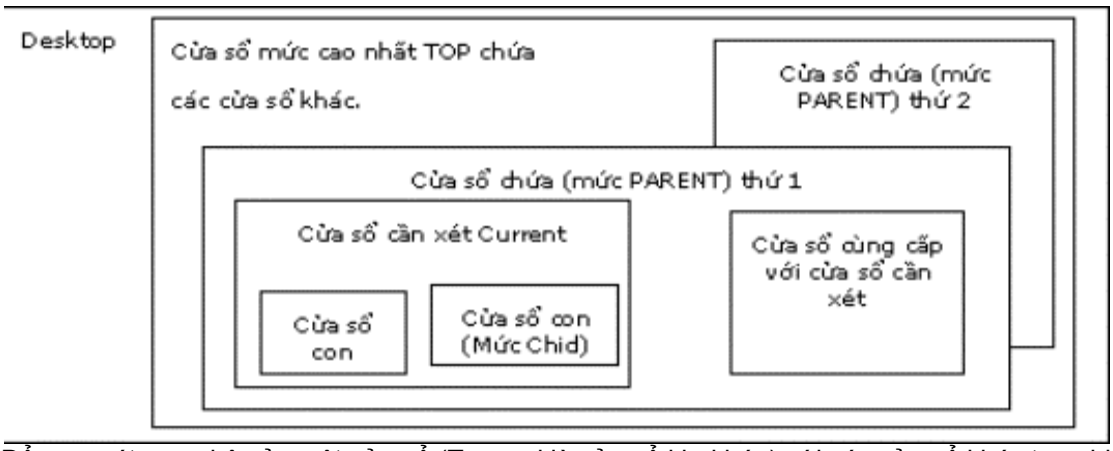
Các hàm API liên quan đến cửa sổ

Trước khi tìm hiểu tiếp phần tiếp theo của API, tôi xin phép được gửi tới các bạn công dụng của các hàm API thông dụng, sau đó chúng ta sẽ tiếp tục nghiên cứu cách sử dụng nó.

Phần 1: Các hàm API liên quan đến cửa sổ



Để xem xét quan hệ của một cửa sổ (Tạm gọi là cửa sổ khai báo) với các cửa sổ khác ta nghiên cứu các mối quan hệ sau:

1. Declare Function AnyPopup Lib "user32" Alias "AnyPopup" () As Long

Công dụng: Đưa ra chỉ số cửa sổ popup hiện đang tồn tại trên màn hình.
Trị trả về: Integer ~ True (Khác zero) nếu có cửa sổ popup.

2. Declare Function AdjustWindowRect Lib "user32" Alias "AdjustWindowRect" (lpRect As RECT, ByVal dwStyle As Long, ByVal bMenu As Long) As Long
3. Declare Function AdjustWindowRectEx Lib "user32" Alias "AdjustWindowRectEx" (lpRect As RECT, ByVal dsStyle As Long, ByVal bMenu As Long, ByVal dwEsStyle As Long) As Long

Công dụng: Điều chỉnh cửa sổ khi có vùng làm việc client (Không tính kích thước của thanh tiêu đề, đường viền và các phần thêm) được khai báo, khi biết kiểu cửa sổ.
Tham số kèm:
LpRect Hình chữ nhật chứa vùng làm việc client.
DwStyle Kiểu cửa sổ.
BMenu Đưa giá trị True (Khác zero) nếu cửa sổ có trình đơn
DwEsStyle kiểu cửa sổ mở rộng.

4. Declare Function ArrangeIconicWindows Lib "user32" Alias "ArrangeIconicWindows" (ByVal hwnd As Long) As Long

Công dụng: Xếp các biểu tượng cửa sổ trong một cửa sổ chứa (Mức Parent).
Trị trả về: Integer chiều cao của hàng biểu tượng. Zero nếu thất bại.
Tham số kèm:
HWnd Cán của cửa sổ chứa (Mức Parent).

5. Declare Function BeginDeferWindowPos Lib "user32" Alias "BeginDeferWindowPos" (ByVal nNumWindows As Long) As Long

Công dụng: Bắt đầu xây dựng danh sách vị trí các cửa sổ mới thành cấu trúc bản đồ nội bộ chứa vị trí các cửa sổ.
Trị trả về: Integer - cán của cấu trúc bản đồ. Zero nếu thất bại.
Tham số kèm:
NNum Windows Số cửa sổ ban đầu để cấp phát chỗ trống.

6. Declare Function DeferWindowPos Lib "user32" Alias "DeferWindowPos" (ByVal hWinPosInfo As Long, ByVal hwnd As Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long

Công dụng: Đinh nghĩa vị trí của cửa sổ mới qua cửa sổ khai báo và đưa vào cấu trúc bản đồ nội bộ chứa vị trí các cửa sổ.
Trị trả về: Integer - Cán mới đối với cấu trúc bản đồ chứa thông tin cập nhật vị trí. Zero nếu thất bại.
Tham số kèm:
HWinPosInfo Cán của cấu trúc bản đồ.
HWnd Cửa sổ cần định vị.
HWndInsertAfter Cán cửa sổ mà cửa sổ hWnd đặt sau nó trong danh sách. Nó có thể là một trong các hằng sau:
HWnd_BOTTOM: Đặt về cuối danh sách.
HWnd_TOP: Đặt cửa sổ ở đầu danh sách
HWnd_TPMOST: Đặt cửa sổ ở đầu danh sách lên trên cùng nhìn thấy được.
X Hoành độ của cửa sổ hWnd theo toạ độ của cửa sổ chứa (Mức Parent) nó.
Y Tung độ của cửa sổ hWnd theo toạ độ cửa sổ chứa (Mức Parent) nó.
cx Chiều rộng cửa sổ mới.
cy Chiều cao cửa sổ mới.
Flags Một số nguyên là một trong các hằng sau:
SWP_DRAWFRAME: Vẽ khung bao quanh cửa sổ.
SWp-HIDEWINDOW: Giấu cửa sổ.
SWP_NOACTIVE: Không kích hoạt cửa sổ.
SWP_NOMOVE: Giữ nguyên vị trí hiện tại.
SWP_NOREDRAW: Không vẽ lại tự động.
SWp_NOSIZE: Giữ nguyên kích thước.
SWp_NOZORDER: Giữ nguyên vị trí hiện hành trong danh sách.

7. Declare Function SetWindowPos Lib "user32" Alias "SetWindowPos" (ByVal hwnd As Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long

Công dụng: Thiết đặt vị trí và trạng thái cửa sổ.
Tham số kèm:
HWnd Cán của cửa sổ cần định vị
HWndInsertAfter Như hàm trên.

8. Declare Function EndDeferWindowPos Lib "user32" Alias "EndDeferWindowPos" (ByVal hWinPosInfo As Long) As Long

Công dụng: Cập nhật các vị trí và tình trạng của tất cả các cửa sổ.
Tham số kèm:
HWinPosInfo Cán của cấu trúc bản đồ lấy từ lệnh DerefWindowPos gần nhất.

9. Declare Function BringWindowToTop Lib "user32" Alias "BringWindowToTop" (ByVal hwnd As Long) As Long

Công dụng: Chuyển cửa sổ lên đầu danh sách làm lộ ra nếu bị khuất.
Tham số kèm:
HWnd Cán của cửa sổ cần tác động.

10. Declare Function ChildWindowFromPoint Lib "user32" Alias "ChildWindowFromPoint" (ByVal hWnd As Long, ByVal xPoint As Long, ByVal yPoint As Long) As Long
11. Declare Function ChildWindowFromPoint Lib "user32" Alias "ChildWindowFromPoint" (ByVal hWndParent As Long, ByVal pt As POINTAPI) As Long

Công dụng: Lấy cán của cửa sổ con (Mức Child) khi đưa điểm của cửa sổ chứa (Mức Parent) nó.
Trị trả về: Integer - Cán của cửa sổ con (Mức Child) đầu tiên thoả mãn. Nếu không thấy cửa sổ con (Mức Child) nào trả về cán của cửa sổ chứa (Mức Parent). Zero nếu điểm nằm ngoài cửa sổ chứa (Mức Parent).
Tham số kèm:
HWnd Cán của cửa sổ chứa (Mức Parent).
Pt Trị của điểm.
XPoint Hoành độ của điểm.
YPoint Tung độ của điểm.

12. Declare Function ClientToScreen Lib "user32" Alias "ClientToScreen" (ByVal hwnd As Long, lpPoint As POINTAPI) As Long

Công dụng: Chuyển toạ độ theo cửa sổ sang toạ độ theo màn hình.

Tham số kèm:

HWnd Cán của cửa sổ làm căn cứ xác định toạ độ.

LpPoint Ddieemr tính theo toạ độ cửa sổ

1. Declare Function CloseWindow Lib "user32" Alias "CloseWindow" (ByVal hwnd As Long) As Long

Công dụng: Thu nhỏ cửa sổ.

Tham số kèm:

HWnd Cán của cửa sổ cần thu.

2. Declare Function CopyRect Lib "user32" Alias "CopyRect" (lpDestRect As RECT, lpSourceRect As RECT) As Long

Công dụng: Sao nội dung hình chữ nhật.

Tham số kèm:

lpDestRect Hình chữ nhật đích sẽ nhận kết quả.

LpSourceRect Hình chữ nhật nguồn bị copy.

3. Declare Function DestroyWindow Lib "user32" Alias "DestroyWindow" (ByVal hwnd As Long) As Long

Công dụng: Phá huỷ cửa sổ (Kể cả các cửa sổ con (Mức Child) của nó).

Trị trả về: Integer khác 0 sẽ thành công. Zero nếu thất bại.

Tham số kèm:

HWnd Cán của cửa sổ sẽ phá huỷ.

4. Declare Function EnableWindow Lib "user32" Alias "EnableWindow" (ByVal hwnd As Long, ByVal fEnable As Long) As Long

Công dụng: Cho hiệu lực hay vô hiệu hoá mọi dữ liệu nhập vào cửa sổ từ bàn phím hoặc chuột.

Trị trả về: Integer True (Khác zero) nếu trước đó cửa sổ được phép. Zero nếu bị vô hiệu hoá.

Tham số kèm:

HWnd Cán của cửa sổ

FEnable Giá trị logic. Nếu là True, thì Window sẽ có hiệu lực Enable. Còn False, sẽ không có hiệu lực Disable.

Declare Function EnumChildWindows Lib "user32" Alias "EnumChildWindows" (ByVal hWndParent As Long, ByVal lpEnumFunc As Long, ByVal lParam As Long) As Long

Công dụng: Liệt kê các cửa sổ con (Mức Child) của một cửa sổ chứa (Mức Parent). Phải có Custom Control CBK.VBX mới sử dụng được.

Trị trả về: Integer True (Khác zero) nếu thành công. Zero nếu thất bại.

Tham số kèm:

HWndParent Cán của cửa sổ chứa (Mức Parent) cần liệt kê

LpEnumFunc Biến trỏ chỉ đến hàm để gọi đối với mỗi cửa sổ con (Mức Child). Sử dụng tính chất ProcAddress của Custon Control CBK.VBX để nhận hàm biến trỏ (function pointer) để gọi lại (callbacks).

LParam Trị chuyển đến cho sự kiện EnumWindows của Custom Control trong lúc liệt kê. Ý nghĩa của trị này do lập trình viên xác định.

5. Declare Function EnumWindowStations Lib "user32" Alias "EnumWindowStationsA" (ByVal lpEnumFunc As Long, ByVal lParam As Long) As Long

Công dụng: Liệt kê danh sách cửa sổ cấp trên, chứa cửa sổ khai báo. Phải có Custom Control CBK.VBX mới sử dụng được.

Trị trả về: Integer True (Khác zero) nếu thành công.

Tham số kèm:

LpEnumFunc Biến trỏ chỉ đến hàm để gọi đối với mỗi cửa sổ con (Mức Child). Sử dụng tính chất ProcAddress của Custon Control CBK.VBX để nhận hàm biến trỏ (function pointer) để gọi lại (callbacks)

LParamTrị chuyển đến cho sự kiện EnumWindows của Custom Control trong lúc liệt kê. Ý nghĩa của trị này do lập trình viên xác định.

6. Declare Function EqualRect Lib "user32" Alias "EqualRect" (lpRect1 As RECT, lpRect2 As RECT) As Long

Công dụng: So sánh 2 cấu trúc hình chữ nhật.
Trị trả về: Integer True (Khác zero) Nếu các toạ độ góc trái trên và góc phải dưới của 2 hình bằng nhau. Zero nếu khác.
Tham số kèm:
LpRec1, lpRec2: Hai hình chữ nhật cần so sánh.

7. Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName As String, ByVal lpWindowName As String) As Long

Công dụng: Tìm cửa sổ đầu tiên trong danh sách cửa sổ thoả mãn điều kiện.
Trị trả về: Integer - Cán của cửa sổ thoả mãn. Zero nếu không có cửa sổ nào.
Tham số kèm:
LpClassName Biến trỏ chỉ đến chuỗi kết thúc bằng null chứa tên lớp đối tượng đối với cửa sổ. Nếu bằng zero chấp nhận bất cứ lớp nào.
LpWindowName Biến trỏ chỉ đến chuỗi kết thúc bằng null chứa tên tiêu đề cửa sổ. Nếu bằng 0, chấp nhận bất cứ tiêu đề nào.

8. Declare Function FindWindowEx Lib "user32" Alias "FindWindowExA" (ByVal hWnd1 As Long, ByVal hWnd2 As Long, ByVal lpsz1 As String, ByVal lpsz2 As String) As Long

Công dụng: Tìm cửa sổ đầu tiên trong danh sách cửa sổ thoả mãn điều kiện.
Trị trả về: Integer - Cán của cửa sổ thoả mãn. Zero nếu không có cửa sổ nào.
Tham số kèm:
hwndParent
Cán cuả cửa sổ chứa (Cấp Parent) có các cửa sổ con để tìm.
Nếu *hwndParent* là NULL, hàm sẽ sử dụng desktop như cửa sổ chứa parent. Hàm sẽ tìm trong số các cửa sổ là cửa sổ con (Cấp Child) của desktop.
Windows 2000 trở lên: Nếu *hwndParent* là HWND_MESSAGE, hàm sẽ tìm tất cả cửa sổ dạng message-only windows.
hwndChildAfter
Là cán của cửa sổ con (cấp child). Tìm kiếm bắt đầu từ cửa sổ con kế tiếp theo thứ tự trục Z. Cửa sổ con phải là một cấp kế tiếp của *hwndParent*, không thể cấp thấp hơn.
Nếu *hwndChildAfter* là NULL, tìm kiếm sẽ bắt đầu với cửa sổ con đầu tiên(Cấp child) của *hwndParent*.
Nhớ rằng nếu cả hai *hwndParent* và *hwndChildAfter* là NULL, hàm sẽ tìm tất cả mức top của dạng message-only windows.
lpszClass Lớp cần tìm kiếm.
lpszWindow
Tiêu đề của cửa sổ cần tìm. Nếu là NULL, Tìm tất cả.

9. Declare Function FlashWindow Lib "user32" Alias "FlashWindow" (ByVal hwnd As Long, ByVal bInvert As Long) As Long

Công dụng: Chiếu sáng cử sổ, ngay cả khi nó chưa được kích hoạt (inactive)
Trị trả về: Integer True (Khác zero) nếu cửa sổ đã được kích hoạt trước khi gọi.
Tham số kèm:
HWnd Cán của cửa sổ cần chiếu sáng.
BInvert Integer - True (Khác zero) nếu bật, False để quay lại trạng thái trước

10. Declare Function GetActiveWindow Lib "user32" Alias "GetActiveWindow" () As Long

Công dụng: Nhận cán của cửa sổ đang kích hoạt.
Trị trả về: Integer - Cán của cửa sổ đang kích hoạt. Zero nếu không có.

11. Declare Function GetClassInfo Lib "user32" Alias "GetClassInfoA" (ByVal hInstance As Long, ByVal lpClassName As String, lpWndClass As WNDCLASS) As Long

Công dụng: Nhận bản sao cấu trúc Wndclass chứa thông tin về lớp khai báo.
Trị trả về: Integer - True (Khác zero) khi thành công. Zero nếu không thấy lớp thoả mãn.
Tham số kèm:
hInstance Cán của đối tượng sở hữu lớp. Dùng NULL để nhận thông tin về các lớp Windows chuẩn.

LpClassName Tên của lớp cần tìm. Có thể dùng ID resource.
LpWndClass WndCLASS - Cấu trúc để chứa kết quả.

12. Declare Function GetClassLong Lib "user32" Alias "GetClassLongA" (ByVal hwnd As Long, ByVal nIndex As Long) As Long

Công dụng: Lấy thông tin lớp.
Trị trả về: Tuỳ theo yêu cầu.
Tham số kèm:
HWnd Cán của cửa sổ để nhận thông tin đối với lớp chứa nó.
NIndex Thông tin cần nhận. Nếu là GLC_MENUNAME lấy tên hay resource ID đối với trình đơn của lớp. Nếu là GLC_WNDPROC để nhận vị trí của hàm cửa sổ lớp (Hàm đờ phôn đối với các cửa sổ trong lớp).

Declare Function CloseWindow Lib "user32" Alias "CloseWindow" (ByVal hwnd As Long) As Long

Công dụng: Thu nhỏ cửa sổ.
Tham số kèm:
HWnd Cán của cửa sổ cần thu.

2. Declare Function CopyRect Lib "user32" Alias "CopyRect" (lpDestRect As RECT, lpSourceRect As RECT) As Long

Công dụng: Sao nội dung hình chữ nhật.
Tham số kèm:
lpDestRect Hình chữ nhật đích sẽ nhận kết quả.
LpSourceRect Hình chữ nhật nguồn bị copy.

3. Declare Function DestroyWindow Lib "user32" Alias "DestroyWindow" (ByVal hwnd As Long) As Long

Công dụng: Phá huỷ cửa sổ (Kể cả các cửa sổ con (Mức Child) của nó).
Trị trả về: Integer khác 0 sẽ thành công. Zero nếu thất bại.
Tham số kèm:
HWnd Cán của cửa sổ sẽ phá huỷ.

4. Declare Function EnableWindow Lib "user32" Alias "EnableWindow" (ByVal hwnd As Long, ByVal fEnable As Long) As Long

Công dụng: Cho hiệu lực hay vô hiệu hoá mọi dữ liệu nhập vào cửa sổ từ bàn phím hoặc chuột.
Trị trả về: Integer True (Khác zero) nếu trước đó cửa sổ được phép. Zero nếu bị vô hiệu hoá.
Tham số kèm:
HWnd Cán của cửa sổ
FEnable Giá trị logic. Nếu là True, thì Window sẽ có hiệu lực Enable. Còn False, sẽ không có hiệu lực Disable.

Declare Function EnumChildWindows Lib "user32" Alias "EnumChildWindows" (ByVal hWndParent As Long, ByVal lpEnumFunc As Long, ByVal lParam As Long) As Long

Công dụng: Liệt kê các cửa sổ con (Mức Child) của một cửa sổ chứa (Mức Parent). Phải có Custom Control CBK.VBX mới sử dụng được.
Trị trả về: Integer True (Khác zero) nếu thành công. Zero nếu thất bại.
Tham số kèm:
HWndParent Cán của cửa sổ chứa (Mức Parent) cần liệt kê
LpEnumFunc Biến trỏ chỉ đến hàm để gọi đối với mỗi cửa sổ con (Mức Child). Sử dụng tính chất ProcAddress của Custon Control CBK.VBX để nhận hàm biến trỏ (function pointer) để gọi lại (callbacks).
LParam Trị chuyển đến cho sự kiện EnumWindows của Custom Control trong lúc liệt kê. Ý nghĩa của trị này do lập trình viên xác định.

5. Declare Function EnumWindowStations Lib "user32" Alias "EnumWindowStationsA" (ByVal lpEnumFunc As Long, ByVal lParam As Long) As Long

Công dụng: Liệt kê danh sách cửa sổ cấp trên, chứa cửa sổ khai báo. Phải có Custom Control CBK.VBX mới sử dụng được.
Trị trả về: Integer True (Khác zero) nếu thành công.
Tham số kèm:

LpEnumFunc Biến trỏ chỉ đến hàm để gọi đối với mỗi cửa sổ con (Mức Child). Sử dụng tính chất ProcAddress của Custon Control CBK.VBX để nhận hàm biến trỏ (function pointer) để gọi lại (callbacks)
LParam Trị chuyển đến cho sự kiện EnumWindows của Custom Control trong lúc liệt kê. Ý nghĩa của trị này do lập trình viên xác định.

6. Declare Function EqualRect Lib "user32" Alias "EqualRect" (lpRect1 As RECT, lpRect2 As RECT) As Long

Công dụng: So sánh 2 cấu trúc hình chữ nhật.
Trị trả về: Integer True (Khác zero) Nếu các toạ độ góc trái trên và góc phải dưới của 2 hình bằng nhau. Zero nếu khác.
Tham số kèm:
LpRec1, lpRec2: Hai hình chữ nhật cần so sánh.

7. Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName As String, ByVal lpWindowName As String) As Long

Công dụng: Tìm cửa sổ đầu tiên trong danh sách cửa sổ thoả mãn điều kiện.
Trị trả về: Integer - Cán của cửa sổ thoả mãn. Zero nếu không có cửa sổ nào.
Tham số kèm:
LpClassName Biến trỏ chỉ đến chuỗi kết thúc bằng null chứa tên lớp đối tượng đối với cửa sổ. Nếu bằng zero chấp nhận bất cứ lớp nào.
LpWindowName Biến trỏ chỉ đến chuỗi kết thúc bằng null chứa tên tiêu đề cửa sổ. Nếu bằng 0, chấp nhận bất cứ tiêu đề nào.

