ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA

TESIS DE GRADO

"MEJORAMIENTO DE IMÁGENES MONOCROMÁTICAS USANDO LA TRANSFORMADA DISCRETA COSENO"

TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN LA ESPECIALIZACIÓN DE ELECTRÓNICA Y TELECOMUNICACIONES

CÓDIGO FUENTE - MANUAL DE USUARIO

WASHINGTON LENIN PAREDES CRUZ

QUITO, SEPTIEMBRE DE 1997

INTRODUCCIÓN

Este documento presenta el listado de dos grupos de programas fuentes, los mismos que al compilarlos con las opciones de compilación indicadas usando el compilador TURBO C++ FOR WINDOWS Version 3.0, generan las aplicaciones Windows MIMTDC.EXE y TDCOSENO.DLL. Se incluye también el manual de usuario del software resultante.

MÓDULO MIMTDC

Este módulo está generado por los archivos indicados en la ventana desplegada en la figura No. 1, cada uno de los cuales se registran a continuación.

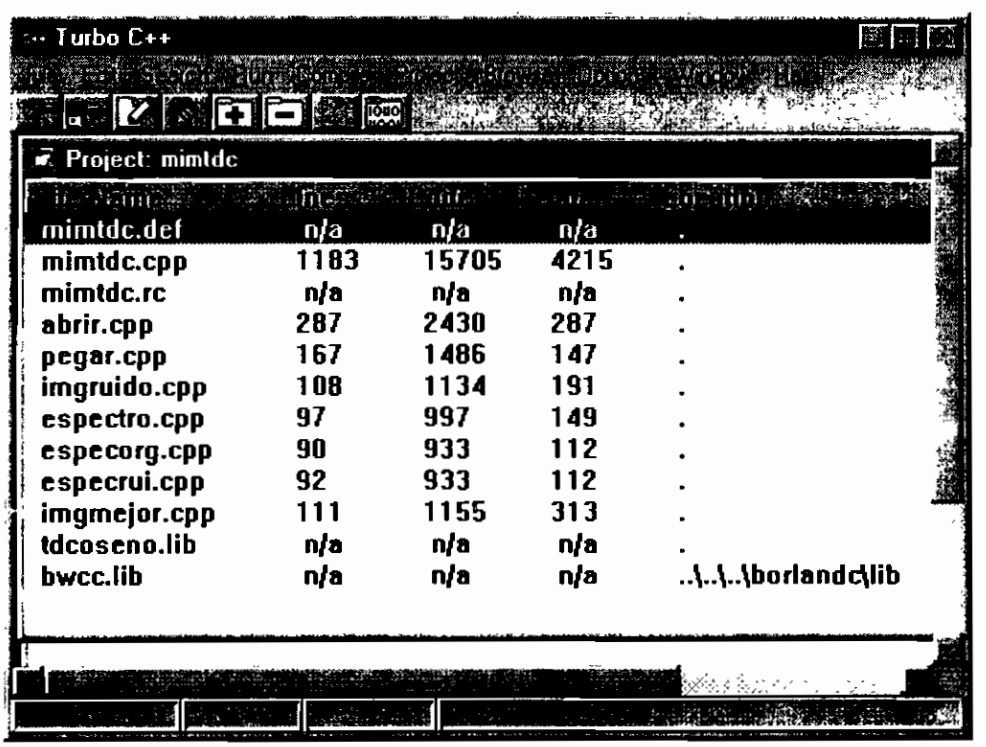

Figura No. 1 Listado de archivos que conforman el proyecto MIMTDC.

MIMTDC.DEF

```
EXETYPE WINDOWS
CODE PRELOAD MOVEABLE DISCARDABLE
DATA PRELOAD MOVEABLE MULTIPLE
```

```
HEAPSIZE 4096
STACKSIZE 6144
```

MIMTDC.H

```
#define CM_RDEPEND125
#define CM_HOMOMOR161
#define CM_DESHACER     112
#define CM_EXPORTAR     111
#define CM_PEGAR        113
#define CM_IMGRUIDO     115
#define CM_ESPECTRO     116
#define CM_IMGMEJOR     118
#define CM_RGAUSS 121
#define CM_RIMPULPN     122
#define CM_RIMPULP123
#define CM_RIMPULN124
#define CM_TDC          126
#define CM_ATRIBUTO     127
#define CM_FGI          131
#define CM_MALTOS 132
#define CM_MBAJOS 133
#define CM_MBANDA 134
#define CM_RAIZ         135
#define CM_AYUDA        140
#define CM_ACERCA 141
```

MIMTDC.RC

```
#include <owlrc.h>
#include "filedial.dlg"
#include "inputdia.dlg"
#include "mimtdc.h"

ESPECTRO ICON frecuenc.ico
ESPACIO ICON espacio.ico
ESPACIOM ICON espaciom.ico
COSENO ICON coseno.ico

1501 BITMAP "../buho.bmp"

ACERCA DIALOG 18, 19, 254, 188
CAPTION "MEJORAMIENTO DE IMAGENES MONOCROMATICAS USANDO LA TDC"
CLASS "BorDlg"
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
BEGIN
      CONTROL   "Button",   501,   "BorBtn",   BBS_BITMAP   |   WS_CHILD   |
WS_VISIBLE | WS_TABSTOP, 1, -2, 12, 12
      CONTROL  "Button",  IDOK,  "BorBtn",  BS_PUSHBUTTON  |  WS_CHILD  |
WS_VISIBLE | WS_TABSTOP, 105, 166, 32, 7
END

#include "menu.rc"
#include "acelera.rc"
```

MENU.RC

```
MENU_1 MENU
BEGIN
      POPUP "&Archivo"
      BEGIN
            MENUITEM "&Nuevo\tCtrl+N", 24329, GRAYED
            MENUITEM SEPARATOR
```

```
            MENUITEM "&Abrir...\tCtrl+A", 24330
            MENUITEM SEPARATOR
            MENUITEM "&Salir\tAlt+F4", 24340
      END

      POPUP "&Edición"
      BEGIN
            MENUITEM "&Deshacer\tAlt+BkSp", 112, GRAYED
            MENUITEM SEPARATOR
            MENUITEM "&Exportar\tCtrl+Ins", 111, GRAYED
            MENUITEM SEPARATOR
            MENUITEM "&Pegar\tShift+Ins", 113
      END

      POPUP "&Ver"
      BEGIN
            MENUITEM "Imagen &Ruidosa\tCtrl+R", 115, GRAYED
            MENUITEM "&Espectro\tCtrl+E", 116, GRAYED
            MENUITEM "Imagen &Mejorada\tCtrl+M", 118, GRAYED
      END

      POPUP "&Imagen"
      BEGIN
            POPUP "&Contaminar", GRAYED
            BEGIN
                  MENUITEM "Ruido &Gaussiano...", 121
                  MENUITEM "Ruido &Sal y Pimienta...", 122
                  MENUITEM "Ruido I&mpulsivo +...", 123
                  MENUITEM "Ruido Im&pulsivo -...", 124
                  MENUITEM SEPARATOR
                  MENUITEM "Ruido Dependiente de la señal...", 125
            END

            MENUITEM SEPARATOR
            MENUITEM "&TDC\tCtrl+T", 126, GRAYED
            MENUITEM "&Atributos\tCtrl+I", 127, GRAYED
            MENUITEM SEPARATOR
            POPUP "&Filtrar", GRAYED
            BEGIN
                  POPUP "Filtro &Lineal Generalizado"
                  BEGIN
                        MENUITEM            "Filtro            &Gaussiano
Inverso...\tShift+G", 131, GRAYED
                        MENUITEM SEPARATOR
                        POPUP "&Máscara"
                        BEGIN
                              MENUITEM  "Pasa  &Altos...\tShift+A",  132,
GRAYED
                              MENUITEM  "Pasa  &Bajos...\tShift+B",  133,
GRAYED
                              MENUITEM  "Pasa  Ba&nda...\tShift+N",  134,
GRAYED
                        END
                  END

                  MENUITEM SEPARATOR
                  MENUITEM "&Raíz...\tShift+R", 135, GRAYED
            END
      END
      POPUP "Homomór&fico"
      BEGIN
            MENUITEM "Fil&trar\tCtrl+H", 161, GRAYED
      END
```

```
      POPUP "Ven&tana"
      BEGIN
            MENUITEM "&Mosaico\tF4", 24336
            MENUITEM "&Cascada\tF5", 24337
            MENUITEM "Cerrar &Todo\tF6", 24338
            MENUITEM "&Organizar iconos", 24335
      END

      POPUP "A&yuda", HELP
      BEGIN
            MENUITEM "A&yuda\tF1", 140
            MENUITEM "A&cerca de ...\tShift+F1", 141
      END
END
```

ACELERA.RC

```
ACCELERATORS_1 ACCELERATORS
BEGIN
      VK_INSERT, 111, VIRTKEY, CONTROL
      VK_BACK, 112, VIRTKEY, ALT
      VK_INSERT, 113, VIRTKEY, SHIFT
      VK_N, 24329, VIRTKEY, CONTROL
      VK_A, 24330, VIRTKEY, CONTROL
      VK_R, 115, VIRTKEY, CONTROL
      VK_E, 116, VIRTKEY, CONTROL
      VK_M, 118, VIRTKEY, CONTROL
      VK_T, 126, VIRTKEY, CONTROL
      VK_I, 127, VIRTKEY, CONTROL
      VK_H, 161, VIRTKEY, CONTROL
      VK_F4, 24336, VIRTKEY
      VK_F5, 24337, VIRTKEY
      VK_F6, 24338, VIRTKEY
      VK_F1, 140, VIRTKEY
      VK_F1, 141, VIRTKEY, SHIFT
      VK_G, 131, VIRTKEY, SHIFT
      VK_A, 132, VIRTKEY, SHIFT
      VK_B, 133, VIRTKEY, SHIFT
      VK_N, 134, VIRTKEY, SHIFT
      VK_R, 135, VIRTKEY, SHIFT
END
```

