**INSTITUTO POLITÉCNICO NACIONAL**

**Escuela Superior de Cómputo (ESCOM)**

**PROFESOR**: GUTIERREZ ALDANA EDUARDO.

**MATERIA**: Teoría de Comunicaciones y Señales.

**Trabajo:** Códigos de programas desarrollados.

**Alumno:**

* Monroy Martos Elioth.

**Grupo:** 3CM6

## Programa 1- Cambiar volumen a la mitad

funciones.h

|  |
| --- |
| #ifndef \_\_FUNCIONES\_H\_\_  #define \_\_FUNCIONES\_H\_\_  *//Librerías de C*  #include <stdio.h>  #include <stdlib.h>  *//Métodos*  void leerCabeceras(char\*\*);  void leerMuestras(short\*);  void dividirMuestras(short\*,float);  void escribirArchivo(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;  *//Archivos*  FILE\* entrada;  FILE\* salida;  *//Variables para muestras*  short muestra;  int total\_muestras;  short headers[37];  #endif |

half.c

|  |
| --- |
| #include"funciones.h"  int main(int argc, char \*argv[]){  *//Leo las cabeceras*  leerCabeceras(argv);  *//Defino variables*  total\_muestras=subchunk2size/blockalign;  short \*muestras=(short \*)malloc(total\_muestras \* **sizeof**(short));  *//Leo las muestras*  leerMuestras(muestras);  *//Divido el volumen a la mitad*  dividirMuestras(muestras,.5);  *//Escribo el archivo*  escribirArchivo(muestras);  }  void leerCabeceras(char \*\* argv){  *//Abrir archivos*  entrada = fopen(argv[1], "rb");  salida=fopen(argv[2],"wb");  **if**(!entrada) {  perror("**\n**File 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 dividirMuestras(short \*muestras, float factor){  int i;  **for** (i = 0; i < total\_muestras; i++)  {  muestras[i]\*=factor;  }  }  void escribirArchivo(short\* muestras){  *//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(&muestras[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);  }  } |

## Programa 2 – Convolución

funciones.h

|  |
| --- |
| #ifndef \_\_FUNCIONES\_H\_\_  #define \_\_FUNCIONES\_H\_\_  *//Librerías de C*  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  *//Métodos*  void leerCabeceras(char\*\*);  void leerMuestras(short\*);  void escribirArchivo(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;  *//Variables para muestras*  short muestra;  int total\_muestras;  short headers[37];  *//Filtro*  #define PI acos(-1.0)*//Defino la constante PI*  #define TOTAL\_COEFICIENTES 20  double filtro[TOTAL\_COEFICIENTES];*//Arreglo para el filtro*  double suma\_filtro;  double e;  void calcularFiltro();  void calcularNuevasMuestras(short\*);  #endif |

filtro\_psb.c

|  |
| --- |
| #include"funciones.h"  int main(int argc, char \*argv[]){  *//Leo las cabeceras*  leerCabeceras(argv);  *//Defino variables*  int i=0;  total\_muestras=subchunk2size/blockalign;  short \*muestras=(short \*)malloc(total\_muestras \* **sizeof**(short));  *//Leo las muestras*  leerMuestras(muestras);  calcularFiltro();  calcularNuevasMuestras(muestras); }  void leerCabeceras(char \*\* argv){  entrada = fopen(argv[1], "rb");  salida=fopen(argv[2],"wb");  **if**(!entrada) {  perror("**\n**File 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++;  }**else**{  fread(&headers,**sizeof**(short),37,entrada);  **break**;  }  }  }  void escribirArchivo(short\* muestras){  *//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(&muestras[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 calcularFiltro(){  e=exp(1);*//Aquí obtengo el valor de e*  int i;  **for** (i= 0; i < TOTAL\_COEFICIENTES; i++){  filtro[i]=(2000\*PI)\*pow(e,((-1)\*2000\*PI\*i)/44100);  suma\_filtro+=filtro[i];  printf("%f**\n**", filtro[i]);}}  void calcularNuevasMuestras(short\* muestras){  *//Algoritmo para la convolución usando Input Side Algorithm*  short \*nuevas\_muestras=(short \*)malloc(total\_muestras \* **sizeof**(short));  int i,j;  *//Primero inicializar a cero el arreglo*  **for** (i = 0; i < total\_muestras; i++){  nuevas\_muestras[i]=0;  }  *//Ahora si el algoritmo*  **for** (i = 0; i < total\_muestras; i++){  **for** (j = 0; j < TOTAL\_COEFICIENTES; j++){  nuevas\_muestras[i+j]+=(muestras[i]\*filtro[j])/suma\_filtro; }  }  escribirArchivo(nuevas\_muestras);  } |

