**VC中窗口ID，句柄，指针三者相互转换函数**

ID--HANDLE--HWND三者之间的互相转换  
id->句柄        hWnd = ::GetDlgItem(hParentWnd,id);  
id->指针        CWnd::GetDlgItem();  
句柄->id        id = GetWindowLong(hWnd,GWL\_ID);  
句柄->指针    CWnd \*pWnd=CWnd::FromHandle(hWnd);  
指针->ID       id = GetWindowLong(pWnd->GetSafeHwnd,GWL\_ID);  
                                            GetDlgCtrlID();  
指针->句柄     hWnd=cWnd.GetSafeHandle() or mywnd->m\_hWnd;

**指针的使用**在编程过程中至关重要，恰到好处并能正确无误的使用指针不但能够提高程序自身的运行效率，而且有助于节省程序执行所需要消耗的资源。指针对应着某个数据在内存空间中的地址，得到了指针就可以自由地修改该数据。**句柄**代表指针的“指针”，也可以将其比作表中数据项的索引值( 表对应某个进程自身的内存空间 )。句柄是间接的引用对象。

**指针和句柄的不同之处**：

* 句柄所指的可以是一个很复杂的结构，并且很有可能与系统有关的，比如上面所说线程的句柄，它指向的就是一个类或者结构，它和系统有很密切的关系。当一个线程由于不可预料的原因而终止时，系统就可以通过句柄来回收它所占用的资料，如CPU，内存等等。反过来想，这些句柄中的某一些，是与系统进行交互用的。
* 指针它也可以指向一个复杂的结构，但通常是由用户自我定义的，所以一些必需的工作都要由用户自己完成，特别是在删除的时候。
* **另外需要注意的是**句柄往往有自己的存在区限，比如一个进程，如果将其传递到另一个进程中，句柄也就失去了意义( **表中数据项的索引值**，索引离开了具体的表也就失去了意义 )。

**具体转换**：

        ( 句柄转为指针 )

        CWnd\* pWnd=FromeHandle(hMyHandle);   
        pWnd->SetWindowText("Hello World!");   
  
        or   
  
        CWnd\* pWnd;   
        pWnd->Attach(hMyHandle);

        MFC类中有的还提供了标准方法，比如Window句柄：

        static CWnd\* PASCAL FromHandle(   
        HWND hWnd   
        );   
        HWND GetSafeHwnd( ) const;   
  
        对于位图：   
        static CBitmap\* PASCAL FromHandle(   
       HBITMAP hBitmap   
         );    
         static CGdiObject\* PASCAL FromHandle(   
       HGDIOBJ hObject   
         );   
        HGDIOBJ GetSafeHandle( ) const;

        当然，更详细的信息需要在具体使用中自我查询。

**建 议**：指针和句柄的使用属于比较复杂、危险性较高的应用，在具体实践中，如果可以，尽量不要使用指针和句柄，最好选择现有的、封装完好的方式来实现，更别提指针同句柄的转换了，它更加危险。

        比如在操作字符串时，尽量使用CString类来实现，通过定义好的构造、析构函数来完成分配和回收，最好不要通过指针来动态操作。

(转摘)