ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA

IMPLANTACIÓN DE ALGORITMOS PARA CONTROL DIGITAL DIRECTO EN UN AMBIENTE MULTITAREA

TESIS PREVIA A LA OBTENCIÓN DEL TITULO DE

INGENIERO EN ELECTRÓNICA Y CONTROL

ANEXO B

LISTADO DEL PROGRAMA

FABRICIO MIGUEL HEREDIA MERINO

Quito, Marzo de 1996

```
// ARCHIVO: ADQUI.CPP

// Adqui demo window object

#include <owl.h>
#include <stdlib.h>
#include "demobase.h"
#include "adqui.h"          // definicion de class para TArtyWindow

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

//FABRI 95
extern int nsample;
extern float tsample;
extern int npuntos;
extern int bits;
extern int inportico,outportico;
extern float ylogicamin,ylogicamax;


//---------- TAdquiWindow ----------------------------------------

TAdquiWindow::TAdquiWindow( PTWindowsObject AParent, LPSTR ATitle ) :
                                                    TBaseDemoWindow( AParent,
ATitle )
{
        StaticControl = new TStatic(this, 100,
                "Pausa: Botón Izquierdo  Reiniciar: Botón Derecho",10,10,10,10,0);
        Iconizado = False;
        AltoTexto = 20;
        Pausa = FALSE;

        Ansample=nsample;
        Atsample=tsample;
        Anpuntos=npuntos;
        Anpuntosmax=npuntos;
        Anpuntosicon=5;
        contador=0;
        punto=0;
        Abits=bits;
        Ainportico=inportico;
        Aoutportico=outportico;
        Aylogicamin=ylogicamin;
        Aylogicamax=ylogicamax;
        xlogicamin=0;
        xlogicamax=Ansample*Atsample*Anpuntos;
        pendiente=(Aylogicamax-Aylogicamin)/(pow(2,Abits)-1);
};

/* Dispose of the objects that this window object created.  There's
        no need to dispose the List pointer, since it will only point to
        one of these two objects which are being disposed by their
        primary pointers */

void TAdquiWindow::GetWindowClass( WNDCLASS& WndClass )
{
```

```
        TBaseDemoWindow::GetWindowClass( WndClass );
        WndClass.hbrBackground = GetStockObject(WHITE_BRUSH);
        WndClass.hCursor = LoadCursor(0,IDC_CROSS);
        WndClass.hIcon = 0;  // we'll paint our own icon when minimized, thank you.
};

/* When the window is resized, scale the line list to fit the new
        window extent, or switch between full size and iconized window
        states.  */
void TAdquiWindow::WMSize( TMessage& Message )
{

        TBaseDemoWindow::WMSize(Message);

        /* Force Windows to repaint the entire window region */
        InvalidateRect(HWindow, NULL, TRUE);

        xfisicamax = Message.LP.Lo;
        yfisicamax = Message.LP.Hi;

        if (IsIconic(HWindow))
        {
                if (!Iconizado)
                {
                        Iconizado = TRUE;
                        contador=0;
                        punto=0;
                        Anpuntos=Anpuntosicon;
                        //List = IconicLineList;
                        escalax=.8*xfisicamax/Anpuntos;
                        escalay=.8*yfisicamax/(pow(2,Abits)-1);
                }
        }
        else
        {
                if (Iconizado)
                {
                        Iconizado = FALSE;
                        //List = BigLineList;
                };
                Anpuntos=Anpuntosmax;
                escalax=.8*xfisicamax/Anpuntos;
                yfisicamax -= AltoTexto;  /* allow room for the text at the bottom */
                escalay=.8*yfisicamax/(pow(2,Abits)-1);
                HDC PantallaDC=GetDC(HWindow);
                DibujaCuadro(PantallaDC);
                ReleaseDC(HWindow,PantallaDC);
        }

//      List->ScaleTo(NewXmax, NewYmax);  /* scale the lines in the list */
        if (StaticControl)
                MoveWindow(StaticControl->HWindow, 0, yfisicamax, xfisicamax, AltoTexto, True);
};

/* Toggle the window's Paused status.  Since the window will
        not receive mouse clicks when iconized, this will not pause the
        iconized lines display.  */
void TAdquiWindow::WMLButtonDown( TMessage& Mensaje )
```

```
{
        if(Pausa)
        {
                if (Mensaje.WParam & MK_CONTROL)
                {
                        if( Mensaje.LP.Lo<.098*xfisicamax || Mensaje.LP.Lo>.902*xfisicamax ||
                                Mensaje.LP.Hi<.098*yfisicamax || Mensaje.LP.Hi>.902*yfisicamax )
                                return;

                        float x,y;
                        char aux1[30];

                        HDC PantallaDC=GetDC(HWindow);
                        Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,1,RGB(255,0,255)) );
                        MoveTo(PantallaDC,.1*xfisicamax,Mensaje.LP.Hi);
                        LineTo(PantallaDC,Mensaje.LP.Lo,Mensaje.LP.Hi);
                        LineTo(PantallaDC,Mensaje.LP.Lo,.9*yfisicamax);

                        x=xlogicamin+(xlogicamax-xlogicamin)*(Mensaje.LP.Lo-
.1*xfisicamax)/(.8*xfisicamax);
                        y=Aylogicamin+(Aylogicamax-Aylogicamin)*(.9*yfisicamax-
Mensaje.LP.Hi)/(.8*yfisicamax);

                        sprintf(aux1,"%3.4g",x);
                        TextOut( PantallaDC,Mensaje.LP.Lo-3.5*strlen(aux1),yfisicamax-
20,aux1,strlen(aux1) );
                        sprintf(aux1,"%3.4g",y);
                        TextOut( PantallaDC,1,Mensaje.LP.Hi-10,aux1,strlen(aux1) );
                        DeleteObject(SelectObject(PantallaDC,Lapiz));
                        ReleaseDC(HWindow,PantallaDC);
                        return;
                }
        }
        Pausa = !Pausa;
};

/* Clear the line list when the user presses the right mouse
        button.  Same comments as above on iconized windows.  */
void TAdquiWindow::WMRButtonDown( TMessage& )
{
        if(!Pausa)
                Pausa=!Pausa;
        contador=0;
        punto=0;
        InvalidateRect(HWindow, NULL, TRUE);
        Pausa=!Pausa;
//      List->ResetLines();
};

void TAdquiWindow::WMMouseMove(TMessage& Mensaje)
{
        if ( !Pausa )
                return;
        if( Mensaje.LP.Lo<.098*xfisicamax || Mensaje.LP.Lo>.902*xfisicamax ||
                        Mensaje.LP.Hi<.098*yfisicamax || Mensaje.LP.Hi>.902*yfisicamax )
                return;

        float x,y;
```

```
        char aux1[40];

        x=xlogicamin+(xlogicamax-xlogicamin)*(Mensaje.LP.Lo-.1*xfisicamax)/(.8*xfisicamax);
        y=Aylogicamin+(Aylogicamax-Aylogicamin)*(.9*yfisicamax-Mensaje.LP.Hi)/(.8*yfisicamax);

        sprintf(mensaje,"(%3.4g,",x);
        sprintf(aux1,"%3.4g",y);
        strcat(mensaje,aux1);
        strcat(mensaje,")          ");

        HDC PantallaDC=GetDC(HWindow);
        TextOut(PantallaDC,1,1,mensaje,strlen(mensaje));
        ReleaseDC(HWindow,PantallaDC);

};

/* When the window is resized, or some other window blots out part
        of our client area, redraw the entire line list.  The PaintDC
        is fetched before Paint is called and is released for us after
        Paint is finished. */
void TAdquiWindow::Paint( HDC PaintDC, PAINTSTRUCT& PaintInfo)
{
        TBaseDemoWindow::Paint(PaintDC, PaintInfo);

        //PantallaDC=GetDC(HWindow);
        DibujaCuadro(PaintDC);
        //ReleaseDC(HWindow,PantallaDC);

//      List->Redraw(PaintDC);
};

void TAdquiWindow::DibujaCuadro(HDC PantallaDC)
{
        float y;

        Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,1,RGB(0,0,0)) );

        MoveTo(PantallaDC,.1*xfisicamax,.1*yfisicamax);
        LineTo(PantallaDC,.1*xfisicamax,.9*yfisicamax);
        LineTo(PantallaDC,.9*xfisicamax,.9*yfisicamax);
        LineTo(PantallaDC,.9*xfisicamax,.1*yfisicamax);
        LineTo(PantallaDC,.1*xfisicamax,.1*yfisicamax);

        //trazado de ejes
        //eje x
        if( Aylogicamin*Aylogicamax<0 )
        {
                y=.1*yfisicamax+.8*yfisicamax*(Aylogicamax-0)/(Aylogicamax-Aylogicamin);
                MoveTo(PantallaDC,.1*xfisicamax,y);
                LineTo(PantallaDC,.9*xfisicamax,y);
                sprintf(mensaje," 0");
                TextOut(PantallaDC,1,y-10,mensaje,strlen(mensaje));
        }
        DeleteObject(SelectObject(PantallaDC,Lapiz));
        //limites
        //eje y
        sprintf(mensaje,"%3.5g",Aylogicamin);
        TextOut(PantallaDC,1,.9*yfisicamax-10,mensaje,strlen(mensaje));
```

```
        sprintf(mensaje,"%3.5g",Aylogicamax);
        TextOut(PantallaDC,1,1*yfisicamax-10,mensaje,strlen(mensaje));
        //eje x
        sprintf(mensaje," 0");
        TextOut(PantallaDC,1*xfisicamax-3.5*strlen(mensaje),yfisicamax-
20,mensaje,strlen(mensaje));
        sprintf(mensaje,"%3.5g",Ansample*Atsample*Anpuntos);
        TextOut(PantallaDC,.9*xfisicamax-3.5*strlen(mensaje),yfisicamax-
20,mensaje,strlen(mensaje));
        //cuadro superior reloj

};

void TAdquiWindow::Adquisicion(HDC PantallaDC)
{
        int i;
        float x,y;

        x=punto*escalax+.1*xfisicamax;
        y=inp(Ainportico);
        y1[punto]=.9*yfisicamax-y*escalay;

        Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,2,RGB(0,0,255)) );
        MoveTo(PantallaDC,x,y1[punto]);
        LineTo(PantallaDC,x,y1[punto]);
        DeleteObject(SelectObject(PantallaDC,Lapiz));

        punto++;
        if( punto>Anpuntos )
        {
                Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,2,RGB(255,255,255)) );
                for(i=0;i<=Anpuntos;i++ )
                {
                        x=i*escalax+.1*xfisicamax;
                        MoveTo(PantallaDC,x,y1[i]);
                        LineTo(PantallaDC,x,y1[i]);
                }
                DeleteObject(SelectObject(PantallaDC,Lapiz));
