```
/* DLGBOX.C -- version 1.00 Demonstrates how to use the
*                                        DialogBoxIndirect and CreateDialogIndirect functions.
*/

#include <windows.h>
#include "dlgbox.h"

/* List of functions used in this module. */
BOOL FAR PASCAL InitDiabox ( HANDLE , HANDLE , int );
LONG FAR PASCAL DlgboxWndProc  (HANDLE, unsigned, WORD, LONG );
BOOL FAR PASCAL DialogBoxWindowProc (HANDLE, unsigned, WORD, LONG);
BOOL                    BuildDialog(HWND);

/* The following functions are found in the DLGTEMP module.
* They are used to build the dialog structure that will be
* passed to the DialogBoxIndirect or CreateDialogIndirect function.
*/

/* -------- Functions defined in another module ----------- */
extern BOOL     FAR PASCAL CreateDialogHeader(LONG,BYTE,int,int,int,int,LPSTR,LPSTR,LPSTR);
extern BOOL     FAR PASCAL CreateDialogItem(int,LONG,BYTE,int,int,int,int,LPSTR,BYTE );
extern HANDLE   FAR PASCAL EndDialogHeader( );

/* -------- Static variables ----------- */
HANDLE   hInst;              /* handle to Instance             */
FARPROC  lpDlgTest;          /* Dialog processing routine      */
HWND     hDlgTest = NULL;    /* handle for modeless dialog box */
BOOL     nModeless;          /* flag set by menu selected      */

HANDLE   hDlgTemp;    /* Handle to created dialog template       */
LPSTR    lpDTemplate; /* Long pointer to created dialog template */

/* ------------------ WinMain ------------------------------ */

int PASCAL WinMain(hInstance, hPrevInstance, lpszCmdLine, cmdShow)
HANDLE  hInstance , hPrevInstance;
LPSTR   lpszCmdLine;
int     cmdShow;
{
  MSG  msg;
  BOOL bResult; /* Modeless dialog message processed flag */

  if ( hPrevInstance )  /* do not allow more than one instance */
     return FALSE;

  InitDiabox ( hInstance, hPrevInstance, cmdShow );

  while ( GetMessage((LPMSG)&msg, NULL, 0, 0 )) {
     bResult = FALSE;

     /* this structure allows checking for multiple
                 modeless dialog boxes */

     if ( hDlgTest != NULL )
        bResult = IsDialogMessage( hDlgTest, (LPMSG)&msg );

     if ( !bResult ) {
        TranslateMessage((LPMSG)&msg);
        DispatchMessage((LPMSG)&msg);
     }
  }
  return msg.wParam ;
}

/* -------------- Initialization section ---------------------- */

BOOL FAR PASCAL InitDiabox( hInstance, hPrevInstance, cmdShow )
HANDLE hInstance;
HANDLE hPrevInstance;
```

```
int    cmdShow;
{
  WNDCLASS wcDiaboxClass;
  HWND     hWnd;

  wcDiaboxClass.lpszClassName      =          (LPSTR) "Diabox";
  wcDiaboxClass.hInstance          =          hInstance;
  wcDiaboxClass.lpfnWndProc        =          DlgboxWndProc;
  wcDiaboxClass.hCursor            =          LoadCursor( NULL, IDC_ARROW );
  wcDiaboxClass.hIcon              =          (HICON)LoadIcon( hInstance, "dlgicon");
  wcDiaboxClass.lpszMenuName       =          (LPSTR)"dlgbox";   /* menu added  */
  wcDiaboxClass.hbrBackground      =          GetStockObject( WHITE_BRUSH );
  wcDiaboxClass.style              =          CS_HREDRAW | CS_VREDRAW;
  wcDiaboxClass.cbClsExtra                    =          0;
  wcDiaboxClass.cbWndExtra                    =          0;

  RegisterClass ( (LPWNDCLASS) &wcDiaboxClass );

  hWnd = CreateWindow( (LPSTR)"Diabox",
                      (LPSTR)"DialogBox",
                      WS_OVERLAPPEDWINDOW,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
                      (HWND)NULL,
                      (HMENU)NULL,
                      (HANDLE)hInstance,
                      (LPSTR)NULL
                     );

  hInst = hInstance;       /*  instance saved for dialog box  */

  ShowWindow( hWnd, cmdShow );
  UpdateWindow( hWnd );

  return( TRUE );
}

/* ------------ parent's window procedure ------------ */

LONG FAR PASCAL DlgboxWndProc( hWnd, message, wParam, lParam )
HWND     hWnd;
unsigned message;
WORD     wParam;
LONG     lParam;
{
  PAINTSTRUCT ps;
  int   iResult;

  switch (message) {
     case WM_COMMAND:
        switch (wParam) {

          case IDM_MODAL:

             if ( hDlgTest == NULL ) { /* allow only one dialog up at a time */
               nModeless = FALSE;
               if ( BuildDialog(hWnd) == TRUE) {
                 lpDlgTest = MakeProcInstance( (FARPROC) DialogBoxWindowProc, hInst );

                 if (lpDlgTest == NULL) {
                   MessageBox( GetFocus(), "MakeProcInstance failed!", "ERROR", MB_OK );
                   GlobalFree( hDlgTemp );
                   return FALSE;
                 }
                 iResult = DialogBoxIndirect ( hInst, hDlgTemp, hWnd, lpDlgTest);
               }
             }
```

```
            break;

         case IDM_MODELESS:

            if ( hDlgTest == NULL ) { /* allow only one dialog up at a time */
              nModeless = TRUE;
              if ( BuildDialog(hWnd) == TRUE) {
                lpDTemplate = GlobalLock( hDlgTemp );
                if (lpDTemplate == NULL) {
                  MessageBox( GetFocus(), "Could not lock memory.", "Diabox", MB_OK );
                  GlobalFree( hDlgTemp );
                  return FALSE;
                }
                lpDlgTest = MakeProcInstance( (FARPROC) DialogBoxWindowProc, hInst );

                if (lpDlgTest == NULL) {
                  MessageBox( GetFocus(), "MakeProcInstance failed!", "ERROR", MB_OK );
                  GlobalFree( hDlgTemp );
                  return FALSE;
                }
                hDlgTest = CreateDialogIndirect( hInst, lpDTemplate, hWnd, lpDlgTest );
              }  /* exit if BuildDialog didn't work */
            }
            break;

         default:
            return DefWindowProc( hWnd, message, wParam, lParam );
            break;
       }

     case WM_PAINT:
        BeginPaint( hWnd, (LPPAINTSTRUCT)&ps );
        EndPaint  ( hWnd, (LPPAINTSTRUCT)&ps );
        break;

     case WM_DESTROY:
        FreeProcInstance ( (FARPROC)lpDlgTest );
        PostQuitMessage( 0 );
        break;

     default:
        return( DefWindowProc( hWnd, message, wParam, lParam ) );
  }

  return( 0L );
}

/* ------------- Build dialog routine --------------------------- */
/*
*  This routine passes all the necessary information to the
*  dialog functions in DLGTEMP.C that will build the dialog table.
*/

BOOL BuildDialog(hWnd)
HWND  hWnd;
{
  BOOL    bResult;
  HCURSOR hOldCursor;      /* Handle to old mouse cursor  */

  /* Build the Dialog header */

  bResult = CreateDialogHeader(
    WS_BORDER  | WS_CAPTION | WS_DLGFRAME | WS_VSCROLL |
    WS_HSCROLL | WS_SYSMENU | WS_GROUP    | WS_TABSTOP  |
    WS_SIZEBOX | WS_VISIBLE | WS_POPUP,    /* window style */
    (BYTE)12, 24, 12, 180, 160,          /* coordinates of Dialog box.*/
    "dlgtemp",                           /* menu        */
    "",                          /* class name  */
    "A test Dialog Box" );               /* Caption bar */
  if (bResult == FALSE) {
    MessageBox( GetFocus(), "Not enough memory.", "Diabox",  MB_OK );
    return FALSE;
  }

  hOldCursor = SetCursor( LoadCursor(NULL, IDC_WAIT) );

  /* Each call to CreateDialogItem is an item to be placed inside the dialog box */

  /* This next comment line is the value that is being passed */
  /* id,  style, class, x,   y, cx, cy,      text, extrabytes */

  CreateDialogItem(  IDCHECK,DI0,   0,  60,  90, 45, 12,  "&Ck box",0x00 );/* ck box   */
  CreateDialogItem( 3001,    DI1,   0,  44, 107,  0,  0,  "dlgicon",0x00 );/* icon     */
  CreateDialogItem( 3002,    DI2,   0,  73, 107, 16, 17,         "",0x00 );/* black box*/
  CreateDialogItem( 3003,    DI3,   0,   6, 107, 25, 15,         "",0x00 );/* rect     */
  CreateDialogItem( 3004,    DI4,   0,   6, 132, 40, 10,"L Justify",0x00 );/* lft text */
  CreateDialogItem( 3005,    DI5,   0, 122,  64, 40, 38,   "M Edit",0x00 );/* muli edt */
  CreateDialogItem( 3006,    DI6,   0,  60,   8, 54, 32,       "lb",0x00 );/* list box */
  CreateDialogItem( 3007,    DI7,   0,  43,   8,  8, 34,      "v s",0x00 );/*vert scrl */