8. Declare Function FindWindowEx Lib "user32" Alias "FindWindowExA" (ByVal hWnd1 As Long, ByVal hWnd2 As Long, ByVal lpsz1 As String, ByVal lpsz2 As String) As Long

Công dụng: Tìm cửa sổ đầu tiên trong danh sách cửa sổ thoả mãn điều kiện.
Trị trả về: Integer - Cán của cửa sổ thoả mãn. Zero nếu không có cửa sổ nào.
Tham số kèm:
hwndParent
Cán cuả cửa sổ chứa (Cấp Parent) có các cửa sổ con để tìm.
Nếu *hwndParent* là NULL, hàm sẽ sử dụng desktop như cửa sổ chứa parent. Hàm sẽ tìm trong số các cửa sổ là cửa sổ con (Cấp Child) của desktop.
Windows 2000 trở lên: Nếu *hwndParent* là HWND_MESSAGE, hàm sẽ tìm tất cả cửa sổ dạng message-only windows.
hwndChildAfter
Là cán của cửa sổ con (cấp child). Tìm kiếm bắt đầu từ cửa sổ con kế tiếp theo thứ tự trục Z. Cửa sổ con phải là một cấp kế tiếp của *hwndParent*, không thể cấp thấp hơn.
Nếu *hwndChildAfter* là NULL, tìm kiếm sẽ bắt đầu với cửa sổ con đầu tiên(Cấp child) của *hwndParent*.
Nhớ rằng nếu cả hai *hwndParent* và *hwndChildAfter* là NULL, hàm sẽ tìm tất cả mức top của dạng message-only windows.
lpszClass Lớp cần tìm kiếm.
lpszWindow
Tiêu đề của cửa sổ cần tìm. Nếu là NULL, Tìm tất cả.

9. Declare Function FlashWindow Lib "user32" Alias "FlashWindow" (ByVal hwnd As Long, ByVal bInvert As Long) As Long

Công dụng: Chiếu sáng cử sổ, ngay cả khi nó chưa được kích hoạt (inactive)
Trị trả về: Integer True (Khác zero) nếu cửa sổ đã được kích hoạt trước khi gọi.
Tham số kèm:
HWnd Cán của cửa sổ cần chiếu sáng.
BInvert Integer - True (Khác zero) nếu bật, False để quay lại trạng thái trước

10. Declare Function GetActiveWindow Lib "user32" Alias "GetActiveWindow" () As Long

Công dụng: Nhận cán của cửa sổ đang kích hoạt.
Trị trả về: Integer - Cán của cửa sổ đang kích hoạt. Zero nếu không có.

11. Declare Function GetClassInfo Lib "user32" Alias "GetClassInfoA" (ByVal hInstance As Long, ByVal lpClassName As String, lpWndClass As WNDCLASS) As Long

Công dụng: Nhận bản sao cấu trúc Wndclass chứa thông tin về lớp khai báo.
Trị trả về: Integer - True (Khác zero) khi thành công. Zero nếu không thấy lớp thoả mãn.

Tham số kèm:
hInstance Cán của đối tượng sở hữu lớp. Dùng NULL để nhận thông tin về các lớp Windows chuẩn.
LpClassName Tên của lớp cần tìm. Có thể dùng ID resource.
LpWndClass WndCLASS - Cấu trúc để chứa kết quả.

12. Declare Function GetClassLong Lib "user32" Alias "GetClassLongA" (ByVal hwnd As Long, ByVal nIndex As Long) As Long

Công dụng: Lấy thông tin lớp.
Trị trả về: Tuỳ theo yêu cầu.
Tham số kèm:
HWnd Cán của cửa sổ để nhận thông tin đối với lớp chứa nó.
NIndex Thông tin cần nhận. Nếu là GLC_MENUNAME lấy tên hay resource ID đối với trình đơn của lớp. Nếu là GLC_WNDPROC để nhận vị trí của hàm cửa sổ lớp (Hàm đờ phôn đối với các cửa sổ trong lớp).

. Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long) As Long

Công dụng: Lấy thông tin từ cấu trúc cửa sổ.
Trị trả về: Theo yêu cầu.
Tham số kèm:
HWnd Cán của cửa sổ cần lấy thông tin.
NIndex Thông tin cần lấy, tuỳ thuộc vào các hằng sau:

GWL_EXSTYLE: Kiểu cửa sổ mở rộng.
GWL_STYLE: Kiểu cửa sổ.
GWL_WNDPROC: Vị trí của hàm xử lý cửa sổ này.
DWL_MSGRESULT: Trị được trả về bởi thông báo bên trong hàm đối thoại.
DWL_DLGPROC: Vị trí của hàm xử lý khung đối thoại đối với cửa sổ này.
DWL_USER: Được định nghĩa bởi ứng dụng.

2. Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long

Công dụng: Thiết đặt thông tin trong cấu trúc cửa sổ.
Trị trả về: Integer - Trị trước đó của dữ kiện cần đặt giá trị.
Tham số kèm:
HWnd Cán của cửa sổ để đặt thông tin.
NIndex Thông tin cần đặt. Xem hàm trên.
DwNewLong Trị mới cần đặt.

3. Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long

Công dụng: Lấy tiêu đề của cửa sổ hay nội dung của ô điều khiển.
Trị trả về: Integer - chiều dài chuỗi được lấy không tính ký tự null đứng cuối.
Tham số kèm:
HWnd Cán của cửa sổ cần lấy.
LpString Biến lưu kết quả là tên chuỗi cần lấy. Phải khai báo tối thiểu aint+1. Dùng công thức sau lấy tên chuỗi: Chuỗi = Left (lpString, len (trim(lpString)).
Aint Chiều dài chuỗi lpString

4. Declare Function GetWindowTextLength Lib "user32" Alias "GetWindowTextLengthA" (ByVal hwnd As Long) As Long

Công dụng: Lấy chiều dài của tiêu đề cửa sổ hay nội dung của một ô điều khiển.
Trị trả về: Chiều dài chuỗi cửa sổ..
Tham số kèm:
HWnd Cán của cửa sổ cần lấy.

5. Declare Function GetWindowWord Lib "user32" Alias "GetWindowWord" (ByVal hwnd As Long, ByVal nIndex As Long) As Integer

Công dụng: Lấy thông tin từ cấu trúc của cửa sổ chỉ định.
Trị trả về: Theo yêu cầu.

Tham số kèm:
HWnd Cán của cửa sổ cần lấy.
NIndex Thông tin cần lấy, phụ thuộc vào một trong các hằng:
GWW_HINSTANCE: Cán của chủ cửa sổ.
GWW_HWNDPARENT: Cán cửa sổ chứa (Mức Parent) nó.
GWW_ID: Số ID của cửa sổ con (Mức Child) bên trong khung đối thoại.

6. Declare Function SetWindowWord Lib "user32" Alias "SetWindowWord" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal wNewWord As Long) As Long

Công dụng: Đặt thông tin trong cấu trúc cửa sổ.
Trị trả về: Integer - Trị trước khi đặt của dữ liệu cần thay.
Tham số kèm:
HWnd Cán của cửa sổ cần đặt.
NIndex Như hàm trên.
DwNewWord - Trị mới cần đặt.

7. Declare Function InflateRect Lib "user32" Alias "InflateRect" (lpRect As RECT, ByVal x As Long, ByVal y As Long) As Long

Công dụng: Thay đổi kích thước của hình chữ nhật.
Tham số kèm:
LpRect Cấu trúc hình chữ nhật cần điều chỉnh
X Chiều rộng được tăng lên hay giảm đi.
Y Chiều cao tăng lên hay giảm đi.

8. Declare Function IntersectRect Lib "user32" Alias "IntersectRect" (lpDestRect As RECT, lpSrc1Rect As RECT, lpSrc2Rect As RECT) As Long

Công dụng: Nạp vào hình chữ nhật đích phần chung của 2 hình chữ nhật đơn.
Trị trả về: Integer (Khác zero)- Nếu hình chữ nhật đích không rỗng. Zero nếu rỗng.
Tham số kèm:
LpDestRect - Hình chữ nhật đích.
LpScr1Rect, lpSrc2Rect: Hai hình chữ nhật giao nhau.

9. Declare Function InvalidateRect Lib "user32" Alias "InvalidateRect" (ByVal hwnd As Long, lpRect As RECT, ByVal bErase As Long) As Long

Công dụng: Làm sai bất hợp lệ tất cả hay một phần vùng làm việc của một cửa sổ. Để vẽ lại đúng lúc, đúng chỗ.
Tham số kèm:
HWnd Cán của cửa sổ cần làm mất hợp lệ.
LpRect hình chữ nhật mô tả phần không hợp lệ.
BErase Cho về True (Khác zero) để xoá vùng chỉ định trước khi vẽ lại.

10. Declare Function IsChild Lib "user32" Alias "IsChild" (ByVal hWndParent As Long, ByVal hwnd As Long) As Long

Công dụng: Xác định cửa sổ cần xét có phải cửa sổ con (Mức Child) thuộc nhánh cửa sổ khác.
Trị trả về: Integer - True (Khác zero) nếu HWnd là cửa sổ con (Mức Child) hay hậu duệ của HWndParent **Tham số kèm:**
HWnd Cán của cửa sổ cần kiểm tra
HWndParent Cán của cửa sổ chứa (Mức Parent).

11. Declare Function IsIconic Lib "user32" Alias "IsIconic" (ByVal hwnd As Long) As Long

Công dụng: Kiểm tra cửa sổ có phải đã thu nhỏ thành biểu tượng không.
Trị trả về: Integer - True (Khác zero) nếu bị thu nhỏ
Tham số kèm:
HWnd Cán của cửa sổ cần kiểm tra.

12. Declare Function IsRectEmpty Lib "user32" Alias "IsRectEmpty" (lpRect As RECT) As Long

Công dụng: Kiểm tra xem hình chữ nhật có rỗng không.
Trị trả về: Integer - True (Khác zero) nếu rỗng. Zero nếu không rỗng.
Tham số kèm:
LpRect Hình chữ nhật cần kiểm tra.

1. Declare Function IsWindow Lib "user32" Alias "IsWindow" (ByVal hwnd As Long) As Long

Công dụng: Xác định xem có phải là cán cửa sổ không.

Trị trả về: Integer - True (Khác zero) nếu đúng là cán cửa sổ.
Tham số kèm:
HWnd Cán cần kiểm tra.

2. Declare Function IsWindowEnabled Lib "user32" Alias "IsWindowEnabled" (ByVal hwnd As Long) As Long

Công dụng: Kiểm tra cửa sổ có hiệu lực (enabled) không.
Trị trả về: Integer - True (Khác zero) nếu có hiệu lực.
Tham số kèm:
HWnd Cán của cửa sổ cần kiểm tra

3. Declare Function IsWindowVisible Lib "user32" Alias "IsWindowVisible" (ByVal hwnd As Long) As Long

Công dụng: Kiểm tra xem cửa sổ xem có thể nhìn thấy nó trên màn hình, kể cả cửa sổ bị cửa sổ khác xếp chồng lên trên.
Trị trả về: Integer - True (Khác zero) nếu nhìn thấy được.
Tham số kèm:
HWnd Cán của cửa sổ cần kiểm tra.

4. Declare Function IsZoomed Lib "user32" Alias "IsZoomed" (ByVal hwnd As Long) As Long

Công dụng: Kiểm tra xem cửa sổ xem có phóng to toàn màn hình không.
Trị trả về: Integer - True (Khác zero) nếu phóng toàn màn hình.
Tham số kèm:
HWnd Cán của cửa sổ cần kiểm tra.

5. Declare Function LockWindowUpdate Lib "user32" Alias "LockWindowUpdate" (ByVal hwndLock As Long) As Long

Công dụng: Khoá cửa sổ, không cho cập nhật. Mỗi lần chỉ có 1 cửa sổ bị khoá.
Trị trả về: Integer - True (Khác zero) nếu thành công. Zero nếu đã có cửa sổ khác bị khoá.
Tham số kèm:
HWndLock Cán của cửa sổ cần khoá.

6. Declare Function MapWindowPoints Lib "user32" Alias "MapWindowPoints" (ByVal hwndFrom As Long, ByVal hwndTo As Long, lppt As Any, ByVal cPoints As Long) As Long

Công dụng: Chuyển đổi các điểm theo các toạ độ sử dụng (client) của một cửa sổ sang các toạ độ cuả cửa sổ khác.
Tham số kèm:
HWndFrom, HWndTo Cán của cửa sổ nguồn và đích. Nếu một cán là toạ độ theo màn hình thì chọn cán là cán của Desktop.
Lppt Điểm chốt POINTAPI của mảng chuyển đổi.
CPoints Số điểm chuyển đổi.

7. Declare Function MoveWindow Lib "user32" Alias "MoveWindow" (ByVal hwnd As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal bRepaint As Long) As Long

Công dụng: Di chuyển và định lại kích thước cửa sổ.
Tham số kèm:
HWnd Cán của cửa sổ cần di chuyển.
X,y Toạ độ mới của đỉnh trái cửa sổ.
NWidth, nHeight Chiều rộng và chiều cao mới của cửa sổ.
BRepaint Integer - True (Khác zero) nếu muốn cửa sổ vẽ lại tự động sau khi di chuyển. False (zero) nếu ứng dụng tự vẽ lại.

8. Declare Function OffsetRect Lib "user32" Alias "OffsetRect" (lpRect As RECT, ByVal x As Long, ByVal y As Long) As Long

Công dụng: Di chuyển và thay đổi kích thước một vùng hình chữ nhật. Lưu ý Các chiều kích thước mới không quá 72767 đơn vị.
Tham số kèm:
LpRect - Hình chữ nhật cần di chuyển và thay đổi kích thước.
X - Khoảng cách dịch chuyển cho góc trái trên hình chữ nhật.
Y - Khoảng cách dịch chuyển cho góc phải dưới hình chữ nhật.

9. Declare Function PostMessage Lib "user32" Alias "PostMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long

Công dụng: Gửi một chỉ lệnh vào hàng đợi message queue của một cửa sổ. Các chỉ lệnh này sẽ được xử lý theo tuần tự.
Trị trả về: Integer - True (Khác zero) nếu thành công.
Tham số kèm:
HWnd Cán của cửa sổ nhận chỉ lệnh.
WMsg Hằng số ID của chỉ lệnh. (Xin tra công dụng của các hằng ở bảng khác)
WParam, lParam Các tham số tuỳ thuộc vào chỉ lệnh.

10. Declare Function PtInRect Lib "user32" Alias "PtInRect" (lpRect As RECT, pt As POINTAPI) As Long

Công dụng: K iểm tra điểm có nằm trong hình chữ nhật không.
Trị trả về: Integer - True (Khác zero)nếu nằm trong. Zero nếu ngoài.
Tham số kèm:
LpRect Hình chữ nhật để kiểm tra.
pt DDieemr cần kiểm tra.

11. Declare Function RedrawWindow Lib "user32" Alias "RedrawWindow" (ByVal hwnd As Long, lprcUpdate As RECT, ByVal hrgnUpdate As Long, ByVal fuRedraw As Long) As Long

Công dụng: Vẽ lại cửa sổ.
Trị trả về: Integer - True (Khác zero)nếu thành công. Zero nếu thất bại.
Tham số kèm:
HWnd Cán của cửa sổ để vẽ lại.
LprcUpdate - Hình chữ nhật bên trong cửa sổ cần vẽ lại.
HrgnUpdate - Cán của miền mô tả khu vực cần vẽ lại.
FuRedraw - Cờ yêu cầu vẽ lại, là một trong các hằng sau:
RDW_ERASE - Nền phẫn vẽ lại phải xoá trước khi vẽ.
RDW_FRAME - Cập nhật khung vẽ lại, nếu khung vẽ trùm lên tiêu đề, thực đơn, dòng trạng thái....
RDW_INTERNALPAINT - Gửi chỉ lệnh WM_PAINT cho cửa sổ.
RDW_INVALIDATE - Yêu cầu vẽ lại khu vực khung HrgnUpdate.
RDW_NOERASE - Không xoá nền của khung cần vẽ lại.
RDW_NOFRAME - Không cập nhật nếu khung vẽ lại trùm lên tiêu đề, thực đơn, dòng trạng thái.
RDW_NOINTERNALPAINT - Cấm các chỉ lệnh WM_PAINT đối với cửa sổ.
RDW_VALIDATE - Thừa nhận khung vẽ lại hợp lệ.
RDW_ERASENOW - Xoá ngay khung vẽ lại.
RDW_UPDATENOW - Cập nhật ngay khung vẽ lại.
RDW_ALLCHIDREN - Thao tác vẽ lại thực hiện luôn trên cả các cửa sổ con (Mức Child) nằm trong khung vẽ lại.
RDW_NOCHIDREN - Không vẽ lại các cửa sổ con (Mức Child), nếu nó nằm trong khung vẽ lại.

12. Declare Function ScreenToClient Lib "user32" Alias "ScreenToClient" (ByVal hwnd As Long, lpPoint As POINTAPI) As Long

Công dụng: Chuyển toạ độ một điểm trên màn hình thành toạ độ tương đối của cửa sổ.
Tham số kèm:
HWnd Cán của cửa sổ làm căn cứ toạ độ.
LpPoint Điểm cần chuyển

1. Declare Function IsWindow Lib "user32" Alias "IsWindow" (ByVal hwnd As Long) As Long

Công dụng: Xác định xem có phải là cán cửa sổ không.
Trị trả về: Integer - True (Khác zero) nếu đúng là cán cửa sổ.
Tham số kèm:
HWnd Cán cần kiểm tra.

2. Declare Function IsWindowEnabled Lib "user32" Alias "IsWindowEnabled" (ByVal hwnd As Long) As Long

Công dụng: Kiểm tra cửa sổ có hiệu lực (enabled) không.

Trị trả về: Integer - True (Khác zero) nếu có hiệu lực.
Tham số kèm:
HWnd Cán của cửa sổ cần kiểm tra

3. Declare Function IsWindowVisible Lib "user32" Alias "IsWindowVisible" (ByVal hwnd As Long) As Long

Công dụng: Kiểm tra xem cửa sổ xem có thể nhìn thấy nó trên màn hình, kể cả cửa sổ bị cửa sổ khác xếp chồng lên trên.
Trị trả về: Integer - True (Khác zero) nếu nhìn thấy được.
Tham số kèm:
HWnd Cán của cửa sổ cần kiểm tra.

4. Declare Function IsZoomed Lib "user32" Alias "IsZoomed" (ByVal hwnd As Long) As Long

Công dụng: Kiểm tra xem cửa sổ xem có phóng to toàn màn hình không.
Trị trả về: Integer - True (Khác zero) nếu phóng toàn màn hình.
Tham số kèm:
HWnd Cán của cửa sổ cần kiểm tra.

5. Declare Function LockWindowUpdate Lib "user32" Alias "LockWindowUpdate" (ByVal hwndLock As Long) As Long

Công dụng: Khoá cửa sổ, không cho cập nhật. Mỗi lần chỉ có 1 cửa sổ bị khoá.
Trị trả về: Integer - True (Khác zero) nếu thành công. Zero nếu đã có cửa sổ khác bị khoá.
Tham số kèm:
HWndLock Cán của cửa sổ cần khoá.

6. Declare Function MapWindowPoints Lib "user32" Alias "MapWindowPoints" (ByVal hwndFrom As Long, ByVal hwndTo As Long, lppt As Any, ByVal cPoints As Long) As Long

Công dụng: Chuyển đổi các điểm theo các toạ độ sử dụng (client) của một cửa sổ sang các toạ độ cuả cửa sổ khác.
Tham số kèm:
HWndFrom, HWndTo Cán của cửa sổ nguồn và đích. Nếu một cán là toạ độ theo màn hình thì chọn cán là cán của Desktop.
Lppt Điểm chốt POINTAPI của mảng chuyển đổi.
CPoints Số điểm chuyển đổi.

7. Declare Function MoveWindow Lib "user32" Alias "MoveWindow" (ByVal hwnd As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal bRepaint As Long) As Long

Công dụng: Di chuyển và định lại kích thước cửa sổ.
Tham số kèm:
HWnd Cán của cửa sổ cần di chuyển.
X,y Toạ độ mới của đỉnh trái cửa sổ.
NWidth, nHeight Chiều rộng và chiều cao mới của cửa sổ.
BRepaint Integer - True (Khác zero) nếu muốn cửa sổ vẽ lại tự động sau khi di chuyển. False (zero) nếu ứng dụng tự vẽ lại.

8. Declare Function OffsetRect Lib "user32" Alias "OffsetRect" (lpRect As RECT, ByVal x As Long, ByVal y As Long) As Long

Công dụng: Di chuyển và thay đổi kích thước một vùng hình chữ nhật. Lưu ý Các chiều kích thước mới không quá 72767 đơn vị.
Tham số kèm:
LpRect - Hình chữ nhật cần di chuyển và thay đổi kích thước.
X - Khoảng cách dịch chuyển cho góc trái trên hình chữ nhật.
Y - Khoảng cách dịch chuyển cho góc phải dưới hình chữ nhật.

9. Declare Function PostMessage Lib "user32" Alias "PostMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long

Công dụng: Gửi một chỉ lệnh vào hàng đợi message queue của một cửa sổ. Các chỉ lệnh này sẽ được xử lý theo tuần tự.
Trị trả về: Integer - True (Khác zero) nếu thành công.
Tham số kèm:
HWnd Cán của cửa sổ nhận chỉ lệnh.
WMsg Hằng số ID của chỉ lệnh. (Xin tra công dụng của các hằng ở bảng khác)

WParam, lParam Các tham số tuỳ thuộc vào chỉ lệnh.

10. Declare Function PtInRect Lib "user32" Alias "PtInRect" (lpRect As RECT, pt As POINTAPI) As Long

Công dụng: K iểm tra điểm có nằm trong hình chữ nhật không.
Trị trả về: Integer - True (Khác zero)nếu nằm trong. Zero nếu ngoài.
Tham số kèm:
LpRect Hình chữ nhật để kiểm tra.
pt DDieemr cần kiểm tra.

11. Declare Function RedrawWindow Lib "user32" Alias "RedrawWindow" (ByVal hwnd As Long, lprcUpdate As RECT, ByVal hrgnUpdate As Long, ByVal fuRedraw As Long) As Long

Công dụng: Vẽ lại cửa sổ.
Trị trả về: Integer - True (Khác zero)nếu thành công. Zero nếu thất bại.
Tham số kèm:
HWnd Cán của cửa sổ để vẽ lại.
LprcUpdate - Hình chữ nhật bên trong cửa sổ cần vẽ lại.
HrgnUpdate - Cán của miền mô tả khu vực cần vẽ lại.
FuRedraw - Cờ yêu cầu vẽ lại, là một trong các hằng sau:
RDW_ERASE - Nền phẩn vẽ lại phải xoá trước khi vẽ.
RDW_FRAME - Cập nhật khung vẽ lại, nếu khung vẽ trùm lên tiêu đề, thực đơn, dòng trạng thái....
RDW_INTERNALPAINT - Gửi chỉ lệnh WM_PAINT cho cửa sổ.
RDW_INVALIDATE - Yêu cầu vẽ lại khu vực khung HrgnUpdate.
RDW_NOERASE - Không xoá nền của khung cần vẽ lại.
RDW_NOFRAME - Không cập nhật nếu khung vẽ lại trùm lên tiêu đề, thực đơn, dòng trạng thái.
RDW_NOINTERNALPAINT - Cấm các chỉ lệnh WM_PAINT đối với cửa sổ.
RDW_VALIDATE - Thừa nhận khung vẽ lại hợp lệ.
RDW_ERASENOW - Xoá ngay khung vẽ lại.
RDW_UPDATENOW - Cập nhật ngay khung vẽ lại.
RDW_ALLCHIDREN - Thao tác vẽ lại thực hiện luôn trên cả các cửa sổ con (Mức Child) nằm trong khung vẽ lại.
RDW_NOCHIDREN - Không vẽ lại các cửa sổ con (Mức Child), nếu nó nằm trong khung vẽ lại.