                punto=0;
                DibujaCuadro(PantallaDC);
        }
}

void TAdquiWindow::EscribirTiempo(HDC PantallaDC)
{
        char s[ 21 ];

        //mostrar reloj
        sprintf( s, "%3.3f",contador*Atsample );
        TextOut( PantallaDC, xfisicamax-36, 4, s, 4);
}

/* Fetch a device context, pass it to the line list object, then
        release the device context back to Windows.  */
void TAdquiWindow::TimerTick()
{

        if (!Pausa)
```

```
    {
        if( fmod(contador,Ansample)==0 || contador==0 )
        {
            HDC PantallaDC=GetDC(HWindow);
            if( contador==0 )
                DibujaCuadro(PantallaDC);
            Adquisicion(PantallaDC);
            EscribirTiempo(PantallaDC);
            ReleaseDC(HWindow,PantallaDC);
        }
        contador++;
    }
};
```

```
//ARCHIVO: DIFER.CPP

// Difer demo window object

#include <owl.h>
#include <stdlib.h>
#include "demobase.h"
#include "difer.h"          // definicion de class para TArtyWindow

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

//FABRI 95
extern int nsample;
extern float tsample;
extern int npuntos;
extern int bits;
extern int inportico,outportico;
extern float ylogicamin,ylogicamax;
extern float referencia;
extern int gradon,gradod;
extern float num[maximo],den[maximo];   //ecuacion de diferencias


//---------- TDiferWindow ----------------------------------------

TDiferWindow::TDiferWindow( PTWindowsObject AParent, LPSTR ATitle ) :
                                                        TBaseDemoWindow( AParent,
ATitle )
{
        StaticControl = new TStatic(this, 100,
                "Pausa: Botón Izquierdo  Reiniciar: Botón Derecho",10,10,10,10,0);

        Iconizado = False;
        AltoTexto = 20;
        Pausa = FALSE;

        int i;

        Ansample=nsample;
        Atsample=tsample;
        Anpuntos=npuntos;
        Anpuntosmax=npuntos;
        Anpuntosicon=5;
        contador=0;
        punto=0;
        Abits=bits;
        Areferencia=referencia;
        Agradon=gradon;
        Agradod=gradod;
        for(i=0;i<=Agradon;i++)
                Anum[i]=num[i];
        for(i=0;i<=Agradod;i++)
                Aden[i]=den[i];

        for(i=0;i<Agradod,i++)
        {
```

```
                c[i]=0;
                r[i]=0;
        }

        Ainportico=inportico;
        Aoutportico=outportico;
        Aylogicamin=ylogicamin;
        Aylogicamax=ylogicamax;
        xlogicamin=0;
        xlogicamax=Ansample*Atsample*Anpuntos;
        pendiente=(Aylogicamax-Aylogicamin)/(pow(2,Abits)-1);
};

/* Dispose of the objects that this window object created.  There's
        no need to dispose the List pointer, since it will only point to
        one of these two objects which are being disposed by their
        primary pointers */

void TDiferWindow::GetWindowClass( WNDCLASS& WndClass )
{
        TBaseDemoWindow::GetWindowClass( WndClass );
        WndClass.hCursor = LoadCursor(0,IDC_CROSS);
        WndClass.hbrBackground = GetStockObject(WHITE_BRUSH);
        WndClass.hIcon = 0;  // we'll paint our own icon when minimized, thank you.
};

/* When the window is resized, scale the line list to fit the new
        window extent, or switch between full size and iconized window
        states.  */
void TDiferWindow::WMSize( TMessage& Message )
{

        TBaseDemoWindow::WMSize(Message);

        /* Force Windows to repaint the entire window region */
        InvalidateRect(HWindow, NULL, TRUE);

        xfisicamax = Message.LP.Lo;
        yfisicamax = Message.LP.Hi;

        if (IsIconic(HWindow))
        {
                if (!Iconizado)
                {
                        Iconizado = TRUE;
                        contador=0;
                        punto=0;
                        Anpuntos=Anpuntosicon;
                        //List = IconicLineList;
                        escalax=.8*xfisicamax/Anpuntos;
                        escalay=.8*yfisicamax/(pow(2,Abits)-1);
                }
        }
        else
        {
                if (Iconizado)
                {
                        Iconizado = FALSE;
```

```
                        //List = BigLineList;
                };
                Anpuntos=Anpuntosmax;
                escalax=.8*xfisicamax/Anpuntos;
                yfisicamax -= AltoTexto;  /* allow room for the text at the bottom */
                escalay=.8*yfisicamax/(pow(2,Abits)-1);
                PantallaDC=GetDC(HWindow);
                DibujaCuadro(PantallaDC);
                ReleaseDC(HWindow,PantallaDC);
        }
//      List->ScaleTo(NewXmax, NewYmax);  /* scale the lines in the list */
        if (StaticControl)
                MoveWindow(StaticControl->HWindow, 0, yfisicamax, xfisicamax, AltoTexto, True);
};

/* Toggle the window's Paused status.  Since the window will
        not receive mouse clicks when iconized, this will not pause the
        iconized lines display.  */
void TDiferWindow::WMLButtonDown( TMessage& Mensaje)
{
        if(Pausa)
        {
                if (Mensaje.WParam & MK_CONTROL)
                {
                        if( Mensaje.LP.Lo<.098*xfisicamax || Mensaje.LP.Lo>.902*xfisicamax ||
                                Mensaje.LP.Hi<.098*yfisicamax || Mensaje.LP.Hi>.902*yfisicamax )
                                return;

                        float x,y;
                        char aux1[30];

                        HDC PantallaDC=GetDC(HWindow);
                        Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,1,RGB(255,0,255)) );
                        MoveTo(PantallaDC,.1*xfisicamax,Mensaje.LP.Hi);
                        LineTo(PantallaDC,Mensaje.LP.Lo,Mensaje.LP.Hi);
                        LineTo(PantallaDC,Mensaje.LP.Lo,.9*yfisicamax);

                        x=xlogicamin+(xlogicamax-xlogicamin)*(Mensaje.LP.Lo-
.1*xfisicamax)/(.8*xfisicamax);
                        y=Aylogicamin+(Aylogicamax-Aylogicamin)*(.9*yfisicamax-
Mensaje.LP.Hi)/(.8*yfisicamax);

                        sprintf(aux1,"%3.4g",x);
                        TextOut( PantallaDC,Mensaje.LP.Lo-3.5*strlen(aux1),yfisicamax-
20,aux1,strlen(aux1) );
                        sprintf(aux1,"%3.4g",y);
                        TextOut( PantallaDC,1,Mensaje.LP.Hi-10,aux1,strlen(aux1) );
                        DeleteObject(SelectObject(PantallaDC,Lapiz));
                        ReleaseDC(HWindow,PantallaDC);
                        return;
                }
        }
        Pausa = !Pausa;
};

/* Clear the line list when the user presses the right mouse
        button.  Same comments as above on iconized windows.  */
void TDiferWindow::WMRButtonDown( TMessage& )
```

```
{
        int i;
        if(!Pausa)
                Pausa=!Pausa;
        contador=0;
        punto=0;
        for(i=0;i<Agrado;i++)
        {
                c[i]=0;
                r[i]=0;
        }
        InvalidateRect(HWindow, NULL, TRUE);
        Pausa=!Pausa;
//      List->ResetLines();
};

void TDiferWindow::WMMouseMove(TMessage& Mensaje)
{
        if ( !Pausa )
                return;
        if( Mensaje.LP.Lo<.098*xfisicamax || Mensaje.LP.Lo>.902*xfisicamax ||
                        Mensaje.LP.Hi<.098*yfisicamax || Mensaje.LP.Hi>.902*yfisicamax )
                return;

        float x,y;
        char aux1[40];

        x=xlogicamin+(xlogicamax-xlogicamin)*(Mensaje.LP.Lo-.1*xfisicamax)/(.8*xfisicamax);
        y=Aylogicamin+(Aylogicamax-Aylogicamin)*(.9*yfisicamax-Mensaje.LP.Hi)/(.8*yfisicamax);

        sprintf(mensaje,"(%3.4g,",x);
        sprintf(aux1,"%3.4g",y);
        strcat(mensaje,aux1);
        strcat(mensaje,")          ");

        HDC PantallaDC=GetDC(HWindow);
        TextOut(PantallaDC,1,1,mensaje,strlen(mensaje));
        ReleaseDC(HWindow,PantallaDC);
};

/* When the window is resized, or some other window blots out part
        of our client area, redraw the entire line list.  The PaintDC
        is fetched before Paint is called and is released for us after
        Paint is finished. */
void TDiferWindow::Paint( HDC PaintDC, PAINTSTRUCT& PaintInfo)
{
        TBaseDemoWindow::Paint(PaintDC, PaintInfo);

        DibujaCuadro(PaintDC);

//      List-> Redraw(PaintDC);
};

void TDiferWindow::DibujaCuadro(HDC PantallaDC)
{
        float y;

        Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,1,RGB(0,0,0)) );
```

```
        MoveTo(PantallaDC,.1*xfisicamax,.1*yfisicamax);
        LineTo(PantallaDC,.1*xfisicamax,.9*yfisicamax);
        LineTo(PantallaDC,.9*xfisicamax,.9*yfisicamax);
        LineTo(PantallaDC,.9*xfisicamax,.1*yfisicamax);
        LineTo(PantallaDC,.1*xfisicamax,.1*yfisicamax);

        //trazado de ejes
        //eje x
        if( Aylogicamin*Aylogicamax<0 )
        {
                y=.1*yfisicamax+.8*yfisicamax*(Aylogicamax-0)/(Aylogicamax-Aylogicamin);
                MoveTo(PantallaDC,.1*xfisicamax,y);
                LineTo(PantallaDC,.9*xfisicamax,y);
                sprintf(mensaje," 0");
                TextOut(PantallaDC,1,y-10,mensaje,strlen(mensaje));
        }

        y=.1*yfisicamax+.8*yfisicamax*(ylogicamax-referencia)/(ylogicamax-ylogicamin);
        MoveTo(PantallaDC,.1*xfisicamax,y);
        LineTo(PantallaDC,.9*xfisicamax,y);
        sprintf(mensaje,"%3.5g",referencia);
        TextOut( PantallaDC,1,y-10,mensaje,strlen(mensaje) );

        DeleteObject(SelectObject(PantallaDC,Lapiz));
        //limites
        //eje y
        sprintf(mensaje,"%3.5g",Aylogicamin);
        TextOut(PantallaDC,1,.9*yfisicamax-10,mensaje,strlen(mensaje));
        sprintf(mensaje,"%3.5g",Aylogicamax);
