Window Programming 1&2

Software Branch

Forth Class

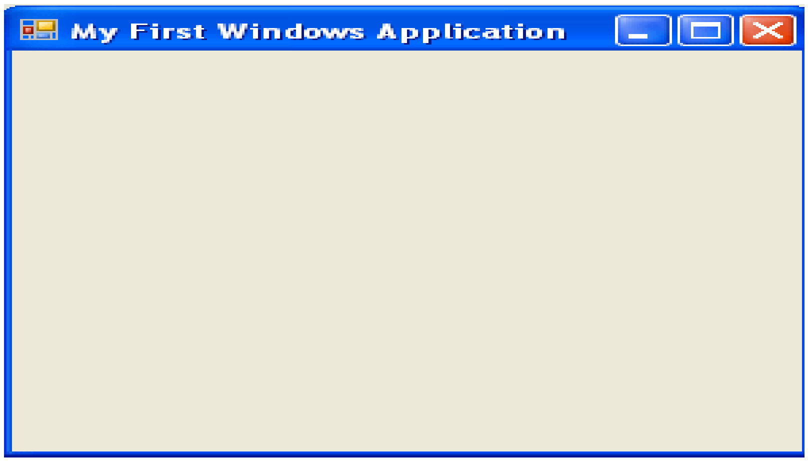
2024_2025

Dr. Yossra Hussain Ali

1- A minimal Windows skeleton

The **WinMain()** function must perform the following general steps:

1. Define a window class.
2. Register that class with Windows.
3. Create a window of that class.
4. Display the window.
5. Begin running the message loop.



2-

Icon Macro	**Shape**
IDI_APPLICATION	Default icon
IDI_ASTERISK	Information icon
IDI_EXCLAMATION	Exclamation point icon
IDI_HAND	Stop sign
IDI_QUESTION	Question mark icon
IDI_WINLOGO	Windows Logo

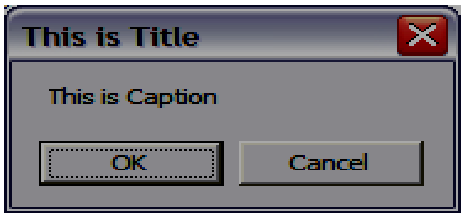
Cursor Macro	Shape
IDC_ARROW	Default arrow pointer
IDC_CROSS	Cross hairs
IDC_IBEAM	Vertical I-beam
IDC_WAIT	Hourglass

Macro Name	Background Type
BLACK_BRUSH	Black
DKGRAY_BRUSH	Dark gray
HOLLOW_BRUSH	See through window
LTGRAY_BRUSH	Light gray
WHITE_BRUSH	White
Display Macros	Effect
SW_HIDE	Removes the window
SW_MINIMIZE	Minimizes the window into an icon
SW_MAXIMIZE	Maximizes the window
SW_RESTORE	Returns a Window to normal size

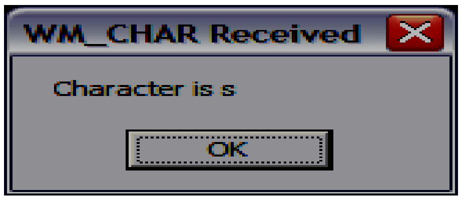
3- Message Boxes

To create a message box, use the **MessagcBox()** API function. Its prototype is shown here:

int MessageBox(HWND hwnd, LPCSTR lpText, LPCSTR lpCaption, UINT MBType);


