buff[0]+(((unsigned long int)buff[1])<<8)
        +(((unsigned long int)buff[2])<<16)+(((unsigned long int)buff[3])<<24);
    return x;
}
```

ヒント３　次の構造体は wave ファイルのヘッダに対応しているので用いなさい。

```
typedef struct {
    char str_riff[4];
    unsigned long int fsize_8;/*ファイルサイズ-8 [byte]*/
    char str_wave[4];
    char str_fmt[4];
    unsigned long int info_block_size;/*format～num_bits_sample の部分の合計サイズ[byte]*/
    unsigned short int wave_format;/*フォーマット ID リニア PCM：1*/
    unsigned short int number_of_channels;/*チャネル数 nc モノラル：1　ステレオ：2*/
```

```
    unsigned long int sampling_frequency;/*サンプリング周波数 fs [Hz] */
    unsigned long int rate;/*平均データ速度 r=fs*sb [byte/s]*/
    unsigned short int blocksize;/*ブロックサイズ sb=(nc*INT(nb/8))[byte/sample time]*/
    unsigned short int num_bits_sample;/*1 サンプル当たりのビット数 nb [bit]*/
    char str_data[4];
    unsigned long int binarydatasize;/*バイナリデータのサイズ[byte]*/
} pcmheader_t;/*ヘッダ構造体の定義*/
```

課題　4

次の課題のプログラムを作りなさい。提出用のファイル名は modifyWaveFile.txt とし，作成したプログラムをこの中に公開しなさい。書式は課題 1 と同様にします。サンプル wave ファイル readytoprint.wav を与えるので，作業結果の wav ファイルを File1.wav，File2.wav，File3.wav，File4.wav，File5.wav として公開しなさい。なおすべてのデータはモノラルと仮定しなさい。また，WAVE ファイルのヘッダを読み込むための関数，ヘッダを書き出すための関数を作り，これを利用するプログラムとしなさい。

（1）wav ファイルを読み込み，読み込みデータ中の先頭 1 秒だけのデータのみで出来ている wav ファイルに変換するプログラム shortenWave.c を作りなさい。
（2）wav ファイルを読み込み，wav ファイルのヘッダ部分，サウンドデータをテキストファイルに変換するプログラム WaveToText.c を作りなさい。ただしサウンドデータは 32767～-32768 の範囲で出力されるようにしなさい。
（2）wav ファイルを読み込み，サンプルデータを逆順にした wav ファイルに変換するプログラム reverseWave.c を作りなさい。「あいうえお」は「おえういあ」に聞こえるでしょう。
（3）wav ファイルを読み込み，読み込んだサウンドを 2 回繰り返す wav ファイルに変換するプログラム repeatWave.c を作りなさい。再生時間は 2 倍になるはずです。
（4）wav ファイルを読み込み，読み込んだサウンドを倍速再生する wav ファイルに変換するプログラム fastPlayWave.c を作りなさい。ただしサンプリング周波数を変更していけません。
（5）wav ファイルを読み込み，読み込んだサウンドを半速再生する wav ファイルに変換するプログラム slowPlayWave.c を作りなさい。ただしサンプリング周波数を変更していけません。

正しい形式の wave 形式かどうかをチェックする「checkwav」というプログラムがこの Web ページの最後の部分に「おまけ」として載せてあるので検査しなさい
（1）windows では～.wav のアイコンを checkwav.exe にドラッグ＆ドロップして実行
（2）UNIX では「>checkwav ～.wav」で実行

ヒント　次の 3 つの関数の役割を調べなさい。

```
void putString4(char *buff)
{
    fwrite(buff,1,4,fpo);
}

void putShortInt(short int x)
{
    unsigned char buff[2];
    buff[0]=x&0xff;
    x>>=8;
    buff[1]=x&0xff;
    fwrite(buff,1,2,fpo);
}

void putLongInt(unsigned long int x)
{
    unsigned char buff[4];
```

```
    buff[0]=(unsigned char)(x&0xff);
    x>>=8;
    buff[1]=(unsigned char)(x&0xff);
    x>>=8;
    buff[2]=(unsigned char)(x&0xff);
    x>>=8;
    buff[3]=(unsigned char)(x&0xff);
    fwrite(buff,1,4,fpo);
}
```

４．正弦波データによる wav ファイルの作成（第６週，第７週）

課題　５

与えられた周波数の単一正弦波トーンを表すモノラルの構成の wav ファイルをつくるプログラムを作りなさい。

（１）トーンの周波数，トーンの継続時間（再生時間），サンプリング周波数，1 サンプルデータ当たりのビット数が固定値で，それぞれ 1200Hz，3sec，8000Hz，8bit であるプログラム createSinWave1.c

（２）トーンの周波数，トーンの継続時間（再生時間），サンプリング周波数，1 サンプルデータ当たりのビット数が設定ファイルで与えられるプログラム createSinWave2.c。

設定ファイルの書式は以下のようである。使用されているキーワードはこのまま使うこと。

