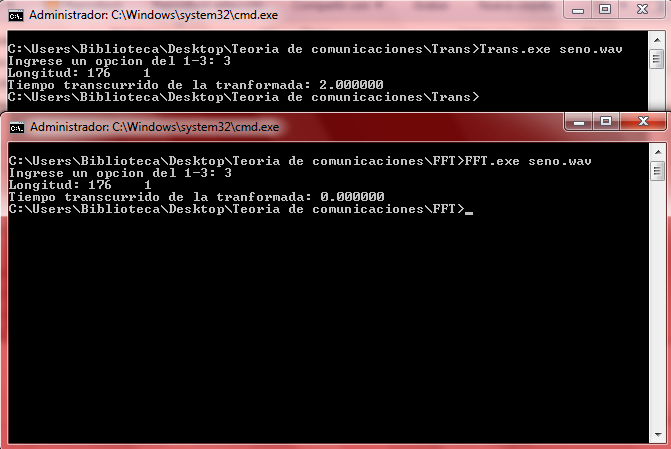
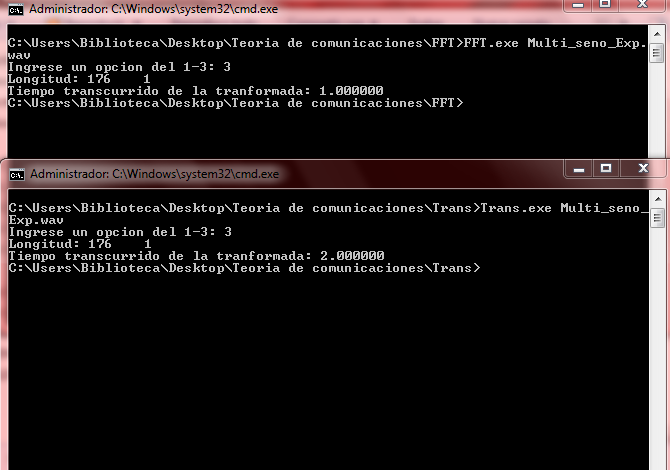
**Transformada Rápida de Fourier**

Es un eficiente algoritmo que permite calcular la [transformada de Fourier discreta](http://es.wikipedia.org/wiki/Transformada_de_Fourier_discreta) (DFT) y su inversa.

En esta práctica ya que la transformada rápida de Fourier es un algoritmo más eficiente, comprobaremos que los procesos los realice en un tiempo menor al de la transformada de Fourier normal haciendo el mismo proceso que la práctica anterior pero tomando el tiempo de ejecución de cada programa.

Veremos la comparativa en las siguientes imágenes





Podemos ver en la comparativa la imagen es de las transformadas del seno y la transformada del resultado en donde dice FFT es la transformada rápida, y podemos ver como tarda menos tiempo en esta que en la transformada normal, esto es porque la transformada hace menos operaciones que la transformada norma haciendo que termine en menos tiempo.

**Código:**

#include <stdio.h>

#include <math.h>

#include <time.h>

void tran(FILE \*s1, FILE \*res, int opc);

double mag(double x, double y);

double fase(double x, double y);

double \*\*argRes, \*argS1, \*\*angulo;

FILE \*hola;

typedef unsigned short int WORD;

typedef unsigned int DWORD;

typedef unsigned char CHAR;

typedef struct

{

CHAR riff[4];

DWORD tamano;

char wave[8];

DWORD nbytescabeza;

WORD formatopcm;

WORD monoestereo;

DWORD frecnyquist;

DWORD bytesporseg;

WORD bytesporcap;

WORD bitsmuestra;

CHAR data[4];

DWORD ndatos;

} DESCRIPTOR;

main(int argc, char \*\* argv){

FILE \*s1, \*res;

int opc;

char nom[50];

s1 = fopen(argv[1],"rb");

while(s1==NULL){

printf("Error. Archivo no encontrado!!!\n");

printf("Ingrese el nombre del archivo wav1: ");

scanf("%s",nom);

s1 = fopen(nom,"rb");

}

printf("Ingrese un opcion del 1-3: ");

scanf("%d",&opc);

res = fopen("resultado.wav","wb");

tran(s1, res, opc);

}

void tran(FILE \*s1, FILE \*res, int opc){

DESCRIPTOR cabS1, cabRes;

hola = fopen("hola.txt", "w");

int i,j;

unsigned short int aux;

unsigned int lenS1, bflong1, bflong2;

double escala;

double maxD = 0, maxI = 0;

double a,b;

fread (&cabS1, sizeof(cabS1), 1, s1);

cabRes = cabS1;

lenS1 = cabS1.ndatos/2;

printf("Longitud: %d\t %d\n",lenS1);

argS1 = (double \*) malloc((lenS1/2)\*sizeof(double));

argRes = (double \*\*) malloc((lenS1/2)\*sizeof(double\*));

angulo = (double \*\*) malloc((lenS1/2)\*sizeof(double\*));

fseek(s1, 44, SEEK\_SET);

bflong1 = lenS1/2;

for(i=0;i<bflong1;i++){

angulo[i] = (double \*) malloc(2\*sizeof(double));

argRes[i] = (double \*) malloc(2\*sizeof(double));

fread(&aux, sizeof(aux),1,s1);

if(aux<=0x7FFF){

escala = aux;

argS1[i] = (escala/0x7FFF);

}else{

aux = (0xFFFF-aux);

escala = aux;

argS1[i] = -(escala/0x7FFF);

