

```

/*****
/*          L V D E M O . C          */
/*-----*/
/* Task      : Demonstration of use of ListView      */
/*-----*/
/* Author      : Michael Tischer and Bruno Jennrich  */
/* developed on : 10/12/1995                          */
/* last update  : 10/20/1995                          */
/*****
#include <windows.h>
#include <commctrl.h>
#include <shellapi.h>
#include <stdio.h>

#include "resource.h"

#include "paintpic.h"

//----- Typedefs -----
typedef struct tagSTOCKSTRUCT
{
    char pszStock[ 20 ];
    float fForeday;
    float fToday;
} STOCKSTRUCT;
typedef STOCKSTRUCT *PSTOCKSTRUCT;

//----- Global Variables -----
HIMAGELIST g_hImagesNormal,           // normal images
            g_hImagesSmall,           // small images
            g_hImagesState;           // state images

FARPROC g_lpOldEditProc; // Address of original EditBox window function

STOCKSTRUCT g_Stocks[10];              // Array for stocks
int          g_iUsedStocks = 0;         // Position of next free stock

/*****
/* EditSubclass - Subclass function for EditControl of a ListView      */
/*-----*/
/* Parameter : default parameters      */
/* Return value: default return value  */
/*-----*/
/* Info: The EditControl requires all keyboard input, since ESC and      */
/*       RETURN will otherwise be interpreted as cancelling the          */
/*       dialog box or pressing the default button.                      */
/*****
LRESULT CALLBACK EditSubclass( HWND    hWnd,
                              UINT     wParam,
                              WPARAM   wp,
                              LPARAM   lp )
{
    // I want all keystrokes -----
    if( wParam == WM_GETDLGCODE ) return DLGC_WANTALLKEYS;

    // Call original EditBox function -----
    return CallWindowProc( g_lpOldEditProc, hWnd, wParam, wp, lp );
}

/*****
/* SetDropHighlight - Identify new item as DropTarget      */
/*-----*/
/* Parameter :      hWnd - Handle of ListView control      */
/*              iNewDropTarget - Index of item to be highlighted */
/*              iOldDropTarget - Index of previously highlighted */
/*              item.                                          */
/* Return value: none */
/*****
void SetDropHighlight( HWND hWnd,
                      int iNewDropTarget,
                      int iOldDropTarget )
{
    if( iNewDropTarget != iOldDropTarget ) // Has anything changed?
    {
        if( iNewDropTarget != -1 )
        {

```

```

        ListView_SetItemState( hListView,
                               iNewDropTarget,
                               LVIS_DROPHILITED,
                               LVIS_DROPHILITED );           // Set highlight
// Prepare redraw of item -----
        ListView_Update(hListView, iNewDropTarget);
    }

    if( iOldDropTarget != -1 )
    {
        ListView_SetItemState( hListView,
                               iOldDropTarget,
                               0,
                               LVIS_DROPHILITED );           // Remove highlight
// Prepare redraw of item -----
        ListView_Update(hListView, iOldDropTarget);
    }
    UpdateWindow( hListView );           // Redraw changes immediately
}
}

```

```

/*****
/* InsertItem - Inserts item or subitem into ListView */
/*-----*/
/* Parameter : hListView - Handle of ListView window */
/*              lpText    - Address of text to be inserted. If */
/*                          lpText == 0 LVN_GETDISPINFO gets */
/*                          the text */
/*              iItem      - Index of item to which a subitem is to */
/*                          be added. (is ignored if */
/*                          SubItem==0) */
/*              iSubItem   - Index of subitem to be set. If */
/*                          iSubItem = 0 a new main item is */
/*                          created. */
/*              iImage     - Index of image to be used (can also be */
/*                          I_IMAGECALLBACK) */
/*              lParam     - User data to be linked to main item */
/* Return value: Index of new item or TRUE/FALSE as success message */
/*              when setting a subitem (iSubItem!=0) */
/*-----*/
/* Note:      SubItems will only be set properly if the */
/*            Column Header of ListView is set before */
/*            inserting new items. */
*****/

