Programas 6 y 7 – FFT y FFTI

funciones.h

```
#ifndef __FUNCIONES_H__
#define __FUNCIONES_H__
   //Librerías de C
   #include <stdio.h>
   #include <stdlib.h>
   #include <math.h>
   #include <string.h>
   //Librería que contiene los máximos y mínimos de los diferentes tipos de datos en c
   #include <limits.h>
   //Libreria para conocer tiempo de ejecución
   #include <time.h>
   //Metodos
   void leerCabeceras(char**);
   void escribirArchivo(short*,short*);
   void leerMuestras(short*);
   void leerMuestras2Canales(short*,short*);
   void convertirFloat(short*,float*,float*);
   void convertirShort(short*,short*,float*,float*);
   //Cabeceras
   int chunkid;
   int chunksize;
   int format;
   int subchunk1id;
   int subchunk1size;
   short audioformat;
   short numchannels;
   int samplerate;
   int byterate;
   short blockalign;
   short bitspersample;
   int subchunk2id;
   int subchunk2size;
   //Archivo
   FILE* entrada;
   FILE* salida;
   //Variables para muestras
   short muestra;
   int total_muestras_originales;
   int total_muestras;
   short headers[37];
   //Métodos TDF
   #define PI acos(-1.0)//Defino la constante PI
```

```
    void calcularFFT(short*);
    void calcularFFTI(short*,short*);
    int calcularNuevoNumeroMuestras(int);
    void intercambiar(float**,int,int);
    //Inversión de bits
    #define SWAP(x,y) do {typeof(x) _x = x;typeof(y) _y = y;x = _y;y = _x;} while(0)
    //Variables para obtener tiempo de ejecución
    clock_t inicio, final;
    double total;
#endif
```

fft.c

```
#include"funciones.h"
int main(int argc, char *argv[]){
    //Leo las cabeceras
    leerCabeceras(argv);
    //Defino variables
    total_muestras_originales=subchunk2size/blockalign;
    printf("Total muestras originales:%d\n",total_muestras_originales);
    //Necesitamos que el total de muestras sea una potencia de 2
    total_muestras=calcularNuevoNumeroMuestras(total_muestras_originales);
    printf("Nuevo total de muestras:%d\n", total_muestras);
    short *muestras=(short *)malloc(total_muestras * sizeof(short));
    //Leo las muestras
    leerMuestras(muestras);
    //Calculo la FFT
    calcularFFT(muestras);
}
void leerCabeceras(char ** argv){
    entrada = fopen(argv[1], "rb");
    salida=fopen(argv[2],"wb");
    if(!entrada) {
        perror("\nFile opening failed");        exit(0);    }
    fread(&chunkid,sizeof(int),1,entrada);
    fread(&chunksize,sizeof(int),1,entrada);
    fread(&format,sizeof(int),1,entrada);
    fread(&subchunk1id,sizeof(int),1,entrada);
    fread(&subchunk1size,sizeof(int),1,entrada);
    fread(&audioformat,sizeof(short),1,entrada);
    fread(&numchannels,sizeof(short),1,entrada);
    fread(&samplerate,sizeof(int),1,entrada);
    fread(&byterate,sizeof(int),1,entrada);
    fread(&blockalign,sizeof(short),1,entrada);
    fread(&bitspersample,sizeof(short),1,entrada);
    fread(&subchunk2id,sizeof(int),1,entrada);
```

```
    fread(&subchunk2size,sizeof(int),1,entrada);
}
void leerMuestras(short *muestras){
    int i=0;
    while (feof(entrada) == 0)
    {
        if(i<total_muestras_originales){
            fread(&muestra,sizeof(short),1,entrada);
            muestras[i]=muestra;
            i++;
        }else{
            fread(&headers,sizeof(short),37,entrada);
            break;
        }