This is Title

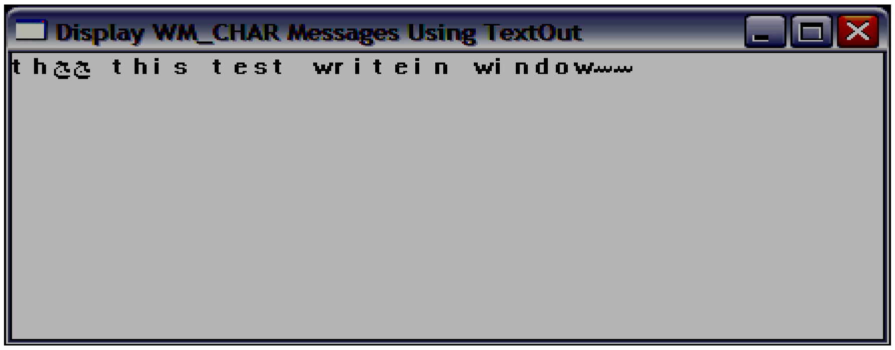
This is Caption


WM_CHAR Received

Character is s

4- Outputting Text to a Window

```
while(GetMessage(&msg, NULL, 0, 0))
        { TranslateMessage(&msg);
          DispatchMessage (&msg);   } return msg. wParam;}
 LRESULT CALLBACK WindowFunc(HWND hwnd, Uunt message,
         WPARAM wParam,  LPARAM  lParam){ HDC hdc; static unsigned j=0;
switch (message) { case WM_CHAR:
                              hdc=GetDC(hwnd);
                              sprintf(str, "%c", (char) wParam);
                             TextOut(hdc, j*10, 0, str, strlen(str));                    j++;
                              ReleaseDC(hwnd, hdc); break;
                         case WM_DESTROY:
                             PostQuitMessage(0); break;
default:  return DefWindowProc(hwnd, message, wParam, lParam);} return 0;}
```


Display WM_CHAR Messages Using TextOut

```
thaa this test writein window
```

5- Device Contexts

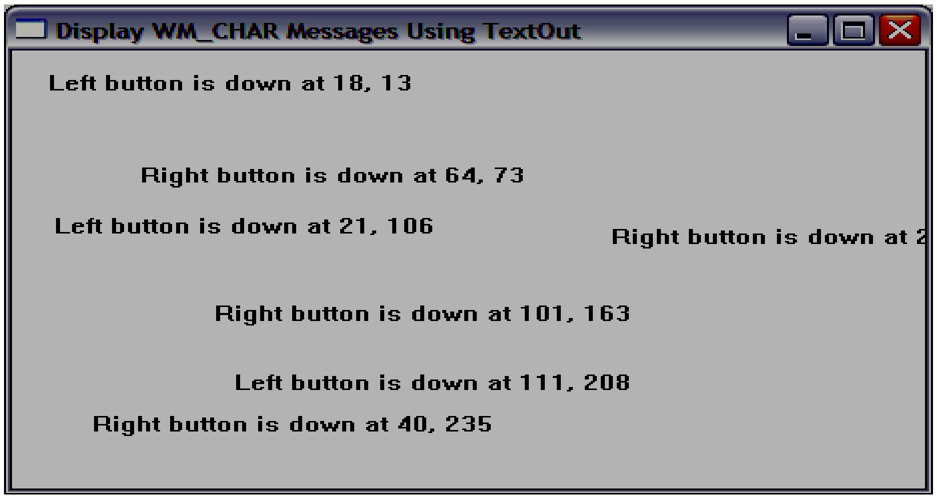
```
switch(message) { case WM_CHAR:
                        hdc = GetDC(hwnd);
                        sprintf(str, "%c", (char) wParam);
                        TextOut(hdc, j*10,0, str, strlen(str));
                         j++;
                        ReleaseDC(hwnd, hdc); break;
```

6- Generating a WM_PAINT Message

```
                case WM_PAINT:
                                hdc = BeginPaint (hwnd, &paintstruct) ;
TextOut(hdc, 0, 0, str, strlen (str ) ) ,-
                                EndPaint (hwnd, &paintstruct ) ; break;
```

7- Responding to Mouse Messages

WM_LBUTTONDOWN	WM_LBUTTONUP	WM-LBUTTONDBLCK
WM_RBUTTONDOWN	WM_RBUTTONUP	WM_RBUTTONDBLCK



```
Left button is down at 18, 13
Right button is down at 64, 73
Left button is down at 21, 106
Right button is down at 2
Right button is down at 101, 163
Left button is down at 111, 208
Right button is down at 40, 235
```