        TextOut(PantallaDC,1,.1*yfisicamax-10,mensaje,strlen(mensaje));
        //eje x
        sprintf(mensaje," 0");
        TextOut(PantallaDC,.1*xfisicamax-3,5*strlen(mensaje),yfisicamax-
20,mensaje,strlen(mensaje));
        sprintf(mensaje,"%3.5g",Ansample*Atsample*Anpuntos);
        TextOut(PantallaDC,.9*xfisicamax-3,5*strlen(mensaje),yfisicamax-
20,mensaje,strlen(mensaje));
        //cuadro superior reloj

};

void TDiferWindow::Difer(HDC PantallaDC)
{
        int i;
        float x,y,error;

        x=punto*escalax+.1*xfisicimax;
        y=inp(Ainportico);
        y1[punto]=.9*yfisicamax-y*escalay;

        Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,2,RGB(0,0,255)) );
        MoveTo(PantallaDC,x,y1[punto]);
        LineTo(PantallaDC,x,y1[punto]);
        DeleteObject(SelectObject(PantallaDC,Lapiz));

        error=Areferencia-(y*pendiente+Aylogicamin);
```

```
//cout<<"in "<<(y*pendiente+Aylogicamin)<<" r "<<Areferencia<<" ";

        if( gradod==gradon )
                r[Agradod]=error;
        else
                r[Agradod-1]=error;

        c[Agradod]=0.;

        for(i=1;i<=Agradod;i++)
                c[Agradod]+=-Aden[i]*c[Agradod-i];
        for(i=0;i<=Agradon;i++)
                c[Agradod]+=Anum[i]*r[Agradon-i];

        y=(c[Agradod]-Aylogicamin)/pendiente;
        if( y>255 ) y=255;
        if( y<0 ) y=0;

        for(i=0;i<Agradod;i++)
                c[i]=c[i+1];
        for(i=0;i<Agradod;i++)
                r[i]=r[i+1];

        y=outp(outportico,y);
        y2[punto]=.9*yfisicamax-y*escalay;

//cout<<" e "<<error<<" sal "<<c[Agradod]<<"\n";

        Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,2,RGB(255,0,0)) );
        MoveTo(PantallaDC,x,y2[punto]);
        LineTo(PantallaDC,x,y2[punto]);
        DeleteObject(SelectObject(PantallaDC,Lapiz));

        punto++;
        if( punto>Anpuntos )
        {
                Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,2,RGB(255,255,255)) );
                for(i=0;i<=Anpuntos;i++ )
                {
                        x=i*escalax+.1*xfisicamax;
                        MoveTo(PantallaDC,x,y1[i]);
                        LineTo(PantallaDC,x,y1[i]);
                        MoveTo(PantallaDC,x,y2[i]);
                        LineTo(PantallaDC,x,y2[i]);
                }
                DeleteObject(SelectObject(PantallaDC,Lapiz));
                punto=0;
                DibujaCuadro(PantallaDC);
        }
}

void TDiferWindow::EscribirTiempo(HDC PantallaDC)
{
        char s[ 21 ];

        //mostrar reloj
        sprintf( s, "%3.3f",contador*Atsample );
        TextOut( PantallaDC, xfisicamax-36, 4, s, 4);
```

```
}

/* Fetch a device context, pass it to the line list object, then
        release the device context back to Windows.  */
void TDiferWindow::TimerTick()
{
        //HDC DC;

        if (!Pausa)
        {
                if( fmod(contador,Ansample)==0 || contador==0 )
                {
                        PantallaDC=GetDC(HWindow);
                        if( contador==0 )
                                DibujaCuadro(PantallaDC);
                        Difer(PantallaDC);
                        EscribirTiempo(PantallaDC);
                        ReleaseDC(HWindow,PantallaDC);
                }
                contador++;
        }
};
```

```
// ARCHIVO:EXPANDIR.CPP

#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>

#include "datos.h"
#include "expandir.h"

/* expandir expande un polinomio
  este polinomio es ingresado factorado en forma de raices
  nraiz = numero de raices inicial del polinomio
          *grado = grado del polinomio resultante (talvez no sea necesario)
          r[]  = multiplicidad de cada raiz
          re[] = parte real de cada raiz
          im[] = parte imaginaria de cada raiz
          coef[] = coeficientes del polinomio expandido
*/


void expandir(int nraiz,int r[], float re[], float im[],int *grado,float coef[])
{
        int h,j,k,g1,g2,g3,g4;
        float aux1[maximo],aux2[maximo],aux3[maximo],aux4[maximo];

        if( nraiz==0 )
        {
                *grado=0;
                coef[0]=1;
        }
        else
        {
                g4=0;
                aux4[0]=1;
                for(j=0;j<nraiz;j++)
                {
                        if( im[j]==0 )   //polo real
                        {
                                if( r[j]>=2 )     //multiplicidad 2 o mayor
                                {
                                        aux2[0]=1;
                                        aux2[1]=-re[j];
                                        potencia(r[j],&g1,aux2,aux1);

                                }
                                else
                                {
                                        g1=1;
                                        aux1[0]=1;
                                        aux1[1]=-re[j];
                                }
                        }
                        else            //polo complejo
                        {
                                g1=2;
                                aux2[0]=aux3[0]=aux1[0]= 1;
                                aux2[1]=aux3[1]=aux1[1]= -2*re[j];
```

```
                        aux2[2]=aux3[2]=aux1[2]= re[j]*re[j]+im[j]*im[j];
                        if( r[j]>=2 )     //multiplicidad 2 o mayor
                        {
                                g2=2;
                                for(k=1;k<r[j];k++)
                                {
                                        multiplica2(2,g2,&g1,aux2,aux3,aux1);
                                        g2=g1;
                                        for(h=0;h<=g2;h++) aux3[h]=aux1[h];
                                }
                        }
                }
                multiplica2(g1,g4,&g3,aux1,aux4,coef);
                g4=g3;
                for(h=0;h<=g4;h++) aux4[h]=coef[h];
        }
        *grado = g3;
    }
}
```

```
// ARCHIVO: LEER2.CPP

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <iostream.h>
#include <math.h>

#include "leer.h"


/* escribefuncion() escribe una cadena en un archivo
        salida = nombre del archivo
        s      = cadena que se va a escribir
*/

void escribefuncion(char *salida,char s[])
{
        FILE *out;

        if((out=fopen(salida,"w"))==NULL)
        {
                cout<<"error:no se puede abrir el archivo para escribir";
                return;
        }
        if( s[0]=='\0' )
                fprintf(out,"p\n");
        else
        {
                if( s[strlen(s)-1]==')' || s[strlen(s)-2]=='^' )
                        fprintf(out,"%s\n",s);
                else
                        fprintf(out,"%sp\n",s);  //genero un error
        }
        fclose(out);
}

/* leefuncion() origen lee archivo lex y busca errores
        entrada = nombre del archivo
        s       = cadena que se va a devolver
*/

void leefuncion(char *entrada,char s[])
{
        FILE *in;
        char aux[50];
        if((in=fopen(entrada,"r"))==NULL)
        {
                strcpy(s,"error:no se puede abrir el archivo para leer");
                return;
        }
        fscanf(in,"%s",aux);
        strcpy(s,aux);
        if( aux[0]=='e' )
        {
                sprintf(aux,"\ncódigos de errores:\n");
                strcat(s,aux);
```

```
                sprintf(aux,"1:no existe suma de factores\n");
                strcat(s,aux);
                sprintf(aux,"2:factor sin elementos\n");
                strcat(s,aux);
                sprintf(aux,"3:factor sin potencia\n");
                strcat(s,aux);
                sprintf(aux,"4:no existe suma de factores\n");
                strcat(s,aux);
                sprintf(aux,"5:signo de potencia perdido\n");
                strcat(s,aux);
                sprintf(aux,"6:no existe potenciación para números\n");
                strcat(s,aux);
                sprintf(aux,"7:caracteres desconocidos");
                strcat(s,aux);
        }
        fclose(in);
}

/* leerorigen() lee desde el archivo de entrada generado por el LEX la funcion
        original en forma de numero de polinomios del numerador (denominador)
        grado de cada polinomio, coeficientes del polinomio y potencia del polinomio
        y escribe en un archivo de salida el polinomio en forma de polos y ceros.
        y su respectiva ganancia: ganancia,numero de raices del numerador (denominador),
        parte real,imaginaria,multiplicidad
        si no existen raices (grado=0) entonces escribe numero de raices cero y ya no escribe
        parte real,imaginaria y multiplicidad
        ademas esta subrutina entrega estos valores a variables de salida para su posterior
        utilizacion sin necesidad de leer el archivo de salida originado

        entrada = nombre del archivo de entrada
        salida  = nombre del archivo de salida
        k       = ganancia de la funcion de transferencia
        nraizn (nraizd) = numero de raices del numerador (denominador)
        ren[]  (red[])  = parte real de las raices del numerador (denominador)
        imn[]  (imd[])  = parte imaginaria de las raices del numerador (denominador)
        rn[]   (rd[])   = multiplicidad de cada raiz del numerador (denominador)
*/


void leerorigen(char *entrada,char *salida,char *modosz,float *k,int *nraizn,float ren[],float imn[],int
rn[],int *nraizd,float red[],float imd[],int rd[],char s[])
{
        FILE *in,*out;
        int i,j;
        float auxf;
        float coeficiente[30];
        int npol,pot,grado;
        char modo,forma;  //modo s=analogo,z=digital.
