Programas 4 y 5 – TDF y TDFI

funciones.h

```
#ifndef __FUNCIONES_H__
#define __FUNCIONES_H__
   //Librerías de C
   #include <stdio.h>
   #include <stdlib.h>
   #include <math.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*);
   //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;
   int modo;
   //Variables para muestras
   short muestra;
   int total_muestras;
   short headers[37];
   //Métodos TDF
   #define PI acos(-1.0)//Defino la constante PI
   void calcularTDF(short*);
   void calcularTDF1(short*);
   void calcularTDF2(short*);
   void calcularTDF3(short*);
   void calcularTDFI(short*,short*);#endif
```

tdf.c

```
#include"funciones.h"
int main(int argc, char *argv[]){
   //Leo las cabeceras
   leerCabeceras(argv);
   //Defino variables
   total_muestras=subchunk2size/blockalign;
   //puts("Voy a crear el arreglo de muestras");
   short *muestras=(short *)malloc(total_muestras * sizeof(short));
   printf("Total de muestras:%d\n", total_muestras);
   //puts("Ya cree el arreglo de muestras");
   //Leo las muestras
   //puts("Voy a leer las muestras del archivo");
   leerMuestras(muestras);
   //puts("Ya lei las muestras del archivo");
   //Calculo la TDF
   if (modo==0){
      //Transformada normal
      //puts("Voy a calcular la TDF");
      calcularTDF(muestras);
   }else if (modo==1){
      //Canal izquierdo señal original, derecho magnitud
      calcularTDF1(muestras);
   }else if(modo==2){
      //Canal izquierdo parte real TDF, derecho magnitud
      calcularTDF2(muestras);
   }else if(modo==3){
      //Canal izquierdo magnitud TDF, derecho fase TDF
      calcularTDF3(muestras);
   }
}
void leerCabeceras(char ** argv){
   entrada = fopen(argv[1], "rb");
   salida=fopen(argv[2],"wb");
   modo=atoi(argv[3]);
   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){
            fread(&muestra,sizeof(short),1,entrada);
            muestras[i]=muestra;
            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 calcularTDF(short* muestras){
   //Aquí va el algoritmo para la TDF
   short *Xre=(short *)malloc(total_muestras * sizeof(short));
   short *Xim=(short *)malloc(total_muestras * sizeof(short));
   //Variables para obtener tiempo de ejecución
   clock_t inicio, final;
   double total;
   inicio = clock();
   //Algoritmo TDF
   int n,k;
   for (k = 0; k < total_muestras; k++)
   {
      Xre[k]=0;
      Xim[k]=0;
      for (n = 0; n < total_muestras; n++)
      {
         Xre[k]+=(muestras[n]/total_muestras)*cos(2*k*n*PI/total_muestras);
         Xim[k]-=(muestras[n]/total_muestras)*sin(2*k*n*PI/total_muestras);
      }
   }
   //Obtener tiempo e imprimir
   final = clock();
   total = (double)(final - inicio) / CLOCKS_PER_SEC;
   printf("Tiempo de ejecucion: %f\n", total);
   //La salida ahora sera un archivo tipo estereo (2 canales)
   //Por lo cual hay que cambiar el número de canales del archivo y todas las demás
cabeceras que dependan de esta
   chunksize-=subchunk2size;
   numchannels*=2;
   byterate*=numchannels;
   blockalign*=numchannels;
   subchunk2size*=numchannels;
   chunksize+=subchunk2size;
   escribirArchivo(Xre,Xim);
}
void calcularTDF1(short *muestras){
   //Canal izquierdo señal original, derecho magnitud
   short *Xre=(short *)malloc(total_muestras * sizeof(short));
   short *Xim=(short *)malloc(total_muestras * sizeof(short));
   short *magnitud=(short *)malloc(total_muestras * sizeof(short));
   //Variables para obtener tiempo de ejecución
   clock_t inicio, final;
   double total;
   inicio = clock();
   //Algoritmo TDF
   int n,k;
```

```
    for (k = 0; k < total_muestras; k++)
    {
       Xre[k]=0;
       Xim[k]=0;
       for (n = 0; n < total_muestras; n++)
       {
          Xre[k]+=(muestras[n]/total_muestras)*cos(2*k*n*PI/total_muestras);
          Xim[k]-=(muestras[n]/total_muestras)*sin(2*k*n*PI/total_muestras);
       }
    }
    //Obtener tiempo e imprimir
    final = clock();
    total = (double)(final - inicio) / CLOCKS_PER_SEC;
    printf("Tiempo de ejecucion: %f\n", total);
    //Ahora calcularé la magnitud de la transformada
    for (k = 0; k < total_muestras; k++)
    {
       magnitud[k]=sqrt(pow(Xre[k],2)+pow(Xim[k],2));
    }
    //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*=numchannels;
    chunksize+=subchunk2size;
    escribirArchivo(muestras,magnitud);
}
void calcularTDF2(short *muestras){
    //Canal izquierdo parte real transformada, derecho magnitud
    short *Xre=(short *)malloc(total_muestras * sizeof(short));
    short *Xim=(short *)malloc(total_muestras * sizeof(short));
    short *magnitud=(short *)malloc(total_muestras * sizeof(short));
    //Variables para obtener tiempo de ejecución
    clock_t inicio, final;
    double total;
    inicio = clock();
    //Algoritmo TDF
    int n,k;
    for (k = 0; k < total_muestras; k++)
    {
       Xre[k]=0;
       Xim[k]=0;
       for (n = 0; n < total_muestras; n++)
       {
```

```
            Xre[k]+=(muestras[n]/total_muestras)*cos(2*k*n*PI/total_muestras);
            Xim[k]-=(muestras[n]/total_muestras)*sin(2*k*n*PI/total_muestras);
        }
    }
    //Obtener tiempo e imprimir
    final = clock();
    total = (double)(final - inicio) / CLOCKS_PER_SEC;
    printf("Tiempo de ejecucion: %f\n", total);
    //Ahora calcularé la magnitud de la transformada
    for (k = 0; k < total_muestras; k++)
    {
       magnitud[k]=sqrt(pow(Xre[k],2)+pow(Xim[k],2));
    }
    //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*=numchannels;
    chunksize+=subchunk2size;
    escribirArchivo(Xre,magnitud);
}
void calcularTDF3(short *muestras){
    //Canal izquierdo magnitud, derecho fase
    short *Xre=(short *)malloc(total_muestras * sizeof(short));
    short *Xim=(short *)malloc(total_muestras * sizeof(short));
    short *magnitud=(short *)malloc(total_muestras * sizeof(short));
    short *fase=(short *)malloc(total_muestras * sizeof(short));
    //Variables para obtener tiempo de ejecución
    clock_t inicio, final;
    double total;
    inicio = clock();
    //Algoritmo TDF
    int n,k;
    for (k = 0; k < total_muestras; k++){
       Xre[k]=0;
       Xim[k]=0;
       for (n = 0; n < total_muestras; n++){
          Xre[k]+=(muestras[n]/total_muestras)*cos(2*k*n*PI/total_muestras);
          Xim[k]-=(muestras[n]/total_muestras)*sin(2*k*n*PI/total_muestras);
       }
    }
    //Obtener tiempo e imprimir
    final = clock();
    total = (double)(final - inicio) / CLOCKS_PER_SEC;
```

```
    printf("Tiempo de ejecucion: %f\n", total);
    //Ahora calcularé la magnitud de la transformada
    for (k = 0; k < total_muestras; k++){
        magnitud[k]=sqrt(pow(Xre[k],2)+pow(Xim[k],2));
    }
    //La fase
    float valor=180.0/PI;
    //printf("Valor %f",valor);
    for (k = 0; k < total_muestras; k++){
        if (magnitud[k]>1000){
            //atan nos devuelve un valor en radianes, hay que pasarlo a grados
            if(Xre[k]==0){
                if (Xim[k]==0){
                    fase[k]=0;
                }else if (Xim[k]<0){//Xim es negativa
                    fase[k]=-90; checarlo
                }else{
                    fase[k]=90;
                }
            }else{
                fase[k]=atan(Xim[k]/Xre[k])*valor;
            }
        }else{
            fase[k]=0;
        }
        fase[k]=(fase[k]*SHRT_MAX)/180;//Para que se pueda ver en goldwave
        printf("%d\n", fase[k]);
    }
    //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*=numchannels;
    chunksize+=subchunk2size;
    escribirArchivo(magnitud,fase);
}
```