8- Responding to a Double-Click

```
switch(message) { case WM_KEYDOWN:
if{(char)wParam==VK_UP) {/*increase interval*/ interval = GetDoubleClickTime();
                                              interval + = 100; SetDoubleClickTime(interval) ;}
if((char)wParam == VK_DOWN) {
                                              interval = GetDoubleClickTime();
                                              interval -= 100;
                                              if(interval < 0) interval = 0;
                                              SetDoubleClickTime(interval); }
   sprintf(str, "New interval is %u milliseconds",interval);
    MessageBox(hwnd, str, "Setting Double-Click Interval", MB_OK); break;
      case WM_RBUTTONDOWN:
             hdc=GetDC(hwnd);
          sprintf(str,"Right   button   is   down   at   %d,   %d",LOWORD(lParam),
HIWORD(lParam));
TextOut(hdc,LOWORD(lParam),HIWORD(lParam),str,strlen(str));ReleaseDC(hwnd,hdc);
break;
    case WM_LBUTTONDOWN:
sprintf(str,"Left      button      is      down      at      %d,
%d",LOWORD(1Param),HIWORD(lParam));
TextOut(hdc, LOWORD(IParam), HIWORD(IParam), str, strlen(str) ) ;
ReleaseDC(hwnd, hdc); break;
    case WM_LBUTTONDBLCLK:
interval   = GetDoubleClickTime ( ) ;
sprintf(str,"Left ButtonXnInterval  is %u milliseconds",   interval);
MessageBox(hwnd, str, "DoubleClick", MB_OK); break;
```

9- Menus Basics

Windows supports three general types:

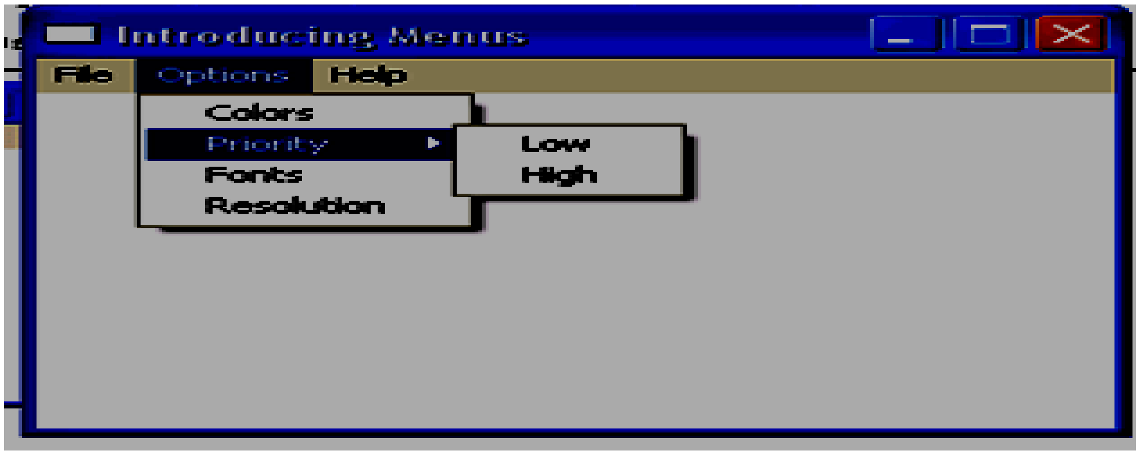
- The menu bar (or main menu)
- Pop-up submenus
- Floating, stand-alone pop-up menus