## Programa 3 – Producto

funciones.h

|  |
| --- |
| #ifndef \_\_FUNCIONES\_H\_\_  #define \_\_FUNCIONES\_H\_\_  *//Librerías de C*  #include <stdio.h>  #include <stdlib.h>  *//Librería que contiene los máximos y mínimos de los diferentes tipos de datos en c*  #include <limits.h>  *//Metodos*  void leerCabeceras(char\*\*);  void leerMuestras(short\*,short\*);  void leerMuestras2Canales(short\*,short\*,short\*,short\*);  void escribirArchivo(short\*,int);  void escribirArchivo2Canales(short\*,short\*,int);  void calcularProducto(short\*,short\*);  void calcularProductoComplejos(short\*,short\*,short\*,short\*);  *//Cabeceras primer archivo*  int chunkid1;  int chunksize1;  int format1;  int subchunk1id1;  int subchunk1size1;  short audioformat1;  short numchannels1;  int samplerate1;  int byterate1;  short blockalign1;  short bitspersample1;  int subchunk2id1;  int subchunk2size1;  *//Cabeceras segundo archivo*  int chunkid2;  int chunksize2;  int format2;  int subchunk1id2;  int subchunk1size2;  short audioformat2;  short numchannels2;  int samplerate2;  int byterate2;  short blockalign2;  short bitspersample2;  int subchunk2id2;  int subchunk2size2;  *//Archivo*  FILE\* entrada1;  FILE\* entrada2;  FILE\* salida;  *//Variables para muestras primer archivo*  int total\_muestras1;  *//Variables segundo archivo*  int total\_muestras2;  *//Variables que se pueden reutilizar*  short muestra;  short headers[37];  #endif |