12. Declare Function ScreenToClient Lib "user32" Alias "ScreenToClient" (ByVal hwnd As Long, lpPoint As POINTAPI) As Long

Công dụng: Chuyển toạ độ một điểm trên màn hình thành toạ độ tương đối của cửa sổ.
Tham số kèm:
HWnd Cán của cửa sổ làm căn cứ toạ độ.
LpPoint Điểm cần chuyển

1. Declare Function ShowWindow Lib "user32" Alias "ShowWindow" (ByVal hwnd As Long, ByVal nCmdShow As Long) As Long

Công dụng: Điều khiển hiện cửa sổ.
Trị trả về: Integer - Nếu cr được nhìn thấy trước đó. Zero nếu ngược lại.
Tham số kèm:
HWnd Cán của cửa sổ cần điều khiển.
NCmdShow - Integer - Là các chỉ lệnh hằng sau:

SW_HIDE: Giấu cửa sổ.
SW_MINIMIZE: Thu nhỏ thành biểu tượng.
SW_RESTORE: Hiện lại như lúc ban đầu, kích hoạt.
SW_SHOW: Hiện lại như lúc chưa giấu, kích hoạt
SW_SHOWMAXIMIZED: Hiện mở rộng tối đa, kích hoạt.
SW_SHOWMINIMIZED: Hiện như biểu tượng, kích hoạt.
SW_SHOWMINNOACTIVE: Thu nhỏ cửa sổ, không làm thay đổi cửa sổ đang kích hoạt.
SW_SHOWNA: Hiện một cửa sổ ở kích thước và vị trí hiện tại, không làm thay đổi cửa sổ đang kích hoạt.

SW_SHOWNOACTIVE: Hiện cửa sổ như trước khi giấu, không làm thay đổi cửa sổ đang kích hoạt.
SW_SHOWNORMAL: Hiện ra bình thường.

2. Declare Function SubtractRect Lib "user32" Alias "SubtractRect" (lprcDst As RECT, lprcSrc1 As RECT, lprcSrc2 As RECT) As Long

Công dụng: Nạp vào cửa sổ đích phần trừ của 2 cửa sổ khác.
Trị trả về: Integer - True (Khác zero) nếu thành công. Zero nếu thất bại.
Tham số kèm:
LpDestRect - Hình chữ nhật đích.
lprcSrc1, lprcSrc2: Hai hình chữ nhật nguồn trừ nhau.

3. Declare Function UnionRect Lib "user32" Alias "UnionRect" (lpDestRect As RECT, lpSrc1Rect As RECT, lpSrc2Rect As RECT) As Long

Công dụng: Nạp vào cửa sổ đích phần cộng của 2 cửa sổ khác.
Trị trả về: Integer - True (Khác zero) nếu thành công. Zero nếu thất bại.
Tham số kèm:
LpDestRect - Hình chữ nhật đích.
lprcSrc1Rect, lprcSrc2Rect: Hai hình chữ nhật nguồn cần cộng.

4. Declare Function UpdateWindow Lib "user32" Alias "UpdateWindow" (ByVal hwnd As Long) As Long

Công dụng: Cập nhật ngay cửa sổ.
Tham số kèm:
HWnd Cán của cửa sổ cần cập nhật.

5. Declare Function ValidateRect Lib "user32" Alias "ValidateRect" (ByVal hwnd As Long, lpRect As RECT) As Long

Công dụng: Hợp lệ hoá cửa sổ, để không cần vẽ lại.
Tham số kèm:
HWnd Cán của cửa sổ cần hợp lệ hoá.
LpRect - Hình chữ nhật cần hợp lệ hoá. Nếu đặt zero thì hợp lệ toàn bộ cửa sổ.

6. Declare Function WindowFromPoint Lib "user32" Alias "WindowFromPoint" (ByVal xPoint As Long, ByVal yPoint As Long) As Long

Công dụng: Lấy cán cửa sổ chứa (Mức Parent) điểm cần khai báo.
Trị trả về: Integer - Cán của cửa sổ chứa (Mức Parent) điểm. Zero nếu không có cửa sổ nào.
Tham số kèm:
XPoint, Ypoint: Điểm theo toạ độ màn hình.

WINDOWS API
Khám phá từ A đến Z
Bản chất của Windows API.

Trong lập trình Visual Basic độc lập hoặc Visual Basic for Application, Microsoft đã cung cấp cho chúng ta một bộ các hàm lập sẵn, hàng trăm hàm API (Aplication Programming Interface) được lưu trong các tệp thư viện liên kết động (Tệp đuôi *.DLL - Dynamic Link Library). Đó là công cụ tuyệt vời cho phép bạn phát triển ứng dụng cực mạnh, tại sao bạn lại bỏ qua và không sử dụng nó?

Tôi sẽ cùng bạn khám phá những gì mà Microsoft cung cấp các hàm Windows API trong bộ Visual Studio. Tuy nhiên vì khuôn khổ cũng như kích thước của bài viết, ta chỉ đi vào các nét chính căn bản nhất, bạn có thể tham khảo trong Help hoặc các bài viết của Nguyễn Hồ Thiên Đăng, Nguyễn Thị Thanh Phương tại WebsiteLH.

I. Hàm API - Nhìn từ góc độ người ít có điều kiện học Tin học

Nếu bạn chưa từng lập trình những chương trình lớn, bạn sẽ phát hoảng khi đọc khai báo (rắc rối và kỳ cục!!!) của API:

Private Declare Function CallNextHookEx Lib "user32" Alias "CallNextHookEx" (ByVal hHook As Long, ByVal ncode As Long, ByVal wParam As Long, lParam As Any) As Long

Tuy nhiên bạn có thể chép phần khai báo trên thật đơn giản bằng Text API Viewer. Theo các chuyên gia về Tin học thì đừng bao giờ sử dụng thẳng hàm API trong thủ tục thiết kế chính của mình. Thay vào đó ta thiết kế một hàm hay thủ tục Visual Basic thay thế hàm API để đơn giản hoá (Gọi là wrapper - Tôi không tìm được từ tiếng Việt tương ứng để nói đủ bản chất của nó.)

Ví dụ một wrapper sau:
```
Public Sub ThoatVaTatMay ()
'Thoát và tắt máy
Dim thoat
Thoat = ExitWindowsEX(2,0)
End Sub
```

Khi đó, trong chương trình Visual Basic của ta khi cần thoát và tắt máy chỉ việc gọi:
ThoatVaTatMay
hoặc Call ThoatVaTatMay
Là máy tính thực hiện thoát và tắt máy.

Chính vì thế, khi học viên dân tộc được học phổ cập API tại Trung tâm Dạy nghề và Phổ cập Tin học Miền núi ABC của chúng tôi đã gọi chức năng tạo Wrapper chẳng khác dán nhãn Tiếng Việt cho từng loại thuốc tây API vậy. Nên khi lập trình ta nên tạo các Wrapper tương ứng với các chức năng mà mình muốn sử dụng. Đó cũng là lời khuyên của Bill Gate cho chúng ta.

Để tạo các wrapper bạn hãy chèn một Module vào Project. Nếu máy của bạn không cài Visual Basic thì bạn phải copy hay đánh thật chính xác những dòng khai báo dạng như khai báo trên,

thật khổ sở nếu như phần tiếng Anh của bạn không thạo lắm, vì một sai sót nhỏ có thể dẫn tới lỗi nặng cho máy. Nếu máy có cài Visual Basic thì quá tốt, chỉ việc khởi động API Viewer để hiện bảng giao tiếp như thế này:

Bạn phải mở các tệp TXT trong thư mục API của thư mục cài Visual Basic trong máy bạn. Ví dụ tệp Win32api.TXT. Máy sẽ hỏi có chuyển thành dạng cơ sở dữ liệu không, thì hãy chọn có để sử dụng API thuận lợi và nhanh chóng hơn. Khi đã copy vào clipboard, bạn có thể dán vào module của mình. Nó sẽ thành dạng tương tự như thế này:

Declare Function CallNextHookEx Lib "user32" Alias "CallNextHookEx" (ByVal hHook As Long, ByVal ncode As Long, ByVal wParam As Long, lParam As Any) As Long

Bạn phải đánh thêm vào trước khai báo trên cụm từ Private để được:
Private Declare Function CallNextHookEx Lib "user32" Alias "CallNextHookEx" (ByVal hHook As Long, ByVal ncode As Long, ByVal wParam As Long, lParam As Any) As Long

Tôi xin phép được nhắc lại 2 từ khoá khai báo trong Visual Basic là Private và Public.

Private - Khai báo dùng riêng trong Module. Có nghĩa là bạn chỉ sử dụng được nó trong Module này. Nếu chèn Module khác sẽ không sử dụng được nó.

Public - Khai báo dùng chung, bạn có thể dùng nó ở bất cứ Module nào.

Bạn biết đấy, ta khai báo các hàm API thì lại dùng Private, còn khai báo các Wrapper lại dùng Public. Đó chính là mẹo mà chúng ta đã học trộm được của Microsoft để tránh lỗi hệ thống.

Các hàm mà ta thiết kế trong Visual Basic có điều khác với các hàm API. Tại sao? Vì hàm ta thiết kế (Tạm gọi là hàm Visual Basic) thường chỉ có một kết quả trả về của hàm làm căn cứ xử lý. Còn hàm API không phải chỉ có một kết quả trả về mà nó còn trả về tiếp các giá trị vào các biến mà ta truyền cho nó. Nghĩa là có gọi nó là hàm thì bản chất nó là một thủ tục, ta gọi hàm để thực hiện và kiểm tra xem thủ tục trong hàm đó có thực hiện được thành công hay không mà thôi.

Điều này chỉ hơi giống như khi bạn lập trình với hàm Visual Basic mà bạn khai báo Public Static - Biến tĩnh dùng chung, để làm biến đổi nó trong hàm của bạn. Ta nghiên cứu một cách tổng quát như sau để hiểu cặn kẽ:

Giả sử ở một Wrapper bạn dùng công thức:
TENBIEN=TEN_HAM_API(Bien1, Bien2, Bien3)

Thực ra đây là một thủ tục. Nếu TENBIEN<>0 thì thủ tục này thành công.

Khi đó các Bien1, Bien2, Bien3 truyền vào, sẽ có một giá trị mới. Lập trình API lúc này không chỉ xử lý TENBIEN, mà bạn có thể xử lý các Bien1, Bien2, Bien3. Đó mới thực sự là sức mạnh của API.

Bản chất lập trình của biến là một vùng các ô nhớ trong RAM được đặt tên để tiện sử dụng. Tại một thời điểm biến chỉ có một giá trị duy nhất. Người ta có thể dùng biến này làm giá trị định vị hoặc kích thước lưu trữ cho biến kia. Các hàm API có thể được gọi nhiều lần để sử dụng kết quả trả về của các biến truyền Bien1, Bien2, Bien3 để xử lý theo quy luật xác định nào đó.

II. Xác định mục đích khi sử dụng WinAPI.

Trong lập trình API có thể phân làm nhiều mục đích sử dụng, tôi chưa từng được học Tin học một cách chính thống, nên có thể khả năng phân tích và tổng hợp những bài viết trên INTERNET của Microsoft khác với những bài học của những học viên học Tin học chính quy. Nhưng tôi nghĩ chúng ta nắm bản chất của vấn đề mới là điều quan trọng.

Có 3 vấn đề chính khi sử dụng và khai thác WinAPI đó là:

a. Kỹ thuật Subclass: Để cải tổ các đối tượng Visual Basic.
b. Kỹ thuật Hook: Câu móc từ chương trình Visual Basic với các chương trình khác. Lấy giá trị nhập vào các chương trình khác của người sử dụng đưa vào chương trình của mình để xử lý.
c. Kỹ thuật Multicasting: Dùng một đối tượng tạo lập để theo dõi, chi phối các đối tượng khác của Visual Basic.

Bạn có thể sử dụng từng Kỹ thuật hoặc cả 3. Tuy nhiên bước đầu chưa thạo, bạn hãy thực hiện từng Kỹ thuật một cho thành thạo. Sau khi kiểm soát được khả năng của mình, bạn sẽ đủ trình độ để sử dụng WinAPI để cải tổ Windows và máy tính.

Xin đọc tất cả các bài viết của Nguyễn Hồ Thiên Đăng, Nguyễn Thị Thanh Phương và Nguyễn Phương Thảo về lập trình Visual Basic trên WebsiteLH.

Khi nắm rõ bản chất của Windows API để lập trình sâu với hệ thống, ta cũng cần hiểu biết sơ bộ về Hệ điều hành Windows cách thức điều khiển của Hệ điều hành đối với ứng dụng để có thể can thiệp như bổ sung chức năng thậm chí biến đổi nó, bắt thực hiện theo hướng của mình, ngay cả khi hướng này ngược hẳn với công dụng truyền thống !!!

Các chương trình ứng dụng trong Windows có thể có nhiều cửa sổ phục vụ cho nó. Cửa sổ có thể là Form thậm chí là Dialog. Mỗi cửa sổ này đều có một handle (Cán) để hệ thống nhận biết do chính hệ điều hành Windows tạo ra. Cán cửa sổ này là chỉ số duy nhất.

Hệ điều hành và chương trình ứng dụng đều duy trì các hàng đợi các chỉ lệnh cần thực hiện. Mỗi ứng dụng đều có hàng đợi (Message Queue). Khi người sử dụng ra lệnh hoặc có một biến cố, các chương trình điều khiển thiết bị nhập (INPUT) sẽ chuyển các thông tin vào thành chỉ lệnh và đặt chỉ lệnh này vào hàng đợi hệ thống (System Message Queue). Hệ điều hành lấy lần lượt các chỉ lệnh trong hàng đợi hệ thống kiểm tra để xác định cửa sổ nào sẽ tiếp nhận thi sẽ đặt vào hàng đợi của nó (thread message) một chỉ lệnh tương ứng. Các chương trình ứng dụng căn cứ vào chỉ lệnh này để thực hiện cũng như xử lý chúng.

Các cửa sổ giống như một động cơ tự động chạy theo một vòng lặp. Tiếp "nhiên liệu" cho các "động cơ" này là hệ điều hành Windows. Hệ điều hành Windows nhận các chỉ lệnh (message) từ hàng đợi của hệ điều hành, dùng một hàm dạng API (Như kỳ 1 - Hàm này bản chất là một thủ tục) để cung cấp chỉ lệnh tới cửa sổ thông qua cán (handle) của cửa sổ.

Có nghĩa là bản thân trong mỗi cửa sổ luôn có một hàm gọi là WinProc (Đôi khi gọi là WinMain()). Hàm này là cốt lõi xử lý của cửa sổ. Trong hàm, nó lặp đi lặp lại liên miên 2 dòng lệnh sau thông qua cấu trúc:

```
Do While 0 <>GetMessage (message, 0, 0,0)
TranslateMessage message
DispatchMessage message
Loop
```

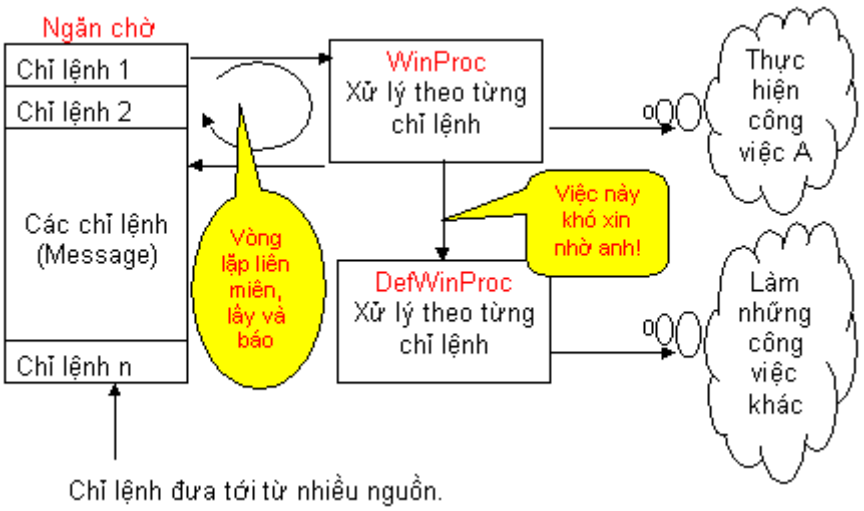
Trong đó message là chỉ lệnh mà Hệ điều hành cung cấp, thông qua cán (handle) của cửa sổ. Đương nhiên, nếu chỉ lệnh có giá trị WM_QUIT thì hàm WinProc trong cửa sổ chấm dứt vòng lặp.

Còn nếu chỉ lệnh message khác giá trị trên, thì 2 dòng lệnh trên sẽ thực hiện. Cụ thể:
TranslateMessage message -> Dịch chỉ lệnh thành dạng dữ liệu khác đặt kết quả này vào hàng đợi của ứng dụng.
DispatchMessage message ->Nhận chỉ lệnh từ hàm GetMessage và gửi cho hệ thống. Hệ thống sẽ đưa chỉ lệnh cho ứng dụng.

Windows có hàng ngàn chỉ lệnh khác nhau đó là các hằng dạng WM_* (Windows đặt tên cho tiện gọi thôi, vì bản chất các hằng này là một con số - Rất khó nhớ. Nếu phân tích chi tiết ra, những con số này lại là dãy số 0 và 1, tức là bật tắt ấy mà).
Một hàm WinProc luôn nhận vào trong nó các biến theo khuôn mẫu sau để xử lý:
Function WinProc(hwnd as Long, wc as WNDCLASSEX, message as MSG, wParam as Long, lparam as Long)

Nếu hàm WinProc không xử lý các chỉ lệnh, nó phải đưa trả chỉ lệnh cho hệ điều hành xử lý thông qua hàm DefWindowProc. Hàm DefWindowProc gởi lại chỉ lệnh WM_CLOSE cho WinProc. Hàm WinProc sẽ lại gởi trả WM_CLOSE cho DefWindowProc một lần nữa như mô tả ở trên.

Bạn có thể thấy, kết quả và cơ chế xử lý rất lằng nhằng. Ta có thể tóm lại sơ bộ như sau:



Các chỉ lệnh đưa tới ngăn chờ trên thông thường từ các nguồn sau:
1. Hệ thống đặt vào
2. Chương trình khác đặt vào
3. Chính chương trình của mình đặt vào thông qua các hàm SendMessage() và PostMessage().

Tuy nhiên nếu bạn chọn sử dụng hàm SendMessage() thì sau khi chỉ lệnh được WinProc lấy ra xử lý thì chương trình mới tiếp tục chạy tiếp lệnh kế sau. Còn bạn dùng PostMessage() chỉ có tác dụng đặt chỉ lệnh vào hàng đợi và thực hiện ngay lệnh kế tiếp.

Từ đây ta nhận thấy việc xử lý hệ thống của Windows thông qua cơ chế trên thì giản đơn đi rất nhiều. Ta sẽ có các hướng giải quyết tiếp theo như sau:
1. Nếu ta thiết kế một hàm WinProc() mới, Ví dụ NewWinProc(), thay thế hàm WinProc() truyền thống
- Rồi ta đổi địa chỉ của WinProc() gốc sang địa chỉ của WinProc() mà ta thiết kế. Trong thủ tục này sử dụng cấu trúc Select Case để tuỳ vào chỉ lệnh message mà xử lý theo ý mình.
- Trường hợp chỉ lệnh message nào không thể viết được thủ tục thi hành (Không cần thay đổi hoặc khó quá) thì ta gọi thủ tục DefWinProc() của Windows xử lý. (Tức là dễ thì làm, khó trả lại ấy mà). Đây chính là Kỹ thuật Subclass.

- Tuy nhiên nếu như hàm WinProc() cũ, xử lý tốt một số tính năng nào đó thì ta có thể tận dụng thủ tục bằng cách gọi lại bằng hàm CallWindowProc().
- Thậm chí ta có thể lồng WinProc() vào trong NewWinProc() để xử lý
(Xem các bài ví dụ của Nguyễn Phương Thảo có sử dụng API).

2. Trường hợp không thể thay được hàm WinProc() bằng NewWinProc() (Ví dụ như bạn lập trình với các chương trình ứng dụng khác như Winword, Excel...) bạn phải chặn các chỉ lệnh trước khi nó được lấy ra khỏi hàng đợi. Đó chính là Kỹ thuật Hooking, một Kỹ thuật cực kỳ mạnh để làm việc với Windows.

Trước khi chỉ lệnh được hàm SendMessage() lấy từ hàng đợi gửi đi, hoặc cũng có thể chỉ lệnh được lấy bằng hàm PickMessage() hay GetMessage(), ta có thể đăng ký với Windows để sử dụng bộ lọc HookFilter. Khi đó những chỉ lệnh cần xử lý đã đăng ký, đều qua HOOK Filter. Ta chỉ việc viết một thủ tục dùng hàm WinProc lấy chỉ lệnh từ HOOK Filter để xử lý. (Xem bài viết chặn các thủ tục in và nhập dữ liệu của Nguyễn Phương Thảo).

3. Các đặc điểm của lớp cửa sổ:

- Mỗi ứng dụng có thể tạo ra nhiều cửa sổ, thông thường các cửa sổ này có những đặc điểm giống nhau và được phân theo từng lớp CLASS. Khi lập trình, bao giờ ta cũng đăng ký lớp với hệ thống thông qua một hàm là RegisterClassEx(). Khi lớp đã được đăng ký (Chỉ 1 lần) thì các thông tin window và địa chỉ hàm WinProc sẽ được lưu trong suốt thời gian mà nó tồn tại.

- Ta có thể thay đổi các thông tin trong bộ đăng ký lớp, khi đó nó sẽ ảnh hưởng đến toàn bộ cửa sổ trong lớp này.