                                                                        //forma
o=riginal,f=factorada
        int n1,n2,n3;
        float k1,k2,auxre[30],auxim[30];

        char s1[30];

        strcpy(s,"");
        if((in=fopen(entrada,"r"))==NULL)
        {
```

```
            strcpy(s,"error:no se puede abrir el archivo para leer");
            return;
        }

        if((out=fopen(salida,"w"))==NULL)
        {
            strcpy(s,"error:no se puede abrir el archivo para escribir");
            fclose(in);
            return;
        }
        fscanf(in,"%c",&modo);
        if( modo=='S' ) modo='s';
        if( modo=='Z' ) modo='z';
        if( modo!='s' && modo!='z' )
        {
            strcpy(s,"error:el archivo abierto no tiene un formato aceptado");
            fclose(in);
            fclose(out);
            return;
        }
        sprintf(s1,"%c",modo);
        strcat(s,s1);
        (*modosz)=modo;
        fscanf(in,"\n%c",&forma);
        if( forma!='o' )
        {
            strcpy(s,"error:el archivo abierto no tiene un formato aceptado");
            fclose(in);
            fclose(out);
            return;
        }
        sprintf(s1,"\n%c",forma);
        strcat(s,s1);

// NUMERADOR
        k1=1;
        n1=0;
        fscanf(in,"\n%i",&npol);
        sprintf(s1,"\n%i",npol);
        strcat(s,s1);
        for(i=0;i<npol;i++)
        {
            fscanf(in,"\n%i",&grado);
            sprintf(s1,"\n%i",grado);
            strcat(s,s1);
            for(j=0;j<=grado;j++)
            {
                fscanf(in," %f",&auxf);
                sprintf(s1," %f",auxf);
                strcat(s,s1);
                coeficiente[j]=auxf;
            }
            fscanf(in," %i",&pot);
            sprintf(s1," %i",pot);
            strcat(s,s1);
            k1=k1*pow(coeficiente[0],pot);
            if(grado>=1)
            {

                fscanf(in,"\n%i",&grado);
                sprintf(s1,"\n%i",grado);
                strcat(s,s1);
                for(j=0;j<=grado;j++)
                {
                    fscanf(in," %f",&auxf);
                    sprintf(s1," %f",auxf);
                    strcat(s,s1);
                    coeficiente[j]=auxf;
                }
                fscanf(in," %i",&pot);
                sprintf(s1," %i",pot);
                strcat(s,s1);
                k2=k2*pow(coeficiente[0],pot);
                if(grado>=1)
                {
                    raices(grado,coeficiente,&n3,auxre,auxim);
                    for(j=n2;j<(n2+n3);j++)
```

```
                        raices(grado,coeficiente,&n3,auxre,auxim);
                        for(j=n1;j<(n1+n3);j++)
                        {
                                ren[j]=auxre[j-n1];
                                imn[j]=auxim[j-n1];
                                rn[j]=pot;
                        }
                        n1=n1+n3;
                }
        }
        multiplicidad(&n1,rn,ren,imn);

// DENOMINADOR
        k2=1.;
        n2=0;
        fscanf(in,"\n%i",&npol);
        sprintf(s1,"\n%i",npol);
        strcat(s,s1);
        for(i=0;i<npol;i++)
        {
                fscanf(in,"\n%i",&grado);
                sprintf(s1,"\n%i",grado);
                strcat(s,s1);
                for(j=0;j<=grado;j++)
                {
                        fscanf(in," %f",&auxf);
                        sprintf(s1," %f",auxf);
                        strcat(s,s1);
                        coeficiente[j]=auxf;
                }
                fscanf(in," %i",&pot);
                sprintf(s1," %i",pot);
                strcat(s,s1);
                k2=k2*pow(coeficiente[0],pot);
                if(grado>=1)
                {
                        raices(grado,coeficiente,&n3,auxre,auxim);
                        for(j=n2;j<(n2+n3);j++)
                        {
                                red[j]=auxre[j-n2];
                                imd[j]=auxim[j-n2];
                                rd[j]=pot;
                        }
                        n2=n2+n3;
                }
        }
        strcat(s,"\n");
        multiplicidad(&n2,rd,red,imd);
        reducir(&n1,&n2,rn,rd,ren,imn,red,imd);

        *nraizn=n1;
        *nraizd=n2;
        *k=k1/k2;

        fprintf(out,"%c",modo);
        fprintf(out,"\nf");
        fprintf(out,"\n%f",*k);
        fprintf(out,"\n%d",n1); //numero de raices del numerador
```

```
        if(n1>=1) fprintf(out,"\n");
        for(i=0;i<n1;i++)
        {
                fprintf(out,"%f ",ren[i]);
                fprintf(out,"%f ",imn[i]);
                fprintf(out,"%d ",rn[i]);
        }

        fprintf(out,"\n%d",n2); //numero de raices del denominador
        if(n2>=1) fprintf(out,"\n");
        for(i=0;i<n2;i++)
        {
                fprintf(out,"%f ",red[i]);
                fprintf(out,"%f ",imd[i]);
                fprintf(out,"%d ",rd[i]);
        }

        fclose(in);
        fclose(out);
}
```

```
// ARCHIVO: MOSTRAR.CPP

#include <iostream.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include<complex.h>

#include "datos.h"
#include "fracpar.h"

int fact2(int i);
void mostrarpolos1(char simbolo,float real,float imag,int r,char s[]);
void mostrarp(char simbolo,int m, float c[], char s[]);
// todas las subrutinas mostrar añaden a una cadena s lo que se quiere utilizar
// para visualizacion


/* mostrarorigen() pone en la cadena s un polinomio que ingresa desde un archivo
        en forma de numero de polinomios del numerador (denominador),
        grado de cada polinomio, coeficientes y potencia a la que esta elevado cada polinomio
*/

void mostrarorigen(char *entrada,char s[])
{
        FILE *in;
        int i,j,contador;
        float auxf,coeficiente[30];
        int npol,pot,grado;
        char modo,forma;  //modo s=analogo,z=digital  aqui debe entrar forma = o
                                                                                //forma
o=riginal,f=factorada
        char s1[50];

        if((in=fopen(entrada,"r"))==NULL)
        {
                printf("\n No se puede abrir el archivo para leer");
                return;
        }
//      cout<<"\n Leyendo Archivo" << entrada;


        fscanf(in,"%c",&modo);
        fscanf(in,"\n%c",&forma);

        strcpy(s,"");
        // NUMERADOR

        fscanf(in,"\n%i",&npol);

        if( npol>=1)    //esta revision no es necesaria, pero para mayor seguridad
        {
                for(i=0;i<npol;i++)
                {
                        strcat(s,"(");
                        fscanf(in,"%i",&grado);

                        for(j=0;j<=grado;j++)
```

```
                        {
                                fscanf(in,"%f",&auxf);
                                coeficiente[j]=auxf;
                        }
                        //lo que sigue es para verificar que los coeficientes sean deistintos de cero
                        contador=0;
                        while( coeficiente[0]==0 && grado>=1 )
                        {
                                grado += -1;
                                contador++;
                                coeficiente[0]=coeficiente[contador];
                        }
                        if( contador>=0 )
                        {
                                for(j=0;j<=grado;j++)
                                coeficiente[j]=coeficiente[j+contador];
                        }
                        //if( coeficiente[0]==0 ) coeficiente[0]=1;  //para evitar errores de funciones
mal ingresadas
                        //NO ES NECESARIO

                        mostrarp(modo,grado,coeficiente,s1);
                        strcat(s,s1);
                        strcat(s,")");

                        fscanf(in,"%i",&pot);
                        if(pot>=2) { sprintf(s1,"^%d",pot);  strcat(s,s1); }
                }
        }

        strcat(s,"\n");
        // DENOMINADOR
        fscanf(in,"%i",&npol);

        if( npol>=1 )  //no es necesario esta revision
        {
                for(i=0;i<npol;i++)
                {
                        strcat(s,"(");
                        fscanf(in,"%i",&grado);

                        for(j=0;j<=grado;j++)
                        {
                                fscanf(in,"%f",&auxf);
                                coeficiente[j]=auxf;
                        }
                        //lo que sigue es para verificar que los primeros coeficientes sean distintos de
cero
                        contador=0;
                        while( coeficiente[0]==0 && grado>=1 )
                        {
                                grado += -1;
                                contador++;
                                coeficiente[0]=coeficiente[contador];
                        }
                        if( contador>=0 )
                        {
                                for(j=0;j<=grado;j++)
```

```
                                        coeficiente[j]=coeficiente[j+contador];
                        }
                        if( coeficiente[0]==0 ) coeficiente[0]=1;  //para evitar errores de funciones mal
ingresadas

                        mostrarp(modo,grado,coeficiente,s1);
                        strcat(s,s1);
                        strcat(s,")");

                        fscanf(in,"%i",&pot);
                        if(pot>=2) { sprintf(s1,"^%d",pot); strcat(s,s1); }
                }
        }
        fclose(in);
}


/* mostrarp() pone en la cadena s un polinomio que entra en forma expandida
        se podria utilizar una bandera para imprimir "s" o "z"
        pero en lugar de ello se mandara el caracter que se quiere visualizar
        simbolo = puede ser una letra cualquiera pero para nuestra aplicacion "s" o "z"
        m = grado del polinomio de entrada
        c[]=coeficientes de entrada del polinomio en forma descendente
        s[]=cadena de salida
        se supone que el primer  coeficiente no debe ser cero
        comprobar esto en rutina de llamada
*/
void mostrarp(char simbolo,int m, float c[], char s[])
{
        int i;
        char signo, s1[20];
        float temp;

        strcpy(s,"");
        temp=c[0];      //***todo antes del lazo se puede meter en el lazo pero
        if ( temp<0 )   //****no se hace para no obtener espacios en blanco al inicio
        {
                if ( m==0 ) { sprintf(s1,"- %g",-temp);  strcat(s,s1); }
                else
                {
                        if( temp!=-1 ) { sprintf(s1,"- %g",-temp); strcat(s,s1); }
                        else { sprintf(s1,"-");  strcat(s,s1); }
                }
        }
        else      //(tem>=0)
        {
                if ( m==0 ) { sprintf(s1,"%g",temp); strcat(s,s1); }
                else
                {
                        if( temp!=1 && temp!=0 ){ sprintf(s1,"%g",temp); strcat(s,s1); }
                }
        }

                                        //********hasta aqui

        if ( m>=1 )
        {
                if ( m==1 && temp!=0 ) { sprintf(s1,"%c",simbolo);  strcat(s,s1); }
```

```