```

```

int InsertItem( HWND    hListView,
                LPSTR   lpText,
                int     iItem,
                int     iSubItem,
                int     iImage,
                LPARAM  lParam )
{
    LV_ITEM lvi;

    if( iSubItem )           // Change subitem of an existing item?
    {
        lvi.mask      = LVIF_TEXT;           // pszText and cchMaxText are valid
        lvi.iItem      = iItem;
        lvi.iSubItem   = iSubItem;
        lvi.pszText    = lpText ? lpText : LPSTR_TEXTCALLBACK;
        lvi.cchTextMax = lpText ? strlen( lpText ) : 0;

        return ListView_SetItem( hListView, &lvi );           // Change subitem
    }
    else
    {
        // All fields of the LV_ITEM structure are valid -----
        lvi.mask      = LVIF_TEXT | LVIF_IMAGE | LVIF_PARAM | LVIF_STATE;
        lvi.iItem      = 0;
        lvi.iSubItem   = 0;                     // generate new main item
        lvi.state      = 0;
        lvi.stateMask  = 0;
        lvi.pszText    = lpText ? lpText : LPSTR_TEXTCALLBACK;
        lvi.cchTextMax = lpText ? strlen( lpText ) : 0;
        lvi.iImage     = iImage;
        lvi.lParam     = lParam;
    }
}

```

```

    return ListView_InsertItem(hListView, &lvi);          // create item
}
}

/*****
/* ADD_STOCK - Function for adding stock */
/*-----*/
/* Parameter : hListView - Handle of ListView window */
/*             pszStock   - Name of stock */
/*             fForeday   - Yesterday's price */
/*             fToday     - Current price */
/*             iPic       - Index of picture */
/* Return value: none */
/*****
void ADD_STOCK( HWND hListView,
               LPSTR pszStock,
               float fForeday,
               float fToday,
               int iPic )
{
    int iIndex;

    // Transfer stock data to structure -----
    lstrcpy( g_Stocks[g_iUsedStocks].pszStock, pszStock );
    g_Stocks[g_iUsedStocks].fForeday = fForeday;
    g_Stocks[g_iUsedStocks].fToday = fToday;

    // Insert item and subitems with TEXTCALLBACK. The structure -----
    // address for each item (subitem) is passed as lParam -----
    iIndex = InsertItem( hListView,
                        LPSTR_TEXTCALLBACK,
                        0,
                        0,
                        iPic,
                        ( LPARAM ) &g_Stocks[g_iUsedStocks] );
    InsertItem( hListView, LPSTR_TEXTCALLBACK, iIndex, 1, 0, 0 );
    InsertItem( hListView, LPSTR_TEXTCALLBACK, iIndex, 2, 0, 0 );
    InsertItem( hListView, LPSTR_TEXTCALLBACK, iIndex, 3, 0, 0 );
    g_iUsedStocks++;

    // Adapt state image to market trend -----
    if( fToday > fForeday )                      // Trend: rising
        ListView_SetItemState( hListView,
                               iIndex,
                               INDEXTOSTATEIMAGEMASK( 1 ),
                               LVIS_STATEIMAGEMASK );

    if( fToday < fForeday )                      // Trend: falling
        ListView_SetItemState( hListView,
                               iIndex,
                               INDEXTOSTATEIMAGEMASK( 2 ),
                               LVIS_STATEIMAGEMASK );

    if( fToday == fForeday )                    // Trend: constant
        ListView_SetItemState( hListView,
                               iIndex,
                               INDEXTOSTATEIMAGEMASK( 3 ),
                               LVIS_STATEIMAGEMASK );
}

/*****
/* AddColumn - Adds new column to Column Header */
/*-----*/
/* Parameter : hListView - Handle of ListView window */
/*             iCol       - Number of column */
/*             lpText     - Column heading */
/*             iFmt       - Column alignment */
/*             iWidth     - Width of column in pixels */
/*             iSubItem   - SubItem to be displayed in column */
/* Return value: Index of new column */
/*****
int AddColumn( HWND hListView,
              int iCol,
              LPSTR lpText,
              int iFmt,
              int iWidth,

