(一)作業題目

任意指定 k 頻聲波，在不同採樣頻率之下，記錄成 WAV 檔

(二)實作方法

- **背景公式：**
  - 某頻率為 k (Hz) 的 cosine 波，在時間為 t (sec) 時，振幅為

f(k,t) = cos (2p k t), k 與 t 皆為實數, e.g. k=440, t=5.67

- **四項標準測試題**(40%)：
  - 以下四項 wav 檔，由助教指定頻率 k，當場產生
  - 產生之後以右鍵檢查檔案內容，並播放聆聽，每項完成得 10%
  - 四個 wav 檔長度皆為 8 秒，單音軌，以下 i 值由 0 開始

S1.wav 設定採樣頻率 44100，各個採樣值為 S1(i)= f(k, i/44100)

S2.wav 設定採樣頻率 22050，各個採樣值為 S2(i)= f(k, i/22050)

S3.wav 設定採樣頻率 11025，各個採樣值為 S3(i)= f(k, i/11025)

S4.wav 模擬採樣頻率 5512.5，檔頭仍設定採樣頻率為 11025，但各個採樣值複製延用一次，S4(i)= f(k, [i/2]/11025), 亦即 S4(0)=S4(1), S4(2)=S4(3)…

我們運用 wavefile 聲音函式庫完成

程式碼如下:

設定 wavefile 的 Header

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <errno.h>
#include "wavfile.h"

struct wavfile_header {
    char    riff_tag[4];
    int     riff_length;
    char    wave_tag[4];
    char    fmt_tag[4];
    int     fmt_length;
    short   audio_format;
    short   num_channels;
    int     sample_rate;
    int     byte_rate;
    short   block_align;
    short   bits_per_sample;
    char    data_tag[4];
    int     data_length;
};

```

建立一個新文件，它會自動把標準的 WAV 標頭放入文件中。(如下圖所示)

wavfile_open(文件名稱,取樣率)

文件建立成功→返回一個指向 FILE 對象的指標。

文件建立不成功→返回 null

```
FILE * wavfile_open( const char *filename,int sampsrate)
{
    struct wavfile_header header;

    int samples_per_second = sampsrate;
    int bits_per_sample = 16;

    strncpy(header.riff_tag,"RIFF",4);
    strncpy(header.wave_tag,"WAVE",4);
    strncpy(header.fmt_tag,"fmt ",4);
    strncpy(header.data_tag,"data",4);

    header.riff_length = 0;
    header.fmt_length = 16;
    header.audio_format = 1;
    header.num_channels = 1;
    header.sample_rate = samples_per_second;
    header.byte_rate = samples_per_second*(bits_per_sample/8);
    header.block_align = bits_per_sample/8;
    header.bits_per_sample = bits_per_sample;
    header.data_length = 0;

    FILE * file = fopen(filename,"wb+");
    if(!file) return 0;
    fwrite(&header,sizeof(header),1,file);
    fflush(file);
    return file;
}
```

把 data 寫入文件， wavfile_write(wavfile_open 返回的 FILE 對象的指標，波形數據，寫入的樣本數)

```
void wavfile_write( FILE *file, short data[], int length )
{
    fwrite(data,sizeof(short),length,file);
}
```

完成寫入 wavfile。當聲音完成後，需要調用 wavfile_close，否則文件會沒辦法使用。

```
void wavfile_close( FILE *file )
{
    int file_length = ftell(file);

    int data_length = file_length - sizeof(struct wavfile_header);
    fseek(file,sizeof(struct wavfile_header) - sizeof(int),SEEK_SET);
    fwrite(&data_length,sizeof(data_length),1,file);

    int riff_length = file_length - 8;
    fseek(file,4,SEEK_SET);
    fwrite(&riff_length,sizeof(riff_length),1,file);

    fclose(file);
}
```

輸入頻率/採樣率

```
int main()
{
    double frequency;
    double frequency2=1480.0;
    double frequency3=2200.0;
    int volume = 10000;
    int sampsrate;
    int wav;

    printf("Frequency : ");
    scanf("%lf", &frequency);
    printf("choose the .wav to generate (1~5) : ");
    scanf("%d", &wav);
```

設定四個 wav 檔，採樣頻率分別為 44100/22050/11025/11025

```
91      if((wav>=1)&&(wav<=6)){
92
93          printf("Frequency : ");
94      scanf("%lf", &frequency);
95
96      switch (wav) {
97          case 1:
98              sampsrate = 44100;
99              break;
100
101         case 2:
102             sampsrate = 22050;
103             break;
104
105         case 3:
106             sampsrate = 11025;
107             break;
108
109         case 4:
110             sampsrate = 11025;
111             break;
```

利用所需頻率的 cosine 波產生波行陣列，而 wav4 設定會將每個採樣值複製一次，即 S4(0)=S4(1), S4(2)=S4(3)…。

