```c
#define UNICODE
#include<windows.h>

// global function declarations
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

// ThreadProc() functions
DWORD WINAPI MyThreadProcOne(LPVOID);
DWORD WINAPI MyThreadProcTwo(LPVOID);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine, int nCmdShow)
{
    WNDCLASSEX wndclass;
    HWND hwnd;
    MSG msg;
    TCHAR AppName[] = TEXT("MULTITHREADING");

    wndclass.cbSize = sizeof(wndclass);
    wndclass.style = CS_HREDRAW | CS_VREDRAW;
    wndclass.cbClsExtra = 0;
    wndclass.cbWndExtra = 0;
    wndclass.lpfnWndProc = WndProc;
    wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
    wndclass.hCursor = LoadCursor(NULL, IDC_ARROW);
    wndclass.hIconSm = LoadIcon(NULL, IDI_APPLICATION);
    wndclass.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);
    wndclass.hInstance = hInstance;
    wndclass.lpszClassName = AppName;
    wndclass.lpszMenuName = NULL;

    RegisterClassEx(&wndclass);

    hwnd = CreateWindow(AppName,
                    TEXT("Example Of Multithreading"),
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                CW_USEDEFAULT,
                        NULL,
                        NULL,
                hInstance,
                NULL);

    ShowWindow(hwnd,nCmdShow);
    UpdateWindow(hwnd);

    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    //Previously for Visual Studio6
    //return (msg.wParam);
    return ((int)msg.wParam);
}

// Window Procedure
LRESULT CALLBACK WndProc(HWND hwnd ,UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    HANDLE hThread1,hThread2;
      DWORD dwID1,dwID2;

    switch (iMsg) {
        case WM_CREATE:
            hThread1=CreateThread(NULL,
                                     0,
                              (LPTHREAD_START_ROUTINE)MyThreadProcOne,
                              (LPVOID)hwnd,
                               0,
                              &dwID1);

            hThread2=CreateThread(NULL,
                                     0,
                              (LPTHREAD_START_ROUTINE)MyThreadProcTwo,
                              (LPVOID)hwnd,
                              0,
                              &dwID2);
            break;

        case WM_DESTROY:
            PostQuitMessage(0);
            break;
    }

    return (DefWindowProc(hwnd,iMsg,wParam,lParam));
}

DWORD WINAPI MyThreadProcOne(LPVOID param)
{
    HDC hdc;
    int i;
    TCHAR str[255];

    hdc = GetDC((HWND)param);
    for (i = 0; i <= 32767; i++) {
        //Previously for Visual Studio6
        //wsprintf(str,"Thread 1 -> Increasing Order Output  = %d",i);
        wsprintf(str,TEXT("Thread 1 -> Increasing Order Output  = %d"),i);
        TextOut(hdc,5,5,str,lstrlen(str));
    }
    ReleaseDC((HWND)param,hdc);
    return(0);
}
```

```c

DWORD WINAPI MyThreadProcTwo(LPVOID param)
{
    HDC hdc;
    int i;
    TCHAR str[255];

    hdc = GetDC((HWND)param);
    for (i = 32767; i >= 0; i--) {
        //Previously for Visual Studio6
        //wsprintf(str,"Thread 2 -> Decreasing Order Output = %d",i);
        wsprintf(str,TEXT("Thread 2 -> Decreasing Order Output = %d"),i);
        TextOut(hdc,5,25,str,lstrlen(str));
    }
    ReleaseDC((HWND)param,hdc);
    return(0);
}

```