

TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA CÔNG NGHỆ THÔNG TIN

---

**GIÁO TRÌNH**



BIÊN SOẠN  
Nguyễn Đăng Quang

THÁNG 09 - 2009

# Mục lục

## Chương 1: GIỚI THIỆU

I. CÁC MÔI TRƯỜNG LẬP TRÌNH.....	1
II. CÁC ĐỐI TƯỢNG ĐIỀU KHIỂN CHUẨN TRÊN WINDOWS .....	2
III. GIỚI THIỆU VISUAL BASIC .....	3
1. Khởi động - cửa sổ khởi động.....	3
2. Màn hình làm việc.....	5
IV. CÁC THAO TÁC CƠ BẢN VỚI ĐỐI TƯỢNG TRÊN FORM.....	10
1. Đưa một đối tượng lên form .....	10
2. Chọn đối tượng.....	11
3. Di chuyển .....	11
4. Hiệu chỉnh.....	11
5. Xóa.....	11
V. GHI NẠP MỘT VISUAL BASIC PROJECT .....	11
1. Thêm form mới vào chương trình.....	11
2. Xóa một form .....	12
3. Ghi Project .....	13
4. Nạp Project.....	14
5. Tạo Project mới.....	14
VI. MỘT CHƯƠNG TRÌNH VÍ DỤ .....	15
<b>Chương 2: Đối tượng và cách sử dụng đối tượng</b>	
I. ĐỐI TƯỢNG .....	17
1. Khái niệm.....	17
2. Các đặc điểm.....	17
3. Truy xuất.....	18
4. Các thuộc tính chung.....	18
5. Các sự kiện chung .....	19

II. ĐỐI TƯỢNG FORM.....	20
1. Thuộc tính .....	20
2. Phương thức .....	20
3. Xử lý sự kiện .....	20
III. ĐỐI TƯỢNG LABEL.....	22
1. Thuộc tính .....	22
2. Xử lý sự kiện .....	22
IV. ĐỐI TƯỢNG TEXTBOX.....	22
1. Thuộc tính .....	23
2. Xử lý sự kiện .....	23
V. ĐỐI TƯỢNG COMMAND BUTTON .....	23
1. Thuộc tính .....	23
2. Xử lý sự kiện .....	23
VI. FOCUS VÀ THỨ TỰ TAB .....	23
1. Focus .....	23
2. Thứ tự TAB .....	24
3. Phím nóng .....	24
4. Ví dụ .....	24
<b>Chương 3: Kiểu dữ liệu – Hằng – Biến</b>	
I. BIÊN .....	27
1. Định nghĩa .....	27
2. Khai báo .....	27
3. Quy tắc đặt tên biến.....	27
4. Truy xuất biến .....	27
5. Phạm vi sử dụng biến .....	28
6. Biến tĩnh .....	29
II. KIỂU DỮ LIỆU.....	30
III. HẰNG .....	30
IV. TOÁN TỬ .....	31

V. MỘT SỐ HÀM CHUẨN .....	31
1. Hàm đại số .....	31
2. Hàm thời gian.....	31
3. Hàm chuyển đổi .....	32
4. Hàm kiểm tra kiểu dữ liệu.....	32
VI. HỘP THÔNG BÁO .....	32
<b>Chương 4: Các cấu trúc điều khiển</b>	
I. LỆNH ĐIỀU KIỆN IF .....	35
II. LỆNH CHỌN LỰA CASE .....	35
III. LỆNH LẶP FOR... NEXT .....	36
IV. LỆNH LẶP DO LOOP .....	37
V. CHƯƠNG TRÌNH CON.....	38
1. Chương trình con Sub .....	39
2. Hàm .....	39
3. Khai báo .....	40
<b>Chương 5: Mảng – Chuỗi – Collection</b>	
I. MẢNG .....	41
1. Định nghĩa.....	41
2. Khai báo .....	41
3. Mảng đối tượng điều khiển .....	42
4. Ví dụ.....	43
5. Mảng động và mảng tĩnh .....	44
6. Một số vấn đề khác .....	46
II. CHUỖI KÝ TỰ .....	48
1. Khai báo .....	48
2. Các hàm xử lý chuỗi .....	48
III. COLLECTION.....	49
1. Giới thiệu .....	49
2. Thao tác trên Collection.....	50