```

```

        int    iSubItem )
{
    LV_COLUMN lvc;

    // All items of the LV_COLUMN structure are valid -----
    lvc.mask      = LVCF_FMT | LVCF_SUBITEM | LVCF_TEXT | LVCF_WIDTH;

    // first column must always be left aligned -----
    lvc.fmt       = iCol ? iFmt : LVCFMT_LEFT;
    lvc.cx        = iWidth;
    lvc.pszText    = lpText;
    lvc.cchTextMax = lstrlen( lpText );
    lvc.iSubItem   = iSubItem;                // subitem to be displayed

    return ListView_InsertColumn(hListView, iCol, &lvc );
}

/*****
/* Compare - Compare function for LVM_SORTITEMSCB
/*-----*/
/* Parameters: lParam1 - lParam of first item to be compared
/*             lParam2 - lParam of second item to be compared
/*             lParamSort - User data
/* Return value: Comparison result of the two items
/*-----*/
/* Note      : lParam1 and lParam2 point to PSTOCKSTRUCTs. In lParam-
/*             Sort the index of the SubItem through which items are
/*             to be sorted is supplied:
/*             LOWORD(lParamSort) = SubItem,
/*             HIWORD(lParamSort) = ascending (TRUE) or
/*                               descending (FALSE)
*****/
int CALLBACK Compare( LPARAM lParam1,
                     LPARAM lParam2,
                     LPARAM lParamSort)
{
    PSTOCKSTRUCT p1 = ( PSTOCKSTRUCT )lParam1;
    PSTOCKSTRUCT p2 = ( PSTOCKSTRUCT )lParam2;
    switch( LOWORD( lParamSort ) )
    {
        case 0:                                // Sort by name
            return HIWORD( lParamSort ) ?
                lstrcmp( p1->pszStock, p2->pszStock ) :
                lstrcmp( p2->pszStock, p1->pszStock );
        case 1:                                // Sort by yesterday's price
            return HIWORD( lParamSort ) ?
                (int)(p1->fForeDay - p2->fForeDay):
                (int)(p2->fForeDay - p1->fForeDay);
        case 2:                                // Sort by current price
            return HIWORD( lParamSort ) ?
                (int)(p1->fToday - p2->fToday):
                (int)(p2->fToday - p1->fToday);
    }
    return 0;
}

/*****
/* DlgProc - Dialog procedure
/*-----*/
/* Parameters: default parameters
/* Return value: default return value
*****/
BOOL WINAPI DlgProc( HWND hWnd, UINT wParam, WPARAM wp, LPARAM lp )
{
    // static variables for simplified access to controls
    static HWND    hListView;                // ListView handle

    static BOOL    bDragging;                // Drag process?
    static int     iDragItem;                // Index of item to be dragged
    static int     iDropTarget;              // Index of DropTarget
    static HIMAGELIST hDragImage;            // Drag Image list
    static POINT    ptHotSpot;               // Coordinates of Drag Hotspot

    switch( wParam )
    {
        case WM_INITDIALOG:
            {

```

```

// Get ListView handle -----
hListView = GetDlgItem( hWnd, IDC_LISTVIEW );

// Reflect default parameters in controls -----
CheckDlgButton( hWnd, IDC_LARGEICON, TRUE );
CheckDlgButton( hWnd, IDC_SORTASCENDING, TRUE );

// Set ImageLists (since ListView has the TVS_SHAREIMAGELIST -
// style at its disposal, no special precautions must be taken -
// for deselection of the system ImageList) -
ListView_SetImageList( hListView, g_hImagesNormal, LVSIL_NORMAL );
ListView_SetImageList( hListView, g_hImagesSmall, LVSIL_SMALL );
ListView_SetImageList( hListView, g_hImagesState, TVSIL_STATE );

// Prepare Column Headings -----
AddColumn( hListView, 0, "Stock", LVCFMT_LEFT, 200, 0 );
AddColumn( hListView, 1, "Yesterday", LVCFMT_CENTER, 100, 1 );
AddColumn( hListView, 2, "Today", LVCFMT_CENTER, 100, 2 );
AddColumn( hListView, 3, "Trend", LVCFMT_CENTER, 100, 3 );

// Insert items -----
// If LVS_SORT... was specified as the ListView style, but the
// items make their text available via LPSTR_TEXTCALLBACK, then
// the ListView control cannot sort the items, so it refuses to
// accept them!
ADD_STOCK( hListView, "IBM", 93.000f, 92.500f, 3 );
ADD_STOCK( hListView, "Computer Assoc.", 42.125f, 42.250f, 3 );
ADD_STOCK( hListView, "Intel Corp.", 63.625f, 62.930f, 3 );
ADD_STOCK( hListView, "Zenith Elec.", 8.625f, 8.625f, 3 );
ADD_STOCK( hListView, "Microsoft Corp.", 86.625f, 86.930f, 3 );
ADD_STOCK( hListView, "Novell Inc.", 15.000f, 14.875f, 3 );
ADD_STOCK( hListView, "Motorola", 64.250f, 63.500f, 3 );

bDragging = FALSE; // Currently no dragging
}
break;
case WM_COMMAND:
    switch( LOWORD( wp ) )
    {
        case IDC_LARGEICON: // Display modes
        case IDC_SMALLICON:
        case IDC_LIST:
        case IDC_REPORT:

        case IDC_SORTNONE: // Sort order
        case IDC_SORTASCENDING:
        case IDC_SORTDESCENDING:

        case IDC_SHOWSELALWAYS: // Styles
        case IDC_EDITLABELS:
        case IDC_NOCOLUMNHEADER:
        case IDC_NOSORTHEADER:
        case IDC_NOLABELWRAP:
        {
            STYLESTRUCT st;

            // Get current ListView style -----
            st.styleOld = GetWindowLong( hListView, GWL_STYLE );
            st.styleNew = st.styleOld;

            switch( LOWORD( wp ) )
            {
                // Set display mode excluding the existing mode -----
                case IDC_LARGEICON:
                    st.styleNew &= ~LVS_TYPEMASK;
                    st.styleNew |= LVS_ICON;
                    break;
                case IDC_SMALLICON:
                    st.styleNew &= ~LVS_TYPEMASK;
                    st.styleNew |= LVS_SMALLICON;
                    break;
                case IDC_LIST:
                    st.styleNew &= ~LVS_TYPEMASK;
                    st.styleNew |= LVS_LIST;
                    break;
                case IDC_REPORT:

```

```

        st.styleNew &= ~LVS_TYPEMASK;
        st.styleNew |= LVS_REPORT;
        break;

// Set styles (first hide and if necessary show) -----
case IDC_SHOWSELALWAYS:
    st.styleNew &= ~LVS_SHOWSELALWAYS;
    if( IsDlgButtonChecked( hWnd, IDC_SHOWSELALWAYS ) )
        st.styleNew |= LVS_SHOWSELALWAYS;
    break;
case IDC_SORTNONE:
    st.styleNew &= ~(LVS_SORTASCENDING | LVS_SORTDESCENDING );
    break;
case IDC_SORTASCENDING:
    st.styleNew &= ~(LVS_SORTASCENDING | LVS_SORTDESCENDING );
    st.styleNew |= LVS_SORTASCENDING;
    break;
case IDC_SORTDESCENDING:
    st.styleNew &= ~(LVS_SORTASCENDING | LVS_SORTDESCENDING );
    st.styleNew |= LVS_SORTDESCENDING;
    break;
case IDC_EDITLABELS:
    st.styleNew &= ~LVS_EDITLABELS;
    if( IsDlgButtonChecked( hWnd, IDC_EDITLABELS ) )
        st.styleNew |= LVS_EDITLABELS;
    break;
case IDC_NOCOLUMNHEADER:
    st.styleNew &= ~LVS_NOCOLUMNHEADER;
    if( IsDlgButtonChecked( hWnd, IDC_NOCOLUMNHEADER ) )
        st.styleNew |= LVS_NOCOLUMNHEADER;
    break;
case IDC_NOSORTHEADER:
    st.styleNew &= ~LVS_NOSORTHEADER;
    if( IsDlgButtonChecked( hWnd, IDC_NOSORTHEADER ) )
        st.styleNew |= LVS_NOSORTHEADER;
    break;
case IDC_NOLABELWRAP:
    st.styleNew &= ~LVS_NOLABELWRAP;
    if( IsDlgButtonChecked( hWnd, IDC_NOLABELWRAP ) )
        st.styleNew |= LVS_NOLABELWRAP;
    break;
}
// Set new style -----
SetWindowLong( hListView, GWL_STYLE, st.styleNew );

// Inform ListView of style changes -----
SendMessage( hListView,
    WM_STYLECHANGED,
    ( WPARAM ) GWL_STYLE,
    ( LPARAM ) &st );
}
break;
case IDCANCEL:
    EndDialog( hWnd, 0 );
    break;
}
break;
case WM_PAINT:
    PaintPicture( (HINSTANCE)GetWindowLong( hWnd, GWL_HINSTANCE ),
        hWnd,
        IDB_PCINTERN,
        GetDlgItem( hWnd, IDC_PCINTERN ) );
break;
case WM_NOTIFY:
    switch( LOWORD( wp ) )
    {
        case IDC_LISTVIEW:
        {
            LPNM_LISTVIEW pnm_lv = (LPNM_LISTVIEW)lp;
            switch( pnm_lv->hdr.code )
            {
                case NM_DBLCLK:
                    // Double-click
                    {
                        LV_HITTESTINFO hi;
                        int iItem;
                        char szBuffer[ MAX_PATH ];