MIMTDC.CPP

```
#include <owl.h>
#include <mdi.h>
#include <filedial.h>
#include <inputdia.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <dir.h>

#include "mimtdc.h"
#include "pegar.h"
#include "abrir.h"
#include "espectro.h"    //GRAFICACION DEL ESPECTRO
#include "especorg.h"    //GRAFICACION DEL ESPECTRO ORIGINAL
#include "especrui.h"    //GRAFICACION DEL ESPECTRO  RUIDOSO
#include "imgmejor.h"    //GRAFICACION DE IMAGEN MEJORADA
#include "imgruido.h"    //GRAFICACION DE IMAGEN RUIDOSA
```

```
      //prototipos de funciones residentes en DLL
void CosDir(int,int,float huge *,float huge *);
void CosInv(int,int,float huge *,float huge *);
void huge Copiar(HWND HWindow);
void huge CursorArena1();
void huge CursorFlecha1();

      //variable globales que identifican a la imagen
char NombreArchivo[MAXPATH];
WORD AnchoImagen,AltoImagen;
WORD BitsPixel;
char Ruido[50];   //variable externa que indica tipo de ruido
char Filtro[50];//variable externa que indica tipo de filtro
BOOL HayRuido;    //variable externa que indica la presencia de ruido
float NivelMax,NivelMin;//Niveles de la imagen
BOOL primer;            //Bandera para el proceso de pegado
HCURSOR hCursor,hOldCursor;

_CLASSDEF(TAcercaDlg)
class TAcercaDlg : public TDialog
{
public:
   TAcercaDlg(PTWindowsObject AParent, LPSTR AName) : TDialog(AParent,
AName) {};
};

_CLASSDEF(TMdiWindow)
class TMdiWindow:public TMDIFrame
{
      HMENU hMenu,hMenuA,hMenuE,hMenuV,hMenuI,hMenuPH;
      HANDLE hMemU,hMemV;      //VARIABLES PUNTERO PARA MEMORIA
      float huge *memU; //MEMORIA PARA IMAGEN ESPACIAL
      float huge *memV; //MEMORIA PARA IMAGEN TRANSFORMADA
      int DesHacer;           //variable usada para deshacer
      float mm,ll;//coef. de ruido dependiente de la señal

public:
      TMdiWindow(LPSTR ATitle, LPSTR MenuName);
      ~TMdi. indow(){LiberarMemoria();};    //destructor
      void LiberarMemoria();
      virtual void SetupWindow();
      virtual void GetWindowClass(WNDCLASS& AWndClass);
      virtual LPSTR GetClassName(){return "TDC";};
//funciones respuesta del menu ARCHIVO
      virtual void CMAbrir(TMessage&)=[CM_FIRST+CM_FILEOPEN];
      virtual void CMNuevo(TMessage&)=[CM_FIRST+CM_FILENEW];
//funciones respuesta del menu EDICION
      virtual void CMPegar(TMessage&)=[CM_FIRST+CM_PEGAR];
      virtual void CMCopiar(TMessage&)=[CM_FIRST+CM_EXPORTAR];
      virtual void CMDeshacer(TMessage&)=[CM_FIRST+CM_DESHACER];
//funciones respuesta del menu IMAGEN
      virtual void CMAtributo(TMessage&)=[CM_FIRST+CM_ATRIBUTO];
      virtual void CMTDC(TMessage&)=[CM_FIRST+CM_TDC];
      virtual void CMRAIZ(TMessage&)=[CM_FIRST+CM_RAIZ];
      virtual void CMFGI(TMessage&)=[CM_FIRST+CM_FGI];
      virtual void CMMBAJOS(RTMessage)=[CM_FIRST+CM_MBAJOS];
      virtual void CMMALTOS(RTMessage)=[CM_FIRST+CM_MALTOS];
      virtual void CMMBANDA(RTMessage)=[CM_FIRST+CM_MBANDA];
      virtual void CMRGAUSS(RTMessage)=[CM_FIRST+CM_RGAUSS];
      virtual void CMRIMPULPN(RTMessage)=[CM_FIRST+CM_RIMPULPN];
      virtual void CMRIMPULP(RTMessage)=[CM_FIRST+CM_RIMPULP];
      virtual void CMRIMPULN(RTMessage)=[CM_FIRST+CM_RIMPULN];
      virtual void CMRDEPEND(RTMessage)=[CM_FIRST+CM_RDEPEND];
//funciones respuesta del menu AYUDA
      virtual void CMAcerca(TMessage&)=[CM_FIRST+CM_ACERCA];
      virtual void CMAyuda(RTMessage)=[CM_FIRST+CM_AYUDA];
```

```
//funciones respuesta del menu VER
      virtual void CMIMGRUIDO(RTMessage)=[CM_FIRST+CM_IMGRUIDO];
      virtual void CMESPECTRO(RTMessage)=[CM_FIRST+CM_ESPECTRO];
      virtual void CMIMGMEJOR(RTMessage)=[CM_FIRST+CM_IMGMEJOR];
//funciones respuesta del menu PROCESO HOMOMORFICO
      virtual void CMHOMOMOR(RTMessage)=[CM_FIRST+CM_HOMOMOR];

      float huge *Ini_Mem(int M,int N,HANDLE *hMem);
      float ProbarPotenciaDos(int);
      void TDCD(int,int,float huge *,float huge *);
      void TDCI(int,int,float huge *,float huge *);
      void Escalar(int,int,float huge *);
      void Mem_a_Archivo(int,int,float huge *,char *);
      void Archivo_a_Mem(int,int,float huge *,char *);
      void Frec_a_Espac();
      void Frec_a_Espac_H();
      int Mensaje(int,float);
      float Adq_Dato(char *,char *,float);
      int Adq_Dato(char *,char *,int);
      void LogNatural(float huge *);
      void Exponencial(float huge *);
      void Toma_Mem();
      void Cepstrum_Inv();
      float Aleatorio();
      void CursorArena();
      void CursorFlecha();
      int Compatibilidad();
      void g_no_lineal(float huge *);
      void f_no_lineal(float huge *);
};

class TMyApp : public TApplication
{
public:
      TMyApp(LPSTR AName, HANDLE hInstance, HANDLE PrevInstance, LPSTR
lpCmdLine, int nCmdShow)
    : TApplication(AName, hInstance, hPrevInstance, lpCmdLine, nCmdShow)
{};
      virtual void InitMainWindow();
      virtual void InitInstance();
};

void TMyApp::InitInstance()
{     TApplication::InitInstance();
      HAccTable=LoadAccelerators(hInstance,"ACCELERATORS_1");
}

                  //DEFINICION DE FUNCIONES

//Constructor, inicializa handle a memoria y llama caja de dialogo
Acerca.
TMdiWindow::TMdiWindow(LPSTR ATitle, LPSTR MenuName):TMDIFrame(ATitle,
MenuName)
{
      HayRuido=FALSE;
      hMemU=0;
      hMemV=0;
      DesHacer=0;
      hCursor=LoadCursor(NULL,IDC_WAIT);        //carga          cursor
predefinido

      //--------------llama identificacion del programa
      TAcercaDlg* AcercaDlg=new TAcercaDlg(this, "ACERCA");
      GetApplication()->ExecDialog(AcercaDlg);

strcpy(Ruido,""); //variable externa que indica tipo de ruido
```

```
strcpy(Filtro,"Ninguno");//variable externa que indica tipo de filtro
mm=ll=0;
}

//libera memoria, para salir del programa o para cambiar de tamaño de
imagen
void TMdiWindow::LiberarMemoria()
{
      if(hMemU)    {
      GlobalUnlock(hMemU);
      GlobalFree(hMemU);
      hMemU=0;
                   }
      if(hMemV)    {
      GlobalUnlock(hMemV);
      GlobalFree(hMemV);
      hMemV=0;
                   }
}
//obtiene los handles del  menu y submenu
void TMdiWindow::SetupWindow()
{
      TMDIFrame::SetupWindow();
      hMenu=GetMenu(HWindow);
      hMenuA=GetSubMenu(hMenu,0);
      hMenuE=GetSubMenu(hMenu,1);
      hMenuV=GetSubMenu(hMenu,2);
      hMenuI=GetSubMenu(hMenu,3);
      hMenuPH=GetSubMenu(hMenu,4);
}

//asigna icono a la ventana principal
void TMdiWindow::GetWindowClass(WNDCLASS& AWndClass)
{
      TMDIFrame::GetWindowClass(AWndClass);
      AWndClass.hIcon=LoadIcon(GetApplication()->hInstance, "COSENO");
}

//calcula transformada discreta coseno directa, llamando a DLL
void TMdiWindow::TDCD(int M,int N,float huge *u,float huge *v)
{
      CursorArena();
            CosDir(M,N,u,v);
      CursorFlecha();
}

//calcula transformada discreta coseno inversa, llamando a DLL
void TMdiWindow::TDCI(int M,int N,float huge *u,float huge *v)
{
      CursorArena();
            CosInv(M,N,u,v);
      CursorFlecha();
}

void TMdiWindow::CursorArena()
{
 CursorArena1();
}

void TMdiWindow::CursorFlecha()
{
 CursorFlecha1();
}
```

```
void CursorArena1()
{
 hOldCursor=SetCursor(hCursor);      //ubica cursor reloj de arena
 ShowCursor(TRUE);
}

void CursorFlecha1()
{
ShowCursor(FALSE);
SetCursor(hOldCursor);  //ubica el cursor anterior flecha

}
//escala la imagen, ubicando en rango 0 - 255 niveles
void TMdiWindow::Escalar(int M,int N,float huge *p)
{
float Max,Min;
float huge *q;
q=p;
register int k,l;
      //CALCULA Y MUESTRA LOS EXTREMOS EN NIVEL DE LA IMAGEN
for(k=0;k<M;k++)
for(l=0;l<N;l++)  {
if(!k&&!l)Max=Min=*q;
if(*q>Max)Max=*q;
if(*q<Min)Min=*q;
q++;              }
char texto[160];
sprintf(texto,"Nivel  Máximo  inicial  =  %d\rNivel  Mínimo  inicial  =
%d\r\rNivel  Máximo  final  =  %d\rNivel  Mínimo  final  =  %d\r\r\rDESEA
ESCALAR                              A                              RANGO
INICIAL?",(int)NivelMax,(int)NivelMin,(int)Max,(int)Min);
int  escalar=MessageBox(HWindow,texto,"INFORMACION  DE  LIMITES  DE  LA
IMAGEN", MB_YESNO|MB_ICONQUESTION);

      //ESCALAMIENTO DE LA IMAGEN MEJORADA
 if(escalar==IDYES)
 for(k=0;k<M;k++)
 for(l=0;l<N;l++) {
 *p=(NivelMax-NivelMin)*(*p-Min);
 *p*=(float)1/(Max-Min);
 *p+=NivelMin;
 p++;              }
}

//pasa una imagen de memoria a archivo
void TMdiWindow::Mem_a_Archivo(int M,int N,float huge *p,char archivo[])
{
 FILE *fp;
 float v;

if(DesHacer&&!strcmp(archivo,"ESPECTRO.WLP")){remove("ESPECTRO.BAK");ren
ame("ESPECTRO.WLP","ESPECTRO.BAK");}
 DWORD l=(DWORD)M*N;
 if(fp=fopen(archivo,"wb"))    {
 for(register DWORD k=0;k<l;k++){
 v=*(p++);
 fwrite(&v,sizeof(float),1,fp);      }
 fclose(fp);                   }
}

//pasa una imagen de archivo a memoria
void TMdiWindow::Archivo_a_Mem(int M,int N,float huge *p,char archivo[])
{
 FILE *fp;
 float v;
 DWORD l=(DWORD)M*N;
```

```
 if(fp=fopen(archivo,"rb"))          {
 for(register DWORD k=0;k<l;k++){
 fread(&v,sizeof(float),l,fp);
 *(p++)=v;          }
 fclose(fp);                    }
}


//funcion respuesta al elemento del menu Acerca de ...
void TMdiWindow::CMAcerca(TMessage&)
{
      TAcercaDlg* AcercaDlg=new TAcercaDlg(this, "ACERCA");
      GetApplication()->ExecDialog(AcercaDlg);
}

//funcion respuesta que carga el sistema de ayudas
void TMdiWindow::CMAyuda(TMessage&)
{
      WinHelp(HWindow,"AYUDATDC.HLP",HELP_INDEX,0L);
}

//pega una imagen desde el portapapeles y habilita menu necesario
void TMdiWindow::CMPegar(TMessage&)
{
if (OpenClipboard(HWindow))     // si el Portapapeles si abri¢ bien */
  {  primer=FALSE;
  if (!(GetClipboardData(CF_BITMAP)))
    {                          // si hay error al obtener el gestor... */
    CloseClipboard();
                                          MessageBox(HWindow,"PORTAPAPELES
VACIO","PORTAPAPELES",MB_OK|MB_ICONSTOP);
    }
    else{
EnableMenuItem(hMenuA,CM_FILEOPEN,MF_GRAYED);//dehabilita Abrir
EnableMenuItem(hMenuA,CM_FILENEW,MF_ENABLED);//habilita Nuevo
EnableMenuItem(hMenuE,CM_PEGAR,MF_GRAYED);//deshabilita pegar
EnableMenuItem(hMenuI,CM_ATRIBUTO,MF_ENABLED);
EnableMenuItem(hMenuI,CM_TDC,MF_ENABLED);//habilita TDC
EnableMenuItem(hMenuPH,CM_HOMOMOR,MF_ENABLED);//habilita          Filtro
Homomórfico
EnableMenuItem(hMenuE,CM_EXPORTAR,MF_ENABLED);//habilita Exportar
EnableMenuItem(hMenuI,0,MF_BYPOSITION|MF_ENABLED);//habilita Contaminar

      GetApplication()->MakeWindow(new     TPegarWindow(this,     "Imagen
Importada"));
      }
      CloseClipboard();
      }
}
            //copia contenido de pantalla a portapapeles
void TMdiWindow::CMCopiar(RTMessage)
{
if(MessageBox(HWindow,"El  uso  de  ésta  opción  permitirá  copiar  al
portapapeles la imagen de la ventana principal, para ello la ventana
hija que contiene la imagen seleccionada debe estar maximizada. Además
la imagen a copiar debe estar completamente vizualizada.\r\r\Otra manera
de proceder es hacer doble-click con el boton izquierdo del ratón sobre
la  ventana  que  contiene  la  imagen  a  copiar.\r\r\r\tDesea  continuar
?","EXPORTAR", MB_YESNO|MB_ICONQUESTION)==IDYES)
      {Copiar(HWindow);}
}

void Copiar(HWND HWindow)
{
  HDC hDC, hMemDC;
  HBITMAP hBitmap, hPrevBitmap;
```

```
  int x,y;
  RECT rcClientArea;

GetClientRect(HWindow, &rcClientArea);
x=AnchoImagen;// rcClientArea.right;
y=AltoImagen;// rcClientArea.bottom;
if(rcClientArea.right>x&&rcClientArea.bottom>y)
    {
hDC= GetDC(HWindow);     // establecer el contexto de visualizaci¢n
hMemDC= CreateCompatibleDC(hDC);     // crea contexto visual. en memoria
hBitmap=                          // crear mapa de bits
       CreateCompatibleBitmap(hDC, x, y);
if (hBitmap)              // si el mapa de bits se cre¢ bien...
  {
  hPrevBitmap= SelectObject(hMemDC, hBitmap);    // seleccionar mapa bits

  BitBlt(hMemDC,0,0,x,y,hDC,0,0,SRCCOPY); // cargarlo
  if (OpenClipboard(HWindow));                    // si el portapapeles se
abri¢ bien
    {
  EmptyClipboard();                          // limpiar el portapapeles
  SetClipboardData(CF_BITMAP, hBitmap);      // pasarle el gestor
  CloseClipboard();                          // cerrar el portapapeles
    }
  SelectObject(hMemDC, hPrevBitmap);             // deseleccionar el mapa
de bits
             MessageBox(HWindow,"Copia        Realizada","COPIA        A
PORTAPAPELES",MB_OK|MB_ICONINFORMATION);
  } // NOTA: No libere el gestor del mapa de bits despu,s de pasarlo.
else
  {                              // si existe alg£n error, mostrar un
mensaje...
  MessageBeep(0);
          MessageBox(HWindow,"Copia      No      Realizada","COPIA      A
PORTAPAPELES",MB_OK|MB_ICONEXCLAMATION);
  }
DeleteDC(hMemDC);              // borrar el contexto de visualizaci¢n en
memoria
ReleaseDC(HWindow, hDC);  // liberar el contexto de visualizaci¢n
    }
else  MessageBox(HWindow,"Asegúrese  que  la  imagen  a  copiar  esté
completamente      visualizada      en      la      ventana","COPIA      A
PORTAPAPELES",MB_OK|MB_ICONSTOP);
}

void TMdiWindow::CMDeshacer(RTMessage)
{
  if(MessageBox(HWindow,"Desea  anular  el  efecto  del\rúltimo  filtro
aplicado ?","DESHACER", MB_YESNO|MB_ICONQUESTION)==IDYES)
{     EnableMenuItem(hMenuE,CM_DESHACER,MF_GRAYED);//deshabilita
Deshacer
      remove("ESPECTRO.WLP");
      rename("ESPECTRO.BAK","ESPECTRO.WLP");
      strcpy(Filtro,"Ninguno");
}
}

//obtiene imagen desde archivo *.BMP hasta archivo y ventana, habilita
menu
void TMdiWindow::CMAbrir(TMessage&)
{
  if (GetApplication()->ExecDialog( new TBMPFileDialog
    (this, SD_FILEOPEN, strcpy(NombreArchivo, "*.bmp"))) == IDOK )
    {
EnableMenuItem(hMenuA,CM_FILEOPEN,MF_GRAYED);//dehabilita Abrir
EnableMenuItem(hMenuA,CM_FILENEW,MF_ENABLED);//habilita Nuevo
```

```
EnableMenuItem(hMenuE,CM_PEGAR,MF_GRAYED);//deshabilita pegar
EnableMenuItem(hMenuI,CM_ATRIBUTO,MF_ENABLED);
EnableMenuItem(hMenuE,CM_EXPORTAR,MF_ENABLED);//habilita Exportar
EnableMenuItem(hMenuI,0,MF_BYPOSITION|MF_ENABLED);//habilita Contaminar
EnableMenuItem(hMenuI,CM_TDC,MF_ENABLED);//habilita TDC
EnableMenuItem(hMenuPH,CM_HOMOMOR,MF_ENABLED);//habilita          Filtro
Homomórfico

      GetApplication()->MakeWindow(new TBitScrollWindow(this, "Abrir"));
    }
}

//pone el programa en condiciones iniciales.
void TMdiWindow::CMNuevo(TMessage&)
{
            //pone en condiciones iniciales elementos del menu
EnableMenuItem(hMenuA,CM_FILEOPEN,MF_ENABLED);//habilita Abrir
EnableMenuItem(hMenuE,CM_PEGAR,MF_ENABLED);//habilita pegar
EnableMenuItem(hMenuA,CM_FILENEW,MF_GRAYED);//deshabilita Nuevo
EnableMenuItem(hMenuE,CM_DESHACER,MF_GRAYED);//deshabilita Deshacer
EnableMenuItem(hMenuI,CM_TDC,MF_GRAYED);//deshabilita TDC
EnableMenuItem(hMenuI,CM_ATRIBUTO,MF_GRAYED);
EnableMenuItem(hMenuE,CM_EXPORTAR,MF_GRAYED);//deshabilita Exportar
EnableMenuItem(hMenuI,0,MF_BYPOSITION|MF_GRAYED);//deshabilita
Contaminar
EnableMenuItem(hMenuI,5,MF_BYPOSITION|MF_GRAYED);//deshabilita filtrar
      EnableMenuItem(hMenuI,CM_FGI,MF_GRAYED);
      EnableMenuItem(hMenuI,CM_MALTOS,MF_GRAYED);
      EnableMenuItem(hMenuI,CM_MBAJOS,MF_GRAYED);
      EnableMenuItem(hMenuI,CM_MBANDA,MF_GRAYED);
      EnableMenuItem(hMenuI,CM_RAIZ,MF_GRAYED);
                  //MENU VER
EnableMenuItem(hMenuV,CM_IMGRUIDO,MF_GRAYED);//deshabilita   ver   imagen
ruidosa
EnableMenuItem(hMenuV,CM_ESPECTRO,MF_GRAYED);//deshabilita ver espectro
EnableMenuItem(hMenuV,CM_IMGMEJOR,MF_GRAYED);//deshabilita   ver   imagen
mejorada

EnableMenuItem(hMenuPH,CM_HOMOMOR,MF_GRAYED);//deshabilita filtro

      TMDIFrame::CloseChildren();

      AnchoImagen=AltoImagen=0;//inicializa tamaño de imagen

      LiberarMemoria(); //libera memoria por si hay cambio de tamaño de
imagen

      DesHacer=0;

      HayRuido=FALSE;          //indica    imagen    sin    contaminacion
artificial

      strcpy(Ruido,"");
      strcpy(Filtro,"Ninguno");
      mm=ll=0;
}

//presenta atributos de la imagen resultante, tamaño, filtros, etc
void TMdiWindow::CMAtributo(TMessage&)
{
char texto[150];
sprintf(texto,"Fichero:   %s\r Ancho:   %d pixeles\r Alto:   %d pixeles\r
Bits/pixel:  %d",NombreArchivo,AnchoImagen,AltoImagen,BitsPixel);
      MessageBox(HWindow,texto,"Caracteristicas Imagen",MB_OK);
}
```

```
//calcula la TDC directa verificando compatibilidad de tamaño
void TMdiWindow::CMTDC(RTMessage)
{
if(Compatibilidad()){
      DesHacer++;
      EnableMenuItem(hMenuI,5,MF_BYPOSITION|MF_ENABLED);//habilita
Filtrar
      EnableMenuItem(hMenuI,CM_FGI,MF_ENABLED);
      EnableMenuItem(hMenuI,CM_MALTOS,MF_ENABLED);
      EnableMenuItem(hMenuI,CM_MBAJOS,MF_ENABLED);
      EnableMenuItem(hMenuI,CM_MBANDA,MF_ENABLED);
      EnableMenuItem(hMenuI,CM_RAIZ,MF_ENABLED);
      Toma_Mem();              //reserva memoria para imagen
      char archivo[]="IMGORG.WLP";
      Archivo_a_Mem(AltoImagen,AnchoImagen,memU,archivo);
      TDCD(AltoImagen,AnchoImagen,memU,memV);
      strcpy(archivo,"ESPECORG.WLP");
      Mem_a_Archivo(AltoImagen,AnchoImagen,memV,archivo);
GetApplication()->MakeWindow(new   TGraficarEspecorg(this,   "FRECUENCIA
ESPACIAL [Imagen Original]"));

if(HayRuido){     strcpy(archivo,"IMGDATO.WLP");
            Archivo_a_Mem(AltoImagen,AnchoImagen,memU,archivo);
            TDCD(AltoImagen,AnchoImagen,memU,memV);
            strcpy(archivo,"ESPECTRO.WLP");
            Mem_a_Archivo(AltoImagen,AnchoImagen,memV,archivo);
            GetApplication()->MakeWindow(new     TGraficarEspectro(this,
"FRECUENCIA ESPACIAL [Imagen Actual]"));
            strcpy(archivo,"ESPECRUI.WLP");
            Mem_a_Archivo(AltoImagen,AnchoImagen,memV,archivo);
            GetApplication()->MakeWindow(new     TGraficarEspecrui(this,
"FRECUENCIA ESPACIAL [Imagen Ruidosa]"));
          }else   {
                   strcpy(archivo,"ESPECTRO.WLP");
                   Mem_a_Archivo(AltoImagen,AnchoImagen,memV,archivo);
                                    GetApplication()->MakeWindow(new
TGraficarEspectro(this, "FRECUENCIA ESPACIAL [Imagen Actual]"));
                   }

EnableMenuItem(hMenuV,CM_ESPECTRO,MF_ENABLED);//habilita ver espectro
            }
}

int TMdiWindow::Compatibilidad()
{
//COMPROBACION SI LAS DIMENSIONES DE LA IMAGEN ES DE
//TAMAÑO 2^m x 2^n CON m Y n ENTEROS
float ResultadoX,ResultadoY;
ResultadoX=ProbarPotenciaDos(AnchoImagen);
ResultadoY=ProbarPotenciaDos(AltoImagen);
if((ResultadoX!=2.0)||(ResultadoY!=2.0)){MessageBox(HWindow,"IMAGEN   NO
COMPATIBLE PARA ALGORITMO DE TDC","TAMAÑO DE IMAGEN POTENCIA DE 2
?",MB_OK|MB_ICONSTOP);return 0;}
else return 1;
}

//funcion respuesta que filtra en dom. frecuencia con filtro raiz
void TMdiWindow::CMRAIZ(RTMessage)
{     float Alpha=1.0;
      char archivo[]="ESPECTRO.WLP";
      float huge *p=memV;
      BOOL Signo=FALSE;
      char Titulo[]="FILTRO RAIZ";
      char Orden[]="Introducir Coeficiente [0.00 - 2.00];";
      Archivo_a_Mem(AltoImagen,AnchoImagen,memV,archivo);
      Alpha=Adq_Dato(Titulo,Orden,Alpha);
```

```
      if(Alpha>0&&Alpha<2){          //Alpha no puede ser cero.
      if(Mensaje(1,Alpha))     {
            for(register int k=0;k<AltoImagen;k++)
            for(register int l=0;l<AnchoImagen;l++)    {
            if(*p<0){Signo=TRUE;*p=-*p;}  //obtiene el modulo
            *p=pow(*p,Alpha);
            if(Signo)*p=-*p;
            Signo=FALSE;
            p++;                                }
            Mem_a_Archivo(AltoImagen,AnchoImagen,memV,archivo);
            Frec_a_Espac();          //calcula  coseno-inversa,  escala  y
grafica
      char Informa[30];
      sprintf(Informa,"Raíz,  Alpha= %4.3f",Alpha);
      strcpy(Filtro,Informa);
                        }
             }
}

//usa filtro pasa bajos en dominio de la frecuencia
void TMdiWindow::CMMBAJOS(RTMessage)
{     float EnergiaTotal=0;
      float Energia=0;
      int r;
      int
RadioPasaBajos=(int)sqrt((DWORD)AltoImagen*AltoImagen+(DWORD)AnchoImagen
*AnchoImagen);
      char archivo[]="ESPECTRO.WLP";
      float huge *p;
      p=memV;
      char Titulo[]="FILTRO PASA BAJOS";
      char Orden[]="Introducir Radio Pasa Bajos:";
      Archivo_a_Mem(AltoImagen,AnchoImagen,memV,archivo);
      RadioPasaBajos=Adq_Dato(Titulo,Orden,RadioPasaBajos);

      if(RadioPasaBajos>=0){         //radio no puede ser negativo
      if(Mensaje(2,(float)RadioPasaBajos)){
            for(register int k=0;k<AltoImagen;k++)
            for(register int l=0;l<AnchoImagen;l++)    {
            EnergiaTotal+=*p**p;
            r=(int)sqrt((DWORD)k*k+(DWORD)l*l);
/* si el radio es mayor al de corte, se asume 0, de otro modo pasa
incrementando la energia transferida*/
            if(r>RadioPasaBajos)*p=0;else Energia+=*p**p;
            p++;                                }
            Mem_a_Archivo(AltoImagen,AnchoImagen,memV,archivo);
            float EnergiaTx=(float)100*Energia/EnergiaTotal;
//informa el resultado
            char Informe[60];
            sprintf(Informe,"Energia Transferida = %5.2f %%\rRadio Pasa
Bajos = %d",EnergiaTx,RadioPasaBajos);
            MessageBox(HWindow,Informe,"ENERGIA TRANSFERIDA",MB_OK);

            Frec_a_Espac();
      char Informa[30];
      sprintf(Informa,"Pasa-bajos,  Radio= %d",RadioPasaBajos);
      strcpy(Filtro,Informa);
                              }
                        }
}


void TMdiWindow::CMMALTOS(RTMessage)
{     float EnergiaTotal=0;
      float Energia=0;
      int r;
```

```
      int RadioPasaAltos=0;
      char archivo[]="ESPECTRO.WLP";
      float huge *p;
      p=memV;
      char Titulo[]="FILTRO PASA ALTOS";
      char Orden[]="Introducir Radio Pasa Altos:";
      Archivo_a_Mem(AltoImagen,AnchoImagen,memV,archivo);
      RadioPasaAltos=Adq_Dato(Titulo,Orden,RadioPasaAltos);
      if(RadioPasaAltos>=0){
      if(Mensaje(3,(float)RadioPasaAltos)){
            for(register int k=0;k<AltoImagen;k++)
            for(register int l=0;l<AnchoImagen;l++)    {
            EnergiaTotal+=*p**p;
            r=(int)sqrt((DWORD)k*k+(DWORD)l*l);
/* si el radio es menor o igual al de corte, se asume 0, de otro modo
pasa incrementando la energia transferida*/
            if(r<=RadioPasaAltos)*p=0;else Energia+=*p**p;
            p++;                                  }
            Mem_a_Archivo(AltoImagen,AnchoImagen,memV,archivo);
            float EnergiaTx=(float)100*Energia/EnergiaTotal;
//informa el resultado
            char Informe[60];
            sprintf(Informe,"Energia Transferida = %5.2f %%\rRadio Pasa
Altos = %d",EnergiaTx,RadioPasaAltos);
            MessageBox(HWindow,Informe,"ENERGIA TRANSFERIDA",MB_OK);

            Frec_a_Espac();
      char Informa[30];
      sprintf(Informa,"Pasa-altos,  Radio= %d",RadioPasaAltos);
      strcpy(Filtro,Informa);
                        }
                  }
}

void TMdiWindow::CMMBANDA(RTMessage)
{     float EnergiaTotal=0;
      float Energia=0;
      int r;
      int
RadioSuperior=(int)sqrt((DWORD)AltoImagen*AltoImagen+(DWORD)AnchoImagen*
AnchoImagen);
      int RadioInferior=0;
      char archivo[]="ESPECTRO.WLP";
      float huge *p;
      p=memV;
      char Titulo[]="FILTRO PASA BANDA";
      char Orden[]="Introducir Radio Inferior:";
      Archivo_a_Mem(AltoImagen,AnchoImagen,memV,archivo);
      RadioInferior=Adq_Dato(Titulo,Orden,RadioInferior);
      strcpy(Orden,"Introducir Radio Superior:");
      RadioSuperior=Adq_Dato(Titulo,Orden,RadioSuperior);
     if(RadioInferior>=0&&RadioSuperior>=0){
      if(Mensaje(4,(float)RadioInferior)&&Mensaje(5,(float)RadioSuperior
))    {
            for(register int k=0;k<AltoImagen;k++)
            for(register int l=0;l<AnchoImagen;l++)    {
            EnergiaTotal+=*p**p;
            r=(int)sqrt((DWORD)k*k+(DWORD)l*l);
// si el radio es menor o igual o mayor respectivamente a los de corte,
se asume 0, de otro modo pasa incrementando la energia transferida
            if(r>RadioSuperior||r<=RadioInferior)*p=0;else
Energia+=*p**p;
            p++;                                  }
            Mem_a_Archivo(AltoImagen,AnchoImagen,memV,archivo);
            float EnergiaTx=(float)100*Energia/EnergiaTotal;
//informa el resultado
```

```
            char Informe[80];
            sprintf(Informe,"Energia Transferida = %5.2f %%\rRadio
Inferior          =          %d\rRadio          Superior          =
%d",EnergiaTx,RadioInferior,RadioSuperior);
            MessageBox(HWindow,Informe,"ENERGIA TRANSFERIDA",MB_OK);

            Frec_a_Espac();
      char Informa[45];
      sprintf(Informa,"Pasa-banda,          R_Sup=     %d,          R_Inf=
%d",RadioSuperior,RadioInferior);
      strcpy(Filtro,Informa);
                        }
                                    }
}

//funcion respuesta para el filtro Gaussiano inverso
void TMdiWindow::CMFGI(RTMessage)
{     float Promedio=0.0;
      float Varianza=0.0;
      float Exponente;
      char archivo[]="ESPECTRO.WLP";
      float huge *p;
      p=memV;
      register int k,l;
      char Titulo[]="FILTRO GAUSSIANO INVERSO";
      Archivo_a_Mem(AltoImagen,AnchoImagen,memV,archivo);
            //CALCULA EL VALOR MEDIO DE LOS COEFICIENTE TRANSFORMADOS
      for(k=0;k<AltoImagen;k++)
      for(l=0;l<AnchoImagen;l++)     {
      Promedio+=*p;p++;              }
      Promedio/=(float)AltoImagen*AnchoImagen;
            //CALCULA LA VARIANZA DE LOS COEF. TRANSFORMADOS
      p=memV;
      for(k=0;k<AltoImagen;k++)
      for(l=0;l<AnchoImagen;l++)     {
      Varianza+=pow((*p-Promedio),2.0);p++;                  }
      Varianza/=(float)AltoImagen*AnchoImagen;
      char Informe[60];
      sprintf(Informe,"Promedio      =      %-9.1f\rVarianza      =      %-
9.1f",Promedio,Varianza);
      MessageBox(HWindow,Informe,Titulo,MB_OK);
            //PROCESO DE FILTRAJE
      p=memV;
      for(k=0;k<AltoImagen;k++)
      for(l=0;l<AnchoImagen;l++)     {
      Exponente=0.5*((float)k*k+(float)l*l);
      Exponente/=(float)Varianza;
      *p*=exp(Exponente);p++;        }

      Mem_a_Archivo(AltoImagen,AnchoImagen,memV,archivo);
            Frec_a_Espac();
      char Informa[30];
      sprintf(Informa,"Gaussiano Inverso");
      strcpy(Filtro,Informa);
}

//funcion respuesta que añade a la imagen ruido gaussiano aditivo
//si nivel sobrepasa de 255, nivel=255
//si es negativo, nivel=0
//de otro modo el resultado de la suma
void TMdiWindow::CMRGAUSS(RTMessage)
{
      HayRuido=TRUE;
      int sigma=0;
      float huge *p;
      register int m,n,k;
```

```
      float suma=0;
      char archivoi[]="IMGORG.WLP";
      char archivof[]="IMGDATO.WLP";
      char Titulo[]="RUIDO GAUSSIANO";
      char Orden[]="Introduzca la desviación estandar [10 - 30]:";
      sigma=Adq_Dato(Titulo,Orden,sigma);
      if(sigma>=0){           //desviacion estandar siempre +
      Toma_Mem();
      Archivo_a_Mem(AltoImagen,AnchoImagen,memU,archivoi);
      p=memU;
      for(m=0;m<AltoImagen;m++)
      for(n=0;n<AnchoImagen;n++)     {
      for(k=0;k<12;k++)suma+=Aleatorio();
      *p+=sigma*(suma-6);
      if(*p>255)*p=255;
      if(*p<0)*p=0;
      suma=0;
      p++;                           }
      Mem_a_Archivo(AltoImagen,AnchoImagen,memU,archivof);
      GetApplication()->MakeWindow(new    TGrafImgRuido(this,    "IMAGEN
RUIDOSA [Ruido aditivo Gaussiano]"));
      EnableMenuItem(hMenuV,CM_IMGRUIDO,MF_ENABLED);//habilita       ver
imagen ruidosa
            //si la imagen se contamina nuevamente
            EnableMenuItem(hMenuE,CM_DESHACER,MF_GRAYED);//deshabilita
Deshacer
            EnableMenuItem(hMenuV,CM_IMGMEJOR,MF_GRAYED);//deshabilita
ver imagen mejorada
            EnableMenuItem(hMenuV,CM_ESPECTRO,MF_GRAYED);//deshabilita
ver espectro
//
      EnableMenuItem(hMenuI,5,MF_BYPOSITION|MF_GRAYED);//deshabilita
filtrar
            EnableMenuItem(hMenuI,CM_FGI,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MALTOS,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MBAJOS,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MBANDA,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_RAIZ,MF_GRAYED);

char Informa[45];
sprintf(Informa,"Gaussiano aditivo,  varianza = %d",(int)pow(sigma,2));
strcpy(Ruido,Informa);
            }
}

//añade ruido impulsivo sal y pimienta 255 o 0
void TMdiWindow::CMRIMPULPN(RTMessage)
{
      HayRuido=TRUE;
      float p=0;
      float huge *ptr;
      register int m,n;
      float b;
      int a=255;
      char archivoi[]="IMGORG.WLP";
      char archivof[]="IMGDATO.WLP";
      char Titulo[]="RUIDO SAL Y PIMIENTA";
      char Orden[]="Introduzca la probabilidad de ocurrencia [0 - 1]:";
      p=Adq_Dato(Titulo,Orden,p);

      if(p>=0&&p<=1){               //probabilidad siempre +
      Toma_Mem();
      Archivo_a_Mem(AltoImagen,AnchoImagen,memU,archivoi);
      ptr=memU;
      for(m=0;m<AltoImagen;m++)
      for(n=0;n<AnchoImagen;n++)     {
```

```
      b=Aleatorio();
      if(b>=1-p&&b<1-0.5*p)*ptr=a;
      if(b>=1-0.5*p&&b<=1)*ptr=0;
      ptr++;                          }
      Mem_a_Archivo(AltoImagen,AnchoImagen,memU,archivof);
      GetApplication()->MakeWindow(new    TGrafImgRuido(this,    "IMAGEN
RUIDOSA [Ruido aditivo Impulsivo pos/neg]"));
      EnableMenuItem(hMenuV,CM_IMGRUIDO,MF_ENABLED);//habilita       ver
imagen ruidosa
            //si la imagen se contamina nuevamente
            EnableMenuItem(hMenuE,CM_DESHACER,MF_GRAYED);//deshabilita
Deshacer
            EnableMenuItem(hMenuV,CM_IMGMEJOR,MF_GRAYED);//deshabilita
ver imagen mejorada
            EnableMenuItem(hMenuV,CM_ESPECTRO,MF_GRAYED);//deshabilita
ver espectro
//
      EnableMenuItem(hMenuI,5,MF_BYPOSITION|MF_GRAYED);//deshabilita
filtrar
            EnableMenuItem(hMenuI,CM_FGI,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MALTOS,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MBAJOS,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MBANDA,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_RAIZ,MF_GRAYED);

char Informa[45];
sprintf(Informa,"Sal y pimienta,  probabilidad= %4.3f",p);
strcpy(Ruido,Informa);
            }
}

//ruido impulso +, siempre 255
void TMdiWindow::CMRIMPULP(RTMessage)
{
      HayRuido=TRUE;
      float p=0;
      float huge *ptr;
      register int m,n;
      float b;
      int a=255;
      char archivoi[]="IMGORG.WLP";
      char archivof[]="IMGDATO.WLP";
      char Titulo[]="RUIDO IMPULSIVO POSITIVO";
      char Orden[]="Introduzca la probabilidad de ocurrencia [0 - 1]:";
      p=Adq_Dato(Titulo,Orden,p);

      if(p>=0&&p<=1){
      Toma_Mem();
      Archivo_a_Mem(AltoImagen,AnchoImagen,memU,archivoi);
      ptr=memU;
      for(m=0;m<AltoImagen;m++)
      for(n=0;n<AnchoImagen;n++)     {
      b=Aleatorio();
      if(b>1-p&&b<=1)*ptr=a;
      ptr++;                          }
      Mem_a_Archivo(AltoImagen,AnchoImagen,memU,archivof);
      GetApplication()->MakeWindow(new    TGrafImgRuido(this,    "IMAGEN
RUIDOSA [ruido aditivo impulsivo positivo]"));
      EnableMenuItem(hMenuV,CM_IMGRUIDO,MF_ENABLED);//habilita       ver
imagen ruidosa
            //si la imagen se contamina nuevamente
      EnableMenuItem(hMenuE,CM_DESHACER,MF_GRAYED);//deshabilita
Deshacer
      EnableMenuItem(hMenuV,CM_IMGMEJOR,MF_GRAYED);//deshabilita     ver
imagen mejorada
```

```
      EnableMenuItem(hMenuV,CM_ESPECTRO,MF_GRAYED);//deshabilita      ver
espectro
//EnableMenuItem(hMenuI,5,MF_BYPOSITION|MF_GRAYED);//deshabilita filtrar
            EnableMenuItem(hMenuI,CM_FGI,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MALTOS,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MBAJOS,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MBANDA,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_RAIZ,MF_GRAYED);
char Informa[45];
sprintf(Informa,"Impulsivo positivo,  probabilidad= %4.3f",p);
strcpy(Ruido,Informa);
            }
}
//ruido impulso - ,siempre 0
void TMdiWindow::CMRIMPULN(RTMessage)
{
      HayRuido=TRUE;
      float p=0;
      float huge *ptr;
      register int m,n;
      float b;
      int a=255;
      char archivoi[]="IMGORG.WLP";
      char archivof[]="IMGDATO.WLP";
      char Titulo[]="RUIDO IMPULSIVO NEGATIVO";
      char Orden[]="Introduzca la probabilidad de ocurrencia [0 - 1]:";
      p=Adq_Dato(Titulo,Orden,p);

      if(p>=0&&p<=1){
      Toma_Mem();
      Archivo_a_Mem(AltoImagen,AnchoImagen,memU,archivoi);
      ptr=memU;
      for(m=0;m<AltoImagen;m++)
      for(n=0;n<AnchoImagen;n++)     {
      b=Aleatorio();
      if(b>1-p&&b<=1)*ptr=0;
      ptr++;                         }
      Mem_a_Archivo(AltoImagen,AnchoImagen,memU,archivof);
      GetApplication()->MakeWindow(new    TGrafImgRuido(this,    "IMAGEN
RUIDOSA [ruido aditivo mpulsivo negativo]"));
      EnableMenuItem(hMenuV,CM_IMGRUIDO,MF_ENABLED);//habilita       ver
imagen ruidosa
            //si la imagen se contamina nuevamente
      EnableMenuItem(hMenuE,CM_DESHACER,MF_GRAYED);//deshabilita
Deshacer
      EnableMenuItem(hMenuV,CM_IMGMEJOR,MF_GRAYED);//deshabilita     ver
imagen mejorada
      EnableMenuItem(hMenuV,CM_ESPECTRO,MF_GRAYED);//deshabilita     ver
espectro
      EnableMenuItem(hMenuI,5,MF_BYPOSITION|MF_GRAYED);//deshabilita
filtrar
            EnableMenuItem(hMenuI,CM_FGI,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MALTOS,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MBAJOS,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MBANDA,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_RAIZ,MF_GRAYED);
char Informa[45];
sprintf(Informa,"Impulsivo negativo,  probabilidad= %4.3f",p);
strcpy(Ruido,Informa);
            }
}
```

```
//ruido dependiente de la señal
void TMdiWindow::CMRDEPEND(RTMessage)
{
            HayRuido=TRUE;
      mm=ll=1.0;
      float f;
      float huge *p;
      register int m,n,k;
      float suma=0;
      char archivoi[]="IMGORG.WLP";
      char archivof[]="IMGDATO.WLP";
      char Titulo[]="RUIDO DEPENDIENTE DE LA SEÑAL";
      char Orden[]="Introduzca el coeficiente l:";
      ll=Adq_Dato(Titulo,Orden,ll);
      strcpy(Orden,"Introduzca el coeficiente m:");
      mm=Adq_Dato(Titulo,Orden,mm);
      if(mm>0&&ll>0){          //coeficientes +
      Toma_Mem();
      Archivo_a_Mem(AltoImagen,AnchoImagen,memU,archivoi);
      p=memU;
      for(m=0;m<AltoImagen;m++)
      for(n=0;n<AnchoImagen;n++)     {
      for(k=0;k<12;k++)suma+=Aleatorio();
      suma-=6;
      *p=pow(*p,ll)+pow(*p,mm)*suma;
      if(*p>255)*p=255;
      if(*p<0)*p=0;
      suma=0;
      p++;                              }
      Mem_a_Archivo(AltoImagen,AnchoImagen,memU,archivof);
      GetApplication()->MakeWindow(new    TGrafImgRuido(this,    "IMAGEN
RUIDOSA [Ruido Dependiente de la señal]"));
      EnableMenuItem(hMenuV,CM_IMGRUIDO,MF_ENABLED);//habilita       ver
imagen ruidosa
            //si la imagen se contamina nuevamente
      EnableMenuItem(hMenuE,CM_DESHACER,MF_GRAYED);//deshabilita
Deshacer
      EnableMenuItem(hMenuV,CM_IMGMEJOR,MF_GRAYED);//deshabilita     ver
imagen mejorada
      EnableMenuItem(hMenuV,CM_ESPECTRO,MF_GRAYED);//deshabilita     ver
espectro
//EnableMenuItem(hMenuI,5,MF_BYPOSITION|MF_GRAYED);//deshabilita filtrar
            EnableMenuItem(hMenuI,CM_FGI,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MALTOS,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MBAJOS,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_MBANDA,MF_GRAYED);
            EnableMenuItem(hMenuI,CM_RAIZ,MF_GRAYED);
char Informa[45];
sprintf(Informa,"Dependiente de la señal");
strcpy(Ruido,Informa);
               }
}

//    Genera un numero aleatorio entre 0 y 1.
float TMdiWindow::Aleatorio()
{
      float num;
      num=(float)rand()/RAND_MAX;
      return num;
}
```

```
/*esta funcion es utilizable solo asegurandose que el puntero memV
apunta informacion valida, nunca luego de un TDC inversa    */
void TMdiWindow::Frec_a_Espac()
{
      DesHacer++; //indica que se uso al menos un filtro
      EnableMenuItem(hMenuE,CM_DESHACER,MF_ENABLED);//habilita Deshacer
      TDCI(AltoImagen,AnchoImagen,memU,memV);
      Escalar(AltoImagen,AnchoImagen,memU);
      char archivo[]="IMGMEJOR.WLP";
      Mem_a_Archivo(AltoImagen,AnchoImagen,memU,archivo);
GetApplication()->MakeWindow(new       TGrafImgMejor(this,       "IMAGEN
FILTRADA"));
EnableMenuItem(hMenuV,CM_IMGMEJOR,MF_ENABLED);//habilita    ver    imagen
mejorada
}

//asigna memoria, siempre que no este asignada
void TMdiWindow::Toma_Mem()
{
      if(!hMemU&&!hMemV){memU=Ini_Mem(AltoImagen,AnchoImagen,&hMemU);
                  memV=Ini_Mem(AltoImagen,AnchoImagen,&hMemV);}
}

//verifica si la imagen es de tamaño adecuado
float TMdiWindow::ProbarPotenciaDos(int tam)
{
float result=(float)tam;
while(result>2.0){
            result=(float)result/2;
            }
return result;
}

//    funcion que presenta un mensaje al usuario
int TMdiWindow::Mensaje(int a,float valor)
{
char texto[60];
if(a==1)sprintf(texto,"FILTRO:    Raiz\rAlpha    =    %3.2f\r\rCORRECTO
?",valor);
if(a==2)sprintf(texto,"FILTRO:   Pasa   Bajos\rRadio   =   %d\r\rCORRECTO
?",(int)valor);
if(a==3)sprintf(texto,"FILTRO:   Pasa   Altos\rRadio   =   %d\r\rCORRECTO
?",(int)valor);
if(a==4)sprintf(texto,"FILTRO:    Pasa    Banda\rRadio    Inferior    =
%d\r\rCORRECTO ?",(int)valor);
if(a==5)sprintf(texto,"FILTRO:    Pasa    Banda\rRadio    Superior    =
%d\r\rCORRECTO ?",(int)valor);

int         c=MessageBox(HWindow,texto,"FILTRO:         CARACTERISTICAS
SELECCCIONADAS",MB_YESNO|MB_ICONQUESTION);
if(c==IDYES)return 1;
else return 0;
}

//    funcion para adquisicion de dato float mediante caja de dialogo
float TMdiWindow::Adq_Dato(char *Titulo,char *Orden,float defecto)
{     char texto[10];
      float dato=defecto;
      sprintf(texto,"%7.3f",dato);
      if(GetApplication()->ExecDialog(new
TInputDialog(this,Titulo,Orden,texto,sizeof(texto)))==IDOK)
      {dato=atof(texto);
      if(dato<0)dato=defecto;}
      else dato=-1.0;         //cuando escoge CANCEL devuelve -1
      return dato;
}
```

```
//funcion para adquisicion de dato int mediante caja de dialogo
int TMdiWindow::Adq_Dato(char *Titulo,char *Orden,int defecto)
{
      char texto[8];
      int dato=defecto;
      sprintf(texto,"%d",dato);
      if(GetApplication()->ExecDialog(new
TInputDialog(this,Titulo,Orden,texto,sizeof(texto)))==IDOK)
        {   dato=atoi(texto);
            if(dato<0)dato=defecto;
        }
      else dato=-1;
      return dato;
}

//calcula el logaritmo natural de coeficientes transformados
void TMdiWindow::LogNatural(float huge *p)
{
      BOOL Signo;
      for(register int k=0;k<AltoImagen;k++)
      for(register int l=0;l<AnchoImagen;l++)    {
      Signo=FALSE;
      if(*p<0){Signo=TRUE;*p=-*p;}
      *p=(float)log(1.0+*p);
      if(Signo)*p=-*p;
      p++;                              }
}

//calcula el exponencial de coef. transformados, operacion inversa a la
anterior
void TMdiWindow::Exponencial(float huge *p)
{
      BOOL Signo;
      for(register int k=0;k<AltoImagen;k++)
      for(register int l=0;l<AnchoImagen;l++)    {
      Signo=FALSE;
      if(*p<0){Signo=TRUE;*p=-*p;}
      *p=(float)exp(*p)-1.0;
      if(Signo)*p=-*p;
      p++;                              }
}

//inicializa puntero de memoria,
 float huge *TMdiWindow::Ini_Mem(int M,int N,HANDLE *hMem)
 {
            *hMem=GlobalAlloc(GMEM_FIXED|GMEM_ZEROINIT,(long
int)M*N*sizeof(float));
            return (float huge *)GlobalLock(*hMem);
 }

//rutinas para visualizacion de ventanas ocultas.
void TMdiWindow::CMIMGRUIDO(RTMessage)
 {
GetApplication()->MakeWindow(new TGrafImgRuido(this, "IMAGEN RUIDOSA"));
 }

void TMdiWindow::CMESPECTRO(RTMessage)
 {
GetApplication()->MakeWindow(new   TGraficarEspectro(this,   "FRECUENCIA
ESPACIAL [Imagen Actual]"));
GetApplication()->MakeWindow(new   TGraficarEspecorg(this,   "FRECUENCIA
ESPACIAL [Imagen Original]"));
if(HayRuido)GetApplication()->MakeWindow(new     TGraficarEspecrui(this,
"FRECUENCIA ESPACIAL [Imagen Ruidosa]"));
 }
```

```
void TMdiWindow::CMIMGMEJOR(RTMessage)
 {
GetApplication()->MakeWindow(new       TGrafImgMejor(this,       "IMAGEN
FILTRADA"));
 }


//usa filtro pasa en dominio de la frecuencia logaritmica
void TMdiWindow::CMHOMOMOR(RTMessage)
{     if(Compatibilidad())    {
      Toma_Mem();                          //reserva memoria para imagen
      char archivo[]="IMGDATO.WLP";
      int Key;
      Archivo_a_Mem(AltoImagen,AnchoImagen,memU,archivo);
if(11==mm)  LogNatural(memU);
      else g_no_lineal(memU);
      TDCD(AltoImagen,AnchoImagen,memU,memV);

      char Informa[45];
      int
RadioPasaBajos=(int)sqrt((DWORD)AltoImagen*AltoImagen+(DWORD)AnchoImagen
*AnchoImagen);
      float Exponente;
      float huge *p;
      p=memV;
      register int k,l;
      if(!HayRuido)     {
            for(k=0;k<AltoImagen;k++)
            for(l=0;l<AnchoImagen;l++)    {
            Exponente=(float)3.5*((DWORD)k*k+(DWORD)l*l);
                  Exponente/=((DWORD)AnchoImagen*AltoImagen);
                  *p*=(0.5+1.5*(1-exp(-Exponente)));
                  p++;                  }
            sprintf(Informa,"Homomórfico: enfatiza alta frecuencia");
                  }
      else  {
int r;
p=memV;
char Titulo[]="FILTRO PASA BAJOS";
char Orden[]="Introducir Radio Pasa Bajos:";
RadioPasaBajos=Adq_Dato(Titulo,Orden,RadioPasaBajos);
      if(RadioPasaBajos>=0){         //radio no puede ser negativo
      if(Key=Mensaje(2,(float)RadioPasaBajos))  {
            for(k=0;k<AltoImagen;k++)
            for(l=0;l<AnchoImagen;l++)    {
            r=(int)sqrt((DWORD)k*k+(DWORD)l*l);
            if(r>RadioPasaBajos)*p=0;
            p++;                        }
                                        }
            sprintf(Informa,"Homomórfico:     Pasa-bajos,     Radio     =
%d",RadioPasaBajos);
                  }
            }

if(RadioPasaBajos>=0&&Key){
            Frec_a_Espac_H();

      strcpy(Filtro,Informa);
                        //cálculo de los espectros\\
char arch[]="IMGDATO.WLP";
Archivo_a_Mem(AltoImagen,AnchoImagen,memU,arch);
TDCD(AltoImagen,AnchoImagen,memU,memV);
strcpy(arch,"ESPECORG.WLP");
Mem_a_Archivo(AltoImagen,AnchoImagen,memV,arch);

strcpy(arch,"IMGMEJOR.WLP");
```

```
Archivo_a_Mem(AltoImagen,AnchoImagen,memU,arch);
TDCD(AltoImagen,AnchoImagen,memU,memV);
strcpy(arch,"ESPECTRO.WLP");
Mem_a_Archivo(AltoImagen,AnchoImagen,memV,arch);
GetApplication()->MakeWindow(new   TGraficarEspecorg(this,   "FRECUENCIA
ESPACIAL [Imagen original]"));
GetApplication()->MakeWindow(new   TGraficarEspectro(this,   "FRECUENCIA
ESPACIAL [Imagen actual]"));
EnableMenuItem(hMenuV,CM_ESPECTRO,MF_ENABLED);//habilita ver espectro
}
                        }
}


void TMdiWindow::Frec_a_Espac_H()
{
      EnableMenuItem(hMenuE,CM_DESHACER,MF_GRAYED);//deshabilita
Deshacer
      TDCI(AltoImagen,AnchoImagen,memU,memV);
if(ll==mm)  Exponencial(memU);
else        f_no_lineal(memU);
      Escalar(AltoImagen,AnchoImagen,memU);
      char archivo[]="IMGMEJOR.WLP";
      Mem_a_Archivo(AltoImagen,AnchoImagen,memU,archivo);
GetApplication()->MakeWindow(new       TGrafImgMejor(this,       "IMAGEN
FILTRADA"));
EnableMenuItem(hMenuV,CM_IMGMEJOR,MF_ENABLED);//habilita   ver   imagen
mejorada
}
void TMdiWindow::g_no_lineal(float huge *p)
{
      float c;
      c=(float)(ll-mm)/ll;
      for(register int k=0;k<AltoImagen;k++)
      for(register int l=0;l<AnchoImagen;l++)   {
      *p=(float)pow(*p,c);
      *p/=(float)c;
      p++;                          }
}


void TMdiWindow::f_no_lineal(float huge *p)
{
      BOOL Signo;
      float c;
      c=(float)ll/(ll-mm);
      for(register int k=0;k<AltoImagen;k++)
      for(register int l=0;l<AnchoImagen;l++)   {
      Signo=FALSE;
      if(*p<0){Signo=TRUE;*p=-*p;}
      *p=(float)pow((float)*p/c,(float)c/ll);
      if(Signo)*p=-*p;
      p++;                          }
}


//_____________     fin bloque de funciones _____________


void TMyApp::InitMainWindow()
{
      MainWindow=     new     TMdiWindow("MEJORAMIENTO     DE     IMAGENES
MONOCROMATICAS USANDO LA TDC", "MENU_1");
}
```

```
int   PASCAL   WinMain(HANDLE   hInstance,   HANDLE   hPrevInstance,   LPSTR
lpCmdLine, int nCmdShow)
{
      TMyApp    MyApp("MDI",    hInstance,    hPrevInstance,    lpCmdLine,
nCmdShow);
      MyApp.Run();
      return MyApp.Status;
}
```

