

```
1  #define UNICODE
2  #include<windows.h>
3
4  // global function declarations
5  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
6
7  // ThreadProc() functions
8  DWORD WINAPI MyThreadProcOne(LPVOID);
9  DWORD WINAPI MyThreadProcTwo(LPVOID);
10
11 int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
12                   LPSTR lpCmdLine, int nCmdShow)
13 {
14     WNDCLASSEX wndclass;
15     HWND hwnd;
16     MSG msg;
17     TCHAR AppName[] = TEXT("MULTITHREADING");
18
19     wndclass.cbSize = sizeof(wndclass);
20     wndclass.style = CS_HREDRAW | CS_VREDRAW;
21     wndclass.cbClsExtra = 0;
22     wndclass.cbWndExtra = 0;
23     wndclass.lpfnWndProc = WndProc;
24     wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
25     wndclass.hCursor = LoadCursor(NULL, IDC_ARROW);
26     wndclass.hIconSm = LoadIcon(NULL, IDI_APPLICATION);
27     wndclass.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);
28     wndclass.hInstance = hInstance;
29     wndclass.lpszClassName = AppName;
30     wndclass.lpszMenuName = NULL;
31
32     RegisterClassEx(&wndclass);
33
34     hwnd = CreateWindow(AppName,
35                        TEXT("Example Of Multithreading"),
36                        WS_OVERLAPPEDWINDOW,
37                        CW_USEDEFAULT,
38                        CW_USEDEFAULT,
39                        CW_USEDEFAULT,
40                        CW_USEDEFAULT,
41                        NULL,
42                        NULL,
43                        hInstance,
44                        NULL);
45
46     ShowWindow(hwnd, nCmdShow);
47     UpdateWindow(hwnd);
48
49     while (GetMessage(&msg, NULL, 0, 0)) {
50         TranslateMessage(&msg);
51         DispatchMessage(&msg);
52     }
```

```

53     //Previously for Visual Studio6
54     //return (msg.wParam);
55     return ((int)msg.wParam);
56 }
57
58 // Window Procedure
59 LRESULT CALLBACK WndProc(HWND hwnd,UINT iMsg, WPARAM wParam, LPARAM lParam)
60 {
61     HANDLE hThread1,hThread2;
62     DWORD dwID1,dwID2;
63
64     switch (iMsg) {
65         case WM_CREATE:
66             hThread1=CreateThread(NULL,
67                                   0,
68                                   (LPTHREAD_START_ROUTINE)MyThreadProcOne,
69                                   (LPVOID)hwnd,
70                                   0,
71                                   &dwID1);
72
73             hThread2=CreateThread(NULL,
74                                   0,
75                                   (LPTHREAD_START_ROUTINE)MyThreadProcTwo,
76                                   (LPVOID)hwnd,
77                                   0,
78                                   &dwID2);
79             break;
80
81         case WM_DESTROY:
82             PostQuitMessage(0);
83             break;
84     }
85
86     return (DefWindowProc(hwnd,iMsg,wParam,lParam));
87 }
88
89 DWORD WINAPI MyThreadProcOne(LPVOID param)
90 {
91     HDC hdc;
92     int i;
93     TCHAR str[255];
94
95     hdc = GetDC((HWND)param);
96     for (i = 0; i <= 32767; i++) {
97         //Previously for Visual Studio6
98         //wsprintf(str,"Thread 1 -> Increasing Order Output = %d",i);
99         wsprintf(str,TEXT("Thread 1 -> Increasing Order Output = %d"),i);
100         TextOut(hdc,5,5,str,lstrlen(str));
101     }
102     ReleaseDC((HWND)param,hdc);
103     return(0);
104 }

```

```
105
106  DWORD WINAPI MyThreadProcTwo(LPVOID param)
107  {
108      HDC hdc;
109      int i;
110      TCHAR str[255];
111
112      hdc = GetDC((HWND)param);
113      for (i = 32767; i >= 0; i--) {
114          //Previously for Visual Studio6
115          //wsprintf(str,"Thread 2 -> Decreasing Order Output = %d",i);
116          wsprintf(str,TEXT("Thread 2 -> Decreasing Order Output = %d"),i);
117          TextOut(hdc,5,25,str,lstrlen(str));
118      }
119      ReleaseDC((HWND)param,hdc);
120      return(0);
121  }
122
```