                if ( m>=2 && temp!=0 ) { sprintf(s1,"%c^%d",simbolo,m); strcat(s,s1);}   //y esto
tambien
                for(i=1;i<=m;i++)
                {
                        temp=c[i];
                        if (temp==0) continue;
                        if ( temp>0 ) signo='+';
                        else { signo='-';temp=-temp;}
                        if (temp!=1)            //coeficente <> 1
                        {
                                if ((m-i)>1)
                                        sprintf(s1," %c %g%c^%d",signo,temp,simbolo,m-i);
                                else
                                {
                                        if ((m-i)==1)
                                                sprintf(s1," %c %g%c",signo,temp,simbolo);
                                        else
                                                sprintf(s1," %c %g",signo,temp);
                                }
                        }
                        else                    //coeficiente = 1
                        {
                                if((m-i)>1)
                                        sprintf(s1," %c %c^%d",signo,simbolo,m-i);
                                else
                                {
                                        if((m-i)==1)
                                                sprintf(s1," %c %c",signo,simbolo);
                                        else
                                                sprintf(s1," %c 1",signo);
                                }
                        }
                        strcat(s,s1);
                }
        }
}


/* mostrarpolos muestra un polinomio en forma de polos y ceros
        simbolo= letra de presentacion "s" o "z"
        k = constante de multiplicacion del polinomio original
        nraiz= numero de raices del polinomio, siempre >=1
        re[]= parte real de las raices del polinomio original
        im[]= parte imaginaria de las raices del polinomio original
        r[]= multiplicidad de cada raiz
        s[] = cadena de salida
*/

void mostrarpolos(char simbolo,float k,int nraiz,float re[],float im[],int r[],char s[])
{
        int i;
        char s1[50];

        strcpy(s,"");
        if( nraiz>=1 )
        {
                if( k!=1 ) sprintf(s,"%g*",k);
```

```
                else strcpy(s,"");
                if( nraiz>=2)
                {
                        for(i=0;i<nraiz-1;i++)
                        {
                                mostrarpolos1(simbolo,re[i],im[i],r[i],s1);
                                strcat(s,s1);
                                strcat(s,"*");
                        }
                }
                mostrarpolos1(simbolo,re[nraiz-1],im[nraiz-1],r[nraiz-1],s1);
                strcat(s,s1);
        }
        else
                sprintf(s,"%g",k);
}


/* mostrarpolos1 muestra un polinomio de grado 1 o 2 en forma de polos y ceros
        esta subrutina tambien sirve para mostrarfp fracciones parciales
        simbolo= letra de presentacion "s" o "z"
        real= parte real de las raices del polinomio original
        imag= parte imaginaria de la raiz del polinomio original
        r= multiplicidad de cada raiz
        s[] = cadena de salida
        para esta subrutina por favor ingresar las raices ya formateadas (aproximadas)
        la parte negativa siempre debe entrar positiva
*/

void mostrarpolos1(char simbolo,float real,float imag,int r,char s[])
{
        char s1[50];
        //char signo;

        strcpy(s,"");
        if( imag==0)        //raiz real
        {
                if( real<0 ) sprintf(s1,"[%c + %g]",simbolo,-real);
                if( real>0 ) sprintf(s1,"[%c - %g]",simbolo,real);
                if (real==0) sprintf(s1,"[%c]",simbolo);
        }
        else            //raiz imaginaria
        {
                if( real<0 )        sprintf(s1,"[(%c + %g)² + %g²]",simbolo,-real,imag);
                if( real>0 )        sprintf(s1,"[(%c - %g)² + %g²]",simbolo,real,imag);
                if( real==0 ) sprintf(s1,"[%c² + %g²]",simbolo,imag);
        }
        strcat(s,s1);
        if( r>1 )           //multiplicidad
        {
                sprintf(s1,"^%d",r);
                strcat(s,s1);
        }
}
```

```
// ARCHIVO: MULTIP2.CPP

/* DERIVADA, SUMA, MULTIPLICACION, DIVISION, POTENCIACION DE POLINOMIOS
   POLINOMIOS REALES!  */

#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>

#include "datos.h"

// multiplicacion de polinomios (descendente) reales
// asegurarse que los coeficinetes[0] sean <> 0
// a[] (grado na) * b[] (grado nb) = c[] (grado nc)

void multiplica2(int na, int nb, int *nc, float a[], float b[], float c[])
{
        int i,j,contador;

        //asegurar primer coeficiente diferente de cero (optimizar)
        contador=0;
        while( a[0] == 0 && na >= 1 )
        {
                na--;
                contador++;
                a[0] = a[contador];
        }

        if(contador > 0 )
        {
                for(i = 0;i<= na;i++)
                        a[i] = a[i + contador];
        }

        //asegurar primer coeficiente diferente de cero
        contador=0;
        while( b[0] == 0 && nb >= 1 )
        {
                nb--;
                contador++;
                b[0] = b[contador];
        }

        if(contador > 0 )
        {
                for(i = 0;i<= nb;i++)
                        b[i] = b[i + contador];
        }

        *nc=na+nb;
        for(i=0;i<=na+nb;i++) c[i]=0;
        for(i=0;i<=na;i++)
        {
                for(j=0;j<=nb;j++)
                {
                        c[i+j]=c[i+j]+a[i]*b[j];
                }
```

```
        }
}

// POTENCIACION DE POLINOMIOS  (REAL)
// eleva un binomio real a[] a una potencia r
// (a0*s+a1)^r=b[] (grado nb)
// funciona para r<=15 (optimamente?)
// tratar que r>=2

void potencia(int r, int *nb, float a[], float b[])
{
        int i;
        float aux1,aux2;

        if (a[0]==0 && a[1]==0)
        {
                cout<<"\nERROR. POTENCIA.CPP:coeficientes = cero, podria generar error";
                exit(1);
        }

        if(a[0]==0)
        {
                *nb=0;
                //cout<<"\ngrado en sub a[0]=0  "<<*nb;
                b[0]=pow(a[1],r);
                return;
        }
        if(a[1]==0)
        {
                *nb=r;
                //cout<<"\ngrado en sub a[1]=0 "<<*nb;
                b[0]=pow(a[0],r);
                for(i=1;i<=r;i++) b[i]=0;
        }

        *nb=r;

        b[0]=1;
        b[r]=1;
        switch(r)
        {
                case 0:
                        break;
                case 1:
                        b[0]=a[0];b[1]=a[1];
                        break;
                case 2:
                        b[1]=2;
                        break;
                case 3:
                        b[1]=3;b[2]=3;
                        break;
                case 4:
                        b[1]=4;b[2]=6;b[3]=4;
                        break;
                case 5:
                        b[1]=5;b[2]=10;b[3]=10;b[4]=5;
                        break;
```

```
                case 6:
                        b[1]=6;b[2]=15;b[3]=20;b[4]=15;b[5]=6;
                        break;
                case 7:
                        b[1]=7;b[2]=21;b[3]=35;b[4]=35;b[5]=21;b[6]=7;
                        break;
                case 8:
                        b[1]=8;b[2]=28;b[3]=56;b[4]=70;b[5]=56;b[6]=28;b[7]=8;
                        break;
                case 9:
                        b[1]=9;b[2]=36;b[3]=84;b[4]=126;b[5]=126;b[6]=84;b[7]=36;b[8]=9;
                        break;
                case 10:

        b[1]=10;b[2]=45;b[3]=120;b[4]=210;b[5]=252;b[6]=210;b[7]=120;b[8]=45;b[9]=10;
                        break;
                case 11:

        b[1]=11;b[2]=55;b[3]=165;b[4]=330;b[5]=462;b[6]=462;b[7]=330;b[8]=165;b[9]=55;b[10]=11;
                        break;
                case 12:

        b[1]=12;b[2]=66;b[3]=220;b[4]=495;b[5]=792;b[6]=924;b[7]=792;b[8]=495;b[9]=220;b[10]=6
6;b[11]=12;
                        break;
                case 13:

        b[1]=13;b[2]=78;b[3]=286;b[4]=715;b[5]=1287;b[6]=1716;b[7]=1716;b[8]=1287;b[9]=715;b[1
0]=286;b[11]=78;b[12]=13;
                        break;
                case 14:

        b[1]=14;b[2]=91;b[3]=364;b[4]=1001;b[5]=2002;b[6]=3003;b[7]=3432;b[8]=3003;b[9]=2002;b
[10]=1001;b[11]=364;b[12]=91;b[13]=14;
                        break;
                case 15:

        b[1]=15;b[2]=105;b[3]=455;b[4]=1365;b[5]=3003;b[6]=5005;b[7]=6435;b[8]=6435;b[9]=5005;
b[10]=3003;b[11]=1365;b[12]=455;b[13]=105;b[14]=15;
                        break;
                default:
                        cout<<"ERROR: se excede el grado (15) del polinomio";
                        exit(1);
        }
        if( r>=2 )
        {
                aux1=1;
                aux2=pow(a[0],r);
                for(i=0;i<=r;i++)
                {
                        b[i] *= aux1*aux2;
                        aux1 *= a[1]/a[0];
                }
        }
}
```

```
// ARCHIVO: ONOFF.CPP

// Adqui demo window object

#include <owl.h>
#include <stdlib.h>
#include "demobase.h"
#include "onoff.h"          // definicion de class para TArtyWindow

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

//FABRI 95
extern int nsample;
extern float tsample;
extern int npuntos;
extern int bits;
extern int inportico,outportico;
extern float ylogicamin,ylogicamax;
extern float referencia;
extern float honoff;
extern float vonoffmin,vonoffmax;


//---------- TOnoffWindow ----------------------------------------

TOnoffWindow::TOnoffWindow( PTWindowsObject AParent, LPSTR ATitle ) :
                                                        TBaseDemoWindow( AParent,
ATitle )
{
        StaticControl = new TStatic(this, 100,
                "Pausa: Botón Izquierdo  Reiniciar: Botón Derecho",10,10,10,10,0);
        Iconizado = False;
        AltoTexto = 20;
        Pausa = FALSE;

        Ansample=nsample;
        Atsample=tsample;
        Anpuntos=npuntos;
        Anpuntosmax=npuntos;
        Anpuntosicon=5;
        contador=0;
        punto=0;
        Abits=bits;
        Areferencia=referencia;
        Ahonoff=honoff;
        Avonoffmin=vonoffmin;
        Avonoffmax=vonoffmax;
        Ainportico=inportico;
        Aoutportico=outportico;
        Aylogicamin=ylogicamin;
        Aylogicamax=ylogicamax;
        xlogicamin=0;
        xlogicamax=Ansample*Atsample*Anpuntos;
        pendiente=(Aylogicamax-Aylogicamin)/(pow(2,Abits)-1);
};
```

```
/* Dispose of the objects that this window object created.  There's
        no need to dispose the List pointer, since it will only point to
        one of these two objects which are being disposed by their
        primary pointers */

void TOnoffWindow::GetWindowClass( WNDCLASS& WndClass )
{
        TBaseDemoWindow::GetWindowClass( WndClass );
        WndClass.hCursor = LoadCursor(0,IDC_CROSS);
        WndClass.hbrBackground = GetStockObject(WHITE_BRUSH);
        WndClass.hIcon = 0;  // we'll paint our own icon when minimized, thank you.