ABRIR.H

```
#include <owl.h>
#include <filedial.h>
#include <string.h>
#include <dir.h>
#include <dos.h>

#define BSA_NAME "BitmapScroll"

//variable globales que identifican a la imagen
extern char NombreArchivo[MAXPATH];
extern WORD AnchoImagen,AltoImagen;
extern WORD BitsPixel;
extern BOOL Acceso;
extern float NivelMax,NivelMin;
extern void huge Copiar(HWND HWindow);
extern void huge CursorArenal();
extern void huge CursorFlechal();

/* TBMPFileDialog, a TFileDialog descendant which uses .bmp as its
    default extension. */
_CLASSDEF(TBMPFileDialog)
class TBMPFileDialog : public TFileDialog {
public:
    TBMPFileDialog(PTWindowsObject AParent, int ResourceId,
                LPSTR AFilePath, PTModule AModule = NULL)
     : TFileDialog(AParent, ResourceId, AFilePath, AModule)
       { strcpy(Extension, ".bmp"); }
};

/* TBitScrollWindow, a TWindow descendant */

_CLASSDEF(TBitScrollWindow)
class TBitScrollWindow : public TWindow {
public:
    char FileName[MAXPATH];
    HBITMAP BitmapHandle;
    WORD PixelHeight , PixelWidth;
    long Mode;
    BOOL primera;

    TBitScrollWindow(PTWindowsObject AParent,LPSTR ATitle);
    virtual ~TBitScrollWindow();
    virtual LPSTR GetClassName();
    virtual void GetWindowClass(WNDCLASS&);
    virtual void Paint(HDC, PAINTSTRUCT&);
    void Abrir();
    virtual void WMSize(TMessage&) = [WM_FIRST + WM_SIZE];
    void AdjustScroller();
    BOOL LoadBitmapFile(LPSTR);
    BOOL OpenDIB(int TheFile);
    void GetBitmapData(int TheFile, HANDLE, long);
    void PonerEnArchivo(HDC hDC);
```

```
                        virtual      void      WMLButtonDblClk(RTMessage
Msg)=[WM_FIRST+WM_LBUTTONDBLCLK];
};
/*  __ahIncr, ordinal 114, is a 'magic' function. Defining this
   function causes Windows to patch the value into the passed
   reference.  This makes it a type of global variable. To use
   the value of AHIncr, use FP_OFF(AHIncr). */

extern "C" {
void FAR PASCAL  __ahIncr();
}
```