```
111     int num_samples = (sampsrate*8);
112     short waveform[num_samples];
113     int length = num_samples;
114
115     if(wav==4){
116         int i;
117         for(i=0;i<length;i+=2) {
118             double t = (double) i / sampsrate;
119             waveform[i] = volume*cos(frequency*t*2*M_PI);
120             waveform[i+1] = waveform[i];
121         }
122     }
```

```
141     else{
142
143         int i;
144         for(i=0;i<length;i++) {
145             double t = (double) i / sampsrate;
146             waveform[i] = volume*cos(frequency*t*2*M_PI);
147         }
```

將波形寫到 sound1.wav 中：

```
151     FILE * f = wavfile_open("sound1.wav",sampsrate);
152     if(!f) {
153         printf("couldn't open the file for writing: %s",strerror(errno));
154         return 1;
155     }
156
157     wavfile_write(f,waveform,length);
158     wavfile_close(f);
159
160     return 0;
161 }
```

Bonus:

■ **提高作業完成度**(20%)：

■ 例如: 可產生 2 個以上頻率的混音、可產生指定頻段內的連續升/降頻…

設定 wav5 在檔案長度一半(4s)之前 做三種頻率的混音，後半段則將其倒過來，達到降頻的結果

```
135     else if(wav==5){
136         int i;
137
138         for(i=0;i<length/2;i+=3) {
139             double t = (double) i / sampsrate;
140             waveform[i] = volume*cos(frequency*t*2*M_PI);
141             waveform[i+1] = volume*cos(frequency2*t*2*M_PI);
142             waveform[i+2] = volume*cos(frequency3*t*2*M_PI);
143         }
144         for(i=length/2;i<length;i++) {
145             double t = (double) i / sampsrate;
146             waveform[i] = volume*cos(frequency*t*2*M_PI);
147             waveform[i+1] = volume*cos(frequency2*t*2*M_PI);
148             waveform[i+2] = volume*cos(frequency*t*2*M_PI);
149         }
```

將 wav6 設定為兩個不同頻率的混音。一個為輸入的頻率，另一個為事先設定好的頻率(frequency2=1480)，將兩者相加即可達到混音的效果。

```
152     else if(wav ==6){
153         int i;
154         for(i=0;i<length;i++) {
155             double t = (double) i / sampsrate;
156             waveform[i] = (volume*cos(frequency*t*2*M_PI)+volume*cos(frequency2*t*2*M_PI))/2;
157             //waveform[i+1] = volume*cos(frequency2*t*2*M_PI);
158         }
159     }
```

■ **進行採樣定理的實驗**(10%)：

- ■ 設計實驗，對比呈現某 k 頻聲波在採樣不足時所發生的狀況
- ■ 預測當採樣不足時，該 k 頻聲波將被記錄為相對低頻的什麼頻率? 並驗證

以 wav4 為例，我們將頻率設定為 5000，採樣頻率自 11025 逐漸降低，發現音頻也隨之逐漸變的低沉且微弱，實測 wav4 在 sample rate=1100 時的人耳可聽的到，但當 sample rate=1000 時聲音會消失

```
109         case 4:
110             sampsrate = 1100;
111             break;
```

(三)分工與進度

王妤霈:30%

謝瑞筑:40%

李羽喬:30%

(四)結果測試

將 S1、S2、S3、S4 做比較，S1、S2、S3 頻率固定(設為 1000)，採樣頻率變動，聽起來聲音沒有太大的差別，S4 頻率一樣設為 1000，但與前三個聲音比較起來，聲音較高。對於 S4，頻率固定，對採樣頻率做調整，採樣頻率越高，聽起來的聲音就越高。

經過多次測試與比較，發現採樣頻率不足時，就會聽不到聲音。

(五)參考資料

https://github.com/RobertDurfee/Wave

https://mropengate.blogspot.com/2015/04/sampling-theoremaliasing-effect.html

(六)心得報告

謝瑞筑：

其實在了解整個 wav 的檔案格式之後，就可以很簡單地寫出這個程式，就跟網路的封包格式一樣，把 Header 的內容先填好，例如取樣頻率、通道數等等，最後再用我們所需要的背景公式把值存到 buffer 裡，並且把 payload 存到 wav 檔裡就能產生我們所需要的檔案了。比較困難的地方其實就是要想還有什麼方式可以做出不同的變化，以及如何利用老師上課教到的一些轉換來實作。

王妤霈：

以前對於改變頻率或採樣頻率會發生什麼改變都是從課堂上得知，並沒有實際操作過，透過這次的作業，自己設定一些參數，就能發現參數改變時，會對原本的聲音造成什麼變化，透過實作驗證上課學到的一些定理，能夠讓自己印象更深刻。

李羽喬：

經過這次的實作才能驗證波型的不同和頻率不同最後對聲音都會造成變化，這次作業用的是 cos 波，過程中要去思考如何在固定振福、時間和採樣頻率的狀況下，並結合老師上課所教的運算公式，分析出不同頻率的產生的音檔差別，受益良多。