};

/* When the window is resized, scale the line list to fit the new
        window extent, or switch between full size and iconized window
        states.  */
void TOnoffWindow::WMSize( TMessage& Message )
{

        TBaseDemoWindow::WMSize(Message);

        /* Force Windows to repaint the entire window region */
        InvalidateRect(HWindow, NULL, TRUE);

        xfisicamax = Message.LP.Lo;
        yfisicamax = Message.LP.Hi;

        if (IsIconic(HWindow))
        {
                if (!Iconizado)
                {
                        Iconizado = TRUE;
                        contador=0;
                        punto=0;
                        Anpuntos=Anpuntosicon;
                        //List = IconicLineList;
                        escalax=.8*xfisicamax/Anpuntos;
                        escalay=.8*yfisicamax/(pow(2,Abits)-1);
                }
        }
        else
        {
                if (Iconizado)
                {
                        Iconizado = FALSE;
                        //List = BigLineList;
                };
                Anpuntos=Anpuntosmax;
                escalax=.8*xfisicamax/Anpuntos;
                yfisicamax -= AltoTexto; /* allow room for the text at the bottom */
                escalay=.8*yfisicamax/(pow(2,Abits)-1);
                PantallaDC=GetDC(HWindow);
                DibujaCuadro(PantallaDC);
                ReleaseDC(HWindow,PantallaDC);
        }
        //      List->ScaleTo(NewXmax, NewYmax);  /* scale the lines in the list */
        if (StaticControl)
                MoveWindow(StaticControl->HWindow, 0, yfisicamax, xfisicamax, AltoTexto, True);
```

```
};

/* Toggle the window's Paused status.  Since the window will
        not receive mouse clicks when iconized, this will not pause the
        iconized lines display.  */
void TOnoffWindow::WMLButtonDown( TMessage& Mensaje)
{
        if(Pausa)
        {
                if (Mensaje.WParam & MK_CONTROL)
                {
                        if( Mensaje.LP.Lo<.098*xfisicamax || Mensaje.LP.Lo>.902*xfisicamax ||
                                Mensaje.LP.Hi<.098*yfisicamax || Mensaje.LP.Hi>.902*yfisicamax )
                                return;

                        float x,y;
                        char aux1[30];

                        PantallaDC=GetDC(HWindow);
                        Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,1,RGB(255,0,255)) );
                        MoveTo(PantallaDC,.1*xfisicamax,Mensaje.LP.Hi);
                        LineTo(PantallaDC,Mensaje.LP.Lo,Mensaje.LP.Hi);
                        LineTo(PantallaDC,Mensaje.LP.Lo,.9*yfisicamax);

                        x=xlogicamin+(xlogicamax-xlogicamin)*(Mensaje.LP.Lo-
.1*xfisicamax)/(.8*xfisicamax);
                        y=Aylogicamin+(Aylogicamax-Aylogicamin)*(.9*yfisicamax-
Mensaje.LP.Hi)/(.8*yfisicamax);

                        sprintf(aux1,"%3.4g",x);
                        TextOut( PantallaDC,Mensaje.LP.Lo-3.5*strlen(aux1),yfisicamax-
20,aux1,strlen(aux1) );
                        sprintf(aux1,"%3.4g",y);
                        TextOut( PantallaDC,1,Mensaje.LP.Hi-10,aux1,strlen(aux1) );
                        DeleteObject(SelectObject(PantallaDC,Lapiz));
                        ReleaseDC(HWindow,PantallaDC);
                        return;
                }
        }
        Pausa = !Pausa;
};

/* Clear the line list when the user presses the right mouse
        button.  Same comments as above on iconized windows.  */
void TOnoffWindow::WMRButtonDown( TMessage& )
{
        if(!Pausa)
                Pausa=!Pausa;
        contador=0;
        punto=0;
        InvalidateRect(HWindow, NULL, TRUE);
        Pausa=!Pausa;
//      List->ResetLines();
};

void TOnoffWindow::WMMouseMove(TMessage& Mensaje)
{
        if ( !Pausa )
```

```
                return;
        if( Mensaje.LP.Lo<.098*xfisicamax || Mensaje.LP.Lo>.902*xfisicamax ||
                        Mensaje.LP.Hi<.098*yfisicamax || Mensaje.LP.Hi>.902*yfisicamax )
                return;

        float x,y;
        char aux1[40];

        x=xlogicamin+(xlogicamax-xlogicamin)*(Mensaje.LP.Lo-.1*xfisicamax)/(.8*xfisicamax);
        y=Aylogicamin+(Aylogicamax-Aylogicamin)*(.9*yfisicamax-Mensaje.LP.Hi)/(.8*yfisicamax);

        sprintf(mensaje,"(%3.4g,",x);
        sprintf(aux1,"%3.4g",y);
        strcat(mensaje,aux1);
        strcat(mensaje,")          ");

        PantallaDC=GetDC(HWindow);
        TextOut(PantallaDC,1,1,mensaje,strlen(mensaje));
        ReleaseDC(HWindow,PantallaDC);
};

/* When the window is resized, or some other window blots out part
        of our client area, redraw the entire line list.  The PaintDC
        is fetched before Paint is called and is released for us after
        Paint is finished. */
void TOnoffWindow::Paint( HDC PaintDC, PAINTSTRUCT& PaintInfo)
{
        TBaseDemoWindow::Paint(PaintDC, PaintInfo);

        DibujaCuadro(PaintDC);

//      List->Redraw(PaintDC);
};

void TOnoffWindow::DibujaCuadro(HDC PantallaDC)
{
        float y;

        Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,1,RGB(0,0,0)) );

        MoveTo(PantallaDC,.1*xfisicamax,.1*yfisicamax);
        LineTo(PantallaDC,.1*xfisicamax,.9*yfisicamax);
        LineTo(PantallaDC,.9*xfisicamax,.9*yfisicamax);
        LineTo(PantallaDC,.9*xfisicamax,.1*yfisicamax);
        LineTo(PantallaDC,.1*xfisicamax,.1*yfisicamax);

        //trazado de ejes
        //eje x
        if( Aylogicamin*Aylogicamax<0 )
        {
                y=.1*yfisicamax+.8*yfisicamax*(Aylogicamax-0)/(Aylogicamax-Aylogicamin);
                MoveTo(PantallaDC,.1*xfisicamax,y);
                LineTo(PantallaDC,.9*xfisicamax,y);
                sprintf(mensaje," 0");
                TextOut(PantallaDC,1,y-10,mensaje,strlen(mensaje));
        }
        y=.1*yfisicamax+.8*yfisicamax*(ylogicamax-referencia)/(ylogicamax-ylogicamin);
        MoveTo(PantallaDC,.1*xfisicamax,y);
```

```
        LineTo(PantallaDC,.9*xfisicamax,y);
        sprintf(mensaje,"%3.5g",referencia);
        TextOut( PantallaDC,1,y-10,mensaje,strlen(mensaje) );
        DeleteObject(SelectObject(PantallaDC,Lapiz));
        //limites
        //eje y
        sprintf(mensaje,"%3.5g",Aylogicamin);
        TextOut(PantallaDC,1,.9*yfisicamax-10,mensaje,strlen(mensaje));
        sprintf(mensaje,"%3.5g",Aylogicamax);
        TextOut(PantallaDC,1,.1*yfisicamax-10,mensaje,strlen(mensaje));
        //eje x
        sprintf(mensaje," 0");
        TextOut(PantallaDC,.1*xfisicamax-3.5*strlen(mensaje),yfisicamax-
20,mensaje,strlen(mensaje));
        sprintf(mensaje,"%3.5g",Ansample*Atsample*Anpuntos);
        TextOut(PantallaDC,.9*xfisicamax-3.5*strlen(mensaje),yfisicamax-
20,mensaje,strlen(mensaje));
        //cuadro superior reloj

};

void TOnoffWindow::Onoff(HDC PantallaDC)
{
    int i;
    float x,y,error;

    x=punto*escalax+.1*xfisicamax;
    y=inp(Ainportico);
    y1[punto]=.9*yfisicamax-y*escalay;

    Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,2,RGB(0,0,255)) );
    MoveTo(PantallaDC,x,y1[punto]);
    LineTo(PantallaDC,x,y1[punto]);
    DeleteObject(SelectObject(PantallaDC,Lapiz));

    error=Areferencia-(y*pendiente+Aylogicamin);

    if( error>Ahonoff )
        y=outp(Aoutportico,(Avonoffmax-Aylogicamin)/pendiente);
    if( error<-Ahonoff )
        y=outp(Aoutportico,(Avonoffmin-Aylogicamin)/pendiente);

    y2[punto]=.9*yfisicamax-y*escalay;
    Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,2,RGB(255,0,0)) );
    MoveTo(PantallaDC,x,y2[punto]);
    LineTo(PantallaDC,x,y2[punto]);
    DeleteObject(SelectObject(PantallaDC,Lapiz));

    punto++;
    if( punto>Anpuntos )
    {
        Lapiz=SelectObject( PantallaDC, CreatePen(PS_SOLID,2,RGB(255,255,255)) );
        for(i=0;i<=Anpuntos;i++ )
        {
            x=i*escalax+.1*xfisicamax;
            MoveTo(PantallaDC,x,y1[i]);
            LineTo(PantallaDC,x,y1[i]);
            MoveTo(PantallaDC,x,y2[i]);
```

```
                        LineTo(PantallaDC,x,y2[i]);
                }
                DeleteObject(SelectObject(PantallaDC,Lapiz));
                punto=0;
                DibujaCuadro(PantallaDC);
        }
}

void TOnoffWindow::EscribirTiempo(HDC PantallaDC)
{
        char s[ 21 ];

        //mostrar reloj
        sprintf( s, "%3.3f",contador*Atsample );
        TextOut( PantallaDC, xfisicamax-36, 4, s, 4);
}

/* Fetch a device context, pass it to the line list object, then
        release the device context back to Windows. */
void TOnoffWindow::TimerTick()
{
        //HDC DC;

        if (!Pausa)
        {
                if( fmod(contador,Ansample)==0 || contador==0 )
                {
                        PantallaDC=GetDC(HWindow);
                        if( contador==0 )
                                DibujaCuadro(PantallaDC);
                        Onoff(PantallaDC);
                        EscribirTiempo(PantallaDC);
                        ReleaseDC(HWindow,PantallaDC);
                }
                contador++;
        }
};
```

```
// ARCHIVO: PID.CPP

#include <math.h>
#include <iostream.h>

/*pid() obtiene el polinomio normalizado en modo an logo o digital a partir
        de los coeficientes del controlador PID
        s:      kp+ki/s+kd*s
        z:      kp+(ki*T/2)(z+1)/(z-1)+(kd/T)(z-1)/z
        modo    = modo digital 'z' o an logo 's'
        tsample = tiempo de muestreo en el modo digital
        kp,ki,kd = constantes del controlador PID
        k                                       = ganancia del polinomio resultante
        nrn(nrd) = numero de raices del numerador (denominador) resultante
        ren(red) = parte real de las raices del numerador (denominador)
        imn(imd) = parte imaginaria de las raices del numerador (denominador)
        rn(rd)  = multiplicidad de las raices del numerador (denominador)
*/
void pid(char modo,float tsample,float kp,float ki,float kd,float *k,int *nrn,float ren[],float imn[],int
rn[],int *nrd,float red[],float imd[],int rd[])
{
        int gn;
        float num[3];
        float det;

        (*nrn)=(*nrd)=0;
        (*k)=1.;
        if( kp==0 && ki==0 && kd==0 )
        {
                cout<<"\nOBSERVACION: se puede producir un error";
                return;
        }
        gn=2;
        red[0]=imd[0]=0.;rd[0]=1;
        if( modo=='s' )
        {
                num[0]=kd;
                num[1]=kp;
                num[2]=ki;
                (*nrd)=1;
                if( ki==0)
                {
                        gn--;
                        (*nrd)--;
                        if( kd==0 )
                        {
                                gn--;
                                num[0]=kp;
                        }
                }
                else
                {
                        if( kd==0 )
                        {
                                gn--;
                                num[0]=kp;
                                num[1]=ki;
                                if( kp==0 )

        {
                if( num[0]==0 ).
