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("\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);
    }
}
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;
}
```