الجامعة التكنولوجية – قسم علوم الحاسوب
2023-2022
(Window Programming)

الاسم:- حسن عيسى عبدالمحسن

المرحلة:- الرابعة

المادة:- برمجة نوافذ 2

الفرع:- البرمجيات

اسم التدريسي:- أ.م.د. يسرى حسين

Menu, Dialog Box, List Box, Edit Box, Modeless

Main.cpp

```
#include <windows.h>
#include <string.h>
#include <stdio.h>
#include "header.h"
#define NUMBOOKS 7
LRESULT CALLBACK WindowFunc (HWND, UINT, WPARAM,
LPARAM);
BOOL CALLBACK DialogFunc (HWND, UINT, WPARAM, LPARAM);
char szWinName [ ] = "MyWin" ; /* name of window class */
HINSTANCE hInst;
HWND hDlg =0; /* dialog box handle */

/* books database */
struct booksTag
{
   char title [40] ;
   unsigned copyright;
   char author[40];
   char publisher [40] ;
}
books[NUMBOOKS] =
{
   {"C: The Complete Reference", 1995,"Hebert Schildt",
 "Osborne/McGraw-Hill"},
   {"MFC Programming from the Ground Up", 1996, "Herbert
Schildt","Osborne/McGraw-Hill"},
   {"Java:The Complete Reference",1997,"Naughton and
Schildt","Osborne/McGraw-Hill " },
   {"Design and Evolution of C+ + ", 1994, "Bjarne Stroustrup",
 "Addison-Wesley" },
   { "Inside OLE", 1995, "Kraig Brockschmidt", "Microsoft Press" },
   {"HTML Sourcebook", 1996, "lan S. Graham", "John Wiley & Sons"
},
   {"Standard C++ Library", 1995, "P. J. Plauger", "Prentice-Hall" }
};
int WINAPI WinMain(HINSTANCE hThisInst, HINSTANCE hPrevInst,
                   LPSTR IpszArgs, int nWinMode)
{
   HWND hwnd;
```

```
   MSG msg;
   WNDCLASSEX wc1;
   HACCEL hAccel;
   wc1.cbSize = sizeof(WNDCLASSEX);
   wc1.hInstance = hThisInst;
   wc1.lpszClassName = szWinName;
   wc1.lpfnWndProc = WindowFunc;
   wc1.style =0;
   wc1.hIcon = LoadIcon(NULL, IDI_APPLICATION);
   wc1.hIconSm= LoadIcon(NULL, IDI_WINLOGO);
   wc1.hCursor = LoadCursor(NULL, IDC_ARROW);
   wc1.lpszMenuName = "MyMenu";
   wc1.cbClsExtra = 0;
   wc1.cbWndExtra = 0;
   wc1.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);
   /* Register the window class. */ if(!RegisterClassEx(&wc1)) return 0;
   /* Now that a window class has been registered, a window can be
created. */
   hwnd=CreateWindow(szWinName,
                "Demonstrate A Modeless Dialog Box",
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                HWND_DESKTOP,
                NULL, hThisInst, NULL);
   hInst = hThisInst; /* save the current instance handle */
   /* load accelerators */ hAccel = LoadAccelerators (hThisInst,
"MyMenu");
   /* Display the window. */ ShowWindow(hwnd, nWinMode) ;
   UpdateWindow(hwnd) ;
   while (GetMessage(&msg, NULL, 0, 0))
   {
      if ( !IsDialogMessage (hDlg, &msg))
         /* is not a dialog message*/
         if (!TranslateAccelerator (hwnd, hAccel, &msg))
         {
            TranslateMessage(&msg) ;
            DispatchMessage( &msg );
         }
   }
```

```
    return msg.wParam;
}

/* This function is called by Windows NT and is passed messages from
the message queue. */
LRESULT CALLBACK WindowFunc(HWND hwnd, UINT message,
                    WPARAM wParam, LPARAM lParam)
{
    int response;
    switch(message)
    {
    case WM_COMMAND:
        switch(LOWORD(wParam))
        {
        case IDM_DIALOG:
            hDlg = CreateDialog(hInst, "MyDB", hwnd,
(DLGPROC)DialogFunc);
            break;

        case IDM_EXIT:
            response = MessageBox(hwnd, "Quit the Program?", "Exit",
MB_YESNO);
            if(response == IDYES) PostQuitMessage(0);
            break;
        case IDM_HELP:
            MessageBox(hwnd, "No Help", "Help", MB_OK) ;
            break;
        }
        break;
    case WM_DESTROY: /* terminate the program */
        PostQuitMessage(0);
        break;
    default:
        return DefWindowProc(hwnd, message, wParam, lParam);
    }
    return 0;
}
/* A simple dialog function. */
BOOL CALLBACK DialogFunc(HWND hdwnd, UINT message,
WPARAM wParam, LPARAM lParam)
{
    long i;
```

```
    char str[255];
    switch(message)
    {
    case WM_INITDIALOG: /* initialize list box */
        for(i = 0; i<NUMBOOKS; i++)
            SendDlgItemMessage(hdwnd, IDD_LB1,LB_ADDSTRING, 0,
(LPARAM)books[i].title);
        /* select first item */
        SendDlgItemMessage(hdwnd, IDD_LB1, LB_SETCURSEL, 0, 0);
        /*initialize the edit box*/
        SetDlgItemText(hdwnd, IDD_EB1, books[0] .title);
        return 1;
    case WM_COMMAND:
        switch(LOWORD(wParam))
        {
        case IDCANCEL:
            DestroyWindow(hdwnd);
            return 1;
        case IDD_COPYRIGHT:
            i = SendDlgItemMessage(hdwnd, IDD_LB1, LB_GETCURSEL,
0, 0);
            sprintf(str,"%u", books[i].copyright);
            MessageBox(hdwnd, str, "Copyright", MB_OK);
            return 1;
        case IDD_AUTHOR:
            i = SendDlgItemMessage(hdwnd, IDD_LB1, LB_GETCURSEL,
0, 0);
            sprintf(str, "%s", books[i].author);
            MessageBox(hdwnd, str, "Author", MB_OK);
            return 1;
        case IDD_PUBLISHER:
            i=SendDlgItemMessage(hdwnd, IDD_LB1, LB_GETCURSEL, 0,
0);
            sprintf(str, "%s", books[i].publisher);
            MessageBox(hdwnd, str, "Publisher", MB_OK);
            return 1;
        case IDD_DONE:/*get current contents of edit box*/
            GetDlgItemText(hdwnd, IDD_EB1, str, 80);
            i=SendDlgItemMessage(hdwnd, IDD_LB1, LB_FINDSTRING, 0,
(LPARAM) str);
            if(i != LB_ERR) /* if match is found */
            {
```

```
            SendDlgItemMessage(hdwnd,IDD_LB1, LB_SETCURSEL,
i,0);
            SendDlgItemMessage(hdwnd,IDD_LB1,LB_GETTEXT, i,
(LPARAM) str);
            /*update text in edit box*/
            SetDlgItemText(hdwnd, IDD_EB1,str);
         }
         else MessageBox(hdwnd, str, "No Title Matching", MB_OK);
         return 1;

      case IDD_LB1: /* process a list box LBN_DBLCLK */
         if(HIWORD(wParam)==LBN_DBLCLK) /* see if user made a
selection */
         {
            i=SendDlgItemMessage(hdwnd, IDD_LB1, LB_GETCURSEL,
0, 0); /*get index*/
            sprintf(str, "%s\n%s\n%s, %u",books[i].title,
books[i].author,books[i].publisher,
                  books[i].copyright);
            MessageBox(hdwnd, str, "Selection Made", MB_OK);

SendDlgItemMessage(hdwnd,IDD_LB1,LB_GETTEXT,i,(LPARAM)str);
            /*update edit box*/ SetDlgItemText(hdwnd, IDD_EB1, str);
            return 1;
         case IDD_SELECT: /* Select Book button has been pressed */
            i = SendDlgItemMessage(hdwnd, IDD_LB1,
LB_GETCURSEL, 0, 0); /* get index */
            sprintf (str, "%s\n%s\n%s, %u", books[i].title, books[i].author,
books[i].publisher,
                  books[i].copyright);
            MessageBox(hdwnd, str, "Selection Made", MB_OK);
            /*get string associated with that index*/
            SendDlgItemMessage(hdwnd, IDD_LB1, LB_GETTEXT, i,
(LPARAM) str);
            /* update edit box */ SetDlgItemText(hdwnd, IDD_EB1, str);
            return 1;
         }
      }
   }
   return 0;
}
```

