配布ファイル：

1. Wave Sample：　 aaa.wav
2. Binary Editor： Sterling.exe
3. サンプルソース： wave.cpp

**WAVE(.WAV) File Format**

* 4バイト 'R' 'I' 'F' 'F'
* 4バイト これ以下のバイト数 (= ファイルサイズ - 8)
* 8バイト 'W' 'A' 'V' 'E' 'f' 'm' 't' ' '(←スペース)
* 4バイト WAVEfmt欄のバイト数，つねに 16 (10 00 00 00)
* 2バイト データ形式 (PCM: 01 00)
* 2バイト チャネル数 (モノ: 01 00 ステレオ: 02 00)
* 4バイト サンプリングレート (44100Hz なら 44 AC 00 00)
* 4バイト バイト／秒 (44100Hz ステレオ 16ビット なら 10 B1 02 00)
* 2バイト バイト／サンプル×チャネル数 (ステレオ 16ビット なら 04 00)
* 2バイト ビット／サンプル (16ビット なら 10 00)
* 4バイト 欄の名前 ('d' 'a' 't' 'a'，'f' 'a' 'c' 't' など)
* 4バイト この欄のバイト数 n
* nバイト 欄の内容

欄は 'd' 'a' 't' 'a' だけが必須。欄 'd' 'a' 't' 'a' の内容には次の形式でデータが入っている。

* ステレオなら L R L R ... の順
* 8ビットデータは 0〜255 (無信号は 128)
* 16ビットデータは -32768〜+32767 (無信号は 0)

数値はすべて Intel バイトオーダで並べる。

**サンプルデータ（aaa.wav）のヘッダー**

アドレス 内容 意味

00 ～ 03 52 49 46 46 'R' 'I' 'F' 'F'

04 ～ 07 20 D3 A1 00 これ以下のバイト数 (= ファイルサイズ - 8)

08 ～ 0F 57 41 56 45 66 6D 74 20 'W' 'A' 'V' 'E' 'f' 'm' 't' ' '(←スペース)

10 ～ 13 10 00 00 00 WAVEfmt欄のバイト数，つねに 16 (10 00 00 00)

14 ～ 15 01 00 データ形式 (PCM: 01 00)

16 ～ 17 01 00 チャネル数、モノ: 01 00 ( == 1 )

18 ～ 1B 44 AC 00 00 サンプリングレート：44100Hz

1C ～ 1F 88 58 01 00 バイト／秒

20 ～ 21 02 00 バイト／サンプル×チャネル数

22 ～ 23 10 00 ビット／サンプル (16ビット)

24 ～ 27 64 61 74 61 欄の名前 ('d' 'a' 't' 'a')

28 ～ 2B FC D2 A1 00 この欄のバイト数

2C ～ この欄の内容

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// File Name: wave.c

// Usage: Wave File にフィルタを施す

#include "stdafx.h"

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

typedef struct {

double r;

double i;

} complex;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Low Pass Filter:

void LowPassFilter(complex \*x, int N)

{

return;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Hi Pass Filter:

void HiPassFilter(complex \*x, int N)

{

return;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Band Pass Filter:

void BandPassFilter(complex \*x, int N)

{

return;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// ファイルから４バイト読む

unsigned long get\_ulong(FILE \*fp)

{

unsigned char s[4];

if (fread(s, 4, 1, fp) != 1) {

fprintf(stderr, "Read error\n");

exit(1);

}

return s[0] + 256LU \* (s[1] + 256LU \* (s[2] + 256LU \* s[3]));

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// ファイルから２バイト読む

unsigned get\_ushort(FILE \*fp)

{

unsigned char s[2];

if (fread(s, 2, 1, fp) != 1) {

fprintf(stderr, "Read error\n");

exit(1);

}

return s[0] + 256U \* s[1];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Wave File fp\_inのヘッダーを取り出して、fp\_outに書き込む

void retrieve\_header(FILE \*fp\_in, FILE \*fp\_out)