ABRIR.CPP

```
#include "abrir.h"
#include <stdio.h>

/* Constructor for a TBitScrollWindow, sets scroll styles and constructs
   the Scroller object.  Also sets the Mode based on whether the display
   is monochrome (two-color) or polychrome. */

TBitScrollWindow::TBitScrollWindow(PTWindowsObject AParent,LPSTR ATitle)
            :TWindow(AParent, ATitle)
{
  HDC DCHandle;
  primera=FALSE;

  Attr.Style |= WS_VSCROLL | WS_HSCROLL;
  Attr.W=256;      //dimensiones de la ventana (anchura)
  Attr.H=256;      //altura
  BitmapHandle = 0;
  Scroller = new TScroller(this, 1, 1, 200, 200);
  DCHandle = CreateDC("Display", NULL, NULL, NULL);
  if ( GetDeviceCaps(DCHandle, NUMCOLORS) < 3 )
    Mode = NOTSRCCOPY;
  else
    Mode = SRCCOPY;
  DeleteDC(DCHandle);
}

/* Change the class name to the application name. */

LPSTR TBitScrollWindow::GetClassName()
{
  return BSA_NAME;
}

/* Allow the iconic picture to be drawn from the client area. */

void TBitScrollWindow::GetWindowClass(WNDCLASS& WndClass)
{
  TWindow::GetWindowClass(WndClass);
  WndClass.style =CS_DBLCLKS; //permite reconocer el doble click.
   WndClass.hIcon   =LoadIcon(GetApplication()->hInstance,"ESPACIO");   /*
Client area will be painted by the app. */
}

TBitScrollWindow::~TBitScrollWindow()
{
  if ( BitmapHandle )
    DeleteObject(BitmapHandle);
}

/* If the the "Open..." menu item is selected, then we prompt the user
   for a new bitmap file.  If the user selects one and it is one that
```

```
   we can read, we display it in the window and change the window's
   caption to reflect the new bitmap file. */

void TBitScrollWindow::Abrir()
{  char CaptionBuffer [MAXPATH+ 12 /*BSA_NAME*/ + 2 /*" "*/ + 1 /*'\0'*/
];

    if ( LoadBitmapFile(NombreArchivo) )
    {
      strcpy(FileName, NombreArchivo);
//      strcpy(NombreArchivo, TempName);
      strcpy(CaptionBuffer, strlwr(FileName));
      SetWindowText(HWindow, CaptionBuffer);
    }
}

/* Adjust the Scroller range so that the the origin is the
  upper-most scrollable point and the corner is the
  bottom-most. */

void TBitScrollWindow::AdjustScroller()
{
  RECT ClientRect;

  GetClientRect(HWindow, &ClientRect);
  Scroller->SetRange(PixelWidth - (ClientRect.right - ClientRect.left),
      PixelHeight - (ClientRect.bottom - ClientRect.top));
  Scroller->ScrollTo(0, 0);
  InvalidateRect(HWindow, NULL, TRUE);
}

/* Reset scroller range. */

void TBitScrollWindow::WMSize(TMessage& Msg)
{
  TWindow::WMSize(Msg);
  if ( (Msg.WParam != SIZEICONIC) )
    AdjustScroller();
}

/* Copys the bitmap bit data from the file into memory. Since
  copying cannot cross a segment (64K) boundary, we are forced
  to do segment arithmetic to compute the next segment.  Created
  a LongType type to simplify the process. */

void TBitScrollWindow::GetBitmapData(int TheFile, HANDLE BitsHandle,long
BitsByteSize)
{
  long Count;
  long Start, ToAddr, Bits;

  Start = 0L;
  Bits = (long)GlobalLock(BitsHandle);
  Count = BitsByteSize - Start;
  while ( Count > 0 )
  {
    ToAddr = MAKELONG(LOWORD(Start),
                  HIWORD(Bits) + (HIWORD(Start) * FP_OFF(__ahIncr)));
    if ( Count > 0x4000 )
      Count = 0x4000;
    _lread(TheFile, (LPSTR)ToAddr, (WORD)Count);
    Start = Start + Count;
    Count = BitsByteSize - Start;
  }
  GlobalUnlock(BitsHandle);
}
```

```
/* Attempt to open a Windows 3.0 device independent bitmap. */

BOOL TBitScrollWindow::OpenDIB(int TheFile)
{
    WORD bitCount;
    WORD size;
    long longWidth;
    HDC DCHandle;
    LPSTR BitsPtr;
    BITMAPINFO *BitmapInfo;
    HANDLE BitsHandle , NewBitmapHandle;
    DWORD NewPixelWidth , NewPixelHeight;
    BOOL retval;

  retval= TRUE;
  _llseek(TheFile, 28, 0);
  _lread(TheFile, (LPSTR)&bitCount, sizeof(bitCount));
  BitsPixel=bitCount;
  if ( bitCount <= 8 )
  {
         size = sizeof(BITMAPINFOHEADER) + ((1 << bitCount) *
sizeof(RGBQUAD));
    BitmapInfo = (BITMAPINFO *)new char[size];
    _llseek(TheFile, sizeof(BITMAPFILEHEADER), 0);
    _lread(TheFile, (LPSTR)BitmapInfo, size);
    NewPixelWidth = BitmapInfo->bmiHeader.biWidth;
    NewPixelHeight = BitmapInfo->bmiHeader.biHeight;
    longWidth = (((NewPixelWidth * bitCount) + 31)/32) * 4;
    BitmapInfo->bmiHeader.biSizeImage = longWidth * NewPixelHeight;
    GlobalCompact(-1);
    BitsHandle = GlobalAlloc(GMEM_MOVEABLE | GMEM_ZEROINIT,
      BitmapInfo->bmiHeader.biSizeImage);
                GetBitmapData(TheFile,     BitsHandle,     BitmapInfo-
>bmiHeader.biSizeImage);
    DCHandle = CreateDC("Display", NULL, NULL, NULL);
    BitsPtr = GlobalLock(BitsHandle);
    NewBitmapHandle =
          CreateDIBitmap(DCHandle, &(BitmapInfo->bmiHeader), CBM_INIT,
BitsPtr,
      BitmapInfo, 0);
    DeleteDC(DCHandle);
    GlobalUnlock(BitsHandle);
    GlobalFree(BitsHandle);
    delete BitmapInfo;
    if ( NewBitmapHandle )
    {
      if ( BitmapHandle )
      DeleteObject(BitmapHandle);
      BitmapHandle = NewBitmapHandle;
      PixelWidth = (WORD)NewPixelWidth;
      PixelHeight = (WORD)NewPixelHeight;
    }
    else
      retval = FALSE;
  }
  else          retval = FALSE;
return retval;
}

/* Test if the passed file is a Windows 3.0 DI bitmap and if so read it.
  Report errors if unable to do so. Adjust the Scroller to the new
  bitmap dimensions. */

BOOL TBitScrollWindow::LoadBitmapFile(LPSTR Name)
{
    int TheFile;
```

```
    long TestWin30Bitmap;
    char ErrorMsg[50] = "";
    BOOL retval;

  TheFile = _lopen(Name, OF_READ);
  if ( TheFile != -1 )
  {
    _llseek(TheFile, 14, 0);
    _lread(TheFile, (LPSTR)&TestWin30Bitmap, sizeof(TestWin30Bitmap));
    if ( TestWin30Bitmap == 40 )
      if ( OpenDIB(TheFile) )
        AdjustScroller();
      else
        strcpy(ErrorMsg,
                "Unable to create Windows 3.0 bitmap from file");
    else
      strcpy(ErrorMsg, "Not a Windows 3.0 bitmap file");
    _lclose(TheFile);
  }
  else
    strcpy(ErrorMsg, "Cannot open bitmap file");
  if ( ErrorMsg[0] == '\0' )
    retval= TRUE;
  else
  { MessageBox(HWindow, ErrorMsg, BSA_NAME, MB_OK);
    retval= FALSE;
  }
  return retval;
}
/* Responds to an incoming "paint" message by redrawing the bitmap.
   (The Scroller's BeginView method, which sets the viewport origin
    relative to the present scroll position, has already been called.)
*/

void TBitScrollWindow::Paint(HDC, PAINTSTRUCT& PaintInfo)
{
  CursorArenal();
  HDC MemoryDC;
  HANDLE OldBitmapHandle;
  RECT ClientRect;

  if(!primera){Abrir();}
  if ( BitmapHandle )
  {
  AnchoImagen=PixelWidth;
  AltoImagen= PixelHeight;
    MemoryDC = CreateCompatibleDC(PaintInfo.hdc);
    OldBitmapHandle = SelectObject(MemoryDC, BitmapHandle);
    if ( Mode == SRCCOPY )
    {
                SetBkColor(PaintInfo.hdc,   GetNearestColor(PaintInfo.hdc,
0x800000L));
      SetTextColor(PaintInfo.hdc, 0xFFFFFFL);
    }
    if ( IsIconic(HWindow) )
    {
      GetClientRect(HWindow, &ClientRect);
      StretchBlt(PaintInfo.hdc, 0, 0,
             ClientRect.right - ClientRect.left,
             ClientRect.bottom - ClientRect.top,
             MemoryDC, 0, 0, PixelWidth, PixelHeight, Mode);
    }
    else
      BitBlt(PaintInfo.hdc, 0, 0, PixelWidth, PixelHeight,
           MemoryDC, 0, 0, Mode);
    if(!primera){primera=TRUE;PonerEnArchivo(MemoryDC);}
```

```
    SelectObject(MemoryDC, OldBitmapHandle);
    DeleteDC(MemoryDC);
  }
  CursorFlecha1();
}


void TBitScrollWindow::PonerEnArchivo(HDC hDC)
{
register int m,n;
COLORREF ColorPixel;
float GrisPixel;
char archivo[]="IMGDATO.WLP";
FILE *fpo;
FILE *fpf;

fpo=fopen("IMGORG.WLP","wb");
fpf=fopen(archivo,"wb");
for(m=0;m<AltoImagen;m++){
for(n=0;n<AnchoImagen;n++){
ColorPixel=GetPixel(hDC,n,m);
GrisPixel=(float)GetRValue(ColorPixel);
      if(!m&&!n)NivelMax=NivelMin=GrisPixel;
      if(GrisPixel>NivelMax)NivelMax=GrisPixel;
      if(GrisPixel<NivelMin)NivelMin=GrisPixel;
fwrite(&GrisPixel,sizeof(float),1,fpo);
fwrite(&GrisPixel,sizeof(float),1,fpf);
                  }
                  }
fclose(fpo);
fclose(fpf);
}

      //Función respuesta al doble-click botón izq.
void TBitScrollWindow::WMLButtonDblClk(RTMessage)
{
HWND HWindow;
Copiar(HWindow);
}
```