Header.h

```
#define IDM_DIALOG 100
#define IDM_EXIT 101
#define IDM_HELP 102
#define IDD_AUTHOR 200
#define IDD_PUBLISHER 201
#define IDD_COPYRIGHT 202
#define IDD_LB1 203
#define IDD_SELECT 204
#define IDD_EB1 205
#define IDD_DONE 206
```

Recourse.rc

```
#include <windows.h>
#include "header.h"
Mymenu MENU
{
   POPUP "&Dialog" {
      MENUITEM "&Dialog\tF2", IDM_DIALOG
      MENUITEM "&Exit\tF3", IDM_EXIT
   }
   MENUITEM "&Help", IDM_HELP }
Mymenu ACCELERATORS
{
   VK_F2, IDM_DIALOG, VIRTKEY
   VK_F3, IDM_EXIT, VIRTKEY
   VK_F1, IDM_HELP, VIRTKEY }

MyDB DIALOG 10, 10, 210, 130
CAPTION "Books Dialog Box"
STYLE WS_POPUP | WS_CAPTION |WS_SYSMENU | WS_VISIBLE
{
   DEFPUSHBUTTON "Author", IDD_AUTHOR, 11, 10, 36, 14,
   WS_CHILD | WS_VISIBLE | WS_TABSTOP
   PUSHBUTTON "Publisher", IDD_PUBLISHER, 11, 34, 36, 14,
   WS_CHILD | WS_VISIBLE | WS_TABSTOP
   PUSHBUTTON "Copyright", IDD_COPYRIGHT, 11, 58, 36, 14,
   WS_CHILD | WS_VISIBLE | WS_TABSTOP
   PUSHBUTTON "Cancel", IDCANCEL, 11, 82, 36, 16,
   WS_CHILD | WS_VISIBLE | WS_TABSTOP
   LISTBOX IDD_LB1, 60, 5, 140, 33,
```

```
    LBS_NOTIFY | WS_VISIBLE |WS_VSCROLL | WS_BORDER
|WS_TABSTOP
    PUSHBUTTON "Select Book", IDD_SELECT, 103, 41, 54, 14,
    WS_CHILD | WS_VISIBLE | WS_TABSTOP
    EDITTEXT IDD_EB1, 65, 73, 130, 12,
    ES_LEFT |WS_VISIBLE |WS_BORDER |ES_AUTOHSCROLL
|WS_TABSTOP
    PUSHBUTTON "Title Search", IDD_DONE, 107, 91, 46, 14,
WS_CHILD |
    WS_VISIBLE | WS_TABSTOP
}
```