```
// SinSetting.txt
// コメントはこのように書く
ToneFrequency           1200     //[Hz]
Duration                   3     //[sec]
SamplingFrequency       8000     //[Hz]
QuantizationBitLength      8     //[bit]
```

提出前にチェック用プログラム powerspectrum で出力ファイルをチェックしなさい。

設定した周波数に近い離散周波数のパワー値が一番大きければよい。

提出はファイル名 createSinWave.txt とし，書式はこれまでと同様にしなさい。また（１）で作成された wav ファイルを createdsinwave.wav の名前で公開しなさい。

ヒント　3 秒間の 1250Hz のトーンをサンプリング周波数 8000Hz で作ることを考えよう。振幅は 15000 とする。

このトーンを時間の関数で表すと，次式となる。

$y = a \sin (2 \pi f t)$，ただし a = 15000，f = 1250

サンプリング周波数が 8000Hz なので，サンプルデータは 1 秒間に 8000 個になる。3 秒間を構成するサンプルデータ総数は 3×8000=24000 個となる。

サンプリング周波数が 8000Hz なので，サンプルデータ間の時間，すなわちサンプリング周期は 1/8000 秒である。先頭から i 番目のサンプルデータは，先頭から i/8000 秒である。

このトーンの i 番目のサンプルデータを数式で表すと次式となる。

y[i] = 15000 * sin (2 * 3.1415926535 * 1250 * i / 8000)

ヒント　設定ファイルを読む関数を作ると簡単になる。

```
/*
// SinSetting.txt
// コメントはこのように書く
ToneFrequency           1200     //[Hz]
Duration                   3     //[sec]
SamplingFrequency       8000     //[Hz]
QuantizationBitLength      8     //[bit]
Amplitude                 80     //[%]
```

```
*/

typedef struct {
    int ToneFrequency;          //1200  [Hz]
    int Duration;               //   3  [sec]
    int SamplingFrequency;      //8000  [Hz]
    int QuantizationBitLength;  //   8  [bit]
    int Amplitude;              //  80  [%]
} setting_t;