producto.c

|  |
| --- |
| #include"funciones.h"  int main(int argc, char \*argv[])  {  *//Leo las cabeceras*  leerCabeceras(argv);  **if**(numchannels1==1){  *//Defino variables para el primer archivo*  total\_muestras1=subchunk2size1/blockalign1;  short \*muestras1=(short \*)malloc(total\_muestras1 \* **sizeof**(short));  *//Defino variables para el segundo archivo*  total\_muestras2=subchunk2size2/blockalign2;  short \*muestras2=(short \*)malloc(total\_muestras2 \* **sizeof**(short));  *//Leo las muestras del primer archivos*  leerMuestras(muestras1,muestras2);  *//Calculo el producto*  calcularProducto(muestras1,muestras2);  }**else** **if**(numchannels1==2){  *//Defino variables para el primer archivo*  total\_muestras1=subchunk2size1/blockalign1;  short \*muestrasRe1=(short \*)malloc(total\_muestras1 \* **sizeof**(short));  short \*muestrasIm1=(short \*)malloc(total\_muestras1 \* **sizeof**(short));  *//Defino variables para el segundo archivo*  total\_muestras2=subchunk2size2/blockalign2;  short \*muestrasRe2=(short \*)malloc(total\_muestras2 \* **sizeof**(short));  short \*muestrasIm2=(short \*)malloc(total\_muestras2 \* **sizeof**(short));  *//Leo las muestras del primer archivos*  leerMuestras2Canales(muestrasRe1,muestrasIm1,muestrasRe2,muestrasIm2);  *//Calculo el producto*  calcularProductoComplejos(muestrasRe1,muestrasIm1,muestrasRe2,muestrasIm2);  }  }  void leerCabeceras(char \*\* argv){  entrada1 = fopen(argv[1], "rb");  entrada2 = fopen(argv[2], "rb");  salida=fopen(argv[3],"wb");  **if**(!entrada1 & !entrada2) {  perror("**\n**File opening failed");  exit(0);  }  *//Lectura de cabeceras primer archivo*  fread(&chunkid1,**sizeof**(int),1,entrada1);  fread(&chunksize1,**sizeof**(int),1,entrada1);  fread(&format1,**sizeof**(int),1,entrada1);  fread(&subchunk1id1,**sizeof**(int),1,entrada1);  fread(&subchunk1size1,**sizeof**(int),1,entrada1);  fread(&audioformat1,**sizeof**(short),1,entrada1);  fread(&numchannels1,**sizeof**(short),1,entrada1);  fread(&samplerate1,**sizeof**(int),1,entrada1);  fread(&byterate1,**sizeof**(int),1,entrada1);  fread(&blockalign1,**sizeof**(short),1,entrada1);  fread(&bitspersample1,**sizeof**(short),1,entrada1);  fread(&subchunk2id1,**sizeof**(int),1,entrada1);  fread(&subchunk2size1,**sizeof**(int),1,entrada1);  *//Lectura de cabeceras segundo archivo*  fread(&chunkid2,**sizeof**(int),1,entrada2);  fread(&chunksize2,**sizeof**(int),1,entrada2);  fread(&format2,**sizeof**(int),1,entrada2);  fread(&subchunk1id2,**sizeof**(int),1,entrada2);  fread(&subchunk1size2,**sizeof**(int),1,entrada2);  fread(&audioformat2,**sizeof**(short),1,entrada2);  fread(&numchannels2,**sizeof**(short),1,entrada2);  fread(&samplerate2,**sizeof**(int),1,entrada2);  fread(&byterate2,**sizeof**(int),1,entrada2);  fread(&blockalign2,**sizeof**(short),1,entrada2);  fread(&bitspersample2,**sizeof**(short),1,entrada2);  fread(&subchunk2id2,**sizeof**(int),1,entrada2);  fread(&subchunk2size2,**sizeof**(int),1,entrada2);  }  void leerMuestras(short \*muestras1, short \*muestras2){  int i=0;  *//Lectura muestras primer archivo*  **while** (feof(entrada1) == 0)  {  **if**(i<total\_muestras1){  fread(&muestra,**sizeof**(short),1,entrada1);  printf("Muestra %d del primer archivo:%hi**\n**",i+1,muestra);  muestras1[i]=muestra;  i++;  *//printf("Muestra %s: %d\n",i,muestras[i-1]);*  }**else**{  fread(&headers,**sizeof**(short),37,entrada1);  **break**;  }  }  *//Lectura muestras segundo archivo*  i=0;  **while** (feof(entrada2) == 0)  {  **if**(i<total\_muestras2){  fread(&muestra,**sizeof**(short),1,entrada2);  printf("Muestra %d segundo archivo:%hi**\n**", muestra);  muestras2[i]=muestra;  i++;  }**else**{  fread(&headers,**sizeof**(short),37,entrada2);  **break**;  }  }  }  void leerMuestras2Canales(short \*muestrasRe1,short\* muestrasIm1,short\* muestrasRe2, short\* muestrasIm2){  int i=0;  *//Lectura de muestras del primer archivo*  **while** (feof(entrada1) == 0)  {  **if**(i<total\_muestras1){  fread(&muestrasRe1[i],**sizeof**(short),1,entrada1);  fread(&muestrasIm1[i],**sizeof**(short),1,entrada1);  i++;    }**else**{  fread(&headers,**sizeof**(short),37,entrada1);  **break**;  }  }  i=0;  *//Lectura de muestras del segundo archivo*  **while** (feof(entrada2) == 0)  {  **if**(i<total\_muestras2){  fread(&muestrasRe2[i],**sizeof**(short),1,entrada2);  fread(&muestrasIm2[i],**sizeof**(short),1,entrada2);  i++;  }**else**{  fread(&headers,**sizeof**(short),37,entrada2);  **break**;  }  }  }  void escribirArchivo(short\* muestras,int decision){  *//Escribo las cabeceras del archivo de salida*  **if** (decision==0){  *//Se escriben las cabeceras del primer archivo*  fwrite(&chunkid1,**sizeof**(int),1,salida);  fwrite(&chunksize1,**sizeof**(int),1,salida);  fwrite(&format1,**sizeof**(int),1,salida);  fwrite(&subchunk1id1,**sizeof**(int),1,salida);  fwrite(&subchunk1size1,**sizeof**(int),1,salida);  fwrite(&audioformat1,**sizeof**(short),1,salida);  fwrite(&numchannels1,**sizeof**(short),1,salida);  fwrite(&samplerate1,**sizeof**(int),1,salida);  fwrite(&byterate1,**sizeof**(int),1,salida);  fwrite(&blockalign1,**sizeof**(short),1,salida);  fwrite(&bitspersample1,**sizeof**(short),1,salida);  fwrite(&subchunk2id1,**sizeof**(int),1,salida);  fwrite(&subchunk2size1,**sizeof**(int),1,salida);  *//Ahora escribo las muestras*  int i=0;  **for**(i=0;i<total\_muestras1;i++){  fwrite(&muestras[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);  }  }**else**{  *//Se escriben las cabeceras del segundo archivo*  fwrite(&chunkid2,**sizeof**(int),1,salida);  fwrite(&chunksize2,**sizeof**(int),1,salida);  fwrite(&format2,**sizeof**(int),1,salida);  fwrite(&subchunk1id2,**sizeof**(int),1,salida);  fwrite(&subchunk1size2,**sizeof**(int),1,salida);  fwrite(&audioformat2,**sizeof**(short),1,salida);  fwrite(&numchannels2,**sizeof**(short),1,salida);  fwrite(&samplerate2,**sizeof**(int),1,salida);  fwrite(&byterate2,**sizeof**(int),1,salida);  fwrite(&blockalign2,**sizeof**(short),1,salida);  fwrite(&bitspersample2,**sizeof**(short),1,salida);  fwrite(&subchunk2id2,**sizeof**(int),1,salida);  fwrite(&subchunk2size2,**sizeof**(int),1,salida);  *//Ahora escribo las muestras*  int i=0;  **for**(i=0;i<total\_muestras2;i++){  fwrite(&muestras[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 escribirArchivo2Canales(short\* muestrasRe, short\* muestrasIm,int decision){  *//Escribo las cabeceras del archivo de salida*  **if** (decision==0){  *//Se escriben las cabeceras del primer archivo*  fwrite(&chunkid1,**sizeof**(int),1,salida);  fwrite(&chunksize1,**sizeof**(int),1,salida);  fwrite(&format1,**sizeof**(int),1,salida);  fwrite(&subchunk1id1,**sizeof**(int),1,salida);  fwrite(&subchunk1size1,**sizeof**(int),1,salida);  fwrite(&audioformat1,**sizeof**(short),1,salida);  fwrite(&numchannels1,**sizeof**(short),1,salida);  fwrite(&samplerate1,**sizeof**(int),1,salida);  fwrite(&byterate1,**sizeof**(int),1,salida);  fwrite(&blockalign1,**sizeof**(short),1,salida);  fwrite(&bitspersample1,**sizeof**(short),1,salida);  fwrite(&subchunk2id1,**sizeof**(int),1,salida);  fwrite(&subchunk2size1,**sizeof**(int),1,salida);  *//Ahora escribo las muestras*  int i=0;  **for**(i=0;i<total\_muestras1;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);  }  }**else**{  *//Se escriben las cabeceras del segundo archivo*  