- Creating RC files

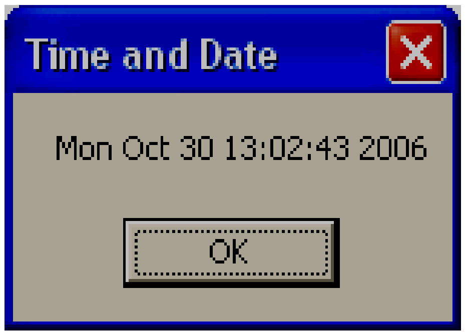
- Creating Header file

Simple Menu

```
# include "menu.h"

MyMenu MENU

{POPUP "&File" {MENUITEM "&Open", IDM_OPEN

                MENUITEM "&Close", IDM_CLOSE

                MENUITEM "&Exit", IDM_EXIT}

 POPUP "&Options" {MENUITEM "&Colors", IDM_COLORS

                   POPUP "&Priority" {MENUITEM "&Low", IDM_LOW

                                      MENUITEM "&High", IDM_HIGH}

                   MENUITEM "&Fonts", IDM_FONT

                   MENUITEM "&Resolution" , IDM_RESOLUTION}

 MENUITEM "&Help", IDM_HELP}
```



Non-Menu Accelerator Keys



Mon Oct 30 13:02:43 2006

Overriding the Class Menu

# include "menu.h"

Placeholder class menu.

PlaceHolder MENU

{POPUP "&File"

{MENUITEM "&Exit\t Ctrl-X", IDM_EXIT}

MENUITEM "&Help", IDM_HELP}

; Menu used by CreateWindow.

MyMenu MENU

10- Dialog Boxes Use Controls

Activating a Dialog Box

- **push button**

To activate a modal dialog box (that is, to cause it to be displayed) you must call the DialogBox( ) API function, whose prototype is shown here:

```
int DialogBox(HINSTANCE hThisInst, LPCSTR lpName, HWND hwnd,
                              DLGPROC lpDFunc);
```