PEGAR.H

```
#include <owl.h>
#include <string.h>
#include <dir.h>
#include <stdio.h>

//variable globales que identifican a la imagen
extern char NombreArchivo[MAXPATH];
extern WORD AnchoImagen,AltoImagen;
extern WORD BitsPixel;
extern BOOL Acceso;
extern float NivelMax,NivelMin;
extern void huge Copiar(HWND HWindow);
extern BOOL primer;
extern void huge CursorArena1();
extern void huge CursorFlecha1();

/* TPegarWindow, a TWindow descendant */

class TPegarWindow : public TWindow {
public:
    HBITMAP hBitmap;
```

```
    WORD alturaPixel, anchuraPixel;
    //BOOL primer;
    TPegarWindow(PTWindowsObject AParent,LPSTR ATitle);
    virtual ~TPegarWindow();
    virtual LPSTR GetClassName();
    virtual void GetWindowClass(WNDCLASS&);
    virtual void Paint(HDC, PAINTSTRUCT&);
    void Pegar();
    virtual void WMSize(TMessage&) = [WM_FIRST + WM_SIZE];
    void AdjustScroller();
    void PonerEnArchivo(HDC hDC);
                        virtual      void      WMLButtonDblClk(RTMessage
Msg)=[WM_FIRST+WM_LBUTTONDBLCLK];
};
```

PEGAR.CPP

```
#include "pegar.h"

/* Constructor for a TPegarWindow, sets scroll styles and constructs
  the Scroller object.*/

TPegarWindow::TPegarWindow(PTWindowsObject AParent,LPSTR ATitle)
            :TWindow(AParent, ATitle)
{
//  primer=FALSE;
  Attr.Style |= WS_VSCROLL | WS_HSCROLL;
  Attr.W=256;      //dimensiones de la ventana (anchura)
  Attr.H=256;      //altura
  hBitmap = 0;
  Scroller = new TScroller(this, 1, 1, 200, 200);
}

/* Change the class name to the application name. */

LPSTR TPegarWindow::GetClassName()
{
  return "PEGAR";
}

/* Allow the iconic picture to be drawn from the client area. */

void TPegarWindow::GetWindowClass(WNDCLASS& WndClass)
{
  TWindow::GetWindowClass(WndClass);
  WndClass.style =CS_DBLCLKS; //permite reconocer el doble click.
    WndClass.hIcon  =LoadIcon(GetApplication()->hInstance,"ESPACIO");  /*
Client area will be painted by the app. */
}

TPegarWindow::~TPegarWindow()
{
  if (hBitmap)
    DeleteObject(hBitmap);
}

void TPegarWindow::Pegar()
{
  BITMAP bmPegar;         // contendr  la informaci¢n del mapa de bi+s

  if (OpenClipboard(HWindow))     // si el Portapapeles si abri¢ bien */
  {                              // ...obtener el gestor del mapa de bits */
  if (!(hBitmap= GetClipboardData(CF_BITMAP)))
    {                            // si hay error al obtener el gestor... */
    CloseClipboard();
```

```
                                        MessageBox(HWindow,"PORTAPAPELES
VACIO","PORTAPAPELES",MB_OK|MB_ICONSTOP);
    }

  GetObject(hBitmap, // obtener informaci¢n de cabecera del mapa bits */
          sizeof(BITMAP),(LPSTR) &bmPegar);
  AnchoImagen=anchuraPixel=bmPegar.bmWidth;
  AltoImagen=alturaPixel=bmPegar.bmHeight;
  BitsPixel=bmPegar.bmBitsPixel;
  CloseClipboard();                    /* liberar el Portapapeles */
  }
}

/* Adjust the Scroller range so that the the origin is the
  upper-most scrollable point and the corner is the
  bottom-most. */

void TPegarWindow::AdjustScroller()
{
  RECT ClientRect;
  GetClientRect(HWindow, &ClientRect);
         Scroller->SetRange(anchuraPixel     -     (ClientRect.right     -
ClientRect.left),
      alturaPixel - (ClientRect.bottom - ClientRect.top));
  Scroller->ScrollTo(0, 0);
  InvalidateRect(HWindow, NULL, TRUE);
}

/* Reset scroller range. */

void TPegarWindow::WMSize(TMessage& Msg)
{
  TWindow::WMSize(Msg);
  if ( (Msg.WParam != SIZEICONIC) )
    AdjustScroller();
}


void TPegarWindow::Paint(HDC MemoryDC, PAINTSTRUCT& PaintInfo)
{
 Cursor Arenal();

 if(!primer)
  {Pegar();

  HANDLE OldhBitmap;
  RECT ClientRect;

  if (hBitmap)
  {    MemoryDC = CreateCompatibleDC(PaintInfo.hdc);
    OldhBitmap = SelectObject(MemoryDC, hBitmap);
    if ( IsIconic(HWindow) )
    {
      GetClientRect(HWindow, &ClientRect);
      StretchBlt(PaintInfo.hdc, 0, 0,
             ClientRect.right - ClientRect.left,
             ClientRect.bottom - ClientRect.top,
             MemoryDC, 0, 0, anchuraPixel, alturaPixel, SRCCOPY);
    }    else
      BitBlt(PaintInfo.hdc, 0, 0, anchuraPixel, alturaPixel,
           MemoryDC, 0, 0, SRCCOPY);
    if(!primer){primer=TRUE;PonerEnArchivo(MemoryDC);}
    SelectObject(MemoryDC, OldhBitmap);
    DeleteDC(MemoryDC);
  }
  }
```

```
else  //graficar desde archivo
  {
register int m,n;
int color;
FILE *fp;
float u;

if(fp=fopen("IMGORG.WLP","rb"))      {
for(m=0;m<AltoImagen;m++)      {
for(n=0;n<AnchoImagen;n++)     {
fread(&u,sizeof(float),1,fp);
color=(int)u;
SetPixel(MemoryDC,n,m,RGB(color,color,color));
                          }
                          }
fclose(fp);               }
  }
CursorFlechal();
}

void TPegarWindow::PonerEnArchivo(HDC hDC)
{
register int m,n;
COLORREF ColorPixel;
float GrisPixel;
char archivo[]="IMGDATO.WLP";
FILE *fpo;
FILE *fpf;

fpo=fopen("IMGORG.WLP","wb");
fpf=fopen(archivo,"wb");
for(m=0;m<AltoImagen;m++){
for(n=0;n<AnchoImagen;n++){
ColorPixel=GetPixel(hDC,n,m);
GrisPixel=(float)GetRValue(ColorPixel);
      if(!m&&!n)NivelMax=NivelMin=GrisPixel;
      if(GrisPixel>NivelMax)NivelMax=GrisPixel;
      if(GrisPixel<NivelMin)NivelMin=GrisPixel;
fwrite(&GrisPixel,sizeof(float),1,fpo);
fwrite(&GrisPixel,sizeof(float),1,fpf);
                  }
                  }
fclose(fpo);
fclose(fpf);
}

      //Función respuesta al doble-click botón izq.
void TPegarWindow::WMLButtonDblClk(RTMessage)
{
HWND HWindow;
Copiar(HWindow);
}
```

IMGRUIDO.H

```
//GRAFICACION DE LA IMAGEN RUIDOSA

#include <owl.h>
#include <stdio.h>
#include <math.h>

extern WORD AltoImagen,AnchoImagen;
extern char Ruido[50];  //variable externa que indica tipo de ruido
extern void huge Copiar(HWND HWindow);
```

```
extern void huge CursorArenal();
extern void huge CursorFlechal();

_CLASSDEF(TGrafImgRuido)
class TGrafImgRuido : public TWindow
{
public:
      TGrafImgRuido(PTWindowsObject AParent, LPSTR ATitle);
      virtual void Paint(HDC PaintDC, PAINTSTRUCT& );
      virtual void GetWindowClass(WNDCLASS& WndClass);
      virtual LPSTR GetClassName();
      virtual void WMSize(TMessage&) = [WM_FIRST + WM_SIZE];
      void AdjustScroller();
      virtual                 void                 WMRButtonDown(TMessage&
Msg)=[WM_FIRST+WM_RBUTTONDOWN];
      virtual                void                WMLButtonDblClk(RTMessage
Msg)=[WM_FIRST+WM_LBUTTONDBLCLK];
};
```

IMGRUIDO.CPP

```
//GRAFICACION DE LA IMAGEN RUIDOSA

#include "imgruido.h"

TGrafImgRuido::TGrafImgRuido(PTWindowsObject  AParent,LPSTR  ATitle)  :
TWindow(AParent, ATitle)
{  Attr.Style |= WS_VSCROLL | WS_HSCROLL;
  Attr.W=(int)AnchoImagen;     //dimensiones de la ventana (anchura)
  Attr.H=(int)AltoImagen;      //altura
  Scroller = new TScroller(this, 1,1, 200,200);
}
LPSTR TGrafImgRuido::GetClassName()
{
  return "IMAGEN MEJORADA";
}

void TGrafImgRuido::GetWindowClass(WNDCLASS& WndClass)
{
  TWindow::GetWindowClass(WndClass);
  WndClass.style =CS_DBLCLKS; //permite reconocer el doble click.
  WndClass.hIcon =LoadIcon(GetApplication()->hInstance,"ESPACIO");
}

void TGrafImgRuido::Paint(HDC hDC, PAINTSTRUCT&)
{
CursorArenal();
       //GRAFICACION DE LA IMAGEN RUIDOSA\\
register int m,n;
int color;
FILE *fp;
float u;

if(fp=fopen("IMGDATO.WLP","rb"))     {
for(m=0;m<AltoImagen;m++)      {
for(n=0;n<AnchoImagen;n++)     {
fread(&u,sizeof(float),1,fp);
color=(int)u;
SetPixel(hDC,n,m,RGB(color,color,color));
                        }
                        }
fclose(fp);                     }
CursorFlechal();
}
```

```
void TGrafImgRuido::AdjustScroller()
{
  RECT ClientRect;

  GetClientRect(HWindow, &ClientRect);
  Scroller->SetRange(AnchoImagen - (ClientRect.right - ClientRect.left),
      AltoImagen - (ClientRect.bottom - ClientRect.top));
  Scroller->ScrollTo(0, 0);
  InvalidateRect(HWindow, NULL, TRUE);
}


void TGrafImgRuido::WMSize(TMessage& Msg)
{
  TWindow::WMSize(Msg);
  if ( (Msg.WParam != SIZEICONIC) )
    AdjustScroller();
}


      //Función respuesta al click botón derecho del ratón
void TGrafImgRuido::WMRButtonDown(RTMessage)
{
CursorArena1();
            //calcula la relación señal a ruido
register int m,n;
float Pixel_o,Pixel_m;
char archivo[]="IMGORG.WLP";
FILE *fo;
FILE *fm;
double SNR_db,e_rms=0;

fo=fopen(archivo,"rb");
fm=fopen("IMGDATO.WLP","rb");

for(m=0;m<AltoImagen;m++){
for(n=0;n<AnchoImagen;n++){
fread(&Pixel_o,sizeof(float),1,fo);
fread(&Pixel_m,sizeof(float),1,fm);
e_rms+=pow(Pixel_o-Pixel_m,2);
                  }
                  }
fclose(fo);
fclose(fm);

e_rms=sqrt(e_rms);
e_rms/=(double)sqrt((float)AltoImagen*AnchoImagen);
if(e_rms<0.001)e_rms=0.001;
SNR_db=20*log10((double)255/e_rms);
char texto[100];
sprintf(texto,"Ruido: %s\r\rSNR = %6.2f dB",Ruido,SNR_db);
CursorFlecha1();
MessageBox(HWindow,texto,"IMAGEN CONTAMINADA",MB_OK|MB_ICONINFORMATION);
}

      //Función respuesta al doble-click botón izq.
void TGrafImgRuido::WMLButtonDblClk(RTMessage)
{
HWND HWindow;
Copiar(HWindow);
}
```

ESPECTRO.H

```
#include <owl.h>
#include <stdio.h>
#include <math.h>

extern WORD AnchoImagen,AltoImagen;
extern char Filtro[50];//variable externa que indica tipo de filtro
extern void huge Copiar(HWND HWindow);
extern void huge CursorArenal();
extern void huge CursorFlechal();


_CLASSDEF(TGraficarEspectro)
class TGraficarEspectro : public TWindow
{
public:
      TGraficarEspectro(PTWindowsObject AParent, LPSTR ATitle);
      virtual LPSTR GetClassName();
      virtual void GetWindowClass(WNDCLASS&);
      virtual void Paint(HDC PaintDC, PAINTSTRUCT& );
      virtual void WMSize(TMessage&) = [WM_FIRST + WM_SIZE];
      void AdjustScroller();
      virtual                 void                WMRButtonDown(TMessage&
Msg)=[WM_FIRST+WM_RBUTTONDOWN];
      virtual               void              WMLButtonDblClk(RTMessage
Msg)=[WM_FIRST+WM_LBUTTONDBLCLK];

};
```

ESPECTRO.CPP

```
#include "espectro.h"

TGraficarEspectro::TGraficarEspectro(PTWindowsObject       AParent,LPSTR
ATitle) : TWindow(AParent, ATitle)
{
  Attr.Style |= WS_VSCROLL | WS_HSCROLL;
  Attr.W=(int)AnchoImagen;     //dimensiones de la ventana (anchura)
  Attr.H=(int)AltoImagen;      //altura
  Scroller = new TScroller(this, 1,1, 200,200);
}

LPSTR TGraficarEspectro::GetClassName()
{
  return "ESPECTRO";
}

void TGraficarEspectro::GetWindowClass(WNDCLASS& WndClass)
{
  TWindow::GetWindowClass(WndClass);
  WndClass.style =CS_DBLCLKS; //permite reconocer el doble click.
   WndClass.hIcon  =LoadIcon(GetApplication()->hInstance,"ESPECTRO");  /*
Client area will be painted by the app. */
}

void TGraficarEspectro::Paint(HDC hDC, PAINTSTRUCT&)
{
CursorArenal();
register int m,n;
float v;
float K;
int color;
FILE *fp;
```

```
float max=0;
            //CALCULA EL MAXIMO NIVEL
if(fp=fopen("ESPECTRO.WLP","rb"))    {
for(m=0;m<AltoImagen;m++){
for(n=0;n<AnchoImagen;n++){
fread(&v,sizeof(float),1,fp);
v=fabs(v);
if(v>max)max=v;
                    }
                    }
fclose(fp);                    }

//          GRAFICACION DE LA IMAGEN
max=log(1+max);
K=(float)255/max;

if(fp=fopen("ESPECTRO.WLP","rb"))    {
for(m=0;m<AltoImagen;m++){
for(n=0;n<AnchoImagen;n++){
fread(&v,sizeof(float),1,fp);
v=log(1+fabs(v));
v*=K;
color=(int)v;
SetPixel(hDC,n,m,RGB(color,color,color));
                    }
                    }
fclose(fp);                    }
CursorFlecha1();
}


void TGraficarEspectro::AdjustScroller()
{
  RECT ClientRect;

  GetClientRect(HWindow, &ClientRect);
  Scroller->SetRange(AnchoImagen - (ClientRect.right - ClientRect.left),
      AltoImagen - (ClientRect.bottom - ClientRect.top));
  Scroller->ScrollTo(0, 0);
  InvalidateRect(HWindow, NULL, TRUE);
}


void TGraficarEspectro::WMSize(TMessage& Msg)
{
  TWindow::WMSize(Msg);
  if ( (Msg.WParam != SIZEICONIC) )
    AdjustScroller();
}

      //Función respuesta al click botón derecho del ratón
void TGraficarEspectro::WMRButtonDown(RTMessage)
{
char texto[70];
sprintf(texto,"Filtro: %s",Filtro);
MessageBox(HWindow,texto,"ESPECTRO:                                    FILTRO
APLICADO",MB_OK|MB_ICONINFORMATION);
}


      //Función respuesta al doble-click botón izq.
void TGraficarEspectro::WMLButtonDblClk(RTMessage)
{
HWND HWindow;
Copiar(HWindow);
}
```