                {
                        cout<<"\nERROR:pid.cpp";
                        return;
                }
                num[1]/=2*num[0];
                num[2]/=num[0];
                det=num[1]*num[1]-num[2];
                if( det>0 )
```

```
                {
                        (*nrn)=2;
                        imn[0]=imn[1]=0;
                        ren[0]=-num[1]+sqrt(det);
                        ren[1]=-ren[0]-2*num[1];
                        rn[0]=rn[1]=1;
                }
                else
                {
                        (*nrn)=1;
                        ren[0]=-num[1];
                        if( det==0 )
                        {
                                rn[0]=2;
                                imn[0]=0;
                        }
                        else
                        {
                                rn[0]=1;
                                imn[0]=sqrt(-det);
                        }
                }
        }
}
```

```
// ARCHIVO: POLCOMPL.CPP

/*
  DERIVADA, SUMA, MULTIPLICACION, POTENCIACION DE POLINOMIOS
  POLINOMIOS CON COEFICIENTES COMPLEJOS!
  RUTINAS SIMILARES A LAS QUE ESTAN EN MULTIP2.CPP
*/

#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <complex.h>

#include "datos.h"

// multiplicacion de polinomios (descendente) reales
// asegurarse que los coeficientes[0] sean <> 0
// a[] (grado na) * b[] (grado nb) = c[] (grado nc]

void multiplica2cmp(int na, int nb, int *nc, complex a[], complex b[], complex c[])
{
        int i,j,contador;

//  if ( norm(a[0]) == 0 || norm(b[0]) == 0 )
//  {
//    cout<<"\n\nADVERTENCIA.puede ocasionar errores:\nMULTIPLICA2 coeficientes = cero \n";
//  }

        //asegurar primer coeficiente diferente de cero (optimizar)
        contador=0;
        while( norm(a[0])==0 && na>=1 )
        {
                na--;
                contador++;
                a[0] = a[contador];
        }

        if(contador > 0 )
        {
                for(i = 0;i<= na;i++)
                        a[i] = a[i + contador];
        }

        //asegurar primer coeficiente diferente de cero
        contador=0;
        while( norm(b[0])==0 && nb>=1 )
        {
                nb--;
                contador++;
                b[0] = b[contador];
        }

        if(contador > 0 )
        {
                for(i = 0;i<= nb;i++)
                        b[i] = b[i + contador];
        }
```

```
        *nc=na+nb;
        for(i=0;i<=na+nb;i++) c[i]=0;
        for(i=0;i<=na;i++)
        {
                for(j=0;j<=nb;j++)
                {
                        c[i+j]=c[i+j]+a[i]*b[j];
                }
        }
}


/* POTENCIACION DE POLINOMIOS  (COMPLEJO)
        eleva un binomio complejo a[] a una potencia r
        (a0*s+a1)^r=b[] (grado nb)
        funciona para r<=15 (optimamente?)
*/

void potenciacmp(int r, int *nb, complex a[], complex b[])
{
        int i;
        complex aux1,aux2;

        if ( norm(a[0]) == 0 && norm(a[1]) == 0 )
        {
                cout<<"\n\nERROR. POTENCIA.CPP:coeficientes = cero, podria generar error";
                exit(1);
        }


        if( norm(a[0]) == 0 )
        {
                *nb=0;
                //cout<<"\ngrado en sub a[0]=0  "<<*nb;
                b[0]=pow(a[1],r);
                return;
        }
        if( norm(a[1])==0 )
        {
                *nb=r;
                //cout<<"\ngrado en sub a[1]=0 "<<*nb;
                b[0]=pow(a[0],r);
                for(i=1;i<=r;i++) b[i]=0;
        }
        *nb=r;
        b[0]=1;
        b[r]=1;
        switch(r)
        {
                case 0:
                        break;
                case 1:
                        b[0]=a[0];b[1]=a[1];
                        break;
                case 2:
                        b[1]=2;
```

```
                    break;
                case 3:
                    b[1]=3;b[2]=3;
                    break;
                case 4:
                    b[1]=4;b[2]=6;b[3]=4;
                    break;
                case 5:
                    b[1]=5;b[2]=10;b[3]=10;b[4]=5;
                    break;
                case 6:
                    b[1]=6;b[2]=15;b[3]=20;b[4]=15;b[5]=6;
                    break;
                case 7:
                    b[1]=7;b[2]=21;b[3]=35;b[4]=35;b[5]=21;b[6]=7;
                    break;
                case 8:
                    b[1]=8;b[2]=28;b[3]=56;b[4]=70;b[5]=56;b[6]=28;b[7]=8;
                    break;
                case 9:
                    b[1]=9;b[2]=36;b[3]=84;b[4]=126;b[5]=126;b[6]=84;b[7]=36;b[8]=9;
                    break;
                case 10:

        b[1]=10;b[2]=45;b[3]=120;b[4]=210;b[5]=252;b[6]=210;b[7]=120;b[8]=45;b[9]=10;
                    break;
                case 11:

        b[1]=11;b[2]=55;b[3]=165;b[4]=330;b[5]=462;b[6]=462;b[7]=330;b[8]=165;b[9]=55;b[10]=11;
                    break;
                case 12:

        b[1]=12;b[2]=66;b[3]=220;b[4]=495;b[5]=792;b[6]=924;b[7]=792;b[8]=495;b[9]=220;b[10]=6
6;b[11]=12;
                    break;
                case 13:

        b[1]=13;b[2]=78;b[3]=286;b[4]=715;b[5]=1287;b[6]=1716;b[7]=1716;b[8]=1287;b[9]=715;b[1
0]=286;b[11]=78;b[12]=13;
                    break;
                case 14:

        b[1]=14;b[2]=91;b[3]=364;b[4]=1001;b[5]=2002;b[6]=3003;b[7]=3432;b[8]=3003;b[9]=2002;b
[10]=1001;b[11]=364;b[12]=91;b[13]=14;
                    break;
                case 15:

        b[1]=15;b[2]=105;b[3]=455;b[4]=1365;b[5]=3003;b[6]=5005;b[7]=6435;b[8]=6435;b[9]=5005;
b[10]=3003;b[11]=1365;b[12]=455;b[13]=105;b[14]=15;
                    break;
                default:
                    cout<<"ERROR: se exede el grado (15) del polinomio";
                    exit(1);
        }
        if( r>=2 )
        {
                aux1=1;
                aux2=pow(a[0],r);
```



```
                for(i=0;i<=r;i++)
                {
                    b[i] *= aux1*aux2;
                    aux1 *= a[1]/a[0];
                }
        }
}
```

```
                for(i=0;i<=r;i++)
                {
                        b[i] *= aux1*aux2;
                        aux1 *= a[1]/a[0];
                }
        }
}
```

```
// ARCHIVO: RAICES.CPP

//Raices .- para encontar raices reales y complejas de p[] de grado n
//          asegurarse en rutina de llamada n>=1

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream.h>

#include "datos.h"    // se incluye maximo numero de iteraciones
                                                                    // y el
error

void segundo3(double u0,double v0,int *numraiz, float re1[],float im1[]);

void raices(int n, float polinomio[],int *numraiz,float re[],float im[])
{
        double a[maximo],aux1[7],q1[maximo]={0}, q2[maximo]={0};
        double real, imag;
        float re2[2],im2[2];
        double u0,v0,u1,v1;
        int  i,n_r,n1 ;   //int salir;
        int contador,niter,nraiz,nraices;
        double q3,q4,q5,q6,q7,q8;
        double p3,p4,p5,p6;

        for(i=0;i<=n;i++) a[i]=polinomio[i];

        if(n==0)                    //asegura que el grado no sea cero
        {
                cout<<"\nERROR: Raices.cpp: grado polinomio=0";
                (*numraiz)=0;
                return;
        }

        if(a[0]==0)                             //asegura que el primer coeficiente no sea cero
        {                                       // puede que no sea necesario, mas bien realizarlo en
                cout<<"\nERROR: Raices.cpp: coeficiente inicial 0";// en el programa principal
                exit(1);
        }

        n_r = 0;
        while(a[n]==0)          //encuentra raices iguales a cero
        {                   //debido a terminos independiente = cero
                re[n_r]=0;
                im[n_r]=0;
                n--;
                n_r++;

        }

        if (n==0)
        {
                (*numraiz)=n_r;
                return;
        }
```

```
// salir = 0;  // bandera para verificar lazo
        switch (n)
        {
                case 1:
                        re[n_r] = -a[1] / a[0];
                        im[n_r] = 0;
                        n_r++;
        break;

case 2:
 u0 = -a[1] / a[0];
        v0 = -a[2] / a[0];
 segundo3(u0,v0,&n1,re2, im2);
        re[n_r]=re2[0];
        im[n_r]=im2[0];
                        n_r++;
        if( n1==2 )
        {
         re[n_r]=re2[1];
         im[n_r]=im2[1];
         n_r++;
        }
        break;

default:

        for(i=0;i<=n;i++) q1[n + 1 - i]=a[i];
                        //cout<<"\nEstoy buscando las raices";


        aux1[1] = n;
        aux1[2] = n + 1;
        aux1[3] = 0;

        nraiz = n;

        for( i = 1;i<= n + 1;i++) q2[n + 2 - i] = q1[i];

NN:
        u0 = .00500101;
        v0 = .010000101;
        contador = 0;
GG:
        u1 = u0;
        u0 = -10 * v0;
        v0 = -10 * u1;
        u1 = u0;
        v1 = v0;
        contador += 1;
        goto BB;
HH:
        aux1[3] = 1;
                        aux1[5] = u1;
        aux1[6] = v1;
BB:
        niter = 0;
FF:
```

```
            p3 = 0;
            p4 = 0;
            p6 = 0;
            q3 = 1;
            q4 = 0;
            p5 = q2[nraiz + 1];
            if( p5 == 0 ) goto CC;
            for(i = 1; i<= nraiz;i++)
            {
            q8 = q2[nraiz+1-i];