{

int i, channels, bits;

unsigned long len;

unsigned char s[10];

printf("\nRetrieving Wave File Header: \n\n");

if (fread(s, 4, 1, fp\_in) != 1) { // "RIFF"の読み書き

printf("Read error\n");

exit(1);

}

if (memcmp(s, "RIFF", 4) != 0) {

printf("Not a 'RIFF' format\n");

exit(1);

}

fwrite("RIFF", 4, 1, fp\_out);

len = get\_ulong(fp\_in);

printf("[RIFF] (%lu bytes)\n", len);

fwrite(&len, 4, 1, fp\_out);

if (fread(s, 8, 1, fp\_in) != 1) { // "WAVEfmt "の読み書き

printf("Read error\n");

exit(1);

}

if (memcmp(s, "WAVEfmt ", 8) != 0) {

printf("Not a 'WAVEfmt ' format\n");

exit(1);

}

fwrite("WAVEfmt ", 8, 1, fp\_out);

len = get\_ulong(fp\_in); // 10 00 00 00 (16)を読み込書き

printf("[WAVEfmt ] (%lu bytes)\n", len);

if (len != 16) {

printf("Length of 'WAVEfmt ' must be 16!\n");

return;

}

fwrite(&len, 4, 1, fp\_out);

len = get\_ushort(fp\_in);

printf(" Data type = %u (1 = PCM)\n", len ); // データ形式：PCM 01 00 を読み書き

fwrite(&len, 2, 1, fp\_out);

channels = get\_ushort(fp\_in); // チャンネル数を読み書き

printf(" Number of channels = %u (1 = mono, 2 = stereo)\n", channels);

fwrite(&channels, 2, 1, fp\_out);

len = get\_ulong(fp\_in);

printf(" Sampling rate = %luHz\n", len); // Sampling Rate を読み書き

fwrite(&len, 4, 1, fp\_out);

len = get\_ulong(fp\_in);

printf(" Bytes / second = %lu\n", len); // Bytes/Second を書き込む

fwrite(&len, 4, 1, fp\_out);

len = get\_ushort(fp\_in);

printf(" Bytes x channels = %u\n", len); // Bytes x Channel を書き込む

fwrite(&len, 2, 1, fp\_out);

bits = get\_ushort(fp\_in); // Bits/Second を書き込む

　　printf(" Bits / sample = %u\n", bits);

fwrite(&bits, 2, 1, fp\_out);

// ファイルのフォーマットをチェックする: モノ、16ビット/サンプルだけ

if( (channels != 1) || (bits != 16) ) {

printf("Wrong File Format!\n");

exit(1);

}

while (fread(s, 4, 1, fp\_in) == 1) {

len = get\_ulong(fp\_in);

s[4] = 0;

printf("[%s] (%lu bytes)\n", s, len);

if (memcmp(s, "data", 4) == 0) break; 　// "data" チャンクを探す

for (i = 0; i < len; i++)

printf("%02x ", fgetc(fp\_in));

printf("\n");

　}

fwrite("data", 4, 1, fp\_out); // "data"を書き込む

fwrite(&len, 4, 1, fp\_out); // data size を書き込む

return;

}

int main(void)

{

char in\_file\_name[20] = "aaa.wav"; // サンプルデータのファイル名

char out\_file\_name[20] = "low\_pass\_aaa.wav";

// char out\_file\_name[20] = "hi\_pass\_aaa.wav";

// char out\_file\_name[20] = "band\_pass\_aaa.wav";

unsigned char s[10];

void (\*p\_Filter)(complex \*, int); // 使用するフィルターを選択

FILE \*in\_file;

FILE \*out\_file;

if ((in\_file = fopen(in\_file\_name, "rb")) == NULL) {

printf("Can not open file\n");

exit(1);

}

if ( (out\_file = fopen(out\_file\_name, "wb")) == NULL) {

printf("Can not open file\n");

exit(1);

}

// サンプルファイルを読み込み、そのまま書き出す

while( fread(s, 1, 1, in\_file) == 1) {

fwrite(&s, 1, 1, out\_file);

}

fclose(in\_file);

fclose(out\_file);

return 0;

}