Scroll bar, Scroll bar Control

```
Main.cpp
#include <windows.h>
#include <string.h>
#include <stdio.h>
#include "header.h"
#define VERTRANGEMAX 200
#define HORZRANGEMAX 50

LRESULT CALLBACK WindowFunc(HWND, UINT, WPARAM,
LPARAM);
BOOL CALLBACK DialogFunc(HWND, UINT, WPARAM, LPARAM);
char szWinName[ ] = "MyWin"; /* name of window class */ HINSTANCE
hInst;

int WINAPI WinMain(HINSTANCE hThisInst, HINSTANCE hPrevInst,
LPSTR IpszArgs, int nWinMode)
{
   HWND hwnd;
   MSG msg;
   WNDCLASSEX wcl;
   HACCEL hAccel;
   wcl.cbSize = sizeof(WNDCLASSEX);
   wcl.hInstance = hThisInst;
   wcl.lpszClassName = szWinName;
   wcl.lpfnWndProc = WindowFunc;
   wcl.style = 0;
   wcl.hIcon = LoadIcon(NULL, IDI_APPLICATION);
   wcl.hIconSm = LoadIcon(NULL, IDI_WINLOGO);
```

```
   wcl.hCursor = LoadCursor(NULL, IDC_ARROW);
   wcl.lpszMenuName = "Mymenu";
   wcl.cbClsExtra =0;
   wcl.cbWndExtra=0;
   wcl.hbrBackground = WHITE_BRUSH;
   if(!RegisterClassEx(&wcl)) return 0;
   hwnd=CreateWindow(szWinName, "Managing Scroll Bars",
                WS_OVERLAPPEDWINDOW, CW_USEDEFAULT,
CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
HWND_DESKTOP, NULL, hThisInst, NULL );
   hInst = hThisInst; /* save the current instance handle */
   /* load accelerators */ hAccel = LoadAccelerators (hThisInst,
"Mymenu");
   /* Display the window. */

   ShowWindow(hwnd, nWinMode);
   UpdateWindow(hwnd);
   while (GetMessage(&msg,NULL,0,0))
   {
      if ( !TranslateAccelerator (hwnd, hAccel, &msg))
      {
         TranslateMessage(&msg) ;
         DispatchMessage( &msg );
      }
   }
   return msg.wParam;
}
/* This function is called by Windows NT and is passedmessages from the
message queue. */

LRESULT CALLBACK WindowFunc (HWND hwnd, UINT message,
                     WPARAM wParam, LPARAM lParam)
{
   int response;
   switch (message)
   {
   case WM_COMMAND:
      switch (LOWORD (wParam) )
      {

      case IDM_DIALOG:
```

```
            DialogBox(hInst,"MyDB", hwnd, (DLGPROC) DialogFunc) ;
            break;
         case IDM_EXIT:
            response=MessageBox(hwnd,"Quit the
Program?","Exit",MB_YESNO) ;
            if (response == IDYES) PostQuitMessage (0) ;
            break;
         case IDM_HELP:
            MessageBox(hwnd,"Try the Scroll Bar","Help",MB_OK) ;
            break;
         }
         break;
      case WM_DESTROY: /* terminate the program */
         PostQuitMessage (0) ;
         break;
      default:
         return DefWindowProc (hwnd,message,wParam,lParam);
      }
      return 0;
}

/* A simple dialog function. */
BOOL CALLBACK DialogFunc (HWND hdwnd, UINT
                  message, WPARAM wParam, LPARAM lParam)
{
      char str[80];
      static int vpos = 0;
      static int hpos= 0;
      static int cntlpos=0;
      static SCROLLINFO si;
      HDC hdc;
      PAINTSTRUCT paintstruct;
      switch (message)
      {
      case WM_COMMAND:
         switch (LOWORD (wParam) )
         {

         case IDCANCEL:
            EndDialog(hdwnd, 0);
            return 1;
         }
```

```
        break;
    case WM_INITDIALOG:
        si.cbSize = sizeof(SCROLLINFO) ;
        si.fMask = SIF_RANGE;
        si.nMin = 0;
        si.nMax = VERTRANGEMAX;
        SetScrollInfo(hdwnd, SB_VERT, &si, 1);
        SetScrollInfo(GetDlgItem(hdwnd, ID_SB1), SB_CTL, &si, 1);
        si.nMax = HORZRANGEMAX;
        SetScrollInfo(hdwnd, SB_HORZ, &si, 1);
        vpos = hpos = cntlpos = 0;
        return 1;
    case WM_PAINT:
        hdc = BeginPaint(hdwnd, &paintstruct);
        sprintf(str, "Vertical: %d", vpos);
        TextOut(hdc, 30, 150, str, strlen(str));
        sprintf(str, "Horizontal: %d", hpos);
        TextOut(hdc, 30, 180, str, strlen(str));
        sprintf(str, "Scroll Bar Control: %d ",cntlpos);