    }
    //Ajuste por si las muestras originales no fueron potencia de dos
    if(total_muestras_originales<total_muestras){
        for (i = total_muestras_originales; i < total_muestras; i++){
            muestras[i]=0;        }
    }
    fclose(entrada);
}
void escribirArchivo(short* muestrasRe,short* muestrasIm){
    //Escribo el archivo
    fwrite(&chunkid,sizeof(int),1,salida);
    fwrite(&chunksize,sizeof(int),1,salida);
    fwrite(&format,sizeof(int),1,salida);
    fwrite(&subchunk1id,sizeof(int),1,salida);
    fwrite(&subchunk1size,sizeof(int),1,salida);
    fwrite(&audioformat,sizeof(short),1,salida);
    fwrite(&numchannels,sizeof(short),1,salida);
    fwrite(&samplerate,sizeof(int),1,salida);
    fwrite(&byterate,sizeof(int),1,salida);
    fwrite(&blockalign,sizeof(short),1,salida);
    fwrite(&bitspersample,sizeof(short),1,salida);
    fwrite(&subchunk2id,sizeof(int),1,salida);
    fwrite(&subchunk2size,sizeof(int),1,salida);
    //Ahora escribo las muestras
    int i=0;
    for(i=0;i<total_muestras;i++){
        fwrite(&muestrasRe[i],sizeof(short),1,salida);
        fwrite(&muestrasIm[i],sizeof(short),1,salida);
    }
    //Y por último los headers de goldwave
    for(i=0;i<37;i++){
        fwrite(&headers[i],sizeof(short),1,salida);
    }
```

```
    fclose(salida);
}
void calcularFFT(short *muestras){
    //Aquí va el algoritmo para la FFT
    float *Xre=(float *)malloc(total_muestras * sizeof(float));
    float *Xim=(float *)malloc(total_muestras * sizeof(float));
    int i;
    //Convierto las muestras de short a float
    convertirFloat(muestras, Xre, Xim);
    //Iniciar relog
    inicio = clock();
    //FFT
    int j, k, fk, m, n, ce, c, w;
    float arg, seno, coseno, tempr, tempi;
    //Bit reversal
    m=log((float)total_muestras)/log(2.0);
    j=w=0;
    for (i = 0; i < total_muestras; i++){
                if (j>i){
                        SWAP(Xre[i],Xre[j]);
                        SWAP(Xim[i],Xim[j]);
                }
                w=total_muestras/2;
                while(w>=2 && j>=w){
                        j-=w;
                        w>>=1;
                }
                j+=w;    }    ce=m;    c=0;
    //Mariposas
    for (i = 0; i < m; i++) {
                for(j = 0; j < (int)pow(2,ce-1); j++){
                        n = (int)pow(2,i);
                        for(k = 0; k < n; k++){
                                fk=k*(int)pow(2,ce-1);
                                coseno=cos((-1)*2*PI*fk/total_muestras);
                                seno=sin((-1)*2*PI*fk/total_muestras);
                                tempr=Xre[c+n];
                                Xre[c+n]=(Xre[c+n]*coseno) - (Xim[c+n]*seno);
                                Xim[c+n]=(Xim[c+n]*coseno) + (tempr*seno);
                                tempr=(Xre[c]+Xre[c+n])/2;
                                tempi=(Xim[c]+Xim[c+n])/2;
                                Xre[c+n]=(Xre[c]-Xre[c+n])/2;
                                Xim[c+n]=(Xim[c]-Xim[c+n])/2;
                                Xre[c]=tempr;
                                Xim[c]=tempi;
                                c++;
                        }
```

```
                    c += n;
                }
                c = 0;
                ce -= 1;
    }