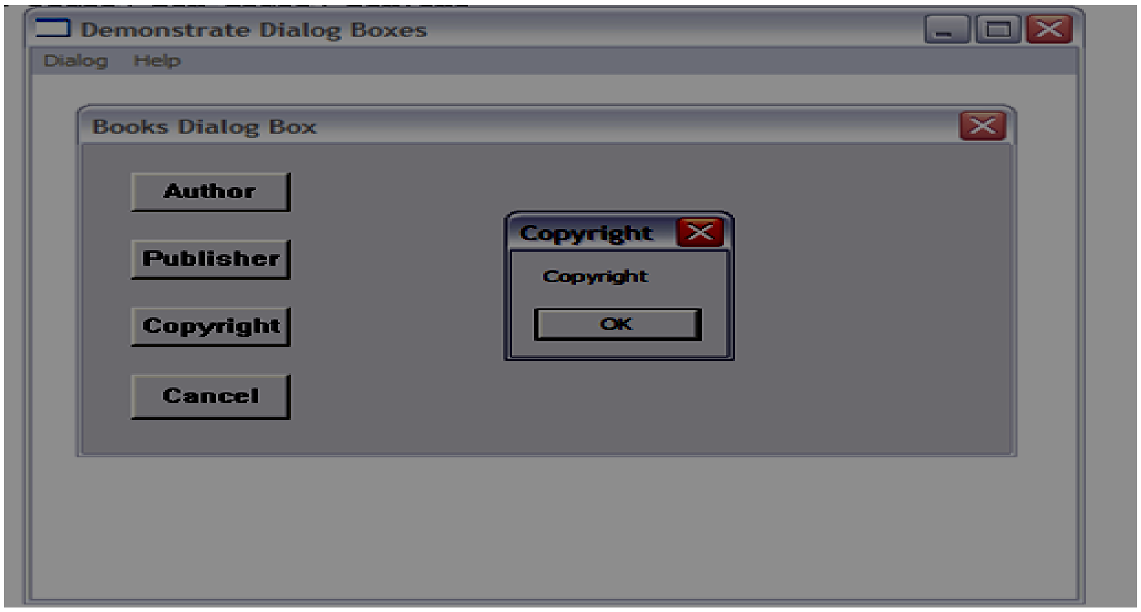
11- Creating a Simple Dialog Box

Dialog-name DIALOG [DISCARDABLE] *X, Y, Width, Height*

Features

{ Dialog-items }

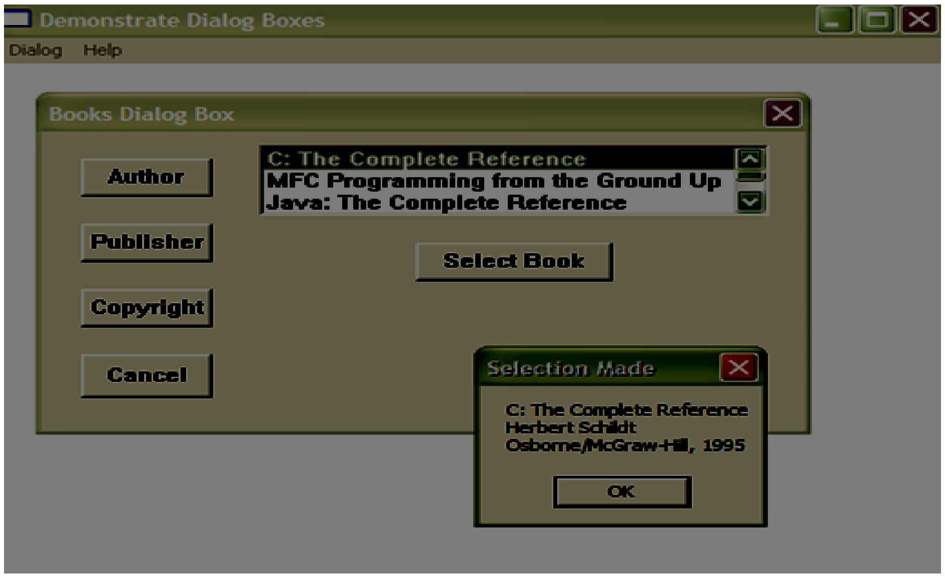
```
  # include <windows.h>
  # include "dialog.h"
  MyMenu MENU
  {POPUP "&Dialog" {MENUITEM "&Dialog\tF2", IDM_DIALOG
                                       MENUITEM "&Exit\tF3", IDM_EXIT }
   MENUITEM "&Help", IDM_HELP }
   MyMenu ACCELERATORS
   {   VK_F2, IDM_DIALOG, VIRTKEY
       VK_F3, IDM_EXIT, VIRTKEY
       VK_F1, IDM_HELP, VIRTKEY   }
MyDB DIALOG 10, 10, 210, 110
CAPTION "Books Dialog Box"
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
{  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   }
```



12- List Box Basics

```
case IDD_LB1: /* process a list box LBN_DBLCLK */
   if(HIWORD(wParam)==LBN_DBLCLK)
    { i = SendDlgltemMessage(hdwnd, IDD_LBl, LB_GETCURSEL, 0, 0);  /* get index */
      sprintftstr, "%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);
    SendDlgltemMessage(hdwnd, IDD_LB1, LB_GETTEXT, i, (LPARAM) str); } return 1;
case IDD_SELECT: /* Select Book button has been pressed */
 i=SendDlgltemMessage(hdwnd, IDD_LB1, LB_GETCURSEL, 0, 0);
sprintf(str, "%s\n%s\n%s, %u", books[i].title, books[i].author, books[i].publisher, books[i].copyright);
```