        TextOut(hdc, 30, 210, str, strlen(str));
        EndPaint(hdwnd, &paintstruct);
        return 1;

    case WM_VSCROLL:
        switch(LOWORD(wParam))
        {
        case SB_LINEDOWN:
            if ( (HWND) lParam==GetDlgItem(hdwnd, ID_SB1))
            {
                /*is control scroll bar*/cntlpos++;
                if(cntlpos>VERTRANGEMAX) cntlpos = VERTRANGEMAX;
            }
            else
            {
                /* is window scroll bar */ vpos++;
                if(vpos>VERTRANGEMAX) vpos = VERTRANGEMAX;
            }
            break;
        case SB_LINEUP:
            if((HWND)lParam==GetDlgItem(hdwnd, ID_SB1))
            {
                /*is control scroll bar*/ cntlpos--;
```

```
            if(cntlpos<0) cntlpos = 0;
        }
        else
        {
            /* is window scroll bar */vpos--;
            if(vpos<0) vpos = 0;
        }
        break;

    case SB_THUMBPOSITION:
        if((HWND)lParam==GetDlgItem(hdwnd, ID_SB1))
        {
            /*is control scroll bar */cntlpos = HIWORD(wParam); /* get
current position */
        }
        else
        {
            /* is window scroll bar */vpos = HIWORD(wParam);
        }
        break;
    case SB_THUMBTRACK:
        if((HWND)lParam==GetDlgItem(hdwnd, ID_SB1))
        {
            /*is control scroll bar*/ cntlpos = HIWORD(wParam); /* get
current position */
        }
        else
        {
            /*is window scroll bar*/vpos = HIWORD(wParam);
        }
        break;
    case SB_PAGEDOWN:
        if ( (HWND) lParam==GetDlgItem(hdwnd, ID_SB1))
        {
            /*is control scroll bar*/cntlpos += 5;
            if(cntlpos>VERTRANGEMAX) cntlpos=VERTRANGEMAX;
        }
        else
        {
            /*is window scroll bar*/vpos += 5;
            if(vpos>VERTRANGEMAX) vpos=VERTRANGEMAX;
        }
```

```
            break;
        case SB_PAGEUP:
            if((HWND)lParam==GetDlgItem(hdwnd, ID_SB1))
            {
                /*is control scroll bar */cntlpos -= 5;
                if(cntlpos<0) cntlpos = 0;
            }
            else  /* is window scroll bar */
            {
                vpos -= 5;
                if(vpos<0) vpos = 0;
            }
            break;
        }
        /* update vertical bar position */ si.fMask = SIF_POS;
        si.nPos = cntlpos;
        SetScrollInfo(hdwnd, SB_VERT, &si, 1);
        hdc = GetDC(hdwnd);
        sprintf(str, "Scroll Bar Control: %d ", cntlpos);
        TextOut(hdc, 30, 210, str, strlen(str));
        ReleaseDC(hdwnd, hdc);
        si.fMask = SIF_POS;
        si.nPos = vpos;
        SetScrollInfo(hdwnd, SB_VERT, &si, 1);
        hdc = GetDC(hdwnd);
        sprintf(str, "Vertical: %d ", vpos);
        TextOut(hdc, 30, 150, str, strlen(str));
        ReleaseDC(hdwnd,hdc);
        return 1;
    case WM_HSCROLL:
        switch(LOWORD(wParam))
        {
        /*Try adding the other event
        handling code for the horizontal scroll bar, here. */
        case SB_LINERIGHT:
            hpos++;
            if(hpos>HORZRANGEMAX) hpos=HORZRANGEMAX;
            break;
        case SB_LINELEFT:
            hpos--;
            if(hpos<0) hpos = 0;
            break;
```

```
        case SB_THUMBPOSITION:
            hpos = HIWORD(wParam);
            break;
        case SB_THUMBTRACK:
            hpos = HIWORD(wParam);
            break;
        }
        /* update horizontal bar position */ si.fMask = SIF_POS;
        si.nPos = hpos ;
        SetScrollInfo(hdwnd, SB_HORZ, &si, 1);
        hdc = GetDC(hdwnd);
        sprintf(str, "Horizontal: %d ", hpos);
        TextOut(hdc, 30, 180, str, strlen(str));
        ReleaseDC(hdwnd, hdc);
        return 1;
    }
    return 0;
}
```