  CreateDialogItem( 3008,    DI8,   0,   6,  47, 47,  8,      "h s",0x00 );/*horz scrl */
  CreateDialogItem( 3009,    DI9,   0,   6,   8, 32, 36,    "group",0x00 );/* group box*/
  CreateDialogItem( IDCANCEL,DI10,  0, 102, 112, 24, 14,    "&Push",0x00 );/*push but  */
  CreateDialogItem( 3011,    DI11,  0,   6,  61, 45, 12,   "&radio",0x00 );/*radio but */
  CreateDialogItem(     IDOK,DI12,  0, 136, 112, 24, 14,      "&Ok",0x00 );/*def buton */
  CreateDialogItem( 3013,    DI13,  0,   6,  90, 44, 12,"&L Ck box",0x00 );/*left ckbox*/
  CreateDialogItem( 3014,    DI14,  0,   6,  76, 30, 12,   "&3auto",0x00 );/*3 autoton */
  CreateDialogItem( 3015,    DI15,  0, 122,  27, 40, 12,   "C Edit",0x00 );/*edit cent */
  CreateDialogItem( 3016,    DI16,  0, 122,  44, 40, 12,   "R Edit",0x00 );/*rt edit   */
  CreateDialogItem(  IDDEDIT,DI17,  0, 122,   8, 40, 12,   "L Edit",0x00 );/* L edit   */
  CreateDialogItem( 3018,    DI18,  0,  60,  45, 54, 32,  "no sort",0x00 );/* lb w/out */
  CreateDialogItem( 3019,    DI19,  0,  66, 132, 50, 10,   "Center",0x00 );/* CEN text */
  CreateDialogItem( 3020,    DI20,  0, 122, 132, 46, 10,"R Justify",0x00 );/* RIG text */

  SetCursor( hOldCursor );

  /* Get the handle to the new dialog table */

  hDlgTemp = EndDialogHeader();  /* end dialog template, return hDlgTemp. */
  if (hDlgTemp == NULL) {
    GlobalFree(hDlgTemp);
    MessageBox( GetFocus(), "End dialog routine.", "Error!", MB_OK );
    return FALSE;
  }
  return TRUE;
}

/* --------------  dialog box window procedure ------------ */
/*
*  This routine processes all the message for the dialog that
*  was dynamically created.
*/

BOOL FAR PASCAL DialogBoxWindowProc ( hDlg, message, wParam, lParam )
HWND     hDlg;
unsigned message;
WORD     wParam;
LONG     lParam;
{
  char    TempEdit[24];
  HWND    hListBox, hListBoxNS;

  switch(message) {

     case WM_INITDIALOG:

        hListBox = GetDlgItem( hDlg, 3006 );
        SendMessage(hListBox, LB_ADDSTRING, 0,(LONG)(LPSTR)"Sorted");
```

```
         SendMessage(hListBox, LB_ADDSTRING, 0,(LONG)(LPSTR)"CCC");
         SendMessage(hListBox, LB_ADDSTRING, 0,(LONG)(LPSTR)"AAA");
         SendMessage(hListBox, LB_ADDSTRING, 0,(LONG)(LPSTR)"BBB");

         hListBoxNS = GetDlgItem( hDlg, 3018 );
         SendMessage(hListBoxNS, LB_ADDSTRING, 0,(LONG)(LPSTR)"NOT Sorted!");
         SendMessage(hListBoxNS, LB_ADDSTRING, 0,(LONG)(LPSTR)"CCC");
         SendMessage(hListBoxNS, LB_ADDSTRING, 0,(LONG)(LPSTR)"AAA");
         SendMessage(hListBoxNS, LB_ADDSTRING, 0,(LONG)(LPSTR)"BBB");

         SetFocus( GetDlgItem( hDlg, IDOK ) );
         return(TRUE);

         break;

      case WM_COMMAND:

         switch( wParam ) {

            case IDCHECK: /*  Check Box       */
               CheckDlgButton(hDlg, IDCHECK, !IsDlgButtonChecked(hDlg,IDCHECK));
               break;

            case IDM_ONE: /*  One menu item  */

               MessageBox ( GetFocus(), (LPSTR)"One selected.", (LPSTR)"Menu item", MB_OK );
               break;

            case IDOK:    /*  Ok button       */

               GetDlgItemText( hDlg, IDDEDIT, TempEdit, 24 );
               MessageBox ( GetFocus(), (LPSTR)TempEdit, (LPSTR)"Edit:", MB_OK );
               /* fall through to IDCANCEL */

            case IDCANCEL: /* Cancel push button */

               hDlgTest = NULL;
               if ( nModeless == FALSE )
                  EndDialog ( hDlg, TRUE );
               else
                  DestroyWindow( hDlg );
               return( TRUE );

            default:
                return( TRUE );
         }  /* end wParam switch */
         break;   /* end wm_command */

      case WM_DESTROY:

         hDlgTest = NULL;

         while (GlobalFlags(hDlgTemp) & GMEM_LOCKCOUNT)
              GlobalUnlock( hDlgTemp );
         if (hDlgTemp)
           GlobalFree( hDlgTemp );

         break;

      default:
         return( FALSE );

   }  /* end message switch */
   return( FALSE );
}
```