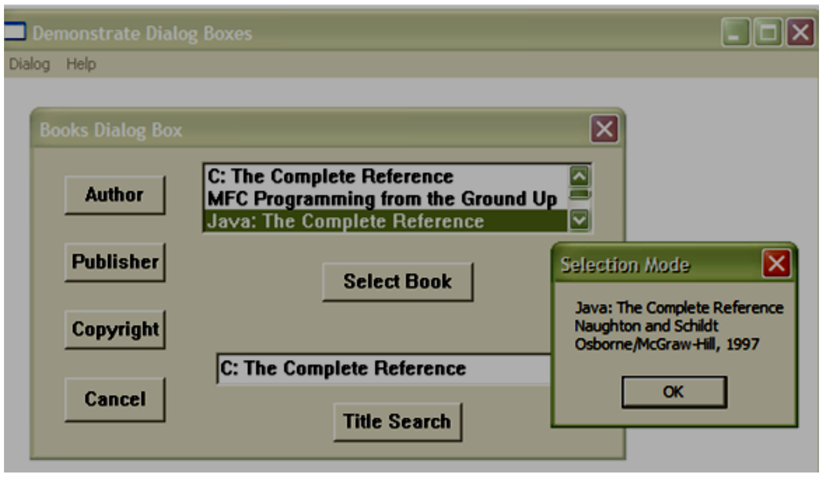
```
 MessageBox(hdwnd, str, "Selection Made", MB_OK);
SendDlgItemMessage (hdwnd, IDD_LB1, LB_GETTEXT, i, (LPARAM) str); return 1;
```



C: The Complete Reference
Herbert Schildt
Osborne/McGraw-Hill, 1995

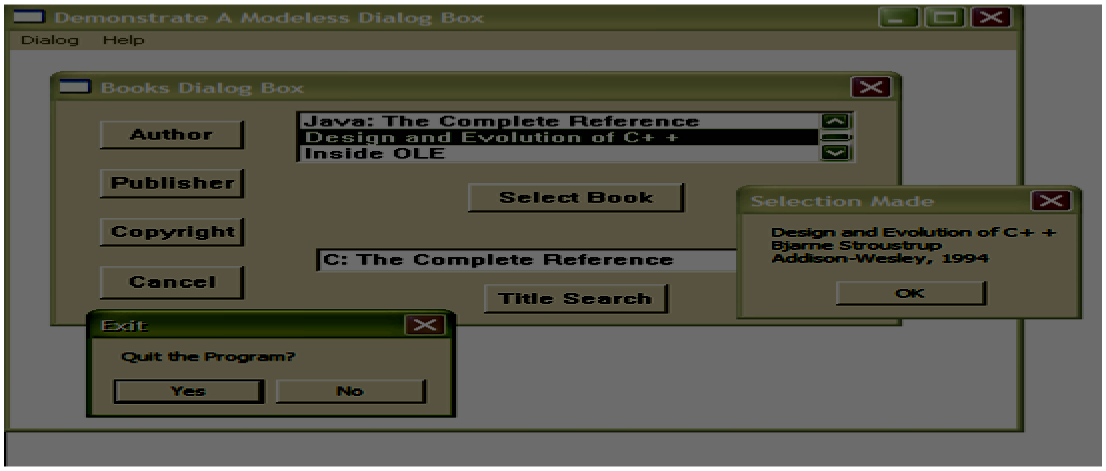
13- Adding an Edit Box

```
MyDB DIALOG 10, 10, 210, 110
CAPTION "Books Dialog Box"
STYLE DS_MODALFRAME | WS_POPUP |WS_CAPTION | WS_SYSMENU
{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_BORDER |WS_VSCROLL | 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_TAB5TOP
PUSHBUTTON "Title Search", IDD_DONE, 107, 91, 46, 14, WS_CHILD |
    WS_VISIBLE | WS_TABSTOP}
```



Java: The Complete Reference
Naughton and Schildt
Osborne/McGraw-Hill, 1997

14- Creating a Modeless Dialog Box


Demonstrate A Modeless Dialog Box
Dialog
Help
Books Dialog Box
Author
Publisher
Copyright
Cancel
Java: The Complete Reference
Design and Evolution of C+ +
Inside OLE
Select Book
C: The Complete Reference
Title Search
Selection Made
OK
Exit
Yes
No

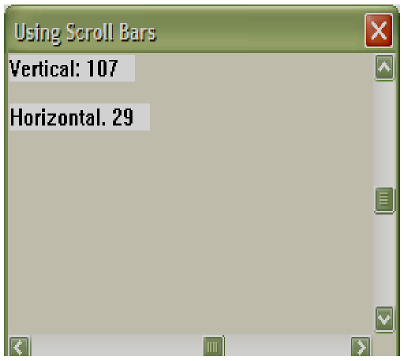
Design and Evolution of C+ +
Bjarne Stroustrup
Addison-Wesley, 1994

Quit the Program?