setting_t getSetting(char *fname)
{
    FILE *fp;
    char buff[512];
    char key[128];
    int value;
    setting_t st;
    fp=fopen(fname,"r");
    if (fp==NULL) {
        printf("** ERROR -- can't open file [%s] **¥n",fname);
        exit(1);
    }
    while (fgets(buff,512,fp)!=NULL) {
        key[0]=0;//key のクリア
        sscanf(buff,"%s %d",key,&value);
        if (!strcmp(key,"ToneFrequency")) st.ToneFrequency=value;
        else if (!strcmp(key,"Duration")) st.Duration=value;
        else if (!strcmp(key,"SamplingFrequency")) st.SamplingFrequency=value;
        else if (!strcmp(key,"QuantizationBitLength")) st.QuantizationBitLength=value;
        else if (!strcmp(key,"Amplitude")) st.Amplitude=value;
    }
    printf("ToneFrequency = %d¥n", st.ToneFrequency);
    printf("Duration = %d¥n", st.Duration);
    printf("SamplingFrequency = %d¥n", st.SamplingFrequency);
    printf("QuantizationBitLength = %d¥n", st.QuantizationBitLength);
    printf("Amplitude = %d¥n", st.Amplitude);
    return st;
}
```

注意　振幅はフルスケールの 80%程度にしなさい。

５．wav ファイルのパワースペクトルを求める（第８週）

FFT（fast Fourier Transform）演算を用いるとサウンドデータのパワースペクトルを求めることが出来る。パワースペクトルとは，音の波形の中にどのような周波数成分のパワーがどれだけ含まれているかを示す量である。次のプログラムは，ファイル fftfunc.c とファイル fftfunc.h とともに，生成した時間領域データ（音の周波数＝1200Hz，サンプリング周波数＝8000Hz）の周波数分析を行ない，パワースペクトルを求めるプログラムである。ただし，FFT を用いたパワースペクトルの計算は，関数 calcPowerspectrum()内に隠されているので，興味があったら内部を読んでみるとよい。実行すると時間領域の波形とパワースペクトルがそれぞれファイル timedomain.txt と powerspectrum.txt として出力されるので，読んで考察しなさい。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
```

```
#include "fftfunc.h"

#define SIZE 512

main()
{
    double timedomaindata[SIZE];/*時間領域波形格納領域*/
    powerspec_t power[SIZE/2+1];/*パワースペクトル格納領域*/
    long int SamplingFrequency=8000;/*サンプリング周波数*/
    int ToneFrequency=1200;/*音の周波数*/
    FILE *fp;
    int i;
    /*時間領域波形の生成*/
    for (i=0;i<SIZE;i++) {
        timedomaindata[i]=30000.*sin(2*3.14159265*ToneFrequency*i/SamplingFrequency);
    }
    /*パワースペクトルの計算*/
    calcPowerspectrum(timedomaindata,SIZE,SamplingFrequency,power);

    /*時間領域波形の出力*/
    if ((fp=fopen("timedomain.txt","w"))==NULL) {
        printf("can't open text file¥n");exit(1);
    }
    for (i=0;i<SIZE;i++) {
        fprintf(fp,"%15lf%20lf¥n",(double)i/SamplingFrequency,timedomaindata[i]);
    }
    fclose(fp);
    /*パワースペクトルの出力*/
    if ((fp=fopen("powerspectrum.txt","w"))==NULL) {
        printf("can't open text file¥n");exit(1);
    }
    for (i=0;i<SIZE/2+1;i++) {
        fprintf(fp,"%5d, %20lf¥n",power[i].frequency,power[i].power);
    }
    fclose(fp);
}
```

課題　6

wav ファイルを読んで，ファイル先頭の 512 個のデータのパワースペクトルを求めるプログラム calculatePowerspectrum.c を作りなさい。
提出はファイル名 calculatePowerspectrum.txt とし，書式はこれまでと同様にしなさい。

６．DTMF の wav ファイルの生成（第９週，第１０週，第１１週）

ＤＴＭＦとは、Dual Tone Multiple Frequency の略で、プッシュホン電話機が電話局に電話番号を送る時に使う、ピポパ音のことです。１つのキャラクタ（プッシュホン電話機の１つのボタン）に対応する音は，２つの周波数の正弦波音の和になっています。例えばキャラクタ「１」の時は周波数 697Hz と 1209Hz の２つの正弦波音の和になっています。(697Hz と 1209Hz の２つの正弦波音を同時に発生する）DTMF の信号と表現するキャラクタは次のように対応しています。

表６．１　NTT の DTMF

周波数	1209	1336	1477
697	1	2	3
770	4	5	6
852	7	8	9
941	*	0	#

表６．２　欧米の DTMF

周波数	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

課題　7

（1）1 キャラクタのみの DTMF 信号 wav ファイル(継続時間 1 秒,モノラル 8kHz サンプリング, 8 ビット）を生成するプログラム makeDTMF1.c を作りなさい。ファイル名および，キャラクタはコマンドラインの引数から与えるものとします。
例 キャラクタ'3'に対応する 697Hz と 1477Hz の正弦波音を発生する wav ファイル dtmf.wav を作る時は次のように使えるようにします。
>makeDTMF dtmf.wav 3

うまく wav ファイルが出来たかどうかは，チェック用プログラム detectDTMF で確認しなさい。
またチェック用プログラム powerspectrum で出力ファイルをチェックし，予想した周波数のパワー値が大きいことを確認しなさい。

（2）複数キャラクタの信号 wav ファイル（設定値およびキャラクタ列はテキストファイルで与えられる）を生成するプログラム makeDTMF2.c を作りなさい。
設定用テキストファイルの内容例は次のとおりである。

```
//コメントはこのように書く
//makeDTMF2.exe用データファイル
SamplingFrequency       8000      //[Hz] 8000,11025,22050,44100のどれか
QuantizationBitLength   8         //[bit]  8 or 16
Amplitude               30        //[%] 0..100
LeaderDuration          1000      //[msec]前後につけるリーダ部空白長さ
ToneDuration            100       //[msec]DTMF信号継続時間
SilenceDuration         100       //[msec]無信号時間
DTMFData              "0426685111" //符号データ
//符号データは「0123456789*#ABCD」中から取り出した複数個のキャラクタで構成される
//電話用のDTMFの規定では
//信号の最小継続時間：  50ms以上
//無信号区間の最小継続時間：  30ms以上
```