            q5 = u1 * q3 - v1 * q4;
            q6 = u1 * q4 + v1 * q3;
            p5 += q8 * q5;
            p6 += q8 * q6;
                                        p3 += i * q3 * q8;
            p4 += - i * q4 * q8;
            q3 = q5;
            q4 = q6;
            }
            aux1[4] = p3 * p3 + p4 * p4;
            if( aux1[4] == 0 ) goto EE;
            real = (p6 * p4 - p5 * p3) / aux1[4];
            u1 += real;
            imag = -(p5 * p4 + p6 * p3) / aux1[4];
            v1 += imag;
            if( ( fabs(real) + fabs(imag) ) < epsilon ) goto DD;
          niter++;
//          cout<<"\niteracion"<<niter;
            if( niter < max_iter ) goto FF;
            if( aux1[3] != 0 ) goto DD;
            if( contador < 5 ) goto GG;
             cout<<"\nERROR: RAICES.CPP NO HAY RAICES PARA 500 INTERACCIONES Y 5
VALORES";
      return;
                DD:
            for(i = 1;i<=aux1[2];i++)
            {
            q8 = q1[n + 2 - i];
            q1[n + 2 - i] = q2[i];
              q2[i] = q8;
            }
            q7 = nraiz;
            nraiz = aux1[1];
            aux1[1] = q7;
            if( aux1[3]==0 ) goto HH;
            goto II;
  EE:
            if( aux1[3]==0 ) goto GG;
            u1 = aux1[5];
            v1 = aux1[6];
  II:
            aux1[3] = 0;
            if( fabs(v1) < 10 * epsilon * fabs(u1) ) goto JJ;
                                    aux1[0] = u1 + u1;
            aux1[4] = u1 * u1 + v1 * v1;
            nraiz += - 2;
            goto KK;
  CC:
```

```
            u1 = 0;
            aux1[1] = aux1[1] - 1;
            aux1[2] = aux1[2] - 1;
    JJ:
            v1 = 0;
            aux1[4] = 0;
            aux1[0] = u1;
            nraiz--;
    KK:
            q2[2] = q2[2] + aux1[0] * q2[1];
            q7 = aux1[0];
            q8 = aux1[4];
            for(i=2;i<=nraiz;i++) q2[i + 1] +=  q7 * q2[i] - q8 * q2[i - 1];

                                    if( fabs(u1)<10*epsilon*fabs(v1) ) u1=0;
                                    re[n_r] = u1;
                                    if( v1 < 0 ) v1 = -v1;
                                    im[n_r] = v1;
                                    n_r ++;

                                    if( nraiz > 0 ) goto NN;

                                    break;
                }
        *numraiz=n_r;
}

// subrutina para resolver una ecuacion de segundo grado x^2 - ux -v
// en la bandera se indica si las raices son reales o imaginarias
// si son reales x1 = primera raiz, x2 = segunda raiz
// si son complejos x1=     parte real, x2 = parte imaginaria


void segundo3(double u0,double v0,int *numraiz, float re1[], float im1[])
{
        float det;

        det = u0 * u0 + 4 * v0;
        if(det >= 0)
        {
                *numraiz=2;
                re1[0] = (u0 + pow(det, .5)) / 2;
                re1[1] = (u0 - pow(det, .5)) / 2;
                im1[0] = 0;
                im1[1]=0;
        }
        else
        {
                *numraiz=1;
                det = -det;
                re1[0] = u0/2;
                im1[0] = pow(det,.5)/2;
        }
}
```

```
// ARCHIVO: REDUCIR.CPP

#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>

#include "datos.h"

/* a la subrutina multiplicidad ingresa el polinomio factorado
   en forma de raices. y ademas su multiplicidad >=1)
   y entonces encuentra raices repetidas (con cierta tolerancia)
   y entrega el polinomio en forma de raices y con su respectiva
   multiplicidad
   como en otras subrutinas las raices complejas no se toman en cuenta
   las raices conjugadas, para evitar errores

   *nraices = numero de raices inicial y final del polinomio
   r[]  = multiplicidad inicial y final de cada raiz
   re[] = parte real de cada raiz
   im[] = parte imaginaria de cada raiz
*/

void multiplicidad(int *nraices,int r[],float re[],float im[])
{
 int i,j;
 float diferencia1,diferencia2,mayor;

        if( *nraices>= 2 )
        {
                for(i=0;i<=(*nraices)-2;i++)
                {
                        for(j=i+1;j<=(*nraices)-1;j++)
                        {
                                diferencia1 = fabs(re[i]-re[j]);
                                diferencia2 = fabs(im[i]-im[j]);

                                mayor = fabs(re[i]);
                                if( fabs(im[i])>mayor ) mayor = fabs(im[i]);
                                if( fabs(re[j])>mayor ) mayor = fabs(re[j]);
                                if( fabs(im[j])>mayor ) mayor = fabs(im[j]);

                                if( mayor==0 ) mayor = 1;

                                diferencia1 = diferencia1/mayor;
                                diferencia2 = diferencia2/mayor;

                                if( diferencia1<=presicion && diferencia2<=presicion )
                                {
                                        r[i]=r[i]+r[j];
                                        (*nraices)--;
                                        if( j<(*nraices) )
                                        {
                                                re[j] = re[(*nraices)];
                                                im[j] = im[(*nraices)];
                                                r[j]  = r[(*nraices)];
                                                j--;                    //fea forma de programar, afecto
a lazos for
```

```
                                        }
                                    }
                                }
                            }
                        }
}

/* la funcion reducir encuentra raices repetidas en el numerador y denominador
        de la funcion de transferencia. y las reduce
        *nrnum  = numero de raices inicial y final del numerador
        *nrden  = nuemro de raices inicial y final del denominador
        rnum[]  = multiplicidad  inicial y final de cada raiz del numerador
        rnum[]  = multiplicidad  inicial y final de cada raiz del denominador
        renum[] = parte real de cada raiz del numerador
        imnum[] = parte imaginaria de cada raiz del numerador
        reden[] = parte real de cada raiz del numerador
        imden[] = parte imaginaria de cada raiz del numerador
*/

void reducir(int *nrnum,int *nrden,int rnum[],int rden[],float renum[],float imnum[],float reden[],float
imden[])
{
        int i,j;
        float diferencia1,diferencia2,mayor;

        if( (*nrnum)>=1 && (*nrden)>=1 )
        {
                for(i=0;i<=(*nrnum)-1;i++)
                {
                        for(j=0;j<=(*nrden)-1;j++)
                        {
                                diferencia1 = fabs(renum[i]-reden[j]);
                                diferencia2 = fabs(imnum[i]-imden[j]);

                                mayor = fabs(renum[i]);
                                if( fabs(imnum[i])>mayor ) mayor = fabs(imnum[i]);
                                if( fabs(reden[j])>mayor ) mayor = fabs(reden[j]);
                                if( fabs(imden[j])>mayor ) mayor = fabs(imden[j]);

                                if( mayor==0 ) mayor = 1;

                                diferencia1 = diferencia1/mayor;
                                diferencia2 = diferencia2/mayor;

                                if( diferencia1<=presicion && diferencia2<=presicion )
                                {
                                        if( rnum[i]>rden[j])
                                        {
                                                rnum[i] += -rden[j];
                                                (*nrden)--;
                                                if( j<(*nrden) )
                                                {
                                                        reden[j] = reden[(*nrden)];
                                                        imden[j] = imden[(*nrden)];
                                                        rden[j]  = rden[(*nrden)];
                                                        j--;                    //fea forma de
programar, afecto a lazos for
```

```
                }
            }
            else
                if( rnum[i]<rden[j])
                {
                    rden[j] += -rnum[i];
                    (*nrnum)--;
                    if( i<(*nrnum) )
                    {
                        renum[i] = renum[(*nrnum)];
                        imnum[i] = imnum[(*nrnum)];
                        rnum[i]  = rnum[(*nrnum)];
                        i--;
                        break;
                    }
                }
                else   // rn=rd
                {
                    (*nrnum)--;
                    (*nrden)--;
                    if( j<(*nrden) )
                    {
                        reden[j] = reden[(*nrden)];
                        imden[j] = imden[(*nrden)];
                        rden[j]  = rden[(*nrden)];
                        j--;                    //fea forma de
programar. afecto a lazos for

                    }
                    if( i<(*nrnum) )
                    {
                        renum[i] = renum[(*nrnum)];
                        imnum[i] = imnum[(*nrnum)];
                        rnum[i]  = rnum[(*nrnum)];
                        i--;
                        break;
                    }
                }
            }
        }
    }
}□
```