    short *Reales=(short *)malloc(total_muestras * sizeof(short));
    short *Imaginarias=(short *)malloc(total_muestras * sizeof(short));
    //Obtener tiempo e imprimir
    final = clock();
    total = (double)(final - inicio) / CLOCKS_PER_SEC;
    printf("Tiempo de ejecucion: %f\n", total);
    //Regreso las muestras calculadas a short
    convertirShort(Reales,Imaginarias,Xre,Xim);
    //La salida ahora sera un archivo tipo estereo (2 canales)
    //Por lo cual hay que cambiar el numero de canales del archivo
    //y todas las demas cabeceras que dependan de esta
    chunksize-=subchunk2size;
    numchannels*=2;
    byterate*=numchannels;
    blockalign*=numchannels;
    subchunk2size=total_muestras*blockalign;
    chunksize+=subchunk2size;
    escribirArchivo(Reales,Imaginarias);
}
int calcularNuevoNumeroMuestras(int total){
    if ((total & (total-1))==0){
        puts("Ya es potencia de 2");
    }else{
        puts("No es potencia de 2");
        int i;
        i=(int)ceil((float)log(total_muestras_originales)/(float)log(2));
        printf("i:%d\n", i);
        total=pow(2,i);    }
    return total;}
void convertirFloat(short *muestras, float *Xre, float *Xim){
    int i;
    for (i = 0; i < total_muestras; i++){
        Xre[i]=(float)muestras[i]/(float)(SHRT_MAX);
        Xim[i]=0.0;    }}
void convertirShort(short *Reales, short *Imaginarias, float *Xre, float *Xim){
    int i;
    for (i = 0; i < total_muestras; i++){
        Reales[i]=Xre[i]*(SHRT_MAX);
        Imaginarias[i]=Xim[i]*(SHRT_MAX);
    }
}
```

ffti.c

```
#include"funciones.h"
int main(int argc, char *argv[])
{
   //Leo las cabeceras
   leerCabeceras(argv);
   //Defino variables
   total_muestras=subchunk2size/blockalign;
   printf("Total muestras %d\n",total_muestras);
   short *muestrasRe=(short *)malloc(total_muestras * sizeof(short));
   short *muestrasIm=(short *)malloc(total_muestras * sizeof(short));
   //Leo las muestras
   leerMuestras2Canales(muestrasRe,muestrasIm);
   //Calculo la TDF
   calcularFFTI(muestrasRe,muestrasIm);
}
void leerCabeceras(char ** argv){
   entrada = fopen(argv[1], "rb");
   salida=fopen(argv[2],"wb");
   if(!entrada) {
      perror("\nFile opening failed");
      exit(0);
   }
   fread(&chunkid,sizeof(int),1,entrada);
   fread(&chunksize,sizeof(int),1,entrada);
   fread(&format,sizeof(int),1,entrada);
   fread(&subchunk1id,sizeof(int),1,entrada);
   fread(&subchunk1size,sizeof(int),1,entrada);
   fread(&audioformat,sizeof(short),1,entrada);
   fread(&numchannels,sizeof(short),1,entrada);
   fread(&samplerate,sizeof(int),1,entrada);
   fread(&byterate,sizeof(int),1,entrada);
   fread(&blockalign,sizeof(short),1,entrada);
   fread(&bitspersample,sizeof(short),1,entrada);
   fread(&subchunk2id,sizeof(int),1,entrada);
   fread(&subchunk2size,sizeof(int),1,entrada);
}
void leerMuestras2Canales(short *muestrasRe,short* muestrasIm){
   int i=0;
   while (feof(entrada) == 0){
      if(i<total_muestras){
         fread(&muestrasRe[i],sizeof(short),1,entrada);
         fread(&muestrasIm[i],sizeof(short),1,entrada);
         i++;
```

```
            //printf("Muestra %s: %d\n",i,muestras[i-1]);
        }else{
            fread(&headers,sizeof(short),37,entrada);
            break;
        }
    }
}
void escribirArchivo(short* muestrasRe,short* muestrasIm){