}

argRes[i][0] = 0;

argRes[i][1] = 0;

angulo[i][0] = cos(((2\*M\_PI)\*i)/bflong1);

angulo[i][1] = sin(((2\*M\_PI)\*i)/bflong1);

//fprintf(hola,"%f %f\n",angulo[i][0], angulo[i][1]);

//fprintf(hola,"%f\n",argS2[i]);

fread(&aux, sizeof(aux),1,s1);

}

clock\_t start = clock();

Mariposa(0, bflong1, bflong1);

for(i=0;i<bflong1;i++){

switch(opc){

case 1:

a = mag(argRes[i][0],argRes[i][1]);

b = argS1[i];

maxD = maxD>=fabs(a)?maxD:fabs(a);

maxI = maxI>=fabs(b)?maxI:fabs(b);

break;

case 2:

escala = mag(argRes[i][0],argRes[i][1]);

b = fase(argRes[i][0],argRes[i][1]);

a = escala;

maxD = maxD>=fabs(a)?maxD:fabs(a);

maxI = maxI>=fabs(b)?maxI:fabs(b);

break;

default:

a = argRes[i][0];

b = argRes[i][1];

maxD = maxD>=fabs(a)?maxD:fabs(a);

maxI = maxI>=fabs(b)?maxI:fabs(b);

fprintf(hola,"Real-Img: %f %f\n",a, b);

break;

}

argRes[i][0] = a;

argRes[i][1] = b;

}

printf("Tiempo transcurrido de la tranformada: %f", ((double)clock() - start));

fseek(res, 0, SEEK\_SET);

fwrite(&cabRes,sizeof(DESCRIPTOR),1,res);

for(i=0;i<bflong1;i++){

if(argRes[i][0]>=0){

aux = (unsigned short int)((argRes[i][0]\*0x7FFF)/maxD);

fwrite(&aux,sizeof(short),1,res);

}else{

argRes[i][0] = (-1)\*argRes[i][0];

aux = (unsigned short int)((argRes[i][0]\*0x7FFF)/maxD);

aux = (0xFFFF-aux);

fwrite(&aux,sizeof(short),1,res);

}

aux=0x0000;

if(argRes[i][1]>=0){

aux = (unsigned short int)((argRes[i][1]\*0x7FFF)/maxI);

fwrite(&aux,sizeof(short),1,res);

}else{

argRes[i][1] = (-1)\*argRes[i][1];

aux = (unsigned short int)((argRes[i][1]\*0x7FFF)/maxI);

aux = (0xFFFF-aux);

fwrite(&aux,sizeof(short),1,res);

}

aux=0x0000;

}

fclose(hola);

fclose(s1);

fclose(res);

}

double mag(double x, double y){

return sqrt(pow(x,2)+pow(y,2));

}

double fase(double x, double y){

if(y!=0){

return atan(x/y);

}else{

return 0;

}

}

int inv(int x, int base){

int res=0;

int b = 1;

//printf("Ent: %d\t %d\n",x,base);

base = base>>1;

while(base>0){

//printf("Inv: %d\t %d\n",b,base);

if(b&x){

res += base;

//printf("Res: %d\n",res);

}

b = b<<1;

base = base>>1;

}

return res;

}

Mariposa(int x, int N, int base){

int inver1=0, inver2=0;

int nang, i;

if(N==2){

//printf("base: %d\n",base);

inver1 = inv(x, base);

//printf("Inv1: %d\t %d\n",inver1, inver2);

inver2 = inv((x+1), base);

nang = (base/N)\*(x%N);

//fprintf(hola,"MB0: %f %f\n",angulo[nang][0], angulo[nang][1]);

argRes[x][0] = argS1[inver1]+ argS1[inver2]\*angulo[nang][0];

argRes[x][1] = argS1[inver2]\*angulo[nang][1];

//printf("Mar1\t");

nang = (base/N)\*((x+1)%N);

//fprintf(hola,"MB1: %f %f\n",angulo[nang][0], angulo[nang][1]);

argRes[x+1][0] = argS1[inver1]+ argS1[inver2]\*angulo[nang][0];

argRes[x+1][1] = argS1[inver2]\*angulo[nang][1];

//printf("Mar2\n");

}else{

Mariposa(x,(N/2),base);

Mariposa((x+(N/2)),(N/2),base);

for(i=x;i<(x+(N/2));i++){

nang = (base/N)\*(i%N);

//fprintf(hola,"M: %d\t %f %f\n",i,angulo[nang][0], angulo[nang][1]);

argRes[i][0] = argRes[i][0]+((argRes[i+(N/2)][0]\*angulo[nang][0])\*(argRes[i+(N/2)][1]\*angulo[nang][1]));

argRes[i][1] = argRes[i][1]+((argRes[i+(N/2)][0]\*angulo[nang][1])\*(argRes[i+(N/2)][1]\*angulo[nang][0]));

nang = (base/N)\*((i+(N/2))%N);

//fprintf(hola,"M: %d\t %f %f\n",(i+(N/2)),angulo[nang][0], angulo[nang][1]);

argRes[i+(N/2)][0] = argRes[i][0]+((argRes[i+(N/2)][0]\*angulo[nang][0])\*(argRes[i+(N/2)][1]\*angulo[nang][1]));

argRes[i+(N/2)][1] = argRes[i][1]+((argRes[i+(N/2)][0]\*angulo[nang][1])\*(argRes[i+(N/2)][1]\*angulo[nang][0]));

}

}

}