Bạn thân mến! Như vậy bạn đã nắm chắc về cơ chế làm việc của Windows và các thủ tục hệ thống của nó. Hi vọng bạn hãy đọc thật kỹ và hiểu rõ về nó, để những bài viết sau chúng ta sẽ mổ xẻ giải phẫu từ những hàm API cơ sở được dễ dàng hơn.

API với Registry

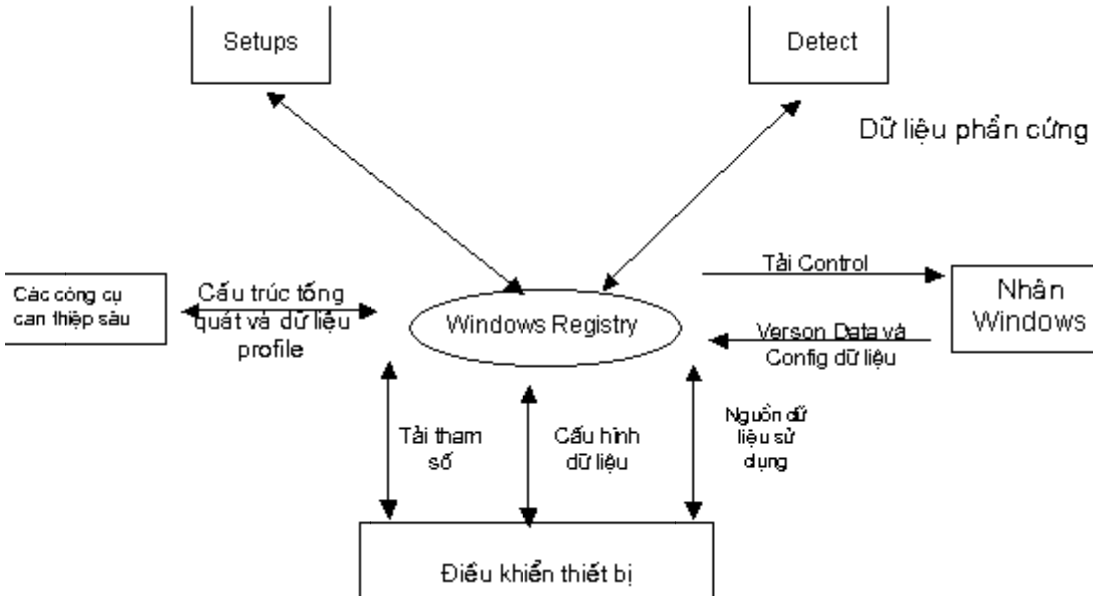
I. Sơ lược về Registry

Registry là nơi lưu trữ tất cả các các loại cấu trúc dữ liệu. Cấu hình hệ thống Windows, cấu hình phần cứng máy tính, cấu hình thông tin về các chương trình ứng dụng dựa trên Win32, và các thiết lập người dùng khác đều được lưu trong Registry.

Ví dụ, bất cứ một phần cứng máy tính nào thay đổi đều làm chức năng Plug and Play (Cắm và chạy) khởi tạo ngay và làm thay đổi luôn cấu hình trong Registry.

Registry lưu trữ tất cả các thiết lập về cấu trúc bộ nhớ, phần cứng, thiết bị ngoại vi, và các thành phần liên quan đến mạng. Bạn sẽ tìm thấy ở đó nhiều hơn những thiết lập cần thiết trong các tệp khởi tạo ban đầu

Từ Win98 về sau, Windows có sử dụng Registry Checker để tự quét Registry, nếu không thấy gì, nó tự lưu backup một lần trong ngày, nếu tìm thấy lỗi sẽ sửa... có thể sửa bằng cách thay thế bản Registry đã backup gần nhất còn tốt. Registry Checker tối ưu hoá và nén file backup thành công mỗi lần khởi động máy. Nó còn làm một loạt các việc linh tinh như loại bỏ những khoảng trống không dùng trong Registry, tối ưu hoá...



Các tệp Registry của Windows.

Registry hiện tại bao gồm 3 tệp chính:

1. Tệp USER.DAT

Dùng để lưu trữ những xác lập người sử dụng đối với các phần mềm.

2. Tệp SYSTEM.DAT

Dùng để lưu trữ những xác lập liên quan tới máy tính và phần cứng.

3. Tệp Policy.pol

System policies được thiết kế để chuẩn bị cho việc ghi đè bất cứ thiết lập đã được chứa trong 2 thành phần registry khác nhau.

System policies có thể chứa dữ liệu bổ sung đặc trưng tới mạng hay môi trường tổ hợp như đã được cài đặt bởi network administrator. Bản thân System policies cũng đã được chứa trong tệp Policy.pol. Không như SYSTEM.DAT và USER.DAT, Policy.pol không phải là thành phần bắt buộc của phần cài đặt Windows.

Lời khuyên của Microsoft về những công cụ xử lý registry

Phương pháp	Thiết lập
Control Panel	Phần lớn thiết lập hệ thống SYSTEM. Ví dụ bạn sử dụng Display Properties để sửa các thành phần của mục appearance
System Policy Editor	Thiết lập người dùng, vài thiết lập hệ thống.
Các chương trình tiện ích thứ 3	Thiết lập chi tiết ứng dụng

Bạn có thể đã sử dụng Registry Editor để thay đổi Registry bằng tay. Tôi thường dùng Norton Registry Editor vì nó còn có thêm chức năng khác, ví dụ như tìm và thay thế đối với các thành phần của Registry.

Từ các phần mềm Registry Editor trên, ta nhận thấy registry được bố trí thành các nhánh lớn. Tại mỗi nhánh có các khoá SUBKEY. Tại các SUBKEY dữ liệu được lưu ở các dạng:

1. String (Dạng chuỗi)

2. Numeric (Dạng số)

3. Binary (Dạng nhị phân)

4. Expanded String (Dạng chuỗi mở rộng)

5. MultiString (Dạng chuỗi tổng hợp)

(Nếu bạn dùng Registry Editor - REGEDIT. EXE thì sẽ gọi tên khác là DWORD)

Ta hãy nghiên cứu sơ đồ mà Microsoft vẽ (Từ đĩa MSDN) về Registry:

Nhìn sơ đồ ta nhận thấy:

Nếu môi trường Windows là sân khấu thì Registry giống như một cuốn sổ tay của đạo diễn. Trong đó các thiết lập liên quan đến phần cứng thì hầu như không thay đổi (Nếu ta không thay phần cứng để phần đề tếch của Windows nhận ra). Nếu thay đổi phần cứng, phần đề tếch này sẽ bắt bạn phải đổi mã điều khiển đối với từng thiết bị.

Xin nói thêm phần này kỹ hơn một chút vì tôi nhận được rất nhiều thư hỏi của các bạn về những vấn đề liên quan đến phần cứng và DRIVER. Một số đông bạn chưa hiểu rõ cách thức làm việc của Windows nên ngay cả những lỗi rất đơn giản như màn hình chỉ có 16 màu (Không chọn được 256 màu hay cao hơn như High Color, True Color) cũng không biết cách giải quyết.

Khi máy tính của bạn có những thiết bị gì, hệ điều hành Windows khi cài đặt hoặc khi đề tếch thiết bị nó sẽ yêu cầu bạn phải đưa ra các tệp DRIVER của thiết bị đó. DRIVER là gì ư? Ồ! Nó chỉ là tệp mã điều khiển thiết bị. Giống như sơ đồ công tắc của thiết bị nào đó. Khi máy tính có công việc liên quan đến điều khiển thiết bị, hệ điều hành Windows sẽ tra trong Registry xem tệp mã nó là tệp nào. Tệp mã sẽ được nạp vào RAM một phần hay toàn bộ, ở một vị trí nào đó. Căn cứ vào lệnh đưa tới từ chương trình, Hệ điều hành sẽ đưa lệnh tới địa chỉ phần mã điều khiển này. Phần mã điều khiển sẽ chỉ tiếp cho con trỏ tới vị trí của lệnh cần thực hiện năm trong phần mã, tương ứng với mã lệnh đưa tới. Khi đó công việc được thực hiện chính xác vì phần mã đúng với thiết bị.

Nếu thiết bị một kiểu, mã DRIVER khác với kiểu được lắp vào máy, thì chẳng khác gì bạn đưa ra một sơ đồ máy khác với máy đang sử dụng. Khi đó sẽ có xung đột, trường hợp này Windows sẽ tự chuyển sang chức năng Đờ phôn ngầm định đối với các khoá, và như vậy bạn sẽ thấy Windows không thể hiện đúng với thực tế của mình. Bạn có thể vào Control Panel chọn mục System để kiểm tra, nếu thấy dấu ? cạnh tên mã thiết bị phần nào thì chọn Update để đổi mã.

Các khoá về mã phần cứng đều được lưu trong tệp SYSTEM.DAT. Do phần này ít thay đổi, vả lại chức năng Plug and Play của Windows quá siêu, nên ta sẽ tạm gác nghiên cứu đến phần này. Sau này tôi sẽ đề nghị một chương trình khoá thiết bị bằng phần mềm ở những chương sau.

Các khoá về phần mềm thường là lưu trữ các thiết lập Option... của từng phần mềm ứng dụng riêng. Ta có thể nhận ra, trước đây khi sử dụng Win 3.1 về trước, các thiết lập này được lưu trên đĩa bằng các tệp đuôi INI. Do Win32 khác xa về cấu trúc, nếu dùng các tệp INI sẽ tương đối bất tiện, ví dụ thực tế dung lượng bỏ phí trên đĩa cực lớn, nên Microsoft đã tích hợp vào các tệp Registry. Tuy nhiên, các tệp WIN.INI, SYSTEM.INI vẫn còn được sử dụng để có thể làm việc với các ứng dụng cũ.

Các khoá "mềm" có thể thay đổi thường xuyên khi ta sử dụng chức năng options để đặt lại. Ta nen lựa chọn tối ưu chứ đừng thí nghiệm vì sẽ có nhiều lỗi không đáng có. Bộ Norton Utilities có nhiều chức năng trong đó có Optimization Wizard (Tối ưu hoá) cũng sắp xếp và tổ chức lại Registry. WinDoctor và System Check thường kiểm tra sai sót hoặc thừa trong Registry để xử lý. Nếu bạn biết chút ít tiếng Anh, thì đó là công cụ thuận tiện đối với bạn.

2. Lập trình với Registry

Khi lập trình đối với Registry, bạn phải thực hiện hết sức thận trọng, sao lưu các tệp này thường xuyên để tránh lỗi đáng tiếc, phải mất công cài lại thì cũng rất mất thời gian.Sử dụng các hàm API đối với Registry cũng xin hết sức thận trọng. Tôi đưa ra một ví dụ để bạn có thể nghiên cứu rõ hơn. Đó là lập trình với Win API chính trong VBA. Ví dụ này sử dụng cách thiết đặt các tham số của WINWORD. Bạn có thể căn cứ vào đây để viết ra các ứng dụng khác tương tự. Phần này bạn chỉ việc copy phần lập trình để nghiên cứu và chạy thử.

2.1 Tạo Form

Để tạo Macro bạn hãy tạo Form tương tự như sau:

'Form thứ nhất

(Name)=EditCnvOptionsForm

Caption=Các lựa chọn sửa chữa Conversions

'Nhãn 1

(Name)=lblConversion

Caption= Conversion:

'Nhãn2

(Name)=lblCnvOption

Caption= Lựa chọn:

'Nhãn 3

(Name)=lblSetting

Caption= Thiết lập:

'Các hộp bỏ trống:

```
(Name)=cboConverters

(Name)=lstOptions

(Name)=txtSetting

(Name)=OptYes

(Name)=OptNo

'Các nút lệnh tuỳ ý Caption

(Name) =cmdSet

(Name) =cmdOK

(Name) =cmdHelp

'Viết lệnh cho Form thứ nhất:

'-------------------------------------------------

'--- Khởi tạo combo box với tên conveter ---

'-------------------------------------------------

Public Function ControlsInit() As Boolean

' Lấp đầy combo với đoạn text converters và graphics filters

ListConverters hCnvExpKeyHandle, HKEY_LOCAL_MACHINE, _

strREG_TEXT_CNV_EXPORT, strREG_CNV_NAME

ListConverters hCnvImpKeyHandle, HKEY_LOCAL_MACHINE, _

strREG_TEXT_CNV_IMPORT, strREG_CNV_NAME

ListConverters hFltExpKeyHandle, HKEY_LOCAL_MACHINE, _

strREG_GRAPH_FLT_EXPORT, strREG_CNV_NAME

ListConverters hFltImpKeyHandle, HKEY_LOCAL_MACHINE, _

strREG_GRAPH_FLT_IMPORT, strREG_CNV_NAME

' có bất cứ dữ liệu nào được đọc từ registry?

If clsCnvTable.ConverterCount = 0 Then
```

```
ControlsInit = False
Exit Function
End If
' lấp đầy combo box với tên converter names
clsCnvTable.AddConverterNamesToCombo cboConverters
' Chọn converter đầu tiên trên combo
cboConverters.ListIndex = 0
ControlsInit = True
End Function
Private Sub CommitChange(ByVal strNewString As String)
Dim strValue As String
Dim hHandleOptKey As Long
Dim res&
Dim strTemp As String
Dim i As Integer
' Nếu chọn phần trên list box khác
If ExtractOptionData(lstOptions.Value) <> strNewString Then
' triển khai chuỗi Value string
strValue = ExtractOptionValue(lstOptions.Value)
' Tìm khoá key handle của converter trên combo
hHandleOptKey = clsCnvTable.OptionsHandle(cboConverters.Value)
' Viết chuỗi dữ liệu mới Data string vào registry
res& = RegSetValueExString(hHandleOptKey, strValue, 0&, REG_SZ, _
strNewString, Len(strNewString))
If res& <> ERROR_SUCCESS Then
```

```
GoTo FatalError

End If

' Cập nhật lựa chọn vào list box

i = lstOptions.ListIndex

lstOptions.RemoveItem i ' bỏ cũ

lstOptions.AddItem strValue & strEQUALS_SIGN & strNewString, i ' Thêm mới

lstOptions.ListIndex = i ' chiếu sáng trùng

End If

Exit Sub

FatalError:

ReportRegError res&

End Sub

'----------------------------------------

' Khai triển dữ liệu conversion option data

' Từ chuỗi định dạng "Option=Data"

'----------------------------------------

Private Function ExtractOptionData(ByVal strDummy) As String

' Khai triển từng dòng sau khi bị biến thành 0

ExtractOptionData = Right$(strDummy, _

Len(strDummy) - InStr(strDummy, strEQUALS_SIGN))

End Function

'----------------------------------------

' Khai triển giá trị lựa chọn conversion

' từ chuỗi định dạng "Option=Data"

'----------------------------------------
```

```
Private Function ExtractOptionValue(ByVal strDummy As String) As String
' Khai triển từng dòng sau khi bằng 0
ExtractOptionValue = Left$(strDummy, InStr(strDummy, strEQUALS_SIGN) - 1)
End Function
'------------------------------------------------------------
' Liệt kê tất cả registered text converters và graphics filters
'------------------------------------------------------------
Private Sub ListConverters(keyHandleDummy As Long, _
ByVal lPredefRegKey As Long, _
ByVal strConverterDir As String, _
ByVal strWhatImLookingFor As String)
' Vị trí
Dim subKeyHandle&
Dim subSubKeyHandle&
Dim strKeyName$, strClassName$
Dim keyLen&, classLen&
Dim res&
Dim i%
' Nháp RegEnumKeyEx temps
Dim subKey As String * MAX_TEXT_BUFF ' tên converter
Dim subSubKey As String * MAX_TEXT_BUFF ' options subkey
Dim className As String * MAX_TEXT_BUFF
Dim keyLastWritten As FILETIME
' RegOpenKeyEx temps
Dim tempSubKey$
```

```
Dim tempAnotherSubKey$
' RegQueryValueEx temps
Dim strDataBuff As String * MAX_TEXT_BUFF
Dim strDataBuffSub As String
Dim lDataBuffSize As Long
' Cờ định hướng Export/Import
' True=Export, False=Import
Dim bExportImport As Boolean
Dim strTemp As String
' khởi tạo về vị trí
i% = 0
If ((strConverterDir = strREG_TEXT_CNV_IMPORT) Or _
(strConverterDir = strREG_GRAPH_FLT_IMPORT)) Then
bExportImport = False
Else
bExportImport = True
End If
' Mở đoạn Converters\Import key
res& = RegOpenKeyEx(lPredefRegKey, strConverterDir, _
0, KEY_ALL_ACCESS, keyHandleDummy)
If res& <> ERROR_SUCCESS Then
ReportRegError res&
Exit Sub
End If
' Đếm liệt kê tất cả converters và tốt nhất là mở các khoá keys
```

```
' cho mỗi converter subkey
Do While RegEnumKeyEx(keyHandleDummy, i%, subKey, MAX_TEXT_BUFF, 0, _
className, MAX_TEXT_BUFF, keyLastWritten) = ERROR_SUCCESS
' Mở nó
res& = RegOpenKeyEx(keyHandleDummy, subKey, 0&, KEY_ALL_ACCESS, _
subKeyHandle&)
' Nếu hỏng RegOpenKey thử lại 1 lần
If res& <> ERROR_SUCCESS Then
ReportRegError res&
Exit Do
Else
' Mở nội dung converter entries với khoá con Options subkey
res& = RegOpenKeyEx(subKeyHandle&, strREG_CNV_OPTIONS, 0&, _
KEY_ALL_ACCESS, subSubKeyHandle&)
lDataBuffSize = MAX_TEXT_BUFF
If res& = ERROR_SUCCESS Then
' thêm từng tên converter vào combo box
res& = RegQueryValueEx(subKeyHandle&, strWhatImLookingFor, _
0&, 0&, strDataBuff, lDataBuffSize)
If res& = ERROR_SUCCESS Then
' Ngắt trim và sửa cỡ của chuỗi string
strDataBuffSub = Left$(strDataBuff, lDataBuffSize - 1)
' và bảng table
clsCnvTable.AddConverter strDataBuffSub, subSubKeyHandle&, bExportImport
Else
```

```
'Thông báo lỗi ReportRegError res&

' converter kế tiếp với khoá Options key

GoTo ResumeDoLoop

End If

Else

GoTo ResumeDoLoop

End If

End If

ResumeDoLoop:

' make sure everything's neat and tidy

Call RegCloseKey(subKeyHandle&)

i% = i% + 1 ' i%++

Loop

' Rõ ràng và ngăn nắp...

Call RegCloseKey(keyHandleDummy)

Exit Sub

FatalError:

' Dọn dẹp xếp đặt

Set clsCnvTable = Nothing

DisplayErrorMsg strERR_INIT_MESSED_UP

End Sub

'----------------------------------------

' Thiết đặt radio buttons tuỳ theo

' thiết đặt trong list box

'----------------------------------------
```

```
Private Sub SetRadioButtons(ByVal strDummy As String)
' Các đối số có thể là "Yes" hoặc "No"
If strDummy = strOPT_YES Or strDummy = strOPT_NO Then
' Bật radio button bên phải lên
If strDummy = strOPT_YES Then
optYes.Value = True
optNo.Value = False
Else
optYes.Value = False
optNo.Value = True
End If
Else
DisplayErrorMsg strERR_WRONG_STRING
End If
End Sub
'-------------------------------------
' Cập nhật nội dung trong list box
'-------------------------------------
Private Sub UpdateOptions()
Dim i As Integer
Dim strListBoxEntry As String
' Các biến RegEnumValue
Dim hOptKey As Long ' Khoá Options reg key
Dim strValue As String * MAX_TEXT_BUFF
Dim lValueSize As Long
```

```
Dim dwTypeCode As Long

Dim strValueData As String * MAX_TEXT_BUFF

Dim lValueDataSize As Long

' Đầu tiên nhận và bỏ nội dung tồn tại

' (Nếu có bất cứ thứ gì trong List box)

If lstOptions.ListCount <> 0 Then

lstOptions.Clear

End If

' nhận key handle từ bảng table

hOptKey = clsCnvTable.OptionsHandle(cboConverters.SelText)

If hOptKey = 0 Then

GoTo FatalError

End If

' bây giờ nhận thiết đặt

dwTypeCode = 0&

lValueSize = MAX_TEXT_BUFF

lValueDataSize = MAX_TEXT_BUFF

' Thiết đặt gì?

i = 0

Do While RegEnumValue(hOptKey, i, strValue, lValueSize, 0&, dwTypeCode, _

strValueData, lValueDataSize) <> ERROR_NO_MORE_ITEMS

' Bỏ qua tất cả không phải chuỗi (non-string) các cặp Value=Data

If dwTypeCode <> REG_SZ Then GoTo ResumeDoLoop

' Đầu tiên ngắt và sửa các cỡ chuỗi

strListBoxEntry = Left$(strValue, lValueSize) & strEQUALS_SIGN & _
```

```
Left$(strValueData, lValueDataSize)
' lần lượt thêm vào list box
lstOptions.AddItem strListBoxEntry
lValueSize = MAX_TEXT_BUFF
lValueDataSize = MAX_TEXT_BUFF
ResumeDoLoop:
i = i + 1
Loop
If lstOptions.ListCount > 0 Then
lstOptions.ListIndex = 0
Else
GoTo FatalError
End If
Exit Sub
FatalError:
DisplayErrorMsg strERR_UPDATE_OPTIONS
End Sub
'--------------------------------------
' Chỉ thay đổi nội dung list box nếu
' lựa chọn mới khác
'--------------------------------------
Private Sub cboConverters_Change()
UpdateOptions
End Sub
Private Sub cmdHelp_Click()
```

```
MsgBox strHLP_DLG_MSG1 & Chr(10) & Chr(13) & _
strHLP_DLG_MSG2 & Chr(10) & Chr(13) & _
strHLP_DLG_MSG3, _
vbOKOnly, strHLP_DLG_CAPTION
End Sub
Private Sub cmdOK_Click()
' Xoá bảng table
Set clsCnvTable = Nothing
Unload formEdOptions
Exit Sub
End Sub
Private Sub cmdSet_Click()
CommitChange txtSetting.Value
End Sub
'--------------------------------------------------
' Đưa ra radio buttons hoặc thiết lập edit box
'--------------------------------------------------
Private Sub lstOptions_Change()
Dim strTemp As String
If lstOptions.Value <> "" Then
' khai triển chuỗi thực tế của option
strTemp = ExtractOptionData(lstOptions.Value)
'Nó là "Yes" hay "No"?
If strTemp = strOPT_YES Or strTemp = strOPT_NO Then
' Cho hiện các radio buttons, ẩn edit box
```

```
'đặt vào nút phải right button
SetRadioButtons strTemp
' Nếu edit box được hiện?
If txtSetting.Visible Then
' ẩn nó
lblSetting.Visible = False
txtSetting.Visible = False
cmdSet.Visible = False
' Hiện các nút!
optYes.Visible = True
optNo.Visible = True
End If
' Nó chỉ là một chuỗi string!
Else
' Cho hiện edit box, ẩn các radio buttons
' cập nhật các chuỗi thiết đặt
txtSetting.Value = strTemp
' các nút hiện?
If optYes.Visible Then
'ẩn chúng
optYes.Visible = False
optNo.Visible = False
' hiện edit box
lblSetting.Visible = True
txtSetting.Visible = True
```

```
cmdSet.Visible = True

End If

End If

End If

End Sub

Private Sub optNo_Click()

CommitChange strOPT_NO

End Sub

Private Sub optYes_Click()

CommitChange strOPT_YES

End Sub

' Khởi tạo dialog và control captions

Private Sub UserForm_Initialize()

' dialog caption

Caption = strMSG_CAPTION

' command buttons

cmdOK.Caption = strCMD_OK

cmdHelp.Caption = strCMD_HELP

cmdSet.Caption = strCMD_SET

' radio buttons

optYes.Caption = strOPT_YES

optNo.Caption = strOPT_NO

' nhãn labels

lblConversion.Caption = strLBL_CONVERSION

lblCnvOption.Caption = strLBL_CNV_OPTION
```

lblSetting.Caption = strLBL_SETTING

' lấp đầy combo box

If ControlsInit = False Then

fEditInitSuccess = False

DisplayErrorMsg strERR_NO_OPTIONS_FOUND

Else

fEditInitSuccess = True

End If

End Sub

2.2 Form thứ 2 có 1 nút lệnh

(Name)=RegOptionsForm

Caption=Thiết lập lựa chọn trong Registry'hay một tên khác tuỳ ý

(Name)=cmdClose

'Chèn 2 Page là

(Name)=Page1

(Name)=Page2

'Trang 1 có các control, bạn căn cứ vào tiếp đầu ngữ của tên để xác định, tôi bỏ giải thích vì quá dài