ESPECORG.H

```
#include <owl.h>
#include <stdio.h>
#include <math.h>

extern WORD AnchoImagen,AltoImagen;
extern void huge Copiar(HWND HWindow);
extern void huge CursorArenal();
extern void huge CursorFlechal();

_CLASSDEF(TGraficarEspecorg)
class TGraficarEspecorg : public TWindow
{
public:
      TGraficarEspecorg(PTWindowsObject AParent, LPSTR ATitle);
      virtual LPSTR GetClassName();
      virtual void GetWindowClass(WNDCLASS&);
      virtual void Paint(HDC PaintDC, PAINTSTRUCT& );
      virtual void WMSize(TMessage&) = [WM_FIRST + WM_SIZE];
      void AdjustScroller();
      virtual                void                WMLButtonDblClk(RTMessage
Msg)=[WM_FIRST+WM_LBUTTONDBLCLK];

};
```

ESPECORG.CPP

```
#include "especorg.h"

TGraficarEspecorg::TGraficarEspecorg(PTWindowsObject       AParent,LPSTR
ATitle) : TWindow(AParent, ATitle)
{
  Attr.Style |= WS_VSCROLL | WS_HSCROLL;
  Attr.W=(int)AnchoImagen;     //dimensiones de la ventana (anchura)
  Attr.H=(int)AltoImagen;      //altura
  Scroller   new TScroller(this, 1,1, 200,200);
}

LPSTR TGraficarEspecorg::GetClassName()
{
  return "ESPECORG";
}


void TGraficarEspecorg::GetWindowClass(WNDCLASS& WndClass)
{
  TWindow::GetWindowClass(WndClass);
  WndClass.style =CS_DBLCLKS; //permite reconocer el doble click.
   WndClass.hIcon  =LoadIcon(GetApplication()->hInstance,"ESPECTRO");  /*
Client area will be painted by the app. */
}

void TGraficarEspecorg::Paint(HDC hDC, PAINTSTRUCT&)
{
CursorArenal();
register int m,n;
float v;
float K;
int color;
FILE *fp;
float max=0;
```

```
            //CALCULA EL MAXIMO NIVEL
if(fp=fopen("ESPECORG.WLP","rb"))    {
for(m=0;m<AltoImagen;m++){
for(n=0;n<AnchoImagen;n++){
fread(&v,sizeof(float),1,fp);
v=fabs(v);
if(v>max)max=v;
                    }
                    }
fclose(fp);                    }

//          GRAFICACION DE LA IMAGEN
max=log(1+max);
K=(float)255/max;

if(fp=fopen("ESPECORG.WLP","rb"))    {
for(m=0;m<AltoImagen;m++){
for(n=0;n<AnchoImagen;n++){
fread(&v,sizeof(float),1,fp);
v=log(1+fabs(v));
v*=K;
color=(int)v;
SetPixel(hDC,n,m,RGB(color,color,color));
                    }
                    }
fclose(fp);                    }
CursorFlechal();
}


void TGraficarEspecorg::AdjustScroller()
{
  RECT ClientRect;

  GetClientRect(HWindow, &ClientRect);
  Scroller->SetRange(AnchoImagen - (ClientRect.right - ClientRect.left),
      AltoImagen - (ClientRect.bottom - ClientRect.top));
  Scroller->ScrollTo(0, 0);
  InvalidateRect(HWindow, NULL, TRUE);
}

void TGraficarEspecorg::WMSize(TMessage& Msg)
{
  TWindow::WMSize(Msg);
  if ( (Msg.WParam != SIZEICONIC) )
    AdjustScroller();
}

      //Función respuesta al doble-click botón izq.
void TGraficarEspecorg::WMLButtonDblClk(RTMessage)
{
HWND HWindow;
Copiar(HWindow);
}
```

ESPECRUI.H

```
//    Washington Lenin Paredes Cruz

#include <owl.h>
#include <stdio.h>
#include <math.h>

extern WORD AnchoImagen,AltoImagen;
extern void huge Copiar(HWND HWindow);
```

```
extern void huge CursorArenal();
extern void huge CursorFlechal();

_CLASSDEF(TGraficarEspecrui)
class TGraficarEspecrui : public TWindow
{
public:
      TGraficarEspecrui(PTWindowsObject AParent, LPSTR ATitle);
      virtual LPSTR GetClassName();
      virtual void GetWindowClass(WNDCLASS&);
      virtual void Paint(HDC PaintDC, PAINTSTRUCT& );
      virtual void WMSize(TMessage&) = [WM_FIRST + WM_SIZE];
      void AdjustScroller();
      virtual                void                WMLButtonDblClk(RTMessage
Msg)=[WM_FIRST+WM_LBUTTONDBLCLK];

};
```

ESPECRUI.CPP

```
#include "especrui.h"

TGraficarEspecrui::TGraficarEspecrui(PTWindowsObject       AParent,LPSTR
ATitle) : TWindow(AParent, ATitle)
{
  Attr.Style |= WS_VSCROLL | WS_HSCROLL;
  Attr.W=(int)AnchoImagen;     //dimensiones de la ventana (anchura)
  Attr.H=(int)AltoImagen;      //altura
  Scroller = new TScroller(this, 1,1, 200,200);
}

LPSTR TGraficarEspecrui::GetClassName()
{
  return "ESPECRUI";
}


void TGraficarEspecrui::GetWindowClass(WNDCLASS& WndClass)
{
  TWindow::GetWindowClass(WndClass);
  WndClass.style =CS_DBLCLKS; //permite reconocer el doble click.
   WndClass.hIcon  =LoadIcon(GetApplication()->hInstance,"ESPECTRO");  /*
Client area will be painted by the app. */
}

void TGraficarEspecrui::Paint(HDC hDC, PAINTSTRUCT&)
{
CursorArenal();
register int m,n;
float v;
float K;
int color;
FILE *fp;
float max=0;
            //CALCULA EL MAXIMO NIVEL
if(fp=fopen("ESPECRUI.WLP","rb"))    {
for(m=0;m<AltoImagen;m++){
for(n=0;n<AnchoImagen;n++){
fread(&v,sizeof(float),1,fp);
v=fabs(v);
if(v>max)max=v;
                  }
                  }
fclose(fp);                    }
```

```
//          GRAFICACION DE LA IMAGEN
max=log(1+max);
K=(float)255/max;

if(fp=fopen("ESPECRUI.WLP","rb"))    {
for(m=0;m<AltoImagen;m++){
for(n=0;n<AnchoImagen;n++){
fread(&v,sizeof(float),1,fp);
v=log(1+fabs(v));
v*=K;
color=(int)v;
SetPixel(hDC,n,m,RGB(color,color,color));
                    }}
fclose(fp);                    }
CursorFlechal();
}


void TGraficarEspecrui::AdjustScroller()
{
  RECT ClientRect;
  GetClientRect(HWindow, &ClientRect);
  Scroller->SetRange(AnchoImagen - (ClientRect.right - ClientRect.left),
      AltoImagen - (ClientRect.bottom - ClientRect.top));
  Scroller->ScrollTo(0, 0);
  InvalidateRect(HWindow, NULL, TRUE);
}


void TGraficarEspecrui::WMSize(TMessage& Msg)
{
  TWindow::WMSize(Msg);
  if ( (Msg.WParam != SIZEICONIC) )
    AdjustScroller();
}

      //Función respuesta al doble-click botón izq.
void TGraficarEspecrui::WMLButtonDblClk(RTMessage)
{
HWND HWindow;
Copiar(HWindow);
}
```

IMGMEJOR.H

```
//GRAFICACION DE LA IMAGEN MEJORADA

#include <owl.h>
#include <stdio.h>
#include <math.h>

extern WORD AltoImagen,AnchoImagen;
extern BOOL HayRuido;
extern void huge Copiar(HWND HWindow);
extern void huge CursorArenal();
extern void huge CursorFlechal();

_CLASSDEF(TGrafImgMejor)
class TGrafImgMejor : public TWindow
{
public:
      TGrafImgMejor(PTWindowsObject AParent, LPSTR ATitle);
      virtual void Paint(HDC PaintDC, PAINTSTRUCT& );
      virtual void GetWindowClass(WNDCLASS& WndClass);
      virtual LPSTR GetClassName();
```

```
      virtual void WMSize(TMessage&) = [WM_FIRST + WM_SIZE];
      void AdjustScroller();
      virtual                    void                    WMRButtonDown(TMessage&
Msg)=[WM_FIRST+WM_RBUTTONDOWN];
      virtual                  void                  WMLButtonDblClk(RTMessage
Msg)=[WM_FIRST+WM_LBUTTONDBLCLK];

};
```

IMGMEJOR.CPP

```
//GRAFICACION DE LA IMAGEN MEJORADA

#include "imgmejor.h"

TGrafImgMejor::TGrafImgMejor(PTWindowsObject  AParent,LPSTR  ATitle)  :
TWindow(AParent, ATitle)
{  Attr.Style |= WS_VSCROLL | WS_HSCROLL;
  Attr.W=(int)AnchoImagen;     //dimensiones de la ventana (anchura)
  Attr.H=(int)AltoImagen;      //altura
  Scroller = new TScroller(this, 1,1, 200,200);}

LPSTR TGrafImgMejor::GetClassName()
{  return "IMAGEN MEJORADA";}

void TGrafImgMejor::GetWindowClass(WNDCLASS& WndClass)
{
  TWindow::GetWindowClass(WndClass);
  WndClass.style =CS_DBLCLKS; //permite reconocer el doble click.
  WndClass.hIcon =LoadIcon(GetApplication()->hInstance,"ESPACIOM");
}

void TGrafImgMejor::Paint(HDC hDC, PAINTSTRUCT&)
{
CursorArenal();
       //GRAFICACION DE LA IMAGEN MEJORADA\\
register int m,n;
int color;
FILE *fp;
float u;
if(fp=fopen("IMGMEJOR.WLP","rb"))    {
for(m=0;m<AltoImagen;m++)      {
for(n=0;n<AnchoImagen;n++)     {
fread(&u,sizeof(float),1,fp);
color=(int)u;
SetPixel(hDC,n,m,RGB(color,color,color));
                               }}
fclose(fp);                          }
CursorFlechal();
}

void TGrafImgMejor::AdjustScroller()
{
  RECT ClientRect;

  GetClientRect(HWindow, &ClientRect);
  Scroller->SetRange(AnchoImagen - (ClientRect.right - ClientRect.left),
      AltoImagen - (ClientRect.bottom - ClientRect.top));
  Scroller->ScrollTo(0, 0);
  InvalidateRect(HWindow, NULL, TRUE);
}
```

```
void TGrafImgMejor::WMSize(TMessage& Msg)
{
  TWindow::WMSize(Msg);
  if ( (Msg.WParam != SIZEICONIC) )
    AdjustScroller();
}
      //Función respuesta al click botón derecho del ratón
void TGrafImgMejor::WMRButtonDown(RTMessage)
{
if(HayRuido)
{     CursorArenal();
      register int m,n;
      float Pixel_o,Pixel_m;
      char archivo[]="IMGORG.WLP";
      FILE *fo;
      FILE *fm;
      double SNR_db,e_rms=0;
      fo=fopen(archivo,"rb");
      fm=fopen("IMGMEJOR.WLP","rb");

      for(m=0;m<AltoImagen;m++){
      for(n=0;n<AnchoImagen;n++){
      fread(&Pixel_o,sizeof(float),1,fo);
      fread(&Pixel_m,sizeof(float),1,fm);
      e_rms+=pow(Pixel_o-Pixel_m,2);
                        }
                        }
      fclose(fo);
      fclose(fm);
      e_rms=sqrt(e_rms);
      e_rms/=(double)sqrt((float)AltoImagen*AnchoImagen);
      if(e_rms<0.001)e_rms=0.001;
      SNR_db=20*log10((double)255/e_rms);
      char texto[20];
      sprintf(texto,"SNR = %6.2f dB",SNR_db);
      CursorFlechal();
      MessageBox(HWindow,texto,"IMAGEN      FILTRADA",      MB_OK      |
MB_ICONINFORMATION);
      }
else   MessageBox(HWindow,"Solamente se calcula la relación señal a
ruido, cuando algún tipo de ruido ha sido generado.","IMAGEN FILTRADA:
RELACION SEÑAL A RUIDO",MB_OK|MB_ICONSTOP);
}

      //Función respuesta al doble-click botón izq.
void TGrafImgMejor::WMLButtonDblClk(RTMessage)
{
HWND HWindow;
Copiar(HWindow);
}
```

Partiendo de este código, se utilizan las siguientes opciones de compilación para obtener el archivo MIMTDC.EXE:

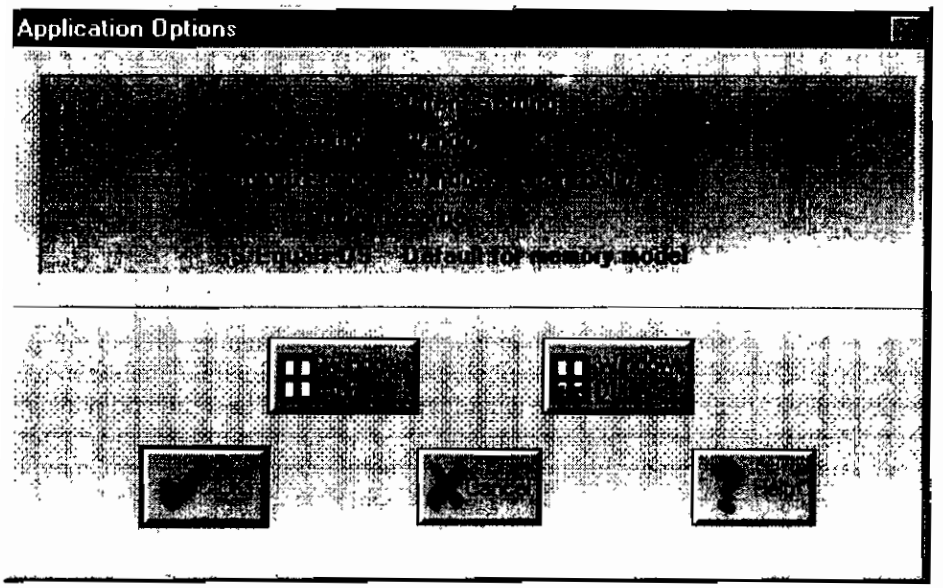

Figura No. 2 Caja de diálogo de Options/Application...

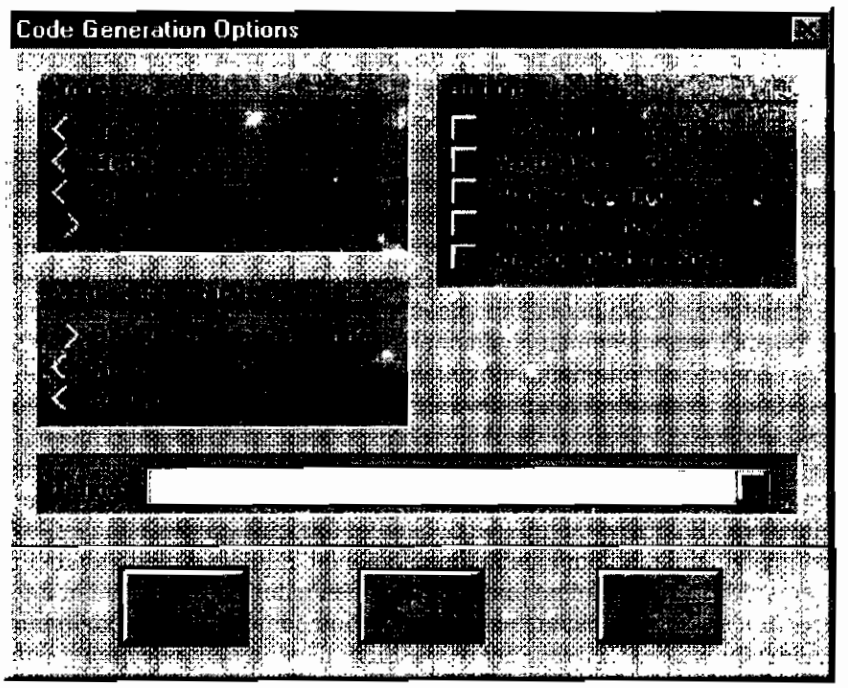

Figura No. 3 Caja de diálogo de Options/Compiler/Code Generation...

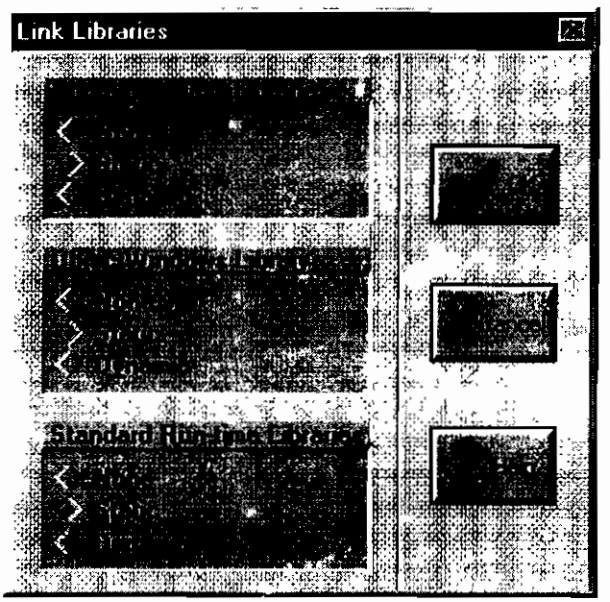

Figura No. 4 Caja de diálogo de Options/Linker/Libraries...

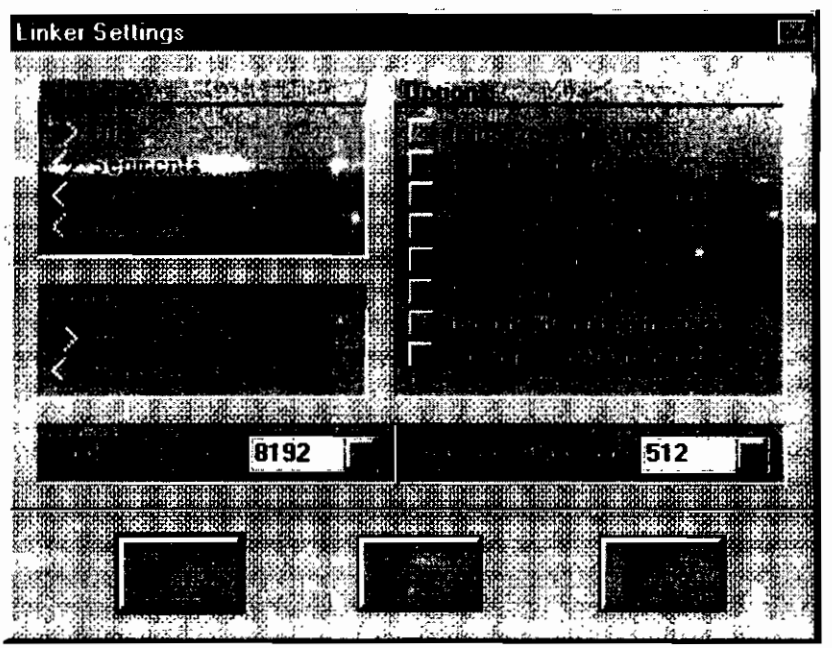

Figura No. 5 Caja de diálogo de Options/Linker/Settings...