fwrite(&chunkid2,**sizeof**(int),1,salida);  fwrite(&chunksize2,**sizeof**(int),1,salida);  fwrite(&format2,**sizeof**(int),1,salida);  fwrite(&subchunk1id2,**sizeof**(int),1,salida);  fwrite(&subchunk1size2,**sizeof**(int),1,salida);  fwrite(&audioformat2,**sizeof**(short),1,salida);  fwrite(&numchannels2,**sizeof**(short),1,salida);  fwrite(&samplerate2,**sizeof**(int),1,salida);  fwrite(&byterate2,**sizeof**(int),1,salida);  fwrite(&blockalign2,**sizeof**(short),1,salida);  fwrite(&bitspersample2,**sizeof**(short),1,salida);  fwrite(&subchunk2id2,**sizeof**(int),1,salida);  fwrite(&subchunk2size2,**sizeof**(int),1,salida);  *//Ahora escribo las muestras*  int i=0;  **for**(i=0;i<total\_muestras2;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 calcularProducto(short\* muestras1, short\* muestras2){  int i;  *//Reviso que archivo tiene más muestras*  **if**(total\_muestras1<=total\_muestras2){  *//El segundo archivo tiene más*  puts("El segundo archivo tiene mas muestras");  short \*resultado=(short \*)malloc(total\_muestras2 \* **sizeof**(short));  **for** (i = 0; i < total\_muestras2; i++)  {  **if**(i<total\_muestras1){  resultado[i]=(muestras1[i]\*muestras2[i])/(SHRT\_MAX+2);  *//printf("Muestra resultado %d: %hi\n",i+1,resultado[i]);*  }**else**{  resultado[i]=0;  }  }  *//Escribo el resultado y cabeceras segundo archivo*  escribirArchivo(resultado,1);  }**else**{  *//El primer archivo tiene más*  puts("El primer archivo tiene mas muestras");  short \*resultado=(short \*)malloc(total\_muestras1 \* **sizeof**(short));  **for** (i = 0; i < total\_muestras1; i++)  {  **if**(i<total\_muestras2){  resultado[i]=(muestras1[i]\*muestras2[i])/(SHRT\_MAX+2);  }**else**{  resultado[i]=0;  }  }  *//Escribo el resultado y cabeceras primer archivo*  escribirArchivo(resultado,0);  }    }  void calcularProductoComplejos(short\* muestrasRe1, short\* muestrasIm1, short\* muestrasRe2, short\* muestrasIm2){  int i;  *//Reviso que archivo tiene más muestras*  **if**(total\_muestras1<=total\_muestras2){  *//El segundo archivo tiene más*  puts("El segundo archivo tiene mas muestras o igual numero de muestras");  short \*resultadoRe=(short \*)malloc(total\_muestras2 \* **sizeof**(short));  short \*resultadoIm=(short \*)malloc(total\_muestras2 \* **sizeof**(short));  *//short resultadoRe[total\_muestras2];*  *//short resultadoIm[total\_muestras2];*  **for** (i = 0; i < total\_muestras2; i++)  {  **if**(i<total\_muestras1){  *//Tenemos que hacer producto de complejos*  resultadoRe[i]=((muestrasRe1[i]\*muestrasRe2[i])-(muestrasIm1[i]\*muestrasIm2[i]))/(SHRT\_MAX\*2); resultadoIm[i]=((muestrasRe1[i]\*muestrasIm2[i])+(muestrasRe2[i]\*muestrasIm1[i]))/(SHRT\_MAX\*2);  printf("Real:%d **\n**",resultadoRe[i]);  printf("Imaginaria: %d**\n**", resultadoIm[i]);  }**else**{  resultadoRe[i]=0;  resultadoIm[i]=0; } }  *//Escribo el resultado y cabeceras segundo archivo*  escribirArchivo2Canales(resultadoRe,resultadoIm,1);  }**else**{  *//El primer archivo tiene más*  puts("El primer archivo tiene mas muestras");  short \*resultadoRe=(short \*)malloc(total\_muestras1 \* **sizeof**(short));  short \*resultadoIm=(short \*)malloc(total\_muestras1 \* **sizeof**(short));  **for** (i = 0; i < total\_muestras1; i++){  **if**(i<total\_muestras2){  *//Tenemos que hacer producto de complejos*  resultadoRe[i]=((muestrasRe1[i]\*muestrasRe2[i])-(muestrasIm1[i]\*muestrasIm2[i]))/(SHRT\_MAX\*2);  resultadoIm[i]=((muestrasRe1[i]\*muestrasIm2[i])+(muestrasRe2[i]\*muestrasIm1[i]))/(SHRT\_MAX\*2);  }**else**{ resultadoRe[i]=0; resultadoIm[i]=0; }  }  *//Escribo el resultado y cabeceras primer archivo*  escribirArchivo2Canales(resultadoRe,resultadoIm,0);  }} |