(Name)=lblWordOption

(Name)=lstWordOpt

(Name)=lblWordSetting

(Name)=txtWordSet

(Name)=lblWordOptDesc

'Trang 2

(Name)=lblEqOption

(Name)=lstEqOpt

(Name)=lblEqSetting

(Name)=txtEqSet

(Name)=lblEqOptDesc

'Viết lệnh cho form thứ 2

'Các biến lấy giá trị ra từ đường dẫn đờ phôn ngầm định của Word 2000 được viết bởi Word không viết nội dung của chúng vào Registry

Dim iAutoSavePath, iDocPath, iPictPath, iProgPath, iStartPath, iToolsPath, iUserPath, iWkgrpPath As String

Dim item__$(), List__$(), list2__$(), Key$, errortext$, sSection()

Dim WItem__$(), EQItem__$(), EQlist__$(), WList__$(), Wkey$, EQKey$, help$, RegOptions$, notset$, errmsg1$, change, errchange$

Dim Space_$, sWordSet, sEqSet

Dim LangList__$(), Lang__$(), CurLang$, NumLanguages, sLangLbl, sDictLbl, sEngLbl, sCancelMessage, sCancelTitle, sRegWord, sRegOpts, sRegProof, sRegEqDir, sRegEqGen

'=======================

' Sub để định vị

'=======================

Private Sub Localize()

Dim HelpTitle$

errortext$ = "Một lỗi sảy ra khi viết vào CSDL registration. Có lẽ lý do của trường hợp này là ảnh hưởng của thuộc tính READ ONLY trong thư mục Windows, hoặc Windows đang chạy ở chế độ Safe mode. Khởi động lại Restart Windows trước khi chạy lại Macro này."

CurLang$ = "1033" ' sử dụng ngôn ngữ đầu tiên trong danh sách

Space_$ = " "

sCancelMessage = "Bạn đã chọn cancel. Tất cả các thay đổi sẽ bị mất. Bạn có muốn tiếp tục và không cất những thay đổi mà bạn thiết lập không? Chọn No để cất những thay đổi."

sCancelTitle = "Thôi không lựa chọn Registry Options"

sSection(0) = "Word 2000 Options"

sSection(1) = "Equation Editor Options"

```
'-------Nhận các đường dẫn nội bộ đờ phôn Built-in Default mà không viết vào Registry-----

iAutoSavePath = Application.Options.DefaultFilePath(wdAutoRecoverPath)

iPictPath = Application.Options.DefaultFilePath(wdPicturesPath)

iProgPath = Application.Options.DefaultFilePath(wdProgramPath)

iStartPath = Application.Options.DefaultFilePath(wdStartupPath)

iToolsPath = Application.Options.DefaultFilePath(wdToolsPath)

iUserPath = Application.Options.DefaultFilePath(wdUserTemplatesPath)

iWkgrpPath = Application.Options.DefaultFilePath(wdWorkgroupTemplatesPath)

iDocPath = System.PrivateProfileString("",
"HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Options", "doc-path")

If iDocPath = "" Then

iDocPath = System.PrivateProfileString("",
"HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell
Folders", "Personal")

End If

'------------Word 2000 Options --------

WItem__$(0, 2) = "Đặt thư mục nơi mà AutoRecovery nháp sẽ cất các tệp vào. Đờ phôn ngầm
định sẽ là thư mục ..\<Windows>\Temp."

WItem__$(1, 2) = "Đặt đờ phôn đuôi để Word lưu backup. Thông thường đờ phôn sẽ là .WBK."

WItem__$(2, 2) = "Đặt cỡ lớn nhất của cache đồ hoạ Word bitmap(bằng KB). Nếu bạn sử dụng
nhiều đồ hoạ trong văn bản mà lớn hơn bitmap cache có thể cải thiện tốc độ cuộn và vẽ lại. Khi
đặt bằng 1 Word sẽ tự động điều chỉnh cỡ bitmap cache."

WItem__$(3, 2) = "Đặt cỡ của cache tệp Word file (bằng KB). Bạn có thể cải thiện tốc độ I/O
(Vào ra) đối với tệp và các tác vụ khác trong Word bằng thay đổi cỡ cache. Nhỏ nhất và đờ phôn
cỡ của cache là 64K."

WItem__$(4, 2) = "Thiết lập đờ phôn định dạng ngày tháng cho trường DATE. Chức năng này
chỉ sử dụng cho mục đích tương thích những phần trước. Word 2000 thường sử dụng." &
Chr(10) & Chr(10) & Space_$ & "Ví dụ: MMMM d, yyyy."

WItem__$(5, 2) = "Đờ phôn của tệp sử dụng văn bản Word. Đờ phôn là .DOC. Bạn phải khởi
động lại Word thì phần thiết lập này mới có tác dụng."

WItem__$(6, 2) = "Đờ phôn thư mục chứa những văn bản Word." & Chr(10) & Chr(10) &
Space_$ & "Đờ phôn là: " & iDocPath
```

WItem__$(7, 2) = "Đờ phôn tệp mẫu Word templates. Đờ phôn là .DOT. Bạn phải khởi động lại Word thì phần thiết lập này mới có tác dụng."

WItem__$(8, 2) = "Lựa chọn đường dẫn người dùng. Chức năng này chỉ sử dụng cho mục đích tương thích những phần trước. Nhớ rằng lựa chọn người dùng trong Word 2000 được cất ở Registry hay bên trong nó."

WItem__$(9, 2) = "Bật tắt Disables/Enables danh sách Font sử dụng gần nhất. Bạn phải khởi động lại Word thì phần thiết lập này mới có tác dụng. Vào số 0 hoặc1:" & Chr(10) & Chr(10) & Space_$ & "0 -Để có danh sách gần nhất Font." & Chr(10) & Space_$ & "1 - để không hiện danh sách này."

WItem__$(10, 2) = "Văn bản mẫu theo danh nghĩa template, được sử dụng khi tạo một đối tượng văn bản chứa Word hay ứng dụng OLE khác."

WItem__$(11, 2) = "Thiết lập đờ phôn đường dẫn khi sử dụng Insert Picture. Bạn phải khởi động lại Word thì phần thiết lập này mới có tác dụng. Bạn có thể cũng sử dụng Tools Options File Locations đế thiết đặt."

WItem__$(12, 2) = "Theo lý thuyết nơi mà các tệp chương trình Word được lưu trữ." & Chr(10) & Chr(10) & Space_$ & "Đường dẫn đờ phôn: " & iProgPath

WItem__$(13, 2) = "Cho phép bạn tô bóng đồ hoạ bằng viết đè bằng một chức năng vẽ đặc biệt lên một vài máy in Hewlett-Packard. Sử dụng lựa chọn này khi in chậm. nếu máy in không hỗ trợ chức năng vẽ đặc biệt, thì khoá chuyển chẳng có tác dụng. Vào 0 cho No, 1 cho Yes."

WItem__$(14, 2) = "Thiết lập đường dẫn khởi động các tệp Word startup, như là templates và WLL's để tải khi khởi động Word." & Chr(10) & Chr(10) & Space_$ & "Đường dẫn đờ phôn ngầm định là: " & iStartPath

WItem__$(15, 2) = "Đặt đờ phôn định dạng thời gian cho trường TIME. Chức năng này chỉ sử dụng cho mục đích tương thích những phần trước. Word 2000 sử dụng định dạng." & Chr(10) & Chr(10) & Space_$ & "Ví dụ: h:mm:ss"

WItem__$(16, 2) = "Thiết lập nơi mà Word sẽ tìm kiếm ở công cụ proofing, lọc filters, chuyển đổi converters, và các cấu thành khác, trong trường hợp này chúng được đăng ký đúng đắn hoặc không tìm thấy trong các thư mục chuẩn."

WItem__$(17, 2) = "Theo lý thuyết từ điển người dùng cho kiểm tra chính tả số tương đương với thứ tự của từ điển người dùng trong danh sách Custom Dictionariestrong hộp thoại Options của tab Spelling."

WItem__$(18, 2) = "Đường dẫn cho tệp mẫu người dùng user templates. Nên nhớ rằng khi thiết lập thay đổi này cho Word, nó sẽ ảnh hưởng đến tất cả các ứng dụng Microsoft Office 2000 , ảnh hưởng tới Office Shortcut Bar." & Chr(10) & Chr(10) & Space_$ & "Ngầm định đờ phôn: " & iUserPath

WItem__$(19, 2) = "Đường dẫn cho nhóm mẫu workgroup templates. Bạn có thể chỉ rõ đường dẫn UNC path, ví dụ: " & Chr(10) & Chr(10) & Space_$ & "\\GROUP\USER\MYTMPLTS"

EQItem__$(0, 2) = "Thư mục chương trình Equation Editor. Equation Editor phải được cài đặt và chạy 1 lần cho chức năng này và thiết lập khác Equation Editor hiện."

```
EQItem__$(1, 2) = "Thiết lập Custom Zoom. Chức năng này chỉ có tác dụng khi Equation Editor được vào và tách rời ra 1 cửa sổ theo đường thiết lập ForceOpen."

EQItem__$(2, 2) = "Forces Equation Editor để mở khi nó tự tách rời thành 1 cửa sổ. Vào 0 hoặc 1:" & Chr(10) & Chr(10) & Space_$ & "0 - Mở trong cùng 1 nơi." & Chr(10) & Space_$ & "1 - Mở thành cửa sổ tách rời."

EQItem__$(3, 2) = "Hiện hoặc ẩn những ký tự không in được non-printing ."

EQItem__$(4, 2) = "Toolbar đã được đăng ký? 1 là yes, 0 là no."

EQItem__$(5, 2) = "Vị trí của toolbar nó được đăng ký:" & Chr(10) & Chr(10) & Space_$ & "1 - đỉnh của cửa sổ EQ Editor." & Chr(10) & Space_$ & "2 - đáy của cửa sổ EQ Editor."

EQItem__$(6, 2) = "Toolbar hiện? 1 là yes, 0 là no."

EQItem__$(7, 2) = "Vị trí của toolbar khi nó không đăng ký, Theo lý thuyết ở toạ độ X,Y với góc trái trên. Đơn vị là pixel."

EQItem__$(8, 2) = "Số của vơ sần Equation Editor."

EQItem__$(9, 2) = "Cỡ phòng chuẩn trên View menu."

help$ = "Để sửa chữa các thiết lập hệ thống liên quan đến Word được lưu trữ ở Windows Registry chọn một mục từ danh sách Section list, sau đó chọn Option. Thay đổi thiết lập trong hộp Setting đánh giá trị mới và chọn Change. Xin hãy cẩn thận."

RegOptions$ = "Microsoft Word Registry Options"

HelpTitle$ = "Microsoft Word RegOptions Help"

notset$ = "Sử dụng ngầm định"

errmsg1$ = "Thiết lập sai giá trị."

errchange$ = "Một vài thiết lập Word liên quan đến Registry, xin thoát và khởi động lại Word."

End Sub

Private Sub cmdClose_Click()

Unload Me

End Sub

Private Sub lstWordOpt_Change()

If Not IsNull(lstWordOpt.Value) Then

i = lstWordOpt.ListIndex
```

```
' ngăn chặn khoảng trắng trong textbox

If WItem__$(i, 3) = " " Then

txtWordSet.Text = ""

Else

txtWordSet.Text = WItem__$(i, 3)

End If

lblWordOptDesc = WItem__$(i, 2)

End If

End Sub

Private Sub lstEqOpt_Change()

If Not IsNull(lstEqOpt.Value) Then

i = lstEqOpt.ListIndex

txtEqSet.Text = EQItem__$(i, 3)

lblEqOptDesc = EQItem__$(i, 2)

End If

End Sub

Private Sub txtWordSet_Enter()

sWordSet = txtWordSet.Value

End Sub

Private Sub txtWordSet_Exit(ByVal Cancel As MSForms.ReturnBoolean)

If Not sWordSet = txtWordSet.Value Then

i = lstWordOpt.ListIndex

WItem__$(i, 3) = txtWordSet.Text

If WItem__$(i, 3) = " " Then WItem__$(i, 3) = ""

System.PrivateProfileString("", WItem__$(i, 0), WItem__$(i, 1)) = WItem__$(i, 3)
```

```
End If

End Sub

Private Sub txtEqSet_Enter()

sEqSet = txtEqSet.Value

End Sub

Private Sub txtEqSet_Exit(ByVal Cancel As MSForms.ReturnBoolean)

If Not sEqSet = txtEqSet.Value Then

i = lstEqOpt.ListIndex

EQItem__$(i, 3) = txtEqSet.Text

If EQItem__$(i, 3) = " " Then EQItem__$(i, 3) = ""

System.PrivateProfileString("", EQItem__$(i, 0), EQItem__$(i, 1)) = EQItem__$(i, 3)

End If

End Sub

Private Sub UserForm_Initialize()

ReDim item__$(0)

ReDim List__$(32)

ReDim sSection(6)

ReDim WItem__$(0)

ReDim EQItem__$(7)

ReDim EQlist__$(9)

ReDim WList__$(19)

Dim i

Dim x

Key$ = ""

errortext$ = ""
```

```
Wkey$ = ""

EQKey$ = ""

help$ = ""

RegOptions$ = ""

notset$ = ""

errmsg1$ = ""

change = 0

errchange$ = ""

Space_$ = ""

CurLang$ = ""

NumLanguages = 0

ReDim List__$(51), list2__$(51)

ReDim WItem__$(19, 3), EQItem__$(9, 3)

sRegWord = "HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word"

sRegOpts = "HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Options"

sRegEqDir = "HKEY_CURRENT_USER\Software\Microsoft\Equation
Editor\3.0\Options\Directories"

sRegEqGen = "HKEY_CURRENT_USER\Software\Microsoft\Equation
Editor\3.0\Options\General"

Localize

mulMainForm.Pages.Item(0).Caption = sSection(0)

mulMainForm.Pages.Item(1).Caption = sSection(1)

mulMainForm.Value = 0

'------------ Word 7.0 Options ---------

WItem__$(0, 0) = sRegOpts

WItem__$(0, 1) = "AutoSave-Path"

WItem__$(1, 0) = sRegOpts
```

WItem__$(1, 1) = "Bak-Extension"

WItem__$(2, 0) = sRegOpts

WItem__$(2, 1) = "BitMapMemory"

WItem__$(3, 0) = sRegOpts

WItem__$(3, 1) = "CacheSize"

WItem__$(4, 0) = sRegOpts

WItem__$(4, 1) = "DateFormat"

WItem__$(5, 0) = sRegOpts

WItem__$(5, 1) = "DOC-Extension"

WItem__$(6, 0) = sRegOpts

WItem__$(6, 1) = "DOC-Path"

WItem__$(7, 0) = sRegOpts

WItem__$(7, 1) = "DOT-Extension"

WItem__$(8, 0) = sRegOpts

WItem__$(8, 1) = "INI-Path"

WItem__$(9, 0) = sRegOpts

WItem__$(9, 1) = "NoFontMRUList"

WItem__$(10, 0) = sRegOpts

WItem__$(10, 1) = "OLEDOT"

WItem__$(11, 0) = sRegOpts

WItem__$(11, 1) = "Picture-Path"

WItem__$(12, 0) = sRegOpts

WItem__$(12, 1) = "ProgramDir"