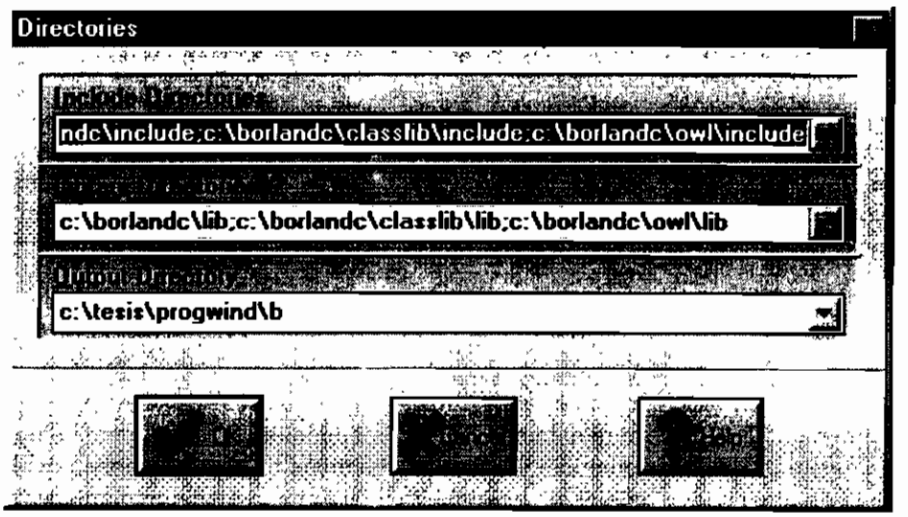

Figura No. 6 Caja de diálogo de Options/Directories...

MÓDULO TDCOSENO

Este módulo está generado por los archivos indicados en la ventana desplegada en la figura No. 7, cada uno de los cuales se registran a continuación.

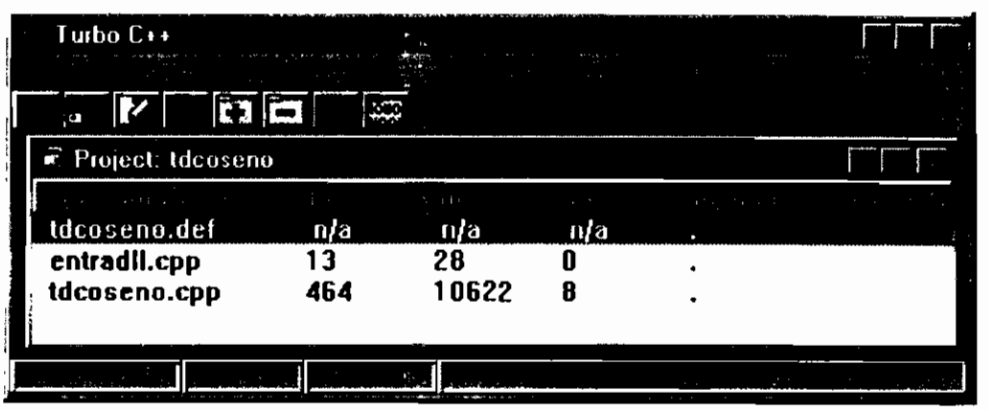

Figura No. 7 Listado de archivos que conforman el proyecto TDCOSENO.

TDCOSENO.DEF

```
LIBRARY TDCOSENO
DESCRIPTION 'DLL Tcoseno'
```

```
EXETYPE     WINDOWS
CODE PRELOAD MOVEABLE DISCARDABLE
DATA PRELOAD MOVEABLE SINGLE
HEAPSIZE 1024
```

ENTRADLL.CPP

```
#include "windows.h"

int FAR PASCAL LibMain(HANDLE hInstance,WORD, WORD,LPSTR)
{
return 1;
}

int FAR PASCAL WEP(int)
{
return 1;
}
```

TDCOSENO.H

```
//TDC DIRECTA E INVERSA

#ifndef __TDCOSENO_H
#define __TDCOSENO_H

#include <windows.h>
#include <string.h>
#include <stdio.h>
#include <math.h>

const float PI= 3.141592654;
const int tm=10;  //tamaño inicial del factor de dimensión

void FAR _export CosDir(int,int,float huge *,float huge *);
void FAR _export CosInv(int,int,float huge *,float huge *);

class TCoseno
{
int fd;           //factor de dimensión
int M;            //tamaño vertical en pixeles
int N;            //tamaño horizontal en pixeles
float huge *W1[tm];
float huge *W2[tm];
float huge *v[tm+1];
float huge *u[tm+1];
HANDLE hW1[tm];
HANDLE hW2[tm];
HANDLE hv[tm];
HANDLE hu[tm];

public:
      TCoseno(int M1,int N1,float huge *u1,float huge *v1);
      //constructor
      ~TCoseno(){liberar_memoria(fd);};   //destructor
      void clcl_TDC_D();
      void clcl_TDC_I();
      int menorMN(int M1,int N1);
      int facdim(int menor);
```

```
            //funciones de TDC directa
      void clcl_transf_dir(int,int, float huge *v, float huge *u);
      void clcl_g_hat(int,int,int, float huge *u, float huge *u2);
      void clcl_V(int,int, int, float huge *v2, float huge *v);
      void TDC_D(int,int,int);

            //funciones comunes
      float e_k(int);
      float coseno(int,int,int);
      void clcl_g(int,int,int, float huge *u,float huge *W1, float huge
*W2);
      void clcl_W1_W2(int,int,float huge *W1, float huge *W2);
      void inicializa_UV(int,int,int);
      void inicializa_W1_W2(int,int,int);
      long int memoria_requerida(int,int,int);
      void liberar_memoria(int);
      void normaliza_V(int,int,float huge *v1);

            //funciones para TDC inversa
      void clcl_G(int,int,int, float huge *v, float huge *v2);
      void clcl_transf_inv(int,int, float huge *u, float huge *v);
      void ordena_pqrs(int,int, int, float huge *u2, float huge *u);
      void TDC_I(int,int,int);
};
#endif
```

TDCOSENO.CPP

```
#include "tdcoseno.h"

void FAR _export CosDir(int M,int N, float huge *u,float huge *v)
{if(M>1&&N>1)     {     //el algoritmo es aplicable a matrices y no a
vectores
      TCoseno *a;
      a=new TCoseno(M,N,u,v);
      a->clcl_TDC_D();
      delete a;
            }
}
voi  FAR _export CosInv(int M,int N, float huge *u,float huge *v)
{if(M>1&&N>1)     {
      TCoseno *a;
      a=new TCoseno(M,N,u,v);
      a->clcl_TDC_I();
      delete a;
            }
}

//----------------------- funciones TCoseno---------------------

TCoseno::TCoseno(int M1,int N1,float huge *u1,float huge *v1)
{
      M=M1;N=N1;
      fd=facdim(menorMN(M,N));
      v[fd]=v1;
      u[fd]=u1;
      inicializa_UV(M/2,N/2,fd);     //solo se inicializan punteros desde
las primeras submatrices
      inicializa_W1_W2(M/2,N/2,fd); //W1 y W2 existen desde los tamaños
M/2, N/2
}
```

```
int TCoseno::menorMN(int M,int N)
{
      if(M>=N)return N;else return M;
}

int TCoseno::facdim(int M)
{
      int fd=0;
      do{   M=M/2;
            fd++;
        }while(M!=1);    //se divide para 2 hasta obtener cociente=1
        return fd;
}

void TCoseno::clcl_TDC_D()
{
      TDC_D(M,N,fd);
      normaliza_V(M,N,v[fd]);//normaliza la matriz en dominio frecuencia
}

void TCoseno::clcl_TDC_I()
{
      normaliza_V(M,N,v[fd]);
      TDC_I(M,N,fd);
}

long int TCoseno::memoria_requerida(int M,int N,int fd)
{
long int memoria=0;
int a;
for(a=fd;a>=0;a--){
      memoria+=(long int)M*N*2;
      M/=2;
      N/=2;
                  }
memoria*=sizeof(float);
return memoria;
}

void TCoseno::inicializa_UV(int M,int N, int fd)
{
int a;
for(a=fd-1;a>=0;a--)    {
      hv[a]=GlobalAlloc(GMEM_FIXED|GMEM_ZEROINIT,(long
int)M*N*sizeof(float));
      hu[a]=GlobalAlloc(GMEM_FIXED|GMEM_ZEROINIT,(long
int)M*N*sizeof(float));
      v[a]=(float huge *)GlobalLock(hv[a]);
      u[a]=(float huge *)GlobalLock(hu[a]);
 M=M/2;
 N=N/2;
                  }
}

void TCoseno::liberar_memoria(int fd)
{
int a;
for(a=fd-1;a>=0;a--)    {
      GlobalUnlock(hv[a]);
      GlobalFree(hv[a]);
      GlobalUnlock(hu[a]);
      GlobalFree(hu[a]);
                        }
```

```
for(a=fd-1;a>=0;a--)     {
      GlobalUnlock(hW1[a]);
      GlobalFree(hW1[a]);
      GlobalUnlock(hW2[a]);
      GlobalFree(hW2[a]);
                  }
}


void TCoseno::inicializa_W1_W2(int M,int N, int fd)
{int a;
for(a=fd-1;a>=0;a--)     {
      hW1[a]=GlobalAlloc(GMEM_FIXED|GMEM_ZEROINIT,M*sizeof(float));
      hW2[a]=GlobalAlloc(GMEM_FIXED|GMEM_ZEROINIT,N*sizeof(float));
      W1[a]=(float huge *)GlobalLock(hW1[a]);
      W2[a]=(float huge *)GlobalLock(hW2[a]);
 clcl_W1_W2(M,N,W1[a],W2[a]);
 M=M/2;
 N=N/2;
                  }
}


void TCoseno::TDC_D(int M,int N,int fd)    {
int a;
 for(a=1;a<5;a++) {
     clcl_g_hat(M/2,N/2,a,u[fd-1],u[fd]);
     clcl_g(M/2,N/2,a,u[fd-1],W1[fd-1],W2[fd-1]);
     fd--;
     if(fd==0)clcl_transf_dir(M/2,N/2,v[0],u[0]);      //calcula   TDC-D,
usando la f¢rmula no normalizada, entrega P,Q1,R1,S1.
     else TDC_D(M/2,N/2,fd);
     fd++;
     clcl_V(M/2,N/2,a,v[fd],v[fd-1]);           //usando      P,Q1,R1,S1
reconstruye V4 sin normalizar.
                    }
                          }

//    cálculo de matriz W1 y W2 .
/*    W1  y  W2  son  matrices  diagonales,  entonces  se  les  a  tomado
solamente
      como vectores, haciendo W1 y W2 vectores  */
void TCoseno::clcl_W1_W2(int M, int N, float huge *W1, float huge *W2)
{     register int m;
      int M4=4*M;
      int N4=4*N;
            //calcula W1\\
      for(m=0;m<M;m++)
      *(W1++)=(float)1/(2*coseno(m,1,M4));
            //calcula W2\\
      for(m=0;m<N;m++)
      *(W2++)=(float)1/(2*coseno(m,1,N4));
}


void TCoseno::clcl_g_hat(int M,int N,int num_mtrz, float huge *u, float
huge *u2)
{     register int m,n;
      int M2=2*M;
      int N2=2*N;
      float huge *p;float huge *q;float huge *r; float huge *s;

      switch(num_mtrz){
      case 1:           //para primera submatr¡z calcula g^=p+q+r+s
      for(m=0;m<M;m++){
      p=(u2+(long int)N2*m);
```

```
        q=(u2+(long int)N2*m+N2-1);
        r=(u2+(long int)N2*(M2-1-m));
        s=(u2+(long int)N2*(M2-1-m)+N2-1);
        for(n=0;n<N;n++)
        *(u++)= *(p++) + *(q--) + *(r++) + *(s--);
                    }
              break;

        case 2:     //para segunda submatr;z calcula g^=p-q+r-s
        for(m=0;m<M;m++){
        p=(u2+(long int)N2*m);
        q=(u2+(long int)N2*m+N2-1);
        r=(u2+(long int)N2*(M2-1-m));
        s=(u2+(long int)N2*(M2-1-m)+N2-1);
        for(n=0;n<N;n++)
        *(u++)= *(p++) - *(q--) + *(r++) - *(s--);
                    }
              break;

        case 3:     //para tercera submatr;z calcula g^=p+q-r-s
        for(m=0;m<M;m++){
        p=(u2+(long int)N2*m);
        q=(u2+(long int)N2*m+N2-1);
        r=(u2+(long int)N2*(M2-1-m));
        s=(u2+(long int)N2*(M2-1-m)+N2-1);
        for(n=0;n<N;n++)
        *(u++)=*(p++) + *(q--) - *(r++) - *(s--);
                    }
              break;

        case 4:     //para cuarta submatr;z calcula g^=p-q-r+s
        for(m=0;m<M;m++){
        p=(u2+(long int)N2*m);
        q=(u2+(long int)N2*m+N2-1);
        r=(u2+(long int)N2*(M2-1-m));
        s=(u2+(long int)N2*(M2-1-m)+N2-1);
        for(n=0;n<N;n++)
        *(u++)= *(p++) - *(q--) - *(r++) + *(s--);
                    }     }
}


void TCoseno::clcl_g(int M,int N,int num_mtrz, float huge *u, float huge
*W1, float huge *W2)     //
{     register int m,n;
      float huge *p;
      switch(num_mtrz){
            //case 1:         //para primera submatr;z hace g=g^
            //break;

            case 2:     //para segunda submatr;z hace g=g^W2
            for(m=0;m<M;m++){
            p=W2;
            for(n=0;n<N;n++)
            *(u++)*=*(p++);}   //*(W2+n)
            break;

            case 3:     //para tercera submatr;z hace g=W1g^
            for(m=0;m<M;m++){
            for(n=0;n<N;n++)
            *(u++)*=*W1;       //*(W1+m)
            W1++;              }
            break;
            case 4:            //para cuarta submatr;z hace g=W1g^W2
            for(m=0;m<M;m++){
            p=W2;
```

```
            for(n=0;n<N;n++)
                  *(u++)*=*W1**(p++);      //*(W1+m)**(W2+n);
            W1++;
                        }
                  }
      }


/*calcula la transformada directa coseno, usando la f¢rmula no
normalizada
      calculando una submatr¡z en la frecuencia en cada paso*/
void TCoseno::clcl_transf_dir(int M,int N,float huge *v, float huge *u)
{
register int m,n,k,l;
int M2=2*M;
int N2=2*N;
float huge *p;
      // la transformada de cada submatr¡z.
      for(k=0;k<M;++k)
      for(l=0;l<N;++l){
      *v=0;
      p=u;
      for(m=0;m<M;++m){
      for(n=0;n<N;++n)
            *v+=*(p++)*coseno(m,k,M2)*coseno(n,l,N2);
                  }
      v++;
                  }
}


float TCoseno::e_k(int k)     //calcula el coeficiente de normalizaci¢n.
{
      if(k==0)return (float)1/sqrt(2.0);
      return 1.0;
}

float TCoseno::coseno(int m,int k,int M)  //calcula la funci¢n coseno,
dando la coord espacial,
{     return cos((2*m+1)*PI*k/M);      //coord frecuencia espacial y el
orden de la matr¡z.
}


void TCoseno::clcl_V(int M, int N, int num_mtrz, float  huge *v2, float
huge *v)    //reconstruye V4 partiendo de P, Q1,R1 y S1.
{     register int k,l;                  //por desplazamiento y suma de
las transformadas parciales.
      int N2=2*N;
      switch(num_mtrz){

      case 1:
      for(k=0;k<M;++k)
      for(l=0;l<N;++l)
      *(v2+(long int)N2*2*k+2*l)=*(v++);  //calcula componentes pares-
pares.
      break;
            //componentes pares-impares
      case 2:
      for(k=0;k<M;++k)
      for(l=0;l<N;++l){
      if(l==(N-1)){     *(v2+(long           int)N2*2*k+2*l+1)=*(v+(long
int)N*k+l);}
      else  *(v2+(long  int)N2*2*k+2*l+1)=*(v+(long  int)N*k+l)+*(v+(long
int)N*k+l+1);     }
      break;
```

```
            //componentes impares-pares
      case 3:
      for(k=0;k<M;++k)
      for(l=0;l<N;++l){
      if(k==(M-1)){      *(v2+(long          int)N2*(2*k+1)+2*l)=*(v+(long
int)N*k+l);}
      else  *(v2+(long                       int)N2*(2*k+1)+2*l)=*(v+(long
int)N*k+l)+*(v+(long int)N*(k+1)+l);
                  }
      break;
            //componentes impares-impares
      case 4:
      for(k=0;k<M;++k)
      for(l=0;l<N;++l){
      if((l==(N-1))&&(k!=(M-1))){    *(v2+(long
int)N2*(2*k+1)+2*l+1)=*(v+(long int)N*k+l)+*(v+(long int)N*(k+1)+l);}
      else          if((l!=(N-1))&&(k==(M-1),){                    *(v2+(long
int)N2*(2*k+1)+2*l+1)=*(v+(long int)N*k+l)+*(v+(long int)N*k+l+1);}
      else if((l==(N-1))&&(k==(M-1))){     *(v2+(long
int)N2*(2*k+1)+2*l+1)=*(v+(long int)N*k+l);}
      else  *(v2+(long                          int)N2*(2*k+1)+2*l+1)=*(v+(long
int)N*k+l)+*(v+(long   int)N*k+l+1)+*(v+(long   int)N*(k+1)+l)+*(v+(long
int)N*(k+1)+l+1);
                  }
                  }
}


void TCoseno::normaliza_V(int M,int N,float huge *v)
{     register int k,l;
      float f;
      int raizM=sqrt(M);
      int raizN=sqrt(N);

      f=(float)2/raizM;
      f*=(float)1/raizN;
      for(k=0;k<M;k++){
      for(l=0;l<N;l++){
      *(v++)*=f*e_k(k)*e_k(l);//normaliza .
                  }
                  }
}


                  //TDC INVERSA

void TCoseno::TDC_I(int M,int N,int fd)    {
int a;
 for(a=1;a<5;a++) {
     clcl_G(M/2,N/2,a,v[fd-1],v[fd]);
     fd--;
     if(fd==0)clcl_transf_inv(M/2,N/2,u[0],v[0]);
     else TDC_I(M/2,N/2,fd);
     fd++;
     clcl_g(M/2,N/2,a,u[fd-1],W1[fd-1],W2[fd-1]);
     ordena_pqrs(M/2,N/2,a,u[fd],u[fd-1]);
                  }
                        }



void TCoseno::clcl_G(int M,int N,int num_mtrz, float huge *v, float huge
*v2)
{     register int k,l;
      int N2=2*N;
```

```
      switch(num_mtrz){
            case 1:                    //componentes pares-pares
            for(k=0;k<M;++k)
            for(l=0;l<N;++l)
            *(v++)=*(v2+(long int)N2*2*k+2*l);
            break;

            case 2:              //componentes pares-impares
            for(k=0;k<M;++k)
            for(l=0;l<N;++l){
      if(l==0){    *(v++)=*(v2+(long int)N2*2*k+2*l+1);}
      else  *(v++)=*(v2+(long                int)N2*2*k+2*l+1)+*(v2+.long
int)N2*2*k+2*l-1);
                        }
            break;

            case 3:              //componentes impares-pares
            for(k=0;k<M;++k)
            for(l=0;l<N;++l){
      if(k==0){    *(v++)=*(v2+(long int)N2*(2*k+l)+2*l);}
      else  *(v++)=*(v2+(long              int)N2*(2*k+1)+2*l)+*(v2+(long
int)N2*(2*k-1)+2*l);
                        }
            break;

            case 4:                    //componentes impares-impares
            for(k=0;k<M;++k)
            for(l=0;l<N;++l){
      if((l==0)&&(k!=0)){      *(v++)=*(v2+(long
int)N2*(2*k+l)+2*l+1)+*(v2+(long int)N2*(2*k-1)+2*l+1);}
      else          if((l!=0)&&(k==0)){                    *(v++)=*(v2+(long
int)N2*(2*k+1)+2*l+1)+*(v2+(long int)N2*(2*k+1)+2*l-1);}

      else if((l==0)&&(k==0)){*(v++)=*(v2+(long int)N2*(2*k+1)+2*l+1);}
      else  *(v++)=*(v2+(long            int)N2*(2*k+1)+2*l+1)+*(v2+(long
int)N2*(2*k+1)+2*l-1)+*(v2+(long         int)N2*(2*k-1)+2*l+1)+*(v2+(long
int)N2*(2*k-1)+2*l-1);
                        }
            }
            }

//funci¢n que calcula la transformada inversa coseno usando la f¢rmula.
void TCoseno::clcl_transf_inv(int M,int N, float huge *u, float huge *v)
{
register int m,n,k,l;
int M2=2*M;
int N2=2*N;
float huge *p;
for(m=0;m<M;++m)
for(n=0;n<N;++n){
*u=0;
p=v;
for(k=0;k<M;++k){
for(l=0;l<N;++l)
*u+=*(p++)*coseno(m,k,M2)*coseno(n,l,N2);
            }
u++;
            }
}


void TCoseno::ordena_pqrs(int M,int N, int num_mtrz, float huge *u2,
float huge *u)
{register int n,m;
      int M2=2*M;
      int N2=2*N;
```

```
        float huge *p;
        float huge *q;
        float huge *r;
        float huge *s;

        switch(num_mtrz){
        case 1:
        for(m=0;m<M;m++){
        p=(u2+(long int)N2*m);
        q=(u2+(long int)N2*m+N2-1);
        r=(u2+(long int)N2*(M2-1-m));
        s=(u2+(long int)N2*(M2-1-m)+N2-1);
        for(n=0;n<N;n++){
        *(p++)=*u;
        *(q--)=*u;
        *(r++)=*u;
        *(s--)=*(u++);
                    }
                    }
        break;
        case 2:
        for(m=0;m<M;m++){
        p=(u2+(long int)N2*m);
        q=(u2+(long int)N2*m+N2-1);
        r=(u2+(long int)N2*(M2-1-m));
        s=(u2+(long int)N2*(M2-1-m)+N2-1);
        for(n=0;n<N;n++){
        *(p++)+=*u;
        *(q--)-=*u;
        *(r++)+=*u;
        *(s--)-=*(u++);
                    }
                    }
        break;
        case 3:
        for(m=0;m<M;m++){
        p=(u2+(long int)N2*m);
        q=(u2+(long int)N2*m+N2-1);
        r=(u2+(long int)N2*(M2-1-m));
        s=(u2+(long int)N2*(M2-1-m)+N2-1);
        for(n=0;n<N;n++){
        *(p++)+=*u;
        *(q--)+=*u;
        *(r++)-=*u;
        *(s--)-=*(u++);
                    }       }
        break;
        case 4:
        for(m=0;m<M;m++){
        p=(u2+(long int)N2*m);
        q=(u2+(long int)N2*m+N2-1);
        r=(u2+(long int)N2*(M2-1-m));
        s=(u2+(long int)N2*(M2-1-m)+N2-1);
        for(n=0;n<N;n++){
        *(p++)+=*u;
        *(q--)-=*u;
        *(r++)-=*u;
        *(s--)+=*(u++);
                    }       }
                    }
}
```