3. Ví dụ khác .....	53
---------------------	----

## **Chương 6: TextBox – ListBox – ComboBox**

I. TEXTBOX .....	55
1. Các thuộc tính bổ sung .....	55
2. Sự kiện .....	55
3. Ví dụ .....	55
II. LISTBOX .....	57
1. Các thuộc tính .....	58
2. Các phương thức .....	59
3. Sự kiện .....	60
4. Một số ví dụ .....	62
III. COMBOBOX .....	65
IV. DRIVELISTBOX, DIRLISTBOX, FILELISTBOX	
1. DriveListBox .....	66
2. DirListBox .....	66
3. FileListBox .....	66

## **Chương 7: Scrollbar – Image – Timer**

I. SCROLLBAR .....	69
1. Các thuộc tính .....	69
2. Sự kiện .....	69
3. Ví dụ .....	69
II. IMAGE .....	70
III. TIMER .....	71
1. Thuộc tính .....	71
2. Sự kiện .....	71
3. Ví dụ .....	71

## **Chương 8: Truy xuất dữ liệu**

I. TRUY XUẤT DỮ LIỆU BẰNG ĐỐI TƯỢNG ĐK CÓ KẾT NỐI CSDL.....	73
1. DataControl.....	73
2. Các thuộc tính .....	73
3. Các đối tượng điều khiển có kết nối cơ sở dữ liệu.....	74
4. Sử dụng Databound Listbox và Combobox.....	75
5. Sử dụng Databound Grid Control (DBGrid) .....	78
II. TRUY XUẤT DỮ LIỆU THÔNG QUA DATA ACCESS OBJECT .....	84
1. Các thao tác cơ bản .....	84
2. Các thuộc tính của Recordset.....	85
3. Các thao tác trên Recordset.....	85
<b>Chương 9: PictureBox –Xử lý mouse</b>	
I. PICTUREBOX .....	91
1. Thuộc tính .....	91
2. Các phương thức đồ họa .....	92
3. Các thuộc tính qui định đơn vị vẽ .....	96
4. Các lệnh ghi nạp ảnh.....	97
II. XỬ LÝ MOUSE.....	98
<b>Chương 10: Menu – Common Dialog</b>	
I. Menu .....	103
1. Định nghĩa menu .....	103
2. Viết lệnh .....	104
II. COMMON DIALOG .....	106
1. Hộp thoại Open, Save .....	106
2. Hộp thoại chọn màu .....	108
<b>Chương 11: Kiểu bản ghi – Tập tin</b>	
I. KIỂU BẢN GHI .....	109
1. Định nghĩa.....	109
2. Khai báo .....	109
II. TẬP TIN.....	109

1. Định nghĩa .....	109
2. Phân loại .....	109
3. Thủ tục truy xuất dữ liệu trên tập tin.....	110
4. Các lệnh trên tập tin truy xuất ngẫu nhiên .....	110
III. CÁC LỆNH TRÊN TẬP TIN VĂN BẢN .....	113

## **Chương 12: Microsoft Windows Common Controls:**

### **ImageList – Listview – ImageCombo**

I. IMAGELIST .....	117
II. LISTVIEW .....	117
1. Các thuộc tính.....	118
2. Các thuộc tính của đối tượng ListItem.....	120
3. Phương thức .....	120
4. Sự kiện.....	123
III. IMAGECOMBO .....	123
1. Các thuộc tính.....	123
2. Các thuộc tính của đối tượng ComboItem .....	124
3. Các phương thức .....	125

## **Chương 13: Microsoft Windows Common Controls:**

### **Toolbar - Statusbar - Dtpicker**

I. TOOLBAR.....	127
1. Sử dụng Toolbar .....	127
2. Định nghĩa Toolbar .....	129
3. Định nghĩa nút Toolbar lúc chạy chương trình .....	130
II. STATUSBAR.....	131
1. Sử dụng.....	131
2. Viết lệnh cho StatusBar.....	133
III. DTPICKER .....	135
1. Thuộc tính .....	135
2. Sự kiện.....	137

## **Chương 14: Microsoft Windows Common Controls:**

### **