    //Escribo el archivo
    fwrite(&chunkid,sizeof(int),1,salida);
    fwrite(&chunksize,sizeof(int),1,salida);
    fwrite(&format,sizeof(int),1,salida);
    fwrite(&subchunk1id,sizeof(int),1,salida);
    fwrite(&subchunk1size,sizeof(int),1,salida);
    fwrite(&audioformat,sizeof(short),1,salida);
    fwrite(&numchannels,sizeof(short),1,salida);
    fwrite(&samplerate,sizeof(int),1,salida);
    fwrite(&byterate,sizeof(int),1,salida);
    fwrite(&blockalign,sizeof(short),1,salida);
    fwrite(&bitspersample,sizeof(short),1,salida);
    fwrite(&subchunk2id,sizeof(int),1,salida);
    fwrite(&subchunk2size,sizeof(int),1,salida);
    //Ahora escribo las muestras
    int i=0;
    for(i=0;i<total_muestras;i++){
        fwrite(&muestrasRe[i],sizeof(short),1,salida);
        fwrite(&muestrasIm[i],sizeof(short),1,salida);
    }
    //Y por último los headers de goldwave
    for(i=0;i<37;i++){
        fwrite(&headers[i],sizeof(short),1,salida);
    }
}
void calcularFFTI(short *Re, short *Im){
    //Aquí va el algoritmo para la FFT
    int i;
    float *Xre=(float *)malloc(total_muestras * sizeof(float));
    float *Xim=(float *)malloc(total_muestras * sizeof(float));
    for (i = 0; i < total_muestras; i++){
            Xre[i]=(float)Re[i]/(float)SHRT_MAX;
            Xim[i]=(float)Im[i]/(float)SHRT_MAX;
    }
    //Inicio relog
    inicio = clock();
    //FFTI
    int j, k, fk, m, n, ce, c, w;
    float arg, seno, coseno, tempr, tempi;
```

```
//Bit reversal
m=log((float)total_muestras)/log(2.0);
j=w=0;
for (i = 0; i < total_muestras; i++){
   if (j>i){
      SWAP(Xre[i],Xre[j]);
      SWAP(Xim[i],Xim[j]);
   }
   w=total_muestras/2;
   while(w>=2 && j>=w){
      j-=w;
      w>>=1;
   }
   j+=w;
}
ce=m;
c=0;
//Mariposas
for (i = 0; i < m; i++) {
   for(j = 0; j < (int)pow(2,ce-1); j++){
      n = (int)pow(2,i);
      for(k = 0; k < n; k++){
         fk=k*(int)pow(2,ce-1);
         coseno=cos(2*PI*fk/total_muestras);
         seno=sin(2*PI*fk/total_muestras);
         tempr=Xre[c+n];
         Xre[c+n]=(Xre[c+n]*coseno) - (Xim[c+n]*seno);
         Xim[c+n]=(Xim[c+n]*coseno) + (tempr*seno);
         tempr=(Xre[c]+Xre[c+n]);
         tempi=(Xim[c]+Xim[c+n]);
         Xre[c+n]=(Xre[c]-Xre[c+n]);
         Xim[c+n]=(Xim[c]-Xim[c+n]);
         Xre[c]=tempr;
         Xim[c]=tempi;
         c++;
      }
      c += n;
   }
   c = 0;
   ce -= 1;
}
//Obtener tiempo e imprimir
final = clock();
total = (double)(final - inicio) / CLOCKS_PER_SEC;
printf("Tiempo de ejecucion: %f\n", total);
short *Reales=(short *)malloc(total_muestras * sizeof(short));
short *Imaginarias=(short *)malloc(total_muestras * sizeof(short));
```

```
    convertirShort(Reales,Imaginarias,Xre,Xim);
    //Escribo el resultado en el archivo
    escribirArchivo(Reales,Imaginarias);
}
void convertirFloat(short *muestras, float *Xre, float *Xim){
    int i;
    for (i = 0; i < total_muestras; i++){
        Xre[i]=(float)muestras[i]/(float)SHRT_MAX;
        Xim[i]=0.0;
        //printf("Muestra Xre: %f\n", Xre[i]);
        //printf("Muestra Xim: %f\n", Xim[i]);
    }
}
void convertirShort(short *Reales, short *Imaginarias, float *Xre, float *Xim){
    int i;
    for (i = 0; i < total_muestras; i++){
        Reales[i]=Xre[i]*SHRT_MAX;
        Imaginarias[i]=Xim[i]*SHRT_MAX;
    }
}
```