## 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("**\n**File 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("**\n**File 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);} |

## 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("**\n**File 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("**\n**File 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;  }  } |

## Extras

## Programa que identifica que tecla del código DTMF fue tocada

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\*);  *//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;  short headers[37];  *//Métodos TDF*  #define PI acos(-1.0)*//Defino la constante PI*  void calcularTDF(short\*);  void obtenerDTMF(short\*,short\*);  float duracion;  int aux1,aux2;  int amp=1000;  int bandera=0;  *//Arreglos de frecuencias*  int f\_1[2] = {697,1209};  int f\_2[2] = {697,1336};  int f\_3[2] = {697,1477};  int f\_A[2] = {697,1633};  int f\_4[2] = {770,1209};  int f\_5[2] = {770,1336};  int f\_6[2] = {770,1477};  int f\_B[2] = {770,1633};  int f\_7[2] = {852,1209};  int f\_8[2] = {852,1336};  int f\_9[2] = {852,1477};  int f\_C[2] = {852,1633};  int f\_ASTE[2] = {941,1209};  int f\_0[2] = {941,1336};  int f\_GATO[2] = {941,1477};  int f\_D[2] = {941,1633};  #endif |

dtmf.c

|  |
| --- |
| #include"funciones.h"  int main(int argc, char \*argv[]){  *//Leo las cabeceras*  leerCabeceras(argv);  *//Defino variables*  total\_muestras=subchunk2size/blockalign;  printf("Total de muestras: %d**\n**", total\_muestras);  short \*muestras=(short \*)malloc(total\_muestras \* **sizeof**(short));  *//Leo las muestras*  leerMuestras(muestras);  *//Calculo la TDF*  calcularTDF(muestras);  }  void leerCabeceras(char \*\* argv){  *//Archivo de entrada*  entrada = fopen(argv[1], "rb");  salida=fopen(argv[2],"wb");  **if**(!entrada) {  perror("**\n**File 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++;  }**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));  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;  **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;  }**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*  }  *//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);  obtenerDTMF(Xre,Xim);}  void obtenerDTMF(short \*Xre,short \*Xim){  *//Obtengo la duración del archivo*  duracion=(float)total\_muestras/(float)samplerate;  printf("Duracion del archivo: %f**\n**", duracion);  int i;  **for**(i=0;i<total\_muestras/2;i++){  **if**(Xre[i]>amp)  printf("Xre[%d] = %d**\n**", i, Xre[i]);  }  *//Aquí reviso que tonos fueron identificados*  *//Tecla: 1, frecuencia baja=697Hz, alta=1209Hz*  aux1=(int)floor((float)f\_1[0]\*duracion);  aux2=(int)floor((float)f\_1[1]\*duracion);  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a un 1");  bandera=1;  }  aux1=(int)floor((float)f\_2[0]\*duracion);  aux2=(int)floor((float)f\_2[1]\*duracion);  *//Tecla: 2, frecuencia baja=697Hz, alta=1336Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a un 2");  bandera=1;  }  aux1=(int)floor((float)f\_3[0]\*duracion);  aux2=(int)floor((float)f\_3[1]\*duracion);  *//Tecla: 3, frecuencia baja=697Hz, alta=1477Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a un 3");  bandera=1;  }  aux1=(int)floor((float)f\_A[0]\*duracion);  aux2=(int)floor((float)f\_A[1]\*duracion);  *//Tecla: A, frecuencia baja=697Hz, alta=1633Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a una A");  bandera=1;  }  aux1=(int)floor((float)f\_4[0]\*duracion);  aux2=(int)floor((float)f\_4[1]\*duracion);  *//Tecla: 4, frecuencia baja=770Hz, alta=1209Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a un 4");  bandera=1;  }  aux1=(int)floor((float)f\_5[0]\*duracion);  aux2=(int)floor((float)f\_5[1]\*duracion);  *//Tecla: 5, frecuencia baja=770Hz, alta=1336Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a un 5");  bandera=1;  }  aux1=(int)floor((float)f\_6[0]\*duracion);  aux2=(int)floor((float)f\_6[1]\*duracion);  *//Tecla: 6, frecuencia baja=770Hz, alta=1477Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a un 6");  bandera=1;  }  aux1=(int)floor((float)f\_B[0]\*duracion);  aux2=(int)floor((float)f\_B[1]\*duracion);  *//Tecla: B, frecuencia baja=770Hz, alta=1633Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a una B");  bandera=1;  }  aux1=(int)floor((float)f\_7[0]\*duracion);  aux2=(int)floor((float)f\_7[1]\*duracion);  *//Tecla: 7, frecuencia baja=852Hz, alta=1209Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a un 7");  bandera=1;  }  aux1=(int)floor((float)f\_8[0]\*duracion);  aux2=(int)floor((float)f\_8[1]\*duracion);  *//Tecla: 8, frecuencia baja=852Hz, alta=1336Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a un 8");  bandera=1;  }  aux1=(int)floor((float)f\_9[0]\*duracion);  aux2=(int)floor((float)f\_9[1]\*duracion);  *//Tecla: 9, frecuencia baja=852Hz, alta=1477Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a un 9");  bandera=1;  }  aux1=(int)floor((float)f\_C[0]\*duracion);  aux2=(int)floor((float)f\_C[1]\*duracion);  *//Tecla: C, frecuencia baja=852Hz, alta=1633Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a una C");  bandera=1;  }  aux1=(int)floor((float)f\_ASTE[0]\*duracion);  aux2=(int)floor((float)f\_ASTE[1]\*duracion);  *//Tecla: \*, frecuencia baja=941Hz, alta=1209Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a un \*");  bandera=1;  }  aux1=(int)floor((float)f\_0[0]\*duracion);  aux2=(int)floor((float)f\_0[1]\*duracion);  *//Tecla: 0, frecuencia baja=941Hz, alta=1336Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a un 0");  bandera=1;  }  aux1=(int)floor((float)f\_GATO[0]\*duracion);  aux2=(int)floor((float)f\_GATO[1]\*duracion);  *//Tecla: #, frecuencia baja=941Hz, alta=1477Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a un #");  bandera=1;  }  aux1=(int)floor((float)f\_D[0]\*duracion);  aux2=(int)floor((float)f\_D[1]\*duracion);  *//Tecla: D, frecuencia baja=941Hz, alta=1633Hz*  **if**(Xre[aux1]>amp && Xre[aux2]>amp){  puts("El tono corresponde a una D");  bandera=1;  }  **if**(!bandera){  puts("Ningun tono detectado**\n**");  }  } |