15- Activating the Standard Scroll Bars

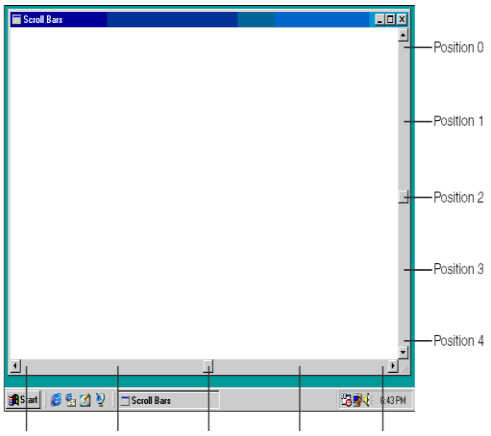
```
case WM_HSCROLL:  switch(LOWORD(wParam)){/*Try adding the other event
handling code for the horizontal scroll bar, here. */
```

```
      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;}
si.fMask = SIF_POS;  si.nPos = hpos ;
SetScrollInfo(hdwnd, SB_HORZ, &si, 1);hdc = GetDC(hdwnd);
 sprintf(str, "Horizontal-. %d   ", hpos); TextOut(hdc, 1, 30, str, strlen(str));
ReleaseDC(hdwnd, hdc);  return 1;} return 0;}
```