WItem__$(13, 0) = sRegOpts

WItem__$(13, 1) = "SlowShading"

```
WItem__$(14, 0) = sRegOpts

WItem__$(14, 1) = "Startup-Path"

WItem__$(15, 0) = sRegOpts

WItem__$(15, 1) = "TimeFormat"

WItem__$(16, 0) = sRegOpts

WItem__$(16, 1) = "Tools-Path"

WItem__$(17, 0) = sRegOpts

WItem__$(17, 1) = "UpdateDictonaryNumber"

WItem__$(18, 0) = sRegOpts

WItem__$(18, 1) = "User-Dot-Path"

WItem__$(19, 0) = sRegOpts

WItem__$(19, 1) = "Workgroup-Dot-Path"

'----------EQEdit

'--Các thư mục

EQItem__$(0, 0) = sRegEqDir

EQItem__$(0, 1) = "Appdir"

'--General

EQItem__$(1, 0) = sRegEqGen

EQItem__$(1, 1) = "CustomZoom"

EQItem__$(2, 0) = sRegEqGen

EQItem__$(2, 1) = "ForceOpen"

EQItem__$(3, 0) = sRegEqGen

EQItem__$(3, 1) = "ShowAll"

EQItem__$(4, 0) = sRegEqGen

EQItem__$(4, 1) = "ToolBarDocked"
```

```
EQItem__$(5, 0) = sRegEqGen

EQItem__$(5, 1) = "ToolBarDockPos"

EQItem__$(6, 0) = sRegEqGen

EQItem__$(6, 1) = "ToolBarShown"

EQItem__$(7, 0) = sRegEqGen

EQItem__$(7, 1) = "ToolBarWinPos"

EQItem__$(8, 0) = sRegEqGen

EQItem__$(8, 1) = "Version"

EQItem__$(9, 0) = sRegEqGen

EQItem__$(9, 1) = "Zoom"

'-------------------------

For i = 0 To 19

WItem__$(i, 3) = System.PrivateProfileString("", WItem__$(i, 0), WItem__$(i, 1))

If WItem__$(i, 3) = "" Then WItem__$(i, 3) = " "

WList__$(i) = WItem__$(i, 1)

Next

'--------------------

For i = 0 To 9

EQItem__$(i, 3) = System.PrivateProfileString("", EQItem__$(i, 0), EQItem__$(i, 1))

If EQItem__$(i, 3) = "" Then EQItem__$(i, 3) = " "

EQlist__$(i) = EQItem__$(i, 1)

Next

'----------------------

lstWordOpt.List() = WItem__$

txtWordSet.Text = WItem__$(0, 3)
```

```
lblWordOptDesc.Caption = WItem__$(0, 2)

lstWordOpt.ColumnWidths = "0;-1"

lstEqOpt.List() = EQItem__$

txtEqSet.Text = EQItem__$(0, 3)

lblEqOptDesc.Caption = EQItem__$(0, 2)

lstEqOpt.ColumnWidths = "0;-1"

End Sub

=============

'2.3 Chèn một Module bằng INSERT MODULE

'Module COMMON

Option Explicit

' Khai báo biến

' Vị trí khoá Key Registry

Public strRegSettingsKey As String

' Cờ

Public fEditInitSuccess As Boolean 'true Khi hộp Edit Conv Options 'được khởi tạo

' Forms

Public formEdOptions As EditCnvOptionsForm 'Options dialog

' Khác

Public clsCnvTable As CConverterTable ' Đối tượng bảng converter/filter

Public Sub EditConversionOptions()

' Bảng converter/filter

Set clsCnvTable = New CConverterTable

If clsCnvTable Is Nothing Then

GoTo FatalError
```

```
End If

' Khởi tạo dialog box

Set formEdOptions = New EditCnvOptionsForm

If formEdOptions Is Nothing Then

' Xoá bảng

Set clsCnvTable = Nothing

GoTo FatalError

End If

If fEditInitSuccess = True Then

formEdOptions.Show

End If

Exit Sub

FatalError:

DisplayErrorMsg strERR_INIT_OPTIONSFORM

End Sub

' Trả về giá trị được lưu trong khoá registry key strId

Public Function StrFetchPref(strId As String) As String

On Error GoTo LReturnNull

StrFetchPref = System.PrivateProfileString("", strRegSettingsKey, strId)

Exit Function

LReturnNull:

StrFetchPref = ""

End Function

' Đưa ra strValue trong registry key strId

'Public Sub StorePref(strId As String, strValue As String)
```

```
Dim strTemp As String

' If string is empty, store "~" as placeholder

If (Len(strValue) = 0) Then

strTemp = "~"

Else

strTemp = strValue

End If

System.PrivateProfileString("", strRegSettingsKey, strId) = strTemp

End Sub

' Trả về chuỗi được lưu trong strId. Nếu rỗng, trả về strDefault

Public Function StrRestorePref(strId As String, strDefault As String) As String

Dim strTemp As String

strTemp = StrFetchPref(strId)

' Nếu chiều dài bằng 0 nó không có trong regsistry, Nên sử dụng giá trị đờ phôn

If Len(strTemp) = 0 Then

strTemp = strDefault

' "~" là chứng cứ để cho phép trả về chuỗi rỗng

ElseIf strTemp = "~" Then

strTemp = ""

End If

StrRestorePref = strTemp

End Function

' Đưa ra iVal trong registry key strId như một chuỗi string

Public Sub StoreValPref(strId As String, iVal As Integer)

StorePref strId, LTrim$(str$(iVal))
```

```
End Sub
' Trả về giá trị được lưu trong khoá key strId
' Nếu sai trả về iDefault.
Public Function FRestorePref(strId As String, fDefault As Boolean) As Boolean
Dim strVal As String
strVal = StrFetchPref(strId)
If Len(strVal) = 0 Then
FRestorePref = fDefault
Else
FRestorePref = Val(strVal)
End If
End Function
' Trả về giá trị được lưu trong khoá key strId
' Nếu sai trả về iDefault.
Public Function IRestorePref(strId As String, iDefault As Integer) As Integer
Dim strVal As String
strVal = StrFetchPref(strId)
If Len(strVal) = 0 Then
IRestorePref = iDefault
Else
IRestorePref = Val(strVal)
End If
End Function
' Hiện lời nhắc lỗi bằng một balloon nếu sử dụng tên thuyết minh Assistant
' hay hiện hộp thoại khi đang ở chế độ dùng dialog.
```

```
Public Sub DisplayErrorMsg(strErrMsg As String)

MsgBox strErrMsg, vbExclamation + vbOKOnly + vbApplicationModal

Err.Clear

End Sub

==================

'2.4 Chèn một MODULE CLASS bằng INSERT MODULE CLASS

'Module CCONVERTERTABLE - CLASS

'------------------------------------------------------------

' Class Name: CConverterTable

' Mô tả: Lớp Class chứa tên và khoá reg bằng tay của đoạn text chuyển và lọc đồ hoạ.

'------------------------------------------------------------

Option Explicit

Option Compare Text

' Hướng dẫn chứa trong converter/filter data

Private Type CNV_FLT_ELEMENT

strCnvName As String ' Tên của converter/filter

hOptKey As Long 'Lựa chọn reg key bằng tay

iNext As Integer ' chỉ số index của phần tử kế tiếp trong danh sách đã nối kết

End Type

' Kể từ VBA không có các pointers, bảng này để sử dụng một bảng về trường lớn để giả vờ một danh sách nối kết

'-----------

' Các hằng constants

'-----------

' Khởi tạo cỡ bảng

Private Const iINIT_TBL_SIZE = 10
```

```
' Cỡ bước nhảy (số gia tăng)
Private Const iINC_STEP = 5
'Giá trị trường.iNext của phần tử cuối cùng trong danh sách nối kết
Private Const iLAST_ELMNT = -1
'----------------------
' Những thành viên riêng
'----------------------
'Chính bảng đó
Private m_arrCnvTable() As CNV_FLT_ELEMENT
' Cỡ hiện tại của bảng
Private m_iCurrTblSize As Integer
' Số hiện tại của các chuyển đổi
Private m_iCurrNumCnv As Integer
' Chỉ số của phần chuyển đầu tiên trong dánh sách
Private m_iFirst As Integer
'-----------------------------------------------
'--- Thêm vào phần thêm của chuyển đổi, đồng thời chuôi handle của nó ---
'--------- tới khoá Options key vào bảng ---------
'-----------------------------------------------
Public Sub AddConverter(ByVal strCnvNameDummy As String, _
ByVal hOptKeyDummy As Long, _
ByVal bExportImportFlag As Boolean)
Dim i, j As Integer
Dim bExpImpOn As Boolean
Dim strTempOld As String
```

```
Dim strTempNew As String
Dim iPrev As Integer ' Phần tử có trước được so sánh với phần tử mới
Dim iFoll As Integer ' phần tử theo sau được so sánh với phần tử mới
Dim iCurr As Integer ' Phần tử hiện tại được thêm vào danh sách
Dim iCompRes As Integer ' Kết quả chuỗi string so sánh
bExpImpOn = False
' Định lại cỡ bảng nếu cần thiết
If m_iCurrNumCnv = m_iCurrTblSize Then
' Cỡ gia tăng
m_iCurrTblSize = m_iCurrTblSize + iINC_STEP
' Định lại cỡ
ReDim Preserve m_arrCnvTable(m_iCurrTblSize) As CNV_FLT_ELEMENT
End If
' True=Export, False=Import
' Kiểm tra trùng tên trong bảng có rồi
For i = 0 To m_iCurrNumCnv - 1
If m_arrCnvTable(i).strCnvName = strCnvNameDummy Then
bExpImpOn = True
Exit For
End If
Next
' Nếu có một tên trùng thì thêm hậu tố
If bExpImpOn = True Then
' Nối thêm hậu tố Export/Import tới tên chuyển
If bExportImportFlag = True Then ' currently called from Exports
```

```
' Phần cũ phải ở Imports
strTempOld = m_arrCnvTable(i).strCnvName & strREG_IMPORT
m_arrCnvTable(i).strCnvName = strTempOld
strTempNew = strCnvNameDummy & strREG_EXPORT
Else
' Phần cũ từ Exports thì đưa vào Import
strTempOld = m_arrCnvTable(i).strCnvName & strREG_EXPORT
m_arrCnvTable(i).strCnvName = strTempOld
strTempNew = strCnvNameDummy & strREG_IMPORT
End If
End If
' Thêm tên chuyển vào bảng
' bộ đếm số gia tăng
m_iCurrNumCnv = m_iCurrNumCnv + 1
' Vào tên chuyển đổi và khoá cán Options key handle
With m_arrCnvTable(m_iCurrNumCnv - 1)
If bExpImpOn = False Then
.strCnvName = strCnvNameDummy
.hOptKey = hOptKeyDummy
Else
.strCnvName = strTempNew
.hOptKey = hOptKeyDummy
End If
End With
' Xếp theo chỉ số
```

```
' danh sách rỗng?
If m_iCurrNumCnv = 1 Then
m_iFirst = 0
m_arrCnvTable(0).iNext = iLAST_ELMNT
Else
'Khởi tạo con trỏ pointers về chỉ số
iPrev = m_iFirst
iFoll = m_iFirst
iCurr = m_iCurrNumCnv - 1
' Liên kết nó với dấu vết
Do
' So sánh nếu mới hơn thì thêm phần tử vào sau
iCompRes = StrComp(m_arrCnvTable(iCurr).strCnvName, _
m_arrCnvTable(iFoll).strCnvName)
' Chèn vào đây
If iCompRes = -1 Then
m_arrCnvTable(iCurr).iNext = iFoll
If iFoll = m_iFirst Then
m_iFirst = iCurr
Else
m_arrCnvTable(iPrev).iNext = iCurr
End If
Exit Do
' Mang đi để tìm kiếm
Else
```

```
iPrev = iFoll
' Cải tiến nếu nó không phải phần tử cuối cùng.,
' Trường hợp khác loại bỏ nó, lặp tới cuối cùng bằng mọi cách
If Not (m_arrCnvTable(iFoll).iNext = iLAST_ELMNT) Then
iFoll = m_arrCnvTable(iFoll).iNext
End If
End If
Loop Until (m_arrCnvTable(iFoll).iNext = iLAST_ELMNT)
' Tải xuống từng bước với toàn bộ danh sách?
If m_arrCnvTable(iFoll).iNext = iLAST_ELMNT Then
' tạo một con trỏ theo sau ở cạnh nó.
m_arrCnvTable(iFoll).iNext = iCurr
'Tạo nó cuối cùng
m_arrCnvTable(iCurr).iNext = iLAST_ELMNT
End If
End If
Exit Sub
End Sub
Public Sub AddConverterNamesToCombo(ByRef cboCombo As Object)
Dim iIndex As Integer
iIndex = m_iFirst
While Not (iIndex = iLAST_ELMNT)
cboCombo.AddItem m_arrCnvTable(iIndex).strCnvName
iIndex = m_arrCnvTable(iIndex).iNext
Wend
```

```
End Sub

Public Property Get ConverterCount() As Integer

' trả về số trong danh sách chuyển đổi hay lọc

ConverterCount = m_iCurrNumCnv

End Property

'----------------------------------------

'--- Bỏ qua tên converter/filter, ---

'---- trả về Options key handle của nó -----

'----------------------------------------

Public Property Get OptionsHandle(ByVal strCnvNameDummy As String) As Long

Dim i As Integer

' Tìm thứ mà nó chuyển thành

For i = 0 To m_iCurrNumCnv - 1

If m_arrCnvTable(i).strCnvName = strCnvNameDummy Then

Exit For

End If

Next

'kiểm tra theo quy cách

If m_arrCnvTable(i).hOptKey = 0 Then

GoTo FatalError

End If

' Trả về Options handle của nó

OptionsHandle = m_arrCnvTable(i).hOptKey

Exit Property

FatalError:
```

```
DisplayErrorMsg strERR_UPDATE_OPTIONS

End Property

'----------------------------

'--- Khởi tạo bảng ---

'----------------------------

Private Sub Class_Initialize()

' Thiết lập cỡ bảng để khởi tạo

m_iCurrTblSize = iINIT_TBL_SIZE

' Khởi tạo bảng với nó

ReDim m_arrCnvTable(m_iCurrTblSize) As CNV_FLT_ELEMENT

' chưa thêm phần chọn

m_iCurrNumCnv = 0

' Hoặc không hiểu đối với phần tử đầu tiên

m_iFirst = -1

End Sub

'----------------------------

'--- Dọn dẹp registry ---

'----------------------------

Private Sub Class_Terminate()

Dim i As Integer

' Đóng tất cả khoá Option reg keys

For i = 0 To m_iCurrNumCnv - 1

Call RegCloseKey(m_arrCnvTable(i).hOptKey)

Next

End Sub
```

==========

'2.5 Chèn một Module bằng INSERT MODULE

'MODULE: EditOptCommon

'---------------------------------------

' Khai báo hằng dùng chung

'---------------------------------------

Option Explicit

' Phần bao trùm

' Hộp thoại và tiêu đề lời nhắc

Global Const strMSG_CAPTION = "Sửa soạn Converter và Filter Options"

' Nhãn lời nhắc khi có lỗi

Global Const strERR_LIST_CONVERTERS = "ListConverters"

' Lời nhắc về lỗi

Global Const strERR_NO_OPTIONS_FOUND = "Không có lựa chọn tìm thấy trong registry."

Global Const strERR_CANNOT_OPEN_REG = "Không thể mở được registry."

Global Const strERR_CANNOT_OPEN_REGKEY = "Không thể mở được khoá registry key."

Global Const strERR_CANNOT_ADD_NAME = "Không thể thêm tên vào danh sách chuyển đổi converter list."

Global Const strERR_INIT_MESSED_UP = "Khởi tạo quy trình ngắt và bỏ qua."

Global Const strERR_WRONG_STRING = "Chuỗi sai quy cách."

Global Const strERR_UPDATE_OPTIONS = "Lựa chọn không thể cập nhật đúng đắn."

Global Const strERR_INIT_OPTIONSFORM = "Lỗi khởi tạo Edit Conversion Options form."

Global Const strERR_INTERNAL = "Macro không thể tiếp tục vì có một lỗi xảy ra."

Global Const strERR_PATH_TOO_LONG = "Đường dẫn thư mục quá dài."

' Lời nhắc lỗi Registry

Global Const strREG_ERR_CAPTION = "Registry hỏng."

```
Global Const strREG_ERR_BAD_DB = "Cơ sở dữ liêu sai."

Global Const strREG_ERR_BAD_KEY = "Sai khoá key."

Global Const strREG_ERR_CANT_OPEN = "Không thể mở registry."

Global Const strREG_ERR_CANT_READ = "Không đọc được registry."

Global Const strREG_ERR_CANT_WRITE = "Không viết vào được registry."

Global Const strREG_ERR_OUT_OF_MEMORY = "Bộ nhớ tràn."

Global Const strREG_ERR_INVALID_PARAMETER = "Sai tham số."

Global Const strREG_ERR_ACCESS_DENIED = "Reigstry bị từ chối truy xuất."

Global Const strREG_ERR_INVALID_PARAMETERS = "Sai các tham số."

Global Const strREG_ERR_NO_MORE_ITEMS = "Không thể nhiều khoản trong khoá key này."

Global Const strREG_ERR_BAD_ACCESS = "Truy nhập registry hỏng."

' Chuỗi trợ giúp Help

Global Const strHLP_DLG_CAPTION = "Edit Conversion Options Help"

Global Const strHLP_DLG_MSG1 = "Hộp lựa chọn Edit Converter and Filter"

Global Const strHLP_DLG_MSG2 = " cho phép bạn tự ý thiết lập duy nhất."

Global Const strHLP_DLG_MSG3 = " cho văn bản chuyển và lọc đồ hoạ."

' Nhãn các nút radio

Global Const strOPT_YES = "Yes"

Global Const strOPT_NO = "No"

' Nhãn các nút lệnh comand

Global Const strCMD_OK = "OK"

Global Const strCMD_CANCEL = "Cancel"

Global Const strCMD_HELP = "Help"

Global Const strCMD_SET = "Set"

' Nhãn hộp label
```

```
Global Const strLBL_CONVERSION = "Conversion:"

Global Const strLBL_CNV_OPTION = "Conversion Option:"

Global Const strLBL_SETTING = "Setting:"

' Chọn nút thư mục text

Global Const strSEL_FLD_BUTTON_TEXT = "Select"

'-------------------------------------------------------------

'********** Đừng khoanh vùng CODE phía sau điểm này *********

'-------------------------------------------------------------

Global Const strEQUALS_SIGN = "="

' Hằng Registry và keys

Global Const REG_SZ As Long = 1

Global Const REG_DWORD As Long = 4

Global Const HKEY_CURRENT_USER = &H80000001

Global Const HKEY_LOCAL_MACHINE = &H80000002

Global Const ERROR_SUCCESS = 0&

Global Const ERROR_BAD_DB = 1&

Global Const ERROR_BAD_KEY = 2&

Global Const ERROR_CANT_OPEN = 3&

Global Const ERROR_CANT_READ = 4&

Global Const ERROR_CANT_WRITE = 5&

Global Const ERROR_OUT_OF_MEMORY = 6&

Global Const ERROR_INVALID_PARAMETER = 7&

Global Const ERROR_ACCESS_DENIED = 8&

Global Const ERROR_INVALID_PARAMETERS = 87&

Global Const ERROR_NO_MORE_ITEMS = 259&
```

```
Global Const SYNCHRONIZE = &H100000

Global Const STANDARD_RIGHTS_READ = &H20000

Global Const STANDARD_RIGHTS_WRITE = &H20000

Global Const STANDARD_RIGHTS_EXECUTE = &H20000

Global Const STANDARD_RIGHTS_REQUIRED = &HF0000

Global Const STANDARD_RIGHTS_ALL = &H1F0000

Global Const KEY_QUERY_VALUE = &H1

Global Const KEY_SET_VALUE = &H2

Global Const KEY_CREATE_SUB_KEY = &H4

Global Const KEY_ENUMERATE_SUB_KEYS = &H8

Global Const KEY_NOTIFY = &H10

Global Const KEY_CREATE_LINK = &H20

Global Const KEY_READ = ((STANDARD_RIGHTS_READ Or KEY_QUERY_VALUE Or
KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY) And Not (SYNCHRONIZE))

Global Const KEY_WRITE = ((STANDARD_RIGHTS_WRITE Or KEY_SET_VALUE Or
KEY_CREATE_SUB_KEY) And (Not SYNCHRONIZE))

Global Const KEY_EXECUTE = (KEY_READ)

Global Const KEY_ALL_ACCESS = ((STANDARD_RIGHTS_ALL Or KEY_QUERY_VALUE Or
KEY_SET_VALUE Or KEY_CREATE_SUB_KEY Or KEY_ENUMERATE_SUB_KEYS Or
KEY_NOTIFY Or KEY_CREATE_LINK) And (Not SYNCHRONIZE))

Global Const REG_OPTION_NON_VOLATILE = 0

' Đường dẫn registry paths tới text converters và graphics filters

Global Const strREG_TEXT_CNV_IMPORT = "SOFTWARE\Microsoft\Shared Tools\Text
Converters\Import"

Global Const strREG_TEXT_CNV_EXPORT = "SOFTWARE\Microsoft\Shared Tools\Text
Converters\Export"

Global Const strREG_GRAPH_FLT_IMPORT = "SOFTWARE\Microsoft\Shared Tools\Graphics
Filters\Import"

Global Const strREG_GRAPH_FLT_EXPORT = "SOFTWARE\Microsoft\Shared Tools\Graphics
Filters\Export"
```

```
' Đường dẫn registry tới User Shell Folders và thiết lập Conversion Wizard

Global Const strREG_PERFORM_BATCH =
"Software\Microsoft\Office\9.0\Word\Wizards\Conversion Wizard"

Global Const strREG_USER_SHELL_FOLDERS =
"Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders"

' Vài tên giá trị và dữ liệu

Global Const strREG_ALWAYS_BATCH = "AlwaysBatch"

Global Const strREG_YES = "Yes"

Global Const strREG_NO = "No"

Global Const strREG_PERSONAL = "Personal"

Global Const strC_DIR = "C:\"

Global Const strREG_CNV_NAME = "Name"

Global Const strREG_CNV_OPTIONS = "Options"

Global Const strREG_IMPORT = " (Import)"

Global Const strREG_EXPORT = " (Export)"

' Cỡ lớn nhất của một text buffer

Global Const THREE_CHARS = 4

Global Const MAX_TEXT_BUFF = 255

Global Const ONE_K_BUFF = 1024

' Chọn thư mục trả về hằng số

Global Const lFILE_OPEN_SUCCESS = 0&

Global Const lFILE_OPEN_ERROR = 99&

' Các khoá chuyển và lọc registry keys

Global hCnvImpKeyHandle As Long

Global hCnvExpKeyHandle As Long

Global hFltImpKeyHandle As Long

Global hFltExpKeyHandle As Long
```

```
' FILETIME Type definition
Type FILETIME
dwLowDateTime As Long
dwHighDateTime As Long
End Type
'--------------------------------
' Các hàm API với Registry
'--------------------------------
' Mở khoá RegOpenKeyEx
Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA" _
(ByVal HKey As Long, ByVal lpSubKey As String, ByVal ulOPTIONS As _
Long, ByVal samDesired As Long, phkResult As Long) As Long
' Tạo khoá RegCreateKeyEx
Declare Function RegCreateKeyEx Lib "advapi32.dll" Alias _
"RegCreateKeyExA" (ByVal HKey As Long, ByVal lpSubKey As String, _
ByVal Reserved As Long, ByVal lpClass As String, ByVal dwOptions _
As Long, ByVal samDesired As Long, ByVal lpSecurityAttributes _
As Long, phkResult As Long, lpdwDisposition As Long) As Long
' Đóng khoá RegCloseKey
Declare Function RegCloseKey Lib "advapi32.dll" (ByVal HKey As Long) _
As Long
' Giá trị khoá RegEnumKeyEx
Declare Function RegEnumKeyEx Lib "advapi32.dll" Alias "RegEnumKeyExA" _
(ByVal HKey As Long, ByVal dwIndex As Long, ByVal lpname As String, _
lpcbName As Long, ByVal lpReserved As Long, ByVal lpClass As String, _
```

```
lpcbClass As Long, lpftLasrWriteTime As FILETIME) As Long
' Giá trị lọc RegQueryValueEx
Declare Function RegQueryValueEx Lib "advapi32.dll" Alias _
"RegQueryValueExA" (ByVal HKey As Long, ByVal lpValueName As String, _
ByVal lpReserved As Long, lpType As Long, ByVal lpData As String, _
lpcbData As Long) As Long
' Giá trị RegEnumValue
Declare Function RegEnumValue Lib "advapi32.dll" Alias "RegEnumValueA" _
(ByVal HKey As Long, ByVal dwValueIndex As Long, ByVal lpValue As String, _
lpValueSize As Long, ByVal lpReserved As Long, lpTypeCode As Long, _
ByVal lpValueData As String, lpcbValueDataSize As Long) As Long
' Đặt chuỗi giá trị vào Registry
Declare Function RegSetValueExString Lib "advapi32.dll" Alias _
"RegSetValueExA" (ByVal HKey As Long, ByVal lpValueName As String, _
ByVal lReserved As Long, ByVal dwValueType As Long, ByVal lpData As String, _
ByVal lcbDataSize As Long) As Long
Public Sub ReportRegError(lErrNum As Long)
Dim strMsg As String
' Nó là cái nào?
Select Case lErrNum
Case ERROR_BAD_DB
strMsg = strREG_ERR_BAD_DB
Case ERROR_BAD_KEY
strMsg = strREG_ERR_BAD_KEY
Case ERROR_CANT_OPEN
```

```
strMsg = strREG_ERR_CANT_OPEN

Case ERROR_CANT_READ

strMsg = strREG_ERR_CANT_READ

Case ERROR_CANT_WRITE

strMsg = strREG_ERR_CANT_WRITE

Case ERROR_OUT_OF_MEMORY

strMsg = strREG_ERR_OUT_OF_MEMORY

Case ERROR_INVALID_PARAMETER

strMsg = strREG_ERR_INVALID_PARAMETER

Case ERROR_ACCESS_DENIED

strMsg = strREG_ERR_ACCESS_DENIED

Case ERROR_INVALID_PARAMETERS

strMsg = strREG_ERR_INVALID_PARAMETERS

Case ERROR_NO_MORE_ITEMS

strMsg = strREG_ERR_NO_MORE_ITEMS

Case Else

strMsg = strREG_ERR_BAD_ACCESS

End Select

' hiện nó

MsgBox strMsg, vbOKOnly, strREG_ERR_CAPTION

End Sub

==========

' 2.6 Chèn một Module INSERT MODULE

'Module RegOptions

Sub RegOptions()
```

```
RegOptionsForm.Show

End Sub
```

Nói về các bus, địa chỉ bộ nhớ, liên kết...

Bạn thân mến! Trước khi đi sâu vào các phần khác của Windows API tôi muốn tóm lược lại một số vấn đề mà chúng ta cần phải thống nhất để dễ làm việc. Đầu tiên tôi xin được trình bày sơ lược về lưu trữ và các bus. Những vấn đề này, rất nhiều bạn đã được học và nghiên cứu ở giảng đường Đại học, nhưng cũng không ít bạn vẫn còn mơ hồ. Tôi xin phép được trình bày theo phương pháp QTN (Đơn giản hoá), một phương pháp mà tôi đã được học và hiện tại đang áp dụng tại Trung tâm Tin học ABC. Tôi không được học Tin học chính quy nên nếu thuật ngữ không đồng nhất với bạn xin được góp ý và chỉ bảo.