```

```

// Get item under cursor -----
GetCursorPos( &hi.pt );
ScreenToClient( hListView, &hi.pt );
iItem = ListView_HitTest(hListView, &hi );

if( iItem >= 0 )                                // Item hit?
{
    // Get text and display -----
    ListView_GetItemText(hListView,
                          iItem,
                          0,
                          szBuffer,
                          sizeof( szBuffer ) )
    MessageBox( hWnd, szBuffer, "Item text", 0 );
}
else MessageBox( hWnd, "Missed!!", "Message", 0 );
}
break;
case LVN_COLUMNCLICK:    // Click on column header in Report
    // Launch sort operation -----
    ListView_SortItems(hListView,
                      Compare,
                      MAKELONG( LOWORD( pnmlv->iSubItem ),
                                IsDlgButtonChecked( hWnd,
                                                    IDC_SORTASCENDING ) ) );
break;
case LVN_GETDISPINFO:    // Get display information of an item
{
    LV_DISPINFO *pnmv = (LV_DISPINFO *) lp;

    if( pnmv->item.mask & LVIF_TEXT )                // Get text?
    {
        static char szValue[ 10 ];
        PSTOCKSTRUCT pss = (PSTOCKSTRUCT)pnmv->item.lParam;

        switch( pnmv->item.iSubItem )                // for which SubItem?
        {
            case 0:
                pnmv->item.pszText = pss->pszStock;
                break;
            case 1:
                sprintf( szValue, "%3.3f", pss->fForeday );
                pnmv->item.pszText = szValue;
                break;
            case 2:
                sprintf( szValue, "%3.3f", pss->fToday );
                pnmv->item.pszText = szValue;
                break;
            case 3:
                // When passing the trend text, the State image
                // is automatically adapted. This is how you could
                // deal with continual changes (e.g., stock ticker)
                if( pss->fToday > pss->fForeday )
                {
                    pnmv->item.pszText = "rising";
                    pnmv->item.mask |= LVIF_STATE;
                    pnmv->item.state = LVIS_STATEIMAGEMASK;
                    pnmv->item.stateMask = INDEXTOSTATEIMAGEMASK( 0 );
                }
                if( pss->fToday < pss->fForeday )
                {
                    pnmv->item.pszText = "falling";
                    pnmv->item.mask |= LVIF_STATE;
                    pnmv->item.state = LVIS_STATEIMAGEMASK;
                    pnmv->item.stateMask = INDEXTOSTATEIMAGEMASK( 1 );
                }
                if( pss->fToday == pss->fForeday )
                {
                    pnmv->item.pszText = "constant";
                    pnmv->item.mask |= LVIF_STATE;
                    pnmv->item.state = LVIS_STATEIMAGEMASK;
                    pnmv->item.stateMask = INDEXTOSTATEIMAGEMASK( 2 );
                }
            }
        break;
    }
}

```

```

        // Specify length of text to be returned -----
        pnmv->item.cchTextMax = lstrlen( pnmv->item.pszText );
    }
}
break;
case LVN_BEGINDRAG:                // Launch drag operation
{
    POINT pt,                      // for screen coordinates of the mouse
        ptItem;                   // Coordinates of drag point

    // Get DragImage -----
    hDragImage = ListView_CreateDragImage( hListView,
                                           pnmlv->iItem,
                                           &ptItem );

    // Calculate position of mouse pointer in DragImage -----
    ptHotSpot.x = pnmlv->ptAction.x - ptItem.x;
    ptHotSpot.y = pnmlv->ptAction.y - ptItem.y;

    // Prepare ImageList for dragging -----
    if ( !ImageList_BeginDrag( hDragImage,
                               0,
                               ptHotSpot.x,
                               ptHotSpot.y ) )

        return FALSE;

    // During the drag no movements in ListView -----
    SetWindowLong( hListView,
                   GWL_STYLE,
                   GetWindowLong( hListView, GWL_STYLE ) |
                   LVS_NOSCROLL );

    GetCursorPos( &pt );
    ImageList_DragEnter( NULL, pt.x, pt.y );    // Launch drag

    SetCapture( hWnd );                // Set capture to dialog
    bDragging = TRUE;
    iDropTarget = -1;                  // up to now no DropTarget
    iDragItem = pnmlv->iItem;           // note DragItem
}
break;
case LVN_BEGINLABELEDIT:            // Subclass EditBox
{
    HWND hEdit;

    hEdit = ListView_GetEditControl( hListView );

    // EditBox should receive all keystrokes -----
    g_lpOldEditProc = (FARPROC)SetWindowLong( hEdit,
                                              GWL_WNDPROC,
                                              (LONG)EditSubclass );
}
return FALSE;                        // Allow Edit
case LVN_ENDLABELEDIT:              // Accept text changes?
{
    LV_DISPINFO FAR *pdi = (LV_DISPINFO FAR *) lp;

    // Changes ended with RETURN? -----
    if( pdi->item.pszText )
    {
        pdi->item.mask = LVIF_TEXT;        // accept changed text
        ListView_SetItem( hListView, &pdi->item );
    }
}
break;
}
break;
}
break;
case WM_MOUSEMOVE:                  // Utilize mouse movements for drag operation
{
    if( bDragging )
    {
        int iItem;
        POINT pt;
    }
}