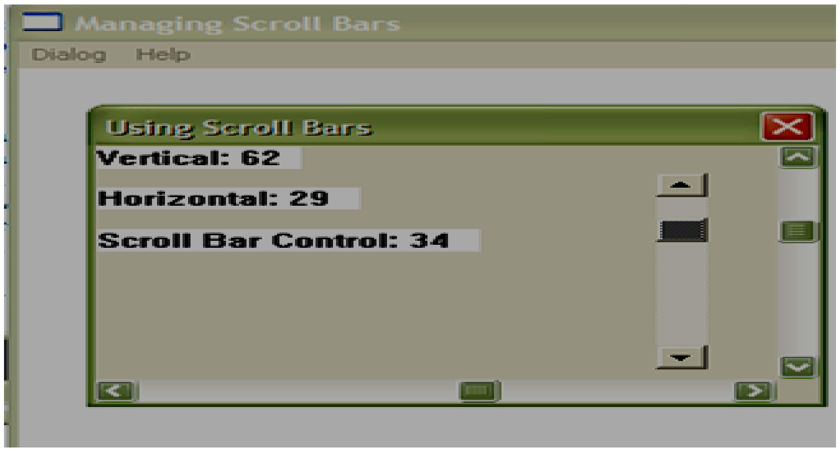


Vertical: 107

Horizontal. 29



16- Sample to try adding a horizontal control scroll bar



Vertical: 62

Horizontal: 29

Scroll Bar Control: 34

17- Check Boxes

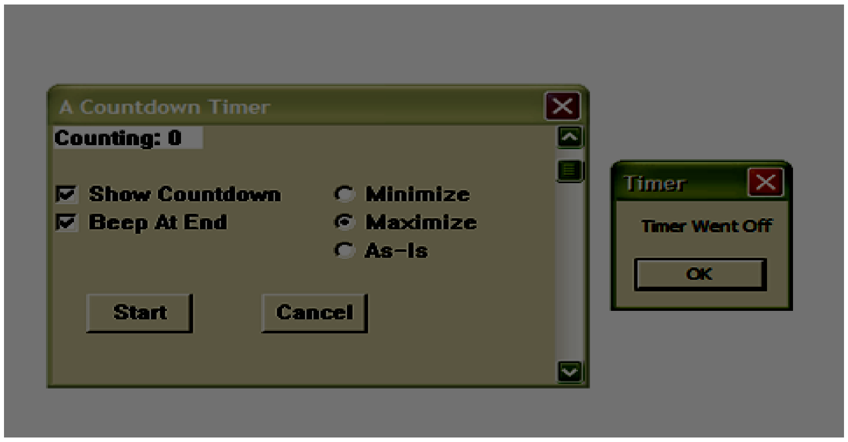
```
        CHECKBOX "string", CBID, X, Y, Width, Height [, Style]

        AUTOCHECKBOX "string", CBID, X, Y, Width, Height [, Style]

#include  "cd.h"

#include <windows.h>

MyMenu MENU{POPUP "&Dialog"{MENUITEM "&Timer\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, 152, 92 CAPTION "A Countdown Timer"

STYLE DS_MODALFRAME | WS_POPUP | WS_VSCROLL |WS_CAPTICN | 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
```

18- Radio Buttons

```
                        AUTORADIOBUTTON "string", RB1D, X, Y, Width, Height [, Style]


AUTORADIOBUTTON "Minimize", IDD_RB1, 80, 20, 50, 10

 AUTORADIQBUTTON "Maximize", IDD_RB2, 80, 30, 50, 10

 AUTORADIOBUTTON "As-Is", IDD_RB3, 80, 40, 50, 10}
```

19- The Countdown Timer Program

```
case WM_TIMER: if(t==0) {KillTimer(hdwnd, DD_TIMER);
            if(SendDlgItemMessage(hdwnd,IDD_CB2,BM_GETCHECK,0,0)==BST_CHECKED)
            MessageBeep (MB_OK) ;  MessageBox(hdwnd, "Timer Went Off", "Timer", MB_OK);
       ShowWindow(hwnd, SW_RESTORE); return 1;}       t- -;
```



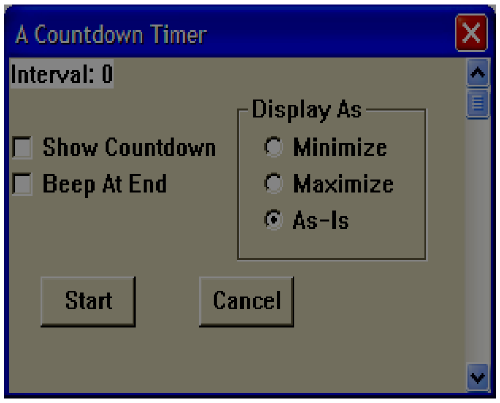
The Countdown Timer Resource and Header Files / Demonstrate scroll bars, check boxes, and radio buttons.

```
#include  "cd.h"

#include <windows.h>

MyMenu MENU{POPUP "&Dialog"{MENUITEM "&Timer\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, 152, 92 CAPTION "A Countdown Timer"
STYLE DS_MODALFRAME | WS_POPUP | WS_VSCROLL |WS_CAPTICN | 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
 AUTORADIQBUTTON "Maximize", IDD_RB2, 80, 30, 50, 10
 AUTORADIOBUTTON "As-Is", IDD_RB3, 80, 40, 50, 10}
```

The header file required by the timer program is shown here. Call this file CD.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_RE2  403
#define            IDD_RB3  404
```



20- Static Controls

CTEXT *"text", ID, X, Y, Width, Height [, Style]*

RTEXT *'text", ID, X, Y, Width, Height [, Style]*

LTEXT *"text", ID, X, Y, Width, Height* [, *Style]*

21- Stand Alone Controls

```
hsbwnd = CreateWindow(
        "SCROLLBAR",
        " ", /* no title */
        SBS_HORZ | WS_CHILD | WS_VISIBLE, /* horizontal scroll bar */
        10, 10, /* position */
       120, 20, /* dimensions
   hwnd, /* parent window */
   NULL, /* no control ID needed for scroll bar */
   hThisInst, NULL
```