Trước tiên vô cùng cảm ơn chân thành tới TS. Quách Tuấn Ngọc (Bộ Giáo dục và Đào tạo), anh Đặng Minh Tuấn (Tác giả VietKey), anh Lưu Hà Xuyên(Tác giả Vietspell), bạn Nguyễn Hồ Thiên Đăng (Thành phố Hồ Chí Minh), chú Do Tuyen (Trường Đại học Los Angeles) và đông đảo các bạn học sinh sinh viên và người học ở trong nước và nước ngoài đã động viên, giúp đỡ, định hướng và cung cấp tài liệu cả tiếng Việt và tiếng Anh để tôi hoàn thành bộ bài viết này.

Xin lưu ý: Tất cả những gì đề cập trong bài viết của tôi bạn đều có thể DownLoad từ các kho của Web Lê Hoàn và trên đĩa CD do cửa hàng Duy Nghi cung cấp.

Địa chỉ cá nhân: ngphthaovn@yahoo.com (Không giải đáp Tin học)

Địa chỉ giải đáp Tin học: fthaoabc@yahoo.com - Do Trung tâm Tin học ABC quản lý.

Các bài viết về API được tham khảo và biên soạn và hoàn thiện từ các chương trình mà tôi được gửi tặng:

1. Bộ đĩa CDROM: Microsoft Studio MSDN.
2. Website của Microsoft. http://msdn.microsoft.com/
3. Đĩa CDROM: Thế giới Vi tính.
4. Giáo án Máy tính và Lập trình Windows của Nguyễn Cường - Trung tâm Tin học ABC
5. Các giáo án Đào tạo Tin học của Trường Đại học Lốt An giơ lét (Hoa Kỳ), Trường Đại học Ham buốc (CHLB Đức), Trường Đại học Tổng hợp Vác sa va (Ba Lan), Trường Đại học Bách Khoa Hà Nội, Học viện Kỹ thuật Quân sự (Việt Nam), Học viện Hàng Không Jucopxki (Nga)...
6. Một loạt các bài viết chuyên đề và tốt nghiệp của các bạn sinh viên và các kỹ sư ở mọi lĩnh vực...
7. Kỹ xảo lập trình Visual Basic của Lê Hữu Đạt
8. Hướng dẫn Lập trình Visual Basic với WIN API của tập đoàn SAMIS

- **Bus và địa chỉ các ô nhớ**

Công việc của máy tính cũng giống như một công việc làm hàng ngày của bạn. Khi bạn ra lệnh cho máy làm một công việc nào đó thì bạn phải nói rõ công việc đó cần phải làm như thế nào. Ta hãy nghiên cứu một công việc đơn giản để dễ hiểu. Ví dụ ra lệnh đơn giản:

- **Mang sách đến bàn học.**

Ta thấy câu lệnh này gồm các phần:

- Lệnh: Mang đi - Trong tin học nó được đưa vào Bus lệnh.

- Mang gì: Sách - Trong tin học nó được đưa vào Bus dữ liệu.

- Mang đi đâu - Trong tin học nó được đưa vào Bus địa chỉ.

Có nghĩa là một lệnh đơn giản thường bao hàm 3 loại trên, ta tạm gọi nó là các dữ kiện để hoàn thành một lệnh. Có một số lệnh không nhất thiết phải có đủ 3 dữ kiện trên như: Tắt điện (2 dữ kiện, lệnh: tắt, dữ liệu: điện) hay Thôi (Chỉ một dữ kiện lệnh: Thôi)....

Vấn đề ra lệnh như trên người ta gọi là ra lệnh trực tiếp. Ra lệnh trực tiếp có ưu điểm là máy sẽ thực hiện rất nhanh nhưng sẽ khó khăn đối với việc tổ chức thành các hệ thống công việc dây chuyền để hình thành nên chương trình ứng dụng. Chính vì vậy các hệ điều hành thường chia địa chỉ bộ nhớ (RAM) ra thành các phần, tại mỗi phần tạo thành các ngăn. Có nhiều loại ngăn phục vụ cho từng mục đích khác nhau.

Ta thấy có một loại ngăn như thế này:

Số chỉ vị trí A1	Tại sao lại là các số chỉ ra một vị trí nào đó? Đây chính là các ngăn ghi địa chỉ của công việc cần tiến hành. Nghĩa là hệ điều hành hoặc bất kỳ một chương trình nào cũng có một ngăn như thế này, để lưu danh sách các công việc cần thực hiện (Địa chỉ của công việc nằm trên danh sách này)
Số chỉ vị trí A2	
.....	
Số chỉ vị trí An	

Một tệp lệnh dạng COM, BIN, BAT hay EXE khi thi hành sẽ được nạp toàn bộ hoặc một phần lên bộ nhớ RAM chiếm một vị trí trong bộ nhớ RAM này. Khi nạp xong, nó sẽ báo địa chỉ nằm của nó lên một ngăn lưu trữ địa chỉ dạng trên.

Khi có biến cố tương ứng, hệ điều hành sẽ tra trong ngăn trên để tìm vị trí lưu công việc phải làm. Nó sẽ tìm đến một dạng ngăn thứ 2.

Độ rộng của ngăn tới vị trí...	Tại ngăn này nó sẽ biết được lệnh phải thực hiện là lệnh gì, vị trí ở đâu và những dữ liệu gì kèm theo để thực hiện nó. (3 dạng dữ kiện như phần trên tôi đã trình bày).
Số chỉ lệnh thực hiện	
Số chỉ địa chỉ	
Số chỉ dữ liệu	
Các số khác nếu có...	
.....	
Số chỉ vị trí lệnh tiếp	

Các ngăn này liên tục được bổ sung do chính chương trình cốt lõi liên tục đưa vào tuỳ thuộc vào người sử dụng tác động hay một biến cố phát sinh.

Từ đây ta lại nói chuyện về các hàm API một cách chung nhất (Về riêng từng hàm, tôi đều có những bài viết riêng). Ta có thể dùng API để nhận một vị trí của một lệnh A nào đó, tìm đến thay đổi hay copy dữ kiện của lệnh đó. Nếu thay đổi, ta sẽ chỉ cho hệ điều hành ra thi hành một lệnh mới ở một vị trí mà ta chỉ định, thực hiện xong lệnh này ta lại chỉ vị trí lại cho hệ điều hành quay về thực hiện tiếp lệnh A trên. Đây thực sự là một trong những trò mà các nhà viết Virus thực hiện.

Nếu copy dữ kiện của lệnh thì ta biết được vị trí của lệnh A sẽ thực hiện, ta có thể:

- Xử lý song song với lệnh A, bằng lệnh A' mà ta chuẩn bị.

- Thay đổi lại một phần dữ kiện của lệnh A hoặc những lệnh sau lệnh A để đạt một ý đồ nào đó.

Nói chung trong các kỹ thuật lập trình người ta nhiều cách để chặn lại như: Chặn từ ngăn gốc, chặn dở chừng chủ yếu là làm sao thuận lợi khi đưa ra kết quả lệnh.

Các hàm API thường sử dụng khi thực hiện những chức năng này thường các hàm có chữ đầu là Get... để nhận, hàm Set... để đặt, hàm Copy... để copy, hàm Send... hay Post... để đưa đi. Bạn hãy đọc kỹ các ví dụ của các bài viết để nắm được hồn cũng như ý đồ của tác giả.

- **Các biến hay địa chỉ của các ô nhớ.**

Hệ điều hành chắc chắn phải giành những chuyên khu để ghi giá trị các biến trong tính toán mà chương trình có sử dụng. Một ngăn chương trình có thể ghi tên biến vào ngăn dạng 1. Khi làm việc với biến, hệ điều hành sẽ đọc tên biến, tìm đến vị trí lưu trữ xem độ lớn của chỗ ghi giá trị và lấy giá trị. Xong việc nó sẽ thực hiện tiếp lệnh kế.

Mỗi biến là một vùng các ô nhớ. Những giá trị cần lưu trung gian có thể được đưa tới vị trí khác, hay thậm chí để tạm ra đĩa trong một thư mục đóng vai trò temp nào đó (Tuỳ theo thiết lập của phần mềm và hệ điều hành).

Trong các chương trình độc lập mà ta xây dựng bao giờ cũng được phân làm 2 loại chính khi liên kết để lấy dữ liệu hoặc gọi các hàm và chương trình ngoại lai (Hàm và chương trình không phải do mình thiết kế, hoặc của Visual Basic). Liên kết tĩnh là liên kết khi ta thiết kế chương trình. Liên kết động xảy ra trong khi chương trình chạy.

Liên kết tĩnh có ưu điểm là thực hiện dễ, nhưng nếu một hàm nào đó sử dụng lặp lại nhiều lần thì khi thi hành nó sẽ chiếm cứ bộ nhớ từng ấy lần, gây nên sự hao phí ký ức giống như để cắt từng ấy nhát bạn phải chuẩn bị từng ấy con dao(!!!). Còn liên kết động thông qua các bộ thư viện liên kết là các tệp tin dạng *.DLL (Dynamic link library - Thường được tạo ra bằng ngôn ngữ C) hoặc tệp dạng *.VBX (Tạo ra bằng ngôn ngữ Visual Basic) hoặc tệp chuẩn hệ thống của Windows dạng *.EXE thì tận dụng được ký ức tạo nên sự mềm dẻo.

- **Cửa sổ - Khái niệm ôm đồm của Windows API**

Thật kỳ quặc khi chính Microsoft đưa ra khái niệm Window khi lập trình API, mà lẽ ra phải gọi đó là các Object hoặc Control. Khi nói đến cửa sổ xin bạn lưu ý nó không chỉ là Form, mà còn có thể là bất cứ một đối tượng độc lập nào trong hộp Tool Box như các thanh cuộn (Scroll Bar), các hộp Text Box... ngay cả chính các biểu tượng Icon cũng được Win API coi là các cửa sổ (!?!). Ngoài ra còn có loại cửa sổ âm thầm mà không nhìn thấy trên màn hình.

Tất cả các cửa sổ kiểu trên đều thuộc vào các lớp. Mỗi lớp đều có các tính chất riêng. Có nghĩa là khi ta đổi vị trí của một cửa sổ từ lớp này sang lớp khác ngoài các tính chất riêng đặc thù của

nó, còn các tính chất chung theo lớp nó sẽ được tiếp nhận ngay các tính năng của lớp mới và giã từ những tính chất của lớp cũ mà nó phụ thuộc.

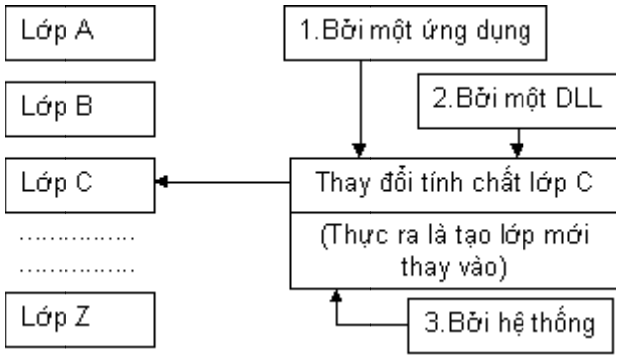
Tìm hiểu các lớp hệ thống thường được sử dụng và đã có sẵn trong Windows (Xin lưu ý tên lớp không có dấu cách)

Tên lớp	Mô tả
BUTTON	Dùng cho các nút lệnh (Command button), nút chọn (Option button), nút kiểm tra (Check box)
COMBOBOX	Dùng cho hộp Combo box
EDIT	Dùng cho các Edit Control
LISTBOX	Dùng cho các List Box
SCROLLBAR	Dùng cho các thanh cuộn
STATIC	Dùng cho các cửa sổ hiển thị văn bản
MDICLIENT	Dùng cho các cửa sổ giao diện nhiều tài liệu MDI

- Xin quan sát các tính chất của mỗi lớp:

Tính chất	Công dụng
Class Style (Kiểu lớp)	Thiết đặt các thuộc tính mẫu của mỗi loại cửa sổ trong lớp.
Class Function (Hàm lớp)	Các hàm mặc định của lớp.

Instance (Thể hiện)	Mô tả phiên bản sở hữu lớp. Thường là lớp hệ thống hoặc thư viện.
Icon (Biểu tượng)	Mô tả biểu tượng mặc nhiên trên Desktop khi cửa sổ của lớp này thu nhỏ Minimize.
Cursor (Con trỏ)	Mô tả con trỏ mặc nhiên khi chuột được định vị trên cửa sổ lớp này.
Background (Nền)	Mô tả màu nền mặc nhiên đối với các cửa sổ trong lớp này.
Menu (Thực đơn)	Mô tả thực đơn mặc nhiên đối với cửa sổ thuộc lớp này.



Bạn có thể thay đổi các tính chất của một lớp trong chương trình của bạn khi chương trình thi hành. Bạn có thể điều chỉnh các tính chất này. Các giá trị tính chất ban đầu của lớp sẽ phục hồi lại tuỳ thuộc vào thời điểm bạn unload bỏ lớp mà bạn đã điều chỉnh. Xem sơ đồ sau:

Thời điểm bạn Unload bỏ lớp mà bạn đã thay đổi tuỳ thuộc vào lớp đó bị thay đổi bởi ứng dụng, thì bạn nên khôi phục nó về ban đầu. Còn các lớp do DLL định nghĩa thì thường được unload khi các ứng dụng sử dụng DLL kết thúc. Khi Windows nạp nó sẽ đưa về các lớp về tình trạng ban đầu.

- **Subclassing**

Xin bạn xem lại bài viết API khám phá từ A đến Z phần 2, tôi đã nói rõ việc xử lý của các hàm Window. Mỗi lớp có hàm Window mặc nhiên để sử dụng cho mọi cửa sổ trong lớp. Một cửa sổ không nhất thiết phải sử dụng hàm mặc nhiên này, ta có thể tạo một lớp con (Subclassing) của một cửa sổ. Xây dựng các tính chất mới trên lớp con này. Hàm lớp mới có thể trực tiếp xử lý một số chỉ lệnh và chuyển giao các chỉ lệnh message cho lớp hiện có.

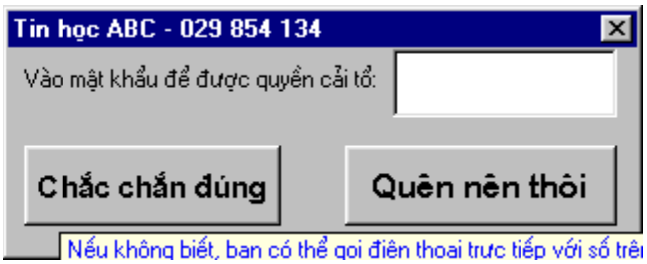
Nghĩa là tất cả các cửa sổ được tạo trên lớp mới này sẽ có hàm Window mặc nhiên cộng với các chức năng mới được tạo ra bởi lớp con.

Đối với Kỹ thuật Subclassing một Window để chặn tất cả các chỉ lệnh gửi đến nó có thể chọn các phương án:

- Kiểm tra message để biết
- Can thiệp vào message trước khi để nó đến đích.
- Bẫy một message sau khi Window gốc đã xử lý, thay đổi kết quả nó trả về cho ứng dụng gọi hay hệ điều hành.
- Can thiệp một message, tự xử lý nó. Không đưa nó cho Window gốc. Thay Window gốc làm mọi việc.

Để một thủ tục kết hợp với mỗi Window và xử lý tất cả các chỉ lệnh message đến ta thường dùng hàm SendMessage hay PostMessage.

Ta xét ví dụ sau để hiểu rõ về API, ta không xét về lập trình thông thường Visual Basic mà chỉ xét đến bản chất API qua ví dụ nhỏ này thôi.



Giả sử bạn tạo một form con như sau: Có Text Box với Name =TxtPass, Nút lệnh 1 có Name =CmdOK, nút lệnh 2 có Name=CmdCancel.

Ta viết lệnh để tìm nhanh chóng chuỗi bên trong hộp TxtPass.

```
Option Explicit

'Khai báo

Public Const LB_FINDSTRINGEXACT = &H1A2

Declare Function GetFocus Lib "user32" Alias "GetFocus" () As Long

Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long,
ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As Long

Sub cmdCancel_Click()

End

End Sub

Sub CmdOK_Click()

Dim hw%, chuoitrave&

TxtPass.SetForcus

hw=txtPass.hwnd
```

chuoitrave&=SendMessage(hw%, LB_FINDSTRINGEXACT,0,0)

'Các lệnh xử lý chuỗi trả về

End Sub

Từ VB5 đến VB6 Kỹ thuật subclassing được đề nghị thông qua từ khóa địa chỉ AddressOf. Thông thường để sử dụng ta phải làm các bước:

1. Chuẩn bị thủ tục thay thế thủ tục Window.

2. Ghi nhớ địa chỉ cũ bằng hàm SetWindowLong, đồng thời đặt địa chỉ mới vào địa chỉ của thủ tục thay thế.

DiaChiCu= SetWindowLong(hWnd, GWL_WNDPROC, AddressOf Tên_Thủ_tục_Thay_Thế)

Sẽ có bạn hỏi AddressOf Tên_Thủ_tục_Thay_Thế là gì? Điều này ta không quan tâm vì khi nạp vào RAM, hệ thống sẽ tự điền và thay thế giúp cho chúng ta, là giá trị tuỳ theo máy cũng như tuỳ theo chương trình của mình, thế mới hay chứ.

Hàm này có khai báo chuẩn như sau:

Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long

Trong đó:

Hwnd - Cán của Window cần đổi thuộc tính

nIndex chọn một trong các hằng sau tuỳ theo mục đích:

Const GWL_HINSTANCE = (-6) - Handle của minh hoạ làm chủ Window

Const GWL_EXSTYLE = (-20) - Kiểu mở rộng của Window

Const GWL_STYLE = (-16) - Kiểu Window

Const GWL_ID = (-12) - ID của một Window con trong một hộp thoại

Const GWL_USERDATA = (-21) - Được định nghĩa bởi ứng dụng

Const GWL_HWNDPARENT = (-8) - Cán của Window cha.

Const GWL_WNDPROC = (-4) - Địa chỉ của thủ tục Window

Const DWL_DLGPROC = 4 - Địa chỉ hàm Dialog của Window

Const DWL_MSGRESULT = 0 - Giá trị trả về của thông điệp được xử lý trong hàm Dialog

Const DWL_USER = 8 - Được định nghĩa bởi ứng dụng.

Đối với tính chất GWL_WNDPROC, hàm SetWindowLong không chỉ đặt một giá trị mới mà còn trả về địa chỉ trước của đề mục đó. Ta lưu địa chỉ cũ để khi hồi phục lại khi kết thúc bằng chính hàm này.

SetWindowLong hWnd, GWL_WNDPROC, DiaChiCu

Những chỉ lệnh message gửi đến cho Window nó sẽ đưa vào cho thủ tục thay thế của bạn. Nếu thủ tục thay thế của bạn muốn thủ tục cũ thực hiện thì dùng hàm gọi thủ tục cũ CallWindowProc như sau:

ViecChoCuLam = CallWindowProc(DiaChiCu, hwnd, uMsg, wParam, lParam)

Trong đó biến ViecChoCuLam là bạn khịa ra để việc gọi hàm thực hiện cho thuận lợi.

Xin lưu ý việc khôi phục địa chỉ phải được thực hiện tránh quên và phải đặt trước lệnh End. Nếu không sẽ treo và dẫn tới nguy hiểm.

Các ví dụ chuyên sâu về SubClassing xin download từ Web Lê Hoàn.

Chúc các bạn đạt được ý nguyện mong muốn.

Xử lý của API khi làm việc với phần cứng và hệ thống

Bạn thân mến!

Để thuận lợi cũng như để mọi người cùng học tập và tham gia xây dựng đề án "Xây dựng bộ gõ tiếng Việt tăng tốc" chúng ta hãy làm quen với các hàm API liên quan đến vấn đề này. Mong rằng đây sẽ là cơ sở cơ bản để chúng ta cùng tiếp cận tìm hiểu cơ chế xây dựng một bộ gõ tiếng Việt đơn giản tại Website Lê Hoàn. Bạn có thể xem các hàm liên quan đến UNICODE hoặc các bài học cơ bản ban đầu Visual Basic tại trang http://www.bangden.com/soncuoc. Chi tiết và ví dụ các hàm bạn có thể tìm ở trang này.

1. Các hàm với chuột, con trỏ

Visual Basic cung cấp sự yểm trợ cho chuột và trỏ chuột (Trỏ chịu sự tác động của chuột thường để điều khiển) cũng như trỏ thanh (Trỏ chịu tác động của bàn phím thường để nhập liệu), tại một thời điểm chỉ có duy nhất một trỏ chuột và một trỏ thanh. Theo ngầm định, vị trí của trỏ chuột và trỏ thanh tính theo toạ độ của màn hình.

Windows cung cấp khả năng hạn chế con chuột vào một vùng khai báo trên màn hình gọi là clipping. Có hai hàm cơ bản là:

1. Declare Function ClipCursor Lib "user32" Alias "ClipCursor" (lpRect As Any) As Long

Công dụng: Giới hạn trỏ chuột đối với vùng chỉ định.

Tham số kèm:

LpREct - Vùng trỏ chuột định vị.

2. Declare Function GetClipCursor Lib "user32" Alias "GetClipCursor" (lprc As RECT) As Long

Công dụng: Nhận hình chữ nhật làm vùng làm việc cho trỏ chuột.

Tham số kèm:

Lprc - Hình chữ nhật nhận vùng làm việc.

Để nhận vị trí trỏ chuột, hay đặt trỏ chuột vào một vị trí ta sử dụng hai hàm:

3. Declare Function GetCursorPos Lib "user32" Alias "GetCursorPos" (lpPoint As POINTAPI) As Long

Công dụng: Xác định vị trí hiện tại của con trỏ chuột.

Tham số kèm:

LpPoint - Cấu trúc tiếp nhận toạ độ con trỏ trên màn hình.

4. Declare Function SetCursorPos Lib "user32" Alias "SetCursorPos" (ByVal x As Long, ByVal y As Long) As Long

Công dụng: Đặt con trỏ chuột vào một vị trí.

Tham số kèm:

X, Y Toạ độ màn hình cần đặt vị trí.

Các con trỏ theo ý muốn có thể được tạo từ cặp bitmap đơn sắc, kích thước có thể tới 32x32 pixel, nó có một cán 16 bit quản lý.