Una vez conformado el proyecto de la aplicación DLL el siguiente paso es compilarlo, para lo cual se utilizan las siguientes opciones de compilación:

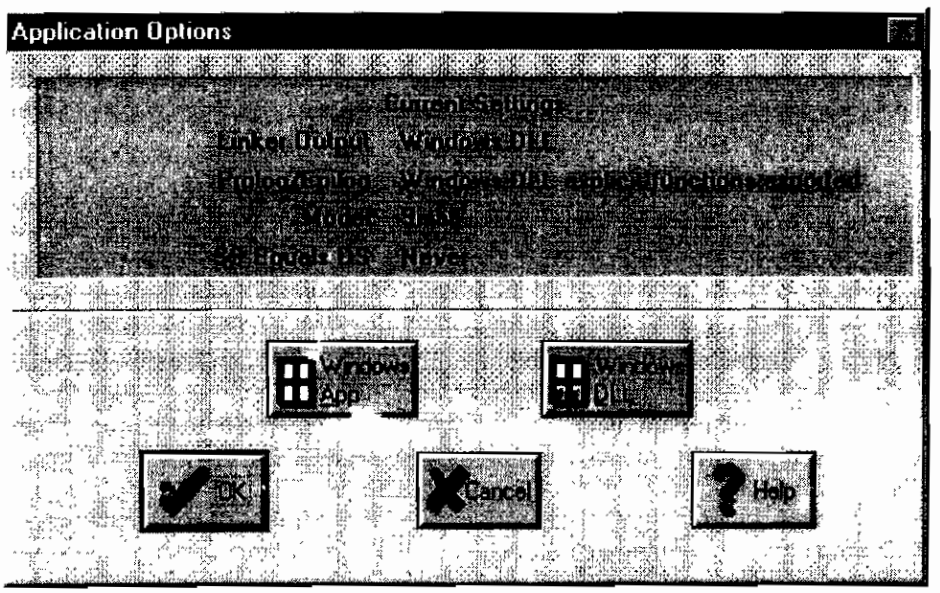

Figura No. 8 Caja de diálogo de Options/Application...

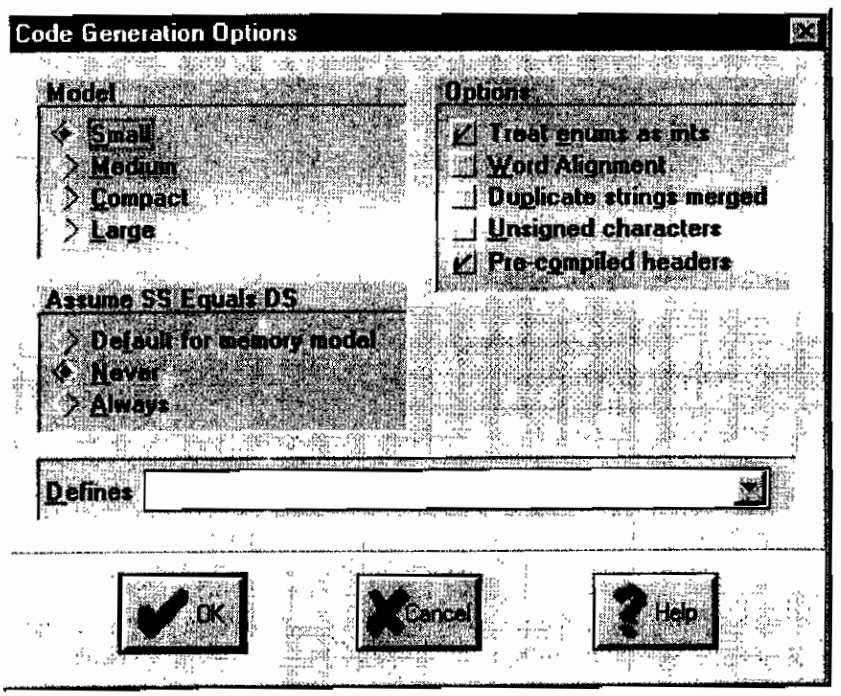

Figura No. 9 Caja de diálogo de Options/Compiler/Code Generation...

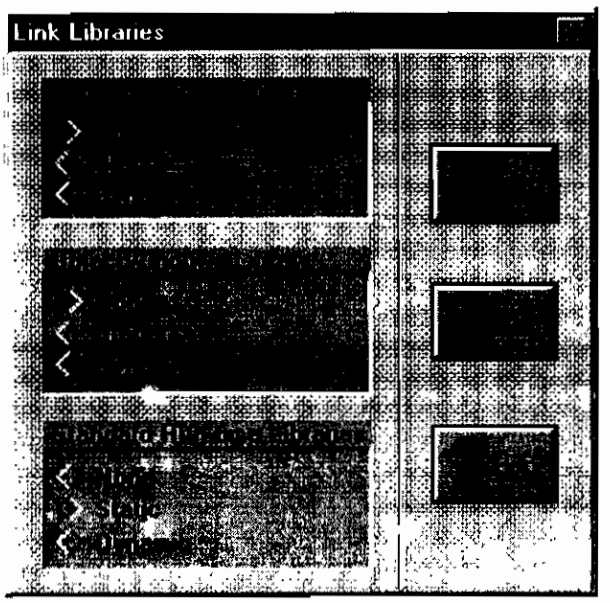

Figura No. 10 Caja de diálogo de Options/Linker/Libraries...

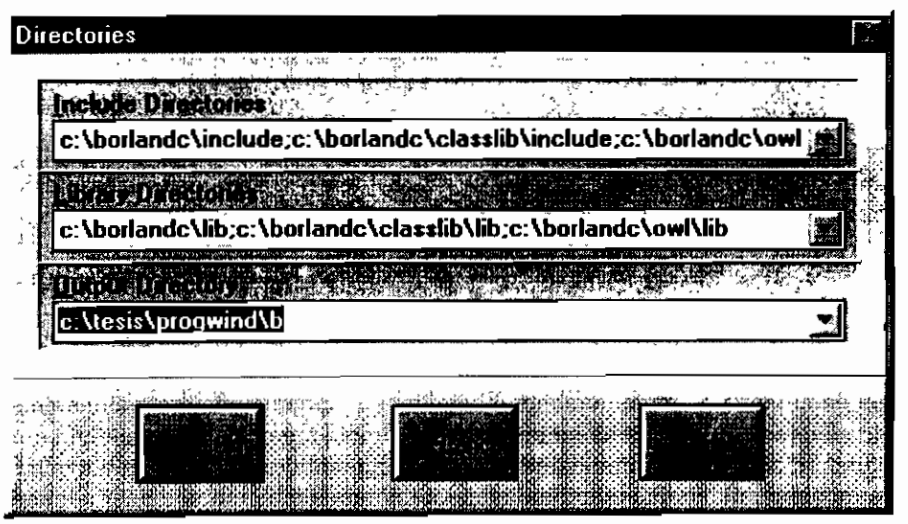

Figura No. 11 Caja de diálogo de Options/Directories

MANUAL DE USUARIO

La información contenida en este apéndice tiene por objeto familiarizar al usuario con el sistema de software implementado en este trabajo de tesis, se inicia indicando los requisitos mínimos necesarios para la ejecución del programa, para en un paso posterior indicar las funciones de menú que debe conocerse para el manejo adecuado del sistema.

INSTALACIÓN

Para la instalación y funcionamiento adecuados del programa se requiere:

- Microsoft Windows 95 o superior.
- 4 MBytes de memoria RAM.
- 6 MBytes de disco fijo.
- Monitor compatible con Windows (SVGA o superior, capaz de desplegar 256 tonalidades de gris).
- Computadora 80486.

Las capacidades en disco y en memoria solicitadas son las que el programa utiliza simultáneamente cuando este se encuentra en ejecución .

El archivo denominado INSTALAR.BAT, tiene bajo su responsabilidad el cargar los archivos necesarios del sistema desarrollado en el disco fijo. El archivo correspondiente para la ejecución del programa es MIMTDC.EXE, el mismo que para su correcto funcionamiento requiere de la asistencia de los módulos TDCOSENO.DLL y BWCC.DLL cargados también durante el proceso de instalación.

FUNCIONAMIENTO

Una vez puesto en marcha el programa se visualizará en pantalla una caja de dialogo con información relativa al programa como se muestra en la figura No. 12 .

Figura No. 12 Caja de diálogo mostrada cuando se inicia la ejecución del programa.

A continuación se describen cada una de las opciones de menú, indicando entre paréntesis la correspondiente combinación de teclas aceleradoras.

Menú Archivo

- Nuevo (Ctrl + N): permite poner el menú y el programa en condiciones iniciales. Es habilitado solamente por *Archivo/Abrir* y *Edición/Pegar*.

- Abrir (Ctrl + A): permite leer un archivo gráfico con formato BMP solamente. En caso contrario puede usarse la opción *Edición/Pegar* con ayuda de otro utilitario gráfico.

- Salir (Alt + F4): abandona la aplicación

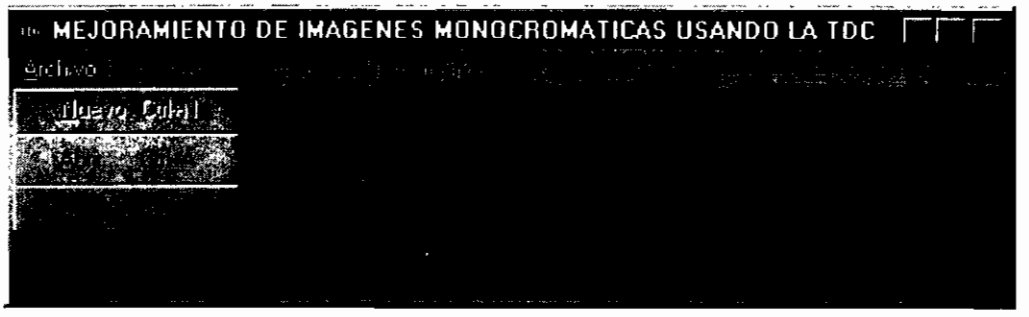

Figura No. 13 Opción de menú Archivo.

Menú Edición

- Deshacer (Alt + BkSp): deshace la acción del último filtro aplicado, opción habilitada solamente cuando se ha aplicado al menos un filtro, y desactivado luego de deshacer.

- Exportar (Ctrl + Ins): permite pegar la imagen desde la ventana principal hasta el portapapeles, opción habilitada solamente cuando hay imagen en pantalla.

- Pegar (Shift + Ins): permite pegar la imagen desde el portapapeles hasta una ventana, habilitado solamente en condiciones iniciales.

Figura No. 14 Opción de menú Edición.

Menú Ver

- Imagen Ruidosa (Ctrl + R): permite solicitar la visualización de la ventana que contiene la imagen ruidosa.

- Espectro (Ctrl + E): permite visualizar el espectro correspondiente de la imagen vigente, habilitado solamente después de haber ejecutado al menos una vez la opción de menú *Imagen/TDC.*
- Imagen Mejorada (Ctrl + M): permite visualizar la imagen mejorada, habilitado después de haber aplicado cualquier filtro.

Figura No. 15 Opción de menú Ver.

Menú Imagen

- Contaminar: permite seleccionar el tipo de ruido a usarse para la contaminación. Habilitado solamente después de usar *Archivo/Abrir* o *Edición/Pegar*.

- TDC (Ctrl + T): calcula la transformada discreta coseno de la imagen vigente. Habilitado solamente por *Archivo/Abrir* o *Edición/Pegar*.

- Atributos (Ctrl + I): muestra las dimensiones y resolución de la imagen presente.

- Filtrar: selecciona alguno de los tipos de filtros indicados. Habilitado por *Imagen/TDC*.

Figura No. 16 Opción de menú Imagen.

Menú Homomórfico

- Filtrar (Ctrl + H): opción de menú utilizada para iniciar el procesamiento homomórfico. Habilitado solamente después de usar *Archivo/Abrir* o *Edición/Pegar*.

Figura No. 17 Opción de menú Homomórfico.

Menú Ventana

Un usuario familiarizado con aplicaciones Windows entenderá el significado de los elementos de esta opción.

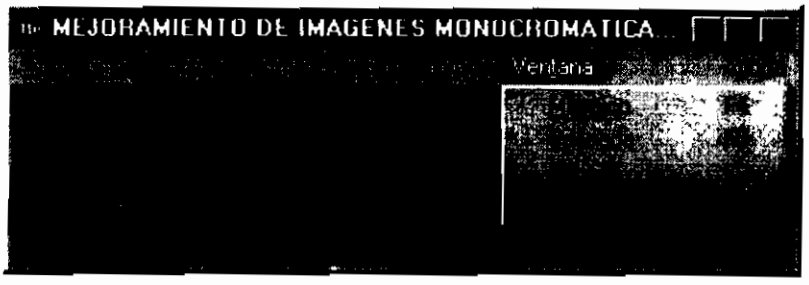

Figura No. 18 Opción de menú Ventana.

Menú Ayuda

- Ayuda (F1): permite el acceso al sistema de ayudas.

- Acerca de ...(Shift + F1): despliega la información de la aplicación.

Figura No. 19 Opción de menú Ayuda.

USO DEL MOUSE

La elección de cualquier elemento de menú se lo hace normalmente con el botón izquierdo del mouse, a la opción de doble clic sobre una ventana con el botón izquierdo se le ha asignado la función de copiar el contenido de la ventana hacia el portapapeles. Un clic con

el botón derecho sobre una ventana hará desplegar una caja de dialogo con la información respectiva de la imagen presentada en aquella ventana.

SISTEMA DE AYUDA

El sistema de ayuda contiene información de mucha utilidad especialmente cuando el usuario no está familiarizado con conceptos de procesamiento digital de imágenes en el área especifica de mejoramiento de imágenes en dominio de la frecuencia. Así mismo contiene información respecto a las funciones de menú, útil para aquellas personas no iniciadas en el manejo de este sistema de software.

UTILIZACIÓN DE LA LIBRERÍA TDCOSENO.DLL

La librería de enlace dinámico TDCOSENO.DLL contiene el código necesario para calcular el par de transformadas coseno bidimensionales de imágenes monocromáticas de tamaño $M \times N$, donde M y N son potencias de 2. Los prototipos de las funciones a invocarse son:

```
void CosDir(int M, int N, float huge *u, float huge *v);   // calcula la TDC directa
void CosInv(int M, int N, float huge *u, float huge *v);   //calcula la TDC inversa
```

en donde M es el alto, N es el ancho de la imagen en pixeles, **u** es un puntero al primer elemento tipo float de un bloque de memoria que contiene la imagen en el dominio espacial, **v** es un puntero al primer elemento tipo float de un bloque de memoria que contiene la imagen en el dominio de la frecuencia espacial; CosDir() toma el dato de **u** y devuelve en **v** , en cambio CosInv() toma el dato de **v** y devuelve en **u**. En ambos casos los bloques de memoria deberán estar previamente reservados.