## Programa para obtener mensaje mediante señal modulada

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\*);  *//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;  int total\_muestras\_originales;  short headers[37];  *//Métodos TDF*  #define PI acos(-1.0)*//Defino la constante PI*  void calcularTDF(short\*,int);  void obtenerMensaje(short\*);  int calcularNuevoNumeroMuestras(int);  float duracion;  int aux1,aux2;  int amp=10;  int aux\_conteo=0;  int bandera=0;  *//Arreglos de frecuencias*  int f\_1[2] = {697,1209};  int f\_2[2] = {697,1336};  int f\_3[2] = {697,1477};  int f\_A[2] = {697,1633};  int f\_4[2] = {770,1209};  int f\_5[2] = {770,1336};  int f\_6[2] = {770,1477};  int f\_B[2] = {770,1633};  int f\_7[2] = {852,1209};  int f\_8[2] = {852,1336};  int f\_9[2] = {852,1477};  int f\_C[2] = {852,1633};  int f\_ASTE[2] = {941,1209};  int f\_0[2] = {941,1336};  int f\_GATO[2] = {941,1477};  int f\_D[2] = {941,1633};  #endif |

modulacion.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 TDF*  short \*aux\_muestras=(short \*)malloc(128 \* **sizeof**(short));  int i;  int j;  **for** (i = 0; i < total\_muestras/32; i++){  **for**(j=0;j<32;j++){  aux\_muestras[j]=muestras[aux\_conteo];  aux\_conteo+=1;  }  calcularTDF(aux\_muestras,32); }  }  void leerCabeceras(char \*\* argv){  *//Archivo de entrada*  entrada = fopen(argv[1], "rb");  **if**(!entrada) {  perror("**\n**File 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);  }  }  void calcularTDF(short\* muestras,int muestras\_recibidas){  *//Aquí va el algoritmo para la TDF*  short \*Xre=(short \*)malloc(muestras\_recibidas \* **sizeof**(short));  short \*Xim=(short \*)malloc(muestras\_recibidas \* **sizeof**(short));  short \*magnitud=(short \*)malloc(muestras\_recibidas \* **sizeof**(short));  short \*fase=(short \*)malloc(muestras\_recibidas \* **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 < muestras\_recibidas; k++)  {  Xre[k]=0;  Xim[k]=0;  **for** (n = 0; n < muestras\_recibidas; n++)  {  Xre[k]+=(muestras[n]/muestras\_recibidas)\*cos(2\*k\*n\*PI/muestras\_recibidas);  Xim[k]-=(muestras[n]/muestras\_recibidas)\*sin(2\*k\*n\*PI/muestras\_recibidas);  }  *//printf("Voy en la iteracion:%d de %d\n",k+1,total\_muestras);*  }  *//Obtener tiempo e imprimir*  final = clock();  total = (double)(final - inicio) / CLOCKS\_PER\_SEC;  printf("Tiempo de ejecucion: %f**\n**", total);  obtenerMensaje(Xre);  }  void obtenerMensaje(short \*Xre){  *//Obtengo la duración del archivo*  duracion=(float)32/(float)samplerate;  printf("Duracion del archivo: %f**\n**", duracion);  int i;  **for**(i=0;i<32/2;i++){  **if**(Xre[i]>amp)  printf("Xre[%d] = %d**\n**", i, Xre[i]);  }  }  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;  } |