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