設定用テキストファイルの名前（たとえば setting.txt），出力される wav ファイルの名前（たとえば dtmf.wav）はコマンドラインから
>makeDTMF2 setting.txt dtmf.wav
のように入力されるようにする。
うまく wav ファイルが出来たかどうかは，チェック用プログラム detectDTMF で確認しなさい。

提出ファイル名は makeDTMF.txt とし，課題（1）（2）の両方を提出しなさい。書式はこれまでと同様にしなさい。また，（1）の出力ファイルは dtmf1.wav，（2）の出力ファイルは dtmf2.wav として公開しなさい。

ここまで演習が進んでくると，wave ファイルの形式に関する手続きの部分を毎回作るのはわずらわしい。よく使う関数群は自分のライブラリとして再利用可能な形にしておくのがよい。小坂の wave ファイル利用ライブラリと使用例を与えるので，読み下して，気に入った学生は利用しても良い。
copyWave.c wave.c wave.h 説明は wave.h に書いてある。また wave.c，wave.h の別な使い方例として createWave.c と createSweepWave.c を公開する。(createWave.c は実は課題５のヒントになっている。)

ヒント　ダイアル文字（0,1,2,...）を 2 つの周波数に変換する関数の例（使い方は研究して下して下さい）

```
/*conversion of a dial symbol to DTMF frequencies*/
#include <stdio.h>
#include <string.h>
#include <ctype.h>

typedef struct {
    double f1; /*周波数 1*/
```

```
    double f2; /*周波数 2*/
} DTMFfrequency_t;

/*ダイアル文字を２つの周波数に変換する関数*/
/*入力：ダイアル文字　出力：2 つの周波数*/
DTMFfrequency_t convertDialSymbol(char sym)
{
    /* DTMF frequency table
     Hz 1209 1336 1477 1633
    697   1    2    3    A
    770   4    5    6    B
    852   7    8    9    C
    941   *    0    #    D
    */
    const static char DTMFchar[]="147*2580369#ABCD";
    const static int duplexFrequency[][2]= {
        {697,1209},{770,1209},{852,1209},{941,1209},
        {697,1336},{770,1336},{852,1336},{941,1336},
        {697,1477},{770,1477},{852,1477},{941,1477},
        {697,1633},{770,1633},{852,1633},{941,1633}
    };
    int num;
    DTMFfrequency_t fr;
    sym=toupper(sym);
    num=strchr(DTMFchar,sym)-DTMFchar;
    if (num<0) {
        fr.f1=fr.f2=0;
    } else {
        fr.f1=duplexFrequency[num][0];
        fr.f2=duplexFrequency[num][1];
    }
    return fr;
}

/*ダイアル文字を２つの周波数に変換する関数*/
/*「*」の替わりに「p」,「#」の替わりに「q」でも動作するように拡張*/
/*入力：ダイアル文字　出力：2 つの周波数*/
DTMFfrequency_t convertDialSymbol_ex(char sym)
{
    sym=toupper(sym);
    if (sym=='P') sym='*';
    if (sym=='Q') sym='#';
    return convertDialSymbol(sym);
}

/*関数 convertDialSymbol_ex()を検査する main*/
/*文字 e,f,g ではエラーの確認*/
int main()
{
    char myDialSymbol[]="123456789pOqABCDefg*#";
    DTMFfrequency_t fr;
    int i=0;
    char ch;
    while (myDialSymbol[i]) {
        ch=myDialSymbol[i];
```

```
        fr=convertDialSymbol_ex(ch);
        printf("symbol:%c --> frequencies: %4.0lf, %4.0lf¥n",ch,fr.f1,fr.f2);
        i++;
    }
    return 0;
}

/* 実行結果

symbol:1 --> frequencies:  697, 1209
symbol:2 --> frequencies:  697, 1336
symbol:3 --> frequencies:  697, 1477
symbol:4 --> frequencies:  770, 1209
symbol:5 --> frequencies:  770, 1336
symbol:6 --> frequencies:  770, 1477
symbol:7 --> frequencies:  852, 1209
symbol:8 --> frequencies:  852, 1336
symbol:9 --> frequencies:  852, 1477
symbol:p --> frequencies:  941, 1209
symbol:0 --> frequencies:  941, 1336
symbol:q --> frequencies:  941, 1477
symbol:A --> frequencies:  697, 1633
symbol:B --> frequencies:  770, 1633
symbol:C --> frequencies:  852, 1633
symbol:D --> frequencies:  941, 1633
symbol:e --> frequencies:    0,    0
symbol:f --> frequencies:    0,    0
symbol:g --> frequencies:    0,    0
symbol:* --> frequencies:  941, 1209
symbol:# --> frequencies:  941, 1477

*/
```