Header.h

```
#define IDM_DIALOG 100
#define IDM_EXIT 101
#define IDM_HELP 102
#define ID_SB1 200
```

Recourse.rc

```
#include <windows.h>
#include "header.h"
Mymenu MENU
{
    POPUP "&Dialog" {
        MENUITEM "&Dialog\tF2", IDM_DIALOG
        MENUITEM "&Exit\tF3", IDM_EXIT
    }
    MENUITEM "&Help", IDM_HELP }
Mymenu ACCELERATORS
{
    VK_F2, IDM_DIALOG, VIRTKEY
    VK_F3, IDM_EXIT, VIRTKEY
    VK_F1, IDM_HELP, VIRTKEY
}
```

```
MyDB DIALOG 18, 18, 252, 222
CAPTION "Using Scroll Bars"
STYLE DS_MODALFRAME | WS_POPUP | WS_VSCROLL |
WS_HSCROLL | WS_CAPTION | WS_SYSMENU
{ SCROLLBAR ID_SB1, 210, 30, 10, 100, SBS_VERT | WS_TABSTOP}
```

Check Box, Radio Button

Main.cpp

```
#include <windows.h>
#include <string.h>
#include <stdio.h>
#include "header.h"
#define VERTRANGEMAX 200

LRESULT CALLBACK WindowFunc(HWND, UINT, WPARAM,
LPARAM);
BOOL CALLBACK DialogFunc(HWND, UINT, WPARAM, LPARAM);
char szWinName[ ] = "MyWin"; /* name of window class */ HINSTANCE
hInst;
HWND hwnd;

int WINAPI WinMain(HINSTANCE hThisInst, HINSTANCE hPrevInst,
LPSTR IpszArgs, int nWinMode)

{
   MSG msg;
   WNDCLASSEX wcl;
   HACCEL hAccel;
   wcl.cbSize = sizeof(WNDCLASSEX);
   wcl.hInstance = hThisInst;
   wcl.lpszClassName = szWinName;
   wcl.lpfnWndProc = WindowFunc;
   wcl.style = 0;
   wcl.hIcon = LoadIcon(NULL, IDI_APPLICATION);
   wcl.hIconSm = LoadIcon(NULL, IDI_WINLOGO);
   wcl.hCursor = LoadCursor(NULL, IDC_ARROW);
   wcl.lpszMenuName = "Mymenu";
   wcl.cbClsExtra =0;
   wcl.cbWndExtra=0;
   wcl.hbrBackground = WHITE_BRUSH;
   if(!RegisterClassEx(&wcl)) return 0;
```

```
   hwnd=CreateWindow(szWinName, "Demonstrating Controls",
                WS_OVERLAPPEDWINDOW, CW_USEDEFAULT,
CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
HWND_DESKTOP, NULL, hThisInst, NULL );
   hInst = hThisInst; /* save the current instance handle */
   /* load accelerators */ hAccel = LoadAccelerators (hThisInst,
"Mymenu");
   /* Display the window. */

   ShowWindow(hwnd, nWinMode);
   UpdateWindow(hwnd);
   while (GetMessage(&msg,NULL,0,0))
   {
      if ( !TranslateAccelerator (hwnd, hAccel, &msg))
      {
         TranslateMessage(&msg) ;
         DispatchMessage( &msg );
      }
   }
   return msg.wParam;
}
/* This function is called by Windows NT and is passedmessages from the
message queue. */

LRESULT CALLBACK WindowFunc (HWND hwnd, UINT message,
                     WPARAM wParam, LPARAM lParam)
{
   int response;
   switch (message)
   {
   case WM_COMMAND:
      switch (LOWORD(wParam) )
      {
      case IDM_DIALOG:
         DialogBox(hInst, "MyDB", hwnd, (DLGPROC) DialogFunc);
         break;
      case IDM_EXIT:
         response=MessageBox (hwnd,"Quit the Program?","Exit",
MB_YESNO) ;
         if (response == IDYES) PostQuitMessage (0) ;
         break;
```

```
        case IDM_HELP:
            MessageBox (hwnd, "Try the Timer", "Help", MB_OK) ;
            break;
        }
        break;
    case WM_DESTROY: /* terminate the program */
        PostQuitMessage (0) ;
        break;
    default:
        return DefWindowProc (hwnd, message, wParam, lParam) ;
    }
    return 0;
}
BOOL CALLBACK DialogFunc (HWND hdwnd, UINT message,
WPARAM wParam, LPARAM lParam)
{
    char str [80] ;
    static int vpos=0;
    static SCROLLINFO si;
    HDC hdc;
    PAINTSTRUCT paintstruct ;
    static int t;
    switch(message)
    {
    case WM_COMMAND:
        switch (LOWORD (wParam) )

        {
        case IDCANCEL:
            EndDialog (hdwnd, 0) ;
            return 1;
        case IDD_START: /* start the timer */
            SetTimer (hdwnd, IDD_TIMER, 1000, NULL) ;
            t = vpos ;

if(SendDlgItemMessage(hdwnd,IDD_RB1,BM_GETCHECK,0,0)==BST_C
HECKED)
            {
                ShowWindow(hdwnd, SW_MINIMIZE);
            }
```

```
if(SendDlgItemMessage(hdwnd,IDD_RB2,BM_GETCHECK,0,0)==BST_C
HECKED)
            ShowWindow(hdwnd, SW_MAXIMIZE);
         return 1;
      }
      break;

   case WM_TIMER:
      if(t==0)
      {
         KillTimer(hdwnd,IDD_TIMER); /*timer went off*/

if(SendDlgItemMessage(hdwnd,IDD_CB2,BM_GETCHECK,0,0)==BST_C
HECKED) MessageBeep (MB_OK) ;
         MessageBox(hdwnd, "Timer Went Off", "Timer", MB_OK);
         ShowWindow(hdwnd, SW_RESTORE);
         return 1;
      }
          t--;/*see if countdown is to be displayed*/
      if(SendDlgItemMessage(hdwnd,IDD_CB1, BM_GETCHECK, 0, 0) ==
BST_CHECKED)
      {
         hdc = GetDC(hdwnd);
         sprintf(str, "Counting: %d ", t);
         TextOut(hdc, 1, 1, str, strlen(str));
         ReleaseDC(hdwnd, hdc);
         return 1;

      case WM_INITDIALOG:
         si.cbSize = sizeof(SCROLLINFO);
         si.fMask = SIF_RANGE;
         si.nMin = 0;
         si.nMax = VERTRANGEMAX;
         SetScrollInfo(hdwnd, SB_VERT, &si, 1); /* check the As-Is radio
button */
         SendDlgItemMessage(hdwnd,IDD_RB3,BM_SETCHECK,
BST_CHECKED, 0);
         return 1;
      case WM_PAINT:
         hdc = BeginPaint(hdwnd, &paintstruct);
         sprintf(str, "Interval: %d", vpos);
```

```
        TextOut(hdc, 1, 1, str, strlen(str));
        EndPaint (hdwnd, &paintstruct);
        return 1;

      case WM_VSCROLL:
        switch (LOWORD(wParam) )
        {
        case SB_LINEDOWN:
          vpos++;
          if(vpos>VERTRANGEMAX) vpos=VERTRANGEMAX;
          break;
        case SB_LINEUP:
          vpos--;
          if(vpos<0) vpos = 0;
          break;
        case SB_THUMBPOSITION:
          vpos = HIWORD(wParam); /*get current position*/break;
        case SB_THUMBTRACK:
          vpos = HIWORD(wParam); /* get current position */break;
        case SB_PAGEDOWN:
          vpos += 5;
          if(vpos>VERTRANGEMAX) vpos=VERTRANGEMAX;
          break;
        case SB_PAGEUP:
          vpos -= 5;
          if(vpos<0) vpos = 0;
        }
        si.fMask = SIF_POS;
        si.nPos = vpos;
        SetScrollInfo(hdwnd, SB_VERT, &si, 1);
        hdc = GetDC(hdwnd);
        sprintf(str, "Interval: %d ", vpos);
        TextOut(hdc, 1, 1, str, strlen(str));
        ReleaseDC(hdwnd, hdc);
      }
      return 1;
    }
    return 0;
}
```