tdfi.c

```
#include"funciones.h"
int main(int argc, char *argv[]){
   //Leo las cabeceras
   leerCabeceras(argv);
   //Defino variables
   total_muestras=subchunk2size/blockalign;
   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
   calcularTDFI(muestrasRe,muestrasIm);}
void leerCabeceras(char ** argv){
   //Primero voy a recordar a leer archivos
   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++;
      }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 calcularTDFI(short* muestrasRe,short* muestrasIm){
    //Aquí va el algoritmo para la TDF
    short *Xre=(short *)malloc(total_muestras * sizeof(short));
    short *Xim=(short *)malloc(total_muestras * sizeof(short));
    //Variables para obtener tiempo de ejecución
    clock_t inicio, final;    double total;    inicio = clock();
    //Algoritmo TDFI
    int n,k;
    double algo;
    for (k = 0; k < total_muestras; k++){
        Xre[k]=0;
        Xim[k]=0;
        for (n = 0; n < total_muestras; n++){
            //Lo que va dentro del coseno y seno.
            algo=(2*k*n*PI)/(total_muestras);
            //Calculo las partes real e imaginarias
            Xre[k]+=muestrasRe[n]*cos(algo)-muestrasIm[n]*sin(algo);
            Xim[k]+=muestrasRe[n]*sin(algo)+muestrasIm[n]*cos(algo);      }    }
    //Obtener tiempo e imprimir
    final = clock();
    total = (double)(final - inicio) / CLOCKS_PER_SEC;
    printf("Tiempo de ejecucion: %f\n", total);
    //La salida seguirá siendo un archivo tipo estereo (2 canales) por lo cual no hay que
alinear nada
    escribirArchivo(Xre,Xim);}
```