Khi ở một vị trí nào đó, ta thao tác chuột như click, d-click, hoặc drag. Các thao tác này sẽ ảnh hưởng đến một cửa sổ X nào đó. Ngay sau đó cửa sổ X này sẽ được nhận focus ngay cả khi trước đó nó đã mất focus. Vì vậy, cửa sổ này còn được gọi là capture (Hay cửa sổ đón chặn hay cửa sổ chộp).

Các hàm liên quan đến trỏ chuột cơ bản gồm:

5. Declare Function CopyCursor Lib "user32" Alias "CopyCursor" (ByVal hcur As Long) As Long

Công dụng: Copy thêm một trỏ chuột.

Tham số kèm:

hCur Cán của trỏ chuột cần sao chép.

6. Declare Function GetCapture Lib "user32" Alias "GetCapture" () As Long

Công dụng: Xác định cửa sổ đón chặn các tình huống chuột.

Trị trả về: Integer - Cán cửa sổ chặn tình huống chuột. Zero nếu không có cửa sổ nào.

7. Declare Function GetCursor Lib "user32" Alias "GetCursor" () As Long

Công dụng: Nhận cán của trỏ chuột hiện tại.

Trị trả về: Integer cán của trỏ chuột hiện tại. Zero nếu không có trỏ chuột hiện tại.

8. Declare Function GetDoubleClickTime Lib "user32" Alias "GetDoubleClickTime" () As Long

Công dụng: Xác định thời gian 2 lần nháy của thủ tục Dclick.

Trị trả về: Thời gian tính theo mili giây.

9. Declare Function LoadCursor Lib "user32" Alias "LoadCursorA" (ByVal hInstance As Long, ByVal lpCursorName As String) As Long

Công dụng: Tải một trỏ chuột.

Tham số kèm:

HInstance Cán của chương trình mô tả trỏ chuột

LpCursorName - Chỉ số ID nguồn. Căn cứ vào chỉ số này mà trỏ sẽ có hình dáng khác nhau. Đây là những hình dáng ngầm định của hệ thống.

IDC_APPSTARTING - Trỏ khi một ứng dụng khởi động

IDC_NO - Trỏ khi không thực hiện việc nào.

IDC_SIZE - Trỏ khi thay đổi cỡ một đối tượng.

IDC_SIZEALL - Trỏ khi thay đổi tất cả cỡ của các đối tượng

IDC_SIZENESW - Trỏ là hình mũi tên 2 chiều chéo góc xuống dốc

IDC_SIZENS - Trỏ là hình mũi tên 2 chiều dọc.

IDC_SIZENWSE - Trỏ là hình mũi tên 2 chiều chéo lên dốc

IDC_SIZEWE - Trỏ là hình mũi tên 2 chiều ngang

IDC_UPARROW - Trỏ là hình mũi tên lên

IDC_WAIT - Trỏ là đồng hồ cát

IDC_ARROW: Trỏ hình mũi tên.

IDC_CROSS: Trỏ hình dấu thập.

IDC_IBEAM: Trỏ hình thanh chữ I

IDC_ICON: Biểu tượng rỗng.

10. Declare Function LoadCursorFromFile Lib "user32" Alias "LoadCursorFromFileA" (ByVal lpFileName As String) As Long

Công dụng: Tải một hình dáng trỏ chuột từ tệp bên ngoài được khai báo

Trị trả về: Integer - Zero nếu thành công.

Tham số kèm:

LpFileName - Cán của tệp mô tả hình dáng trỏ.

11. Declare Function ReleaseCapture Lib "user32" Alias "ReleaseCapture" () As Long

Công dụng: Tải bỏ việc chặn đối với chuột.

12. Declare Function SetCapture Lib "user32" Alias "SetCapture" (ByVal hwnd As Long) As Long

Công dụng: Thiết lập việc chặn con chuột đối với cửa sổ.

Trị trả về: Integer là cán của cửa sổ trước đó đang chặn chuột. Zero nếu không có cửa sổ nào chặn trước đó.

Tham số kèm:

hwnd - Cán của cửa sổ cần thiết lập để chặn.

13. Declare Function SetCursor Lib "user32" Alias "SetCursor" (ByVal hCursor As Long) As Long

Công dụng: Đặt con trỏ hiện tại.

Trị trả về: Integer - Trị số mô tả trỏ trước đó.

Tham số kèm:

HCursor - Cán của con trỏ cần đặt làm hiện tại.

14. Declare Function SetDoubleClickTime Lib "user32" Alias "SetDoubleClickTime" (ByVal wCount As Long) As Long

Công dụng: Đặt thời gian giữa hai lần bấm của thao tác Dclick.

Tham số kèm:

WCount - Thời gian cần đặt, tính theo miligiây.

15. Declare Function ShowCursor Lib "user32" Alias "ShowCursor" (ByVal bShow As Long) As Long

Công dụng: Hiện hay giấu con trỏ trên màn hình.

Trị trả về: Số nguyên DP báo số lần hiển thị. Mỗi lần sử dụng hàm này để hiện trỏ DP sẽ được cộng 1. Mỗi lần giấu đi DP sẽ trừ 1. Nếu DP âm trỏ không được hiện.

Tham số kèm:

BShow - Trị số điều khiển. Nếu dương, để hiện. Số 0 để dấu.

16. Declare Function SwapMouseButton Lib "user32" Alias "SwapMouseButton" (ByVal bSwap As Long) As Long

Công dụng: Kiểm tra sự tráo đổi chức năng 2 phím chuột.

Trị trả về: Integer - True nếu đã bị hoán đổi. False nếu vẫn chưa bị đổi.

Tham số kèm:

BSwap - Đặt là True để đổi. Đặt là False để không đổi.

2. Các mã quét và phím ảo:

Các phím trên bàn phím tạo ra các mã quét (scan code) tuỳ thuộc vào vị trí của chúng. Khi ta tác động vào phím, bàn phím sẽ gửi mã quét của phím đó cho máy tính. Do có nhiều loại bàn phím khác nhau nên Windows đưa ra khái niệm phím ảo (virtual key) là phím không phụ thuộc vào vị trí của từng loại bàn phím, giúp cho việc thống nhất xử lý. Ví như bạn nhấn phím Enter, ở bất cứ bàn phím nào, ở vị trí nào vẫn là phím đó, chứ không liên quan đến vị trí như mã quét.

Do đó ta có thể hiểu vấn đề này như sau:

Khi một phím được nhấn, Windows sẽ nhận được tín hiệu và chuyển đổi thông tin mã quét của bàn phím thành một mã phím ảo. Phần xử lý phím ảo sẽ chịu trách nhiệm chuyển tiếp hình ảnh của ký tự được nhấn, sau khi đã căn cứ vào trạng thái của phím Shift.

Các hàm xử lý bàn phím bao gồm:

17. Declare Function GetAsyncKeyState Lib "user32" Alias "GetAsyncKeyState" (ByVal vKey As Long) As Integer

Công dụng: Nhận tình trạng của phím ảo được khai báo.

Trị trả về: Integer - Bit 0 là 1 nếu phím đã được nhấn kể từ lần gọi hàm này gần đây nhất. Ngược lại là Zero. Bit 15 là 1 nếu phím đang nhấn xuống, hay zero nếu thả ra.

Tham số kèm:

VKey Mã của phím cần kiểm tra (Xem bảng mã key code các phím)

18. Declare Function GetKBCodePage Lib "user32" Alias "GetKBCodePage" () As Long

Công dụng: Nhận trang mã Windows cần chuyển đổi giữa các bộ ký tự.

Trị trả về: Integer - Là các trị số chỉ trang mã.

19. Declare Function GetKeyboardState Lib "user32" Alias "GetKeyboardState" (pbKeyState As Byte) As Long

Công dụng: Nhận trạng thái phím ảo của bàn phím.

Trị trả về: Đọc trạng thái ghi vào biến cần lưu.

Tham số kèm:

PbKeyState - Biến dạng byte (Là chuỗi có chiều dài có thể lưu trữ 256 ký tự) nhằm lưu giá trị để nhận tình trạng các phím. Giá trị này phải được hiểu là nhị phân, căn cứ vào bit 0 của các đoạn tương ứng với các phím Caplock, NumLock, Scroll Lock, nếu là 1 sẽ là đang bật, còn bit 7 các phím thường, nếu nhấn xuống là 1, giá trị 0 là thả ra.

20. Declare Function GetKeyboardType Lib "user32" Alias "GetKeyboardType" (ByVal nTypeFlag As Long) As Long

Công dụng: Nhận kiểu bàn phím đang dùng.

Trị trả về: Integer - Zero nếu có lỗi. Các số khác chỉ loại bàn phím đăng ký.

Nếu cờ bằng 0, trị trả về:

1 - Tương ứng bàn phím 83 phím

2 - Tương ứng 102 phím

3 - Tương ứng 84 phím

4 - Tương ứng 101 -102 phím (IBM)

5 - Tương ứng Nokia 1050

6 - Tương ứng Nokia 9140

7 - Tương ứng tiếng Nhật.

Nếu cờ bằng 1, trị trả về tuỳ thuộc hãng sản xuất.

Nếu cờ bằng 2, trị trả về:

1, 3, 5 - Tương ứng bàn phím 10 phím chức năng.

2 - Tương ứng bàn phím 12 hoặc 18 phím chức năng.

4 - Tương ứng bàn phím 12 phím chức năng.

6 - Tương ứng bàn phím 24 phím chức năng.

7 - Do hãng sản xuất quy định.

Tham số kèm:

NTypeFlag - Cờ xác định kiểu dữ liệu cần lấy.

21. Declare Function GetKeyNameText Lib "user32" Alias "GetKeyNameTextA" (ByVal lParam As Long, ByVal lpBuffer As String, ByVal nSize As Long) As Long

Công dụng: Nhận tên của phím.

Trị trả về: Chiều dài của tên phím nhận được trong lpBuffer

Tham số kèm:

LParam Là một số mà bit từ 0 - 5 đặt là 0. Bít 16 - 23 đặt mã quét của phím cần xác định tên, bít 24 - Bít mở rộng trên những bàn phím nâng cao. Bit 25 - Khi gán bằng 1, bỏ qua sự khác biệt giữa các phím trái, phải.

LpBuffer Chuỗi được gán trước để nhận tên phím. Nên có nSize + 1 bytes.

NSize Chiều dài tối đa của chuỗi.

22. Declare Function GetKeyState Lib "user32" Alias "GetKeyState" (ByVal nVirtKey As Long) As Integer

Công dụng: Nhận trạng thái phím ảo.

Trị trả về: Bit 0 là 1 khi các phím như Caplock, NumLock, ScrollLock là bật, nếu bit 0 là 0, các phím này tắt.

Bit 15 là 1, nếu phím đang bị nhấn, bit 15 là 0 nếu vừa nhả.

Tham số kèm:

NVirtKey - Mã phím ảo để kiểm tra

23. Declare Function MapVirtualKey Lib "user32" Alias "MapVirtualKeyA" (ByVal wCode As Long, ByVal wMapType As Long) As Long

Công dụng: Thực hiện những sự chuyển đổi mã quét và ký tự tuỳ thuộc vào kiểu ánh xạ cung cấp.

Trị trả về: Tuỳ thuộc vào kiểu ánh xạ yêu cầu, trị trả về theo bảng sau

WCode là	Giá trị của VMapType		
	0	1	2
Mã phím ảo	Mã quét tương ứng		
Mã quét		Mã phím ảo tương ứng	
Mã phím ảo			Trị ASCII tương ứng.

Tham số kèm:

WCode - Giá trị nguồn cần chuyển đổi.

WMapType - Điều khiển kiểu chuyển đổi.

24. Declare Function MapVirtualKeyEx Lib "user32" Alias "MapVirtualKeyExA" (ByVal uCode As Long, ByVal uMapType As Long, ByVal dwhkl As Long) As Long

Công dụng Hàm dịch mã phím ảo thành một mã quét hoặc giá trị ký tự, hoặc dịch một mã quét thành mã phím ảo. Chức năng dịch các mã dùng cho ngôn ngữ nhập và định dạng vị trí.
Tham số kèm:

Parameters

uCode

Mã phím ảo hoặc mã quét của một phím. Giá trị này phải tương ứng với uMapType.

uMapType

[in] Kiểu dữ liệu cần dịch. Giá trị này phải phù hợp với tham số uCode trên.

Các giá trị 0, 1, 2 như bảng trên.

Riêng giá trị bằng 3 chỉ sử dụng cho **Windows NT/2000 trở lên:** *uCode* là một mã quét và được dịch thành mã phím ảo để phân biệt giữa những phím bên tay trái và những phím bên tay phải. Nếu điều này không dịch, hàm trả về 0

dwhkl

[in] Định vị vị trí nhập để sử dụng cho việc dịch các mã đặc biệt. Tham số này có thể là bất kỳ giá trị định vị vị trí trước đó mà đã nhận được từ hàm **LoadKeyboardLayout**

Trị trả về

Hoặc là một ã quét, môt mã phím ảo hoặc một giá trị ký tự tuỳ thuộc vào *uCode* và *uMapType*. Nếu điều này không được dịch, hàm trả về zero.
Một ứng dụng có thể dùng **MapVirtualKeyEx** để dịch các mã quét thành các hằng mã phím ảo như VK_SHIFT, VK_CONTROL, và VK_MENU, và ngược lại. Quá trình dịch không do not phân

biệt giữa các phím trá hay phải của các phím SHIFT, CTRL, hoặc ALT. Còn **Windows NT/2000 trở lên** thì phân biệt, nó sẽ chuyển được thành các hằng mã phím ảo như:
VK_LSHIFT
VK_RSHIFT
VK_LCONTROL
VK_RCONTROL
VK_LMENU
VK_RMENU
Sự phân biệt này ảnh hưởng tới các hàm **GetKeyboardState**, **SetKeyboardState**, **GetAsyncKeyState**, **GetKeyState**, **MapVirtualKey**, và chính **MapVirtualKeyEx** . Xin xem bảng mã phím ảo Virtual-Key Codes.

25. Declare Function OemKeyScan Lib "user32" Alias "OemKeyScan" (ByVal wOemChar As Long) As Long

Công dụng: Nhận mã quét và trạng thái Shift đối với một ký tự ASCII trong bộ ký tự OEM.

Trị trả về: Từ thấp chứa mã quét. Từ cao đưa ra các cờ hiệu thông qua các bit. Bit 0 nếu bằng 1, phím Shift bị nhấn. Bit 1 là 1 cho biết Ctrl bị nhấn.

Tham số kèm:

WOemChar - Giá trị ASCII của ký tự cần chuyển đổi.

26. Declare Function SetKeyboardState Lib "user32" Alias "SetKeyboardState" (lppbKeyState As Byte) As Long

Công dụng: Đặt trạng thái phím ảo. Thường sử dụng để đặt trạng thái phím CaspLock, NumLock, ScrollLock.

Tham số kèm:

lppbKeyState - Biến con trỏ 256 byte chứa trạng thái bàn phím.

27. Declare Function ToAscii Lib "user32" Alias "ToAscii" (ByVal uVirtKey As Long, ByVal uScanCode As Long, lpbKeyState As Byte, lpwTransKey As Long, ByVal fuState As Long) As Long

Công dụng: Xác định trị ASCII của một phím ảo dựa trên các trạng thái phím Shift và phím điều khiển.

Trị trả về:

Nếu phím cần kiểm tra là phím chết, giá trị trả về là số âm. Trường hợp khác là một trong các giá trị sau:

0 - Phím ảo cần kiểm tra không được dịch trong trạng thái hiện hành của bàn phím.

1 - Một ký tự được copy vào buffer

2 - Hai ký tự được copy vào bufer, trường hợp này xảy ra khi một phím nhấn chết, và gõ một phím khác.

Tham số kèm:

UVirtKey - Phím ảo cần chuyển đổi.

UScanCode - Mã quét của phím hoặc phím cần chuyển đổi. Bit cao của giá trị được thiết lập nếu phím nhả.

LpbKeyState - Biến trỏ tới mảng 256 byte chỉ trạng thái bàn phím hiện tại. Mỗi byte trong mảng chứa trạng thái của một phím. Nếu bit cao của một byte được thiết lập là phím được nhấn xuống. Với bit thấp được thiết lập, tín hiệu được đảo là ON. Trường hợp này chỉ dùng cho CAPSLOCK. Còn SCROLLOCK và NUMLOCK bị bỏ qua.

LpwTransKey - Biến con trỏ chỉ tới buffer mà nhận giá trị kết quả sau khi dịch.

FuState - Cờ hiệu, trạng thái thực đơn được kích hoạt. Nếu là 1 là được kích hoạt, ngược lại 0.

28. Declare Function ToAsciiEx Lib "user32" Alias "ToAsciiEx" (ByVal uVirtKey As Long, ByVal uScanCode As Long, lpKeyState As Byte, lpChar As Integer, ByVal uFlags As Long, ByVal dwhkl As Long) As Long

Như trên riêng dwhkl dùng để định vị trí nhập, dùng cho chuyển mã.Tham số này nhận được bất cứ vị trí định vị nào trước đó đã được trả về từ hàm **LoadKeyboardLayout**

29. Declare Function VkKeyScan Lib "user32" Alias "VkKeyScanA" (ByVal cChar As Byte) As Integer

Công dụng: Dịch ký tự phím ảo tùy thuộc trạng thái phím shift hiện tại.

Trị trả về: Nếu hàm thành công, byte thấp của giá trị trả về chứa mã phím ảo, và byte cao chứa trạng thái Shift, căn cứ vào bảng sau

Bit	Nghĩa là
1	Bất cứ SHIFT key bị nhấn
2	Bất cứ CTRL key bị nhấn.
4	Bất cứ ALT key bị nhấn.
8	The Hankaku key bị nhấn
16	Dành riêng (Được định nghĩa bởi driver).
32	Dành riêng (Được định nghĩa bởi driver).

Nếu hàm không tìm thấy phím nào để dịch thì bỏ qua mã ký tựcả hai byte thấp và cao đều chứa –1.

30. Declare Function VkKeyScanEx Lib "user32" Alias "VkKeyScanExA" (ByVal ch As Byte, ByVal dwhkl As Long) As Integer

Như trên riêng dwhkl dùng để định vị trí nhập, dùng cho chuyển mã.Tham số này nhận được bất cứ vị trí định vị nào trước đó đã được trả về từ hàm **LoadKeyboardLayout**

3. Tiếp theo là các hàm điều khiển nhập liệu:

31. Declare Function GetInputState Lib "user32" Alias "GetInputState" () As Long

Công dụng: Nhận trạng thái nhập liệu, kiểm tra xem có tình huống chuột hoặc bàn phím đang chờ xử lý không.

Trị trả về: True - Nếu có trường hợp cần xử lý đang chờ.

32. Declare Function GetQueueStatus Lib "user32" Alias "GetQueueStatus" (ByVal fuFlags As Long) As Long

Công dụng: Nhận loại chỉ lệnh cần xử lý trong hàng đợi đợi ứng dụng.

Trị trả về: Từ cao là tập cờ 16 bit, khai báo các chỉ lệnh ở hàng đợi. Từ thấp cho biết các loại chỉ lệnh được thêm vào.

Tham số kèm:

FUFlags - Tập cờ yêu cầu cần kiểm tra chỉ lệnh là một từ thông qua các bit. Là một trong các hằng số sau:

QS_ALLEVENTS = Tất cả các mức (QS_INPUT hoặc QS_POSTMESSAGE hoặc QS_TIMER hoặc QS_PAINT hoặc QS_HOTKEY)

QS_ALLINPUT = Tấ cả phần nhập liệu (QS_SENDMESSAGE hoặc QS_PAINT hoặc QS_TIMER hoặc QS_POSTMESSAGE Hoặc QS_MOUSEBUTTON Hoặc QS_MOUSEMOVE Hoặc QS_HOTKEY Hoặc QS_KEY)

QS_HOTKEY - Phím nóng.

QS_INPUT - Phần nhập (QS_MOUSE Hoặc QS_KEY)

QS_KEY - Các chỉ lệnh phím.

QS_MOUSE = Các chỉ lệnh chuột (QS_MOUSEMOVE Hoặc QS_MOUSEBUTTON)

QS_MOUSEBUTTON - Chỉ lệnh liên quan đến nút chuột.

QS_MOUSEMOVE - Di chuyển bằng chuột

QS_PAINT - Chỉ lệnh vẽ.

QS_POSTMESSAGE - Chỉ lệnh được phát đi.

QS_SENDMESSAGE - Chỉ lệnh được chuyển từ ứng dụng khác.

QS_TIMER - Timer.

33. Declare Function GetQueuedCompletionStatus Lib "kernel32" Alias "GetQueuedCompletionStatus" (ByVal CompletionPort As Long, lpNumberOfBytesTransferred As Long, lpCompletionKey As Long, lpOverlapped As Long, ByVal dwMilliseconds As Long) As Long

Công dụng: Nhận trạng thái chỉ lệnh đã hoàn thành xếp hàng trong hàng đợi hệ thống

Trị trả về: @

Tham số kèm:

CompletionPort - Cán cổng hoàn thành. Để tạo một cổng hoàn thành sử dụng hàm **CreateIoCompletionPort**

LpNumberOfBytesTransferred - Biến con trỏ chỉ vào một biến nhận về số byte trao đổi trong tác vụ vào ra đã hoàn thành.

LpCompletionKey - Biến con trỏ chỉ vào biến nhận về phím đã hoàn thành tác vụ vào ra. Một phím hoàn thành là một giá trị nhận từ hàm **CreateIoCompletionPort**

LpOverlapped - Biến con trỏ chỉ tới biến ghi lại vị trí của địa chỉ cấu trúc **OVERLAPPED** được thiết lapaj khi tác vụ hoàn thành được bắt đầu.

DwMilliseconds - Số mili giây gọi để chờ cho tác vụ hoàn thành.

34. Declare Function IsDBCSLeadByte Lib "kernel32" Alias "IsDBCSLeadByte" (ByVal bTestChar As Byte) As LongDeclare

35. Function IsDBCSLeadByte Lib "kernel32" Alias "IsDBCSLeadByte" (ByVal TestChar As Byte) As Long

Công dụng: Kiểm tra xem có phải là ký tự đầu tiên trong bộ ký tự 2 byte.

Trị trả về: True nếu là byte đầu của ký tự thuộc bộ ký tự 2 byte.

Tham số kèm:

BTestChar - Ký tự cần kiểm tra.