Header.h

```
#define IDM_DIALOG 100
#define IDM_EXIT 101
#define IDM_HELP 102
#define IDD_START 300
#define IDD_TIMER 301
#define IDD_CB1 400
#define IDD_CB2 401
#define IDD_RB1 402
#define IDD_RB2 403
#define IDD_RB3 404
```

Recourse.rc

```
#include <windows.h>
#include "header.h"
Mymenu MENU
{
   POPUP "&Dialog" {
      MENUITEM "&Dialog\tF2", IDM_DIALOG
      MENUITEM "&Exit\tF3", IDM_EXIT
   }
   MENUITEM "&Help", IDM_HELP }
Mymenu ACCELERATORS
{
   VK_F2, IDM_DIALOG, VIRTKEY
   VK_F3, IDM_EXIT, VIRTKEY
   VK_F1, IDM_HELP, VIRTKEY }
MyDB DIALOG 18, 18, 142, 92
CAPTION "A Countdown Timer"
STYLE DS_MODALFRAME | WS_POPUP | WS_VSCROLL |
WS_CAPTION | WS_SYSMENU
{
   PUSHBUTTON "Start", IDD_START, 10, 60, 30, 14, WS_CHILD |
WS_VISIBLE | WS_TABSTOP
   PUSHBUTTON "Cancel", IDCANCEL, 60, 60, 30, 14,WS_CHILD |
WS_VISIBLE | WS_TABSTOP
   AUTOCHECKBOX "Show Countdown", IDD_CB1, 1, 20, 70, 10
   AUTOCHECKBOX "Beep At End", IDD_CB2, 1, 30, 50, 10
   AUTORADIOBUTTON "Minimize", IDD_RB1, 80, 20, 50, 10
   AUTORADIOBUTTON "Maximize", IDD_RB2, 80, 30, 50, 10
   AUTORADIOBUTTON "As-Is", IDD_RB3, 80, 40, 50, 10
}
```