```



```

LV_HITTESTINFO lvhtst;

// set current mouse position for HitTest -----
pt.x = lvhtst.pt.x = LOWORD( lp );
pt.y = lvhtst.pt.y = HIWORD( lp );

// Convert dialog coordinates to ListView coordinates... -----
ClientToScreen( hWnd, &lvhtst.pt );
ScreenToClient( hListView, &lvhtst.pt );

// Get index of hit item -----
iItem = ListView_HitTest( hListView, &lvhtst );

if( iItem != iDropTarget )
{
    // Hide drag image -----
    ImageList_DragShowNoLock(FALSE);

    // highlight hit item, normal display for left item -----
    SetDropHighlight( hListView,
                     iItem,
                     iDropTarget );

    iDropTarget = iItem; // note DropTarget
    ImageList_DragShowNoLock(TRUE); // Display DragImage again
}
ClientToScreen( hWnd, &pt ); // and set DragImage to screen
ImageList_DragMove( pt.x, pt.y ); // coordinates
}
}
break;
case WM_LBUTTONDOWN:
    if( bDragging ) // Drag operation?
    {
        POINT pt, ptOrg;

        pt.x = LOWORD( lp ); // Note mouse coordinates
        pt.y = HIWORD( lp );

        // Convert dialog coordinates to ListView coordinates -----
        ClientToScreen( hWnd, &pt );
        ScreenToClient( hListView, &pt );

        ImageList_DragLeave(NULL);
        ImageList_EndDrag(); // End drag

        ImageList_Destroy( hDragImage ); // Destroy DragImage

        SetDropHighlight( hListView, // Remove DropHighlight
                         -1,
                         iDropTarget );

        // get current left, upper corner of ListView... -----
        ListView_GetOrigin( hListView, &ptOrg );

        // ...and place DragItem at mouse position plus origin -----
        ListView_SetItemPosition( hListView,
                                iDragItem,
                                pt.x + ptOrg.x - ptHotSpot.x,
                                pt.y + ptOrg.y - ptHotSpot.y );

        bDragging = FALSE; // End dragging
        ReleaseCapture(); // Release mouse

        // Allow movements within the ListView again -----
        SetWindowLong( hListView,
                      GWL_STYLE,
                      GetWindowLong( hListView, GWL_STYLE ) &
                      ~LVS_NOSCROLL );
    }
    break;
}
return FALSE;
}

```

```

/*****

```

```

/* WinMain - Start function */
/*-----*/
/* Parameters: default parameters */
/* Return value: default return value */
/*****
int WINAPI WinMain( HINSTANCE hInst,
                   HINSTANCE hPrev,
                   LPSTR lpCmdLine,
                   int nCmdShow )

{
    SHFILEINFO sfi;          // SHFILEINFO structure for SystemImage lists
    InitCommonControls();    // Initialize CommonControl

    g_hImagesNormal= (HIMAGELIST)SHGetFileInfo( "",          // Normal icons
                                                0,
                                                &sfi,
                                                sizeof( sfi ),
                                                SHGFI_SYSICONINDEX |
                                                SHGFI_LARGEICON );

    g_hImagesSmall = (HIMAGELIST)SHGetFileInfo( "",          // small icons
                                                0,
                                                &sfi,
                                                sizeof( sfi ),
                                                SHGFI_SYSICONINDEX |
                                                SHGFI_SMALLICON );

    g_hImagesState = ImageList_LoadBitmap( hInst,
                                           MAKEINTRESOURCE( IDB_TREND ),
                                           16,
                                           0,
                                           RGB( 0, 128, 128 ) );

    DialogBox( hInst,
               MAKEINTRESOURCE( IDD_DIALOG ),
               NULL,
               DlgProc );

    return 0;
}

```