７．　DTMF の wav ファイルの解析（第１２週，第１３週，第１４週）

課題　８

「６」で用いたチェック用 detectDTMF は，パワースペクトルの構成を見ながら，どのキャラクタに対する DTMF 信号であるかをチェックしている。FFT サイズをどのくらいに設定したらよいか考えながら，自分で detectDTMF を作りなさい。

解析対象の wave ファイルは次のものを行ない，実行結果にどこのレベルまで対応できたか書きなさい。

難易度	レベル１（やさしい）	レベル２	レベル３	レベル４（難しい）
特徴	純粋 DTMF	DTMF+50Hz ノイズ	DTMF+ノイズ	仕様内高速 DTMF

提出ファイル名は detectDTMF.txt としなさい。書式はこれまでと同様にしなさい。

８．　FIR ディジタルフィルタ（第１５週）

ある wav ファイル（例えば input.wav）のデータから新しい wav ファイル（例えば output.wav）のデータを次のように生成することを考えよう。

次の作業を wav ファイル input.wav 内のサンプルデータの個数回繰り返す｛
  input.wav から 1 サンプルデータ読み出し，x とする。
  「y=func(x)」を計算する
  y を output.wav に書き出す
｝

ただし関数は double func(double x)であり，例えばこの関数が

```
double func(double x)
{
    static double x3=0;
    static double x2=0;
    static double x1=0;
    static double x0=0;
    double y;
    x3=x2;x2=x1;x1=x0;x0=x;
    y=0.25*x0+0.25*x1+0.25*x2+0.25*x3;
    return y;
}
```

ならば，この関数は，言葉で表すと「出力値は，入力値および過去 3 回の入力値の平均値である」あるいは「「出力値は，現在を含む 4 回の連続した入力値の平均値である」となる。static 宣言された変数は最初の 1 回目の関数呼び出し時にだけ初期化（この例では 0 が代入される）され，2 回目以降の関数呼び出し時は，前回の呼び出し時の終了時の値を保っている。

さて，入力された変数を格納する変数を x0,x1,x2,x3 としたが，配列にすることが出来る。また，各変数にかける値がこの例ではすべて 0.25 であったが一般的にはこれも配列とすることが出来る。

この関数をかきなおすと，次のようになる。

```
double func(double x)
{
    static double xb[4]={0,0,0,0};
    static double w[4]={0.25,0.25,0.25,0.25};
    double y;
    int i;
    for (i=3; 0<i; i--) {
        xb[i]=xb[i-1];
    }
    xb[0]=x;
    y=0.0;
    for (i=0; i<4; i++) {
        y=+w[i]*xb[i];
    }
    return y;
}
```

この関数はディジタルフィルタと呼ばれるれ，出力値は，現在以前の N 個の入力値の一次結合となっている。w[n]は重み係数（タップ係数）と呼ばれ，すべての要素が同じ値とは限らない。

課題９　wav ファイルを読み込み，エコーをつけた wav ファイルに変換するプログラム addEcho.c を作りなさい。

ここで使用するディジタルフィルタの係数は全長 2400 とし，800 番目の係数 w[799]=0.6，1600 番目の係数 w[1599]=0.3，2400 番目の係数 w[2399]=0.1 とし，その他の係数はすべて 0.0 として作りなさい。

データのシフト量が多くなったので，xb にリングバッファ（配列だが，先頭とお尻が連続している）の利用を検討しなさい。提出ファイル名は digitalFilter.txt，digitalFilter.wav としなさい。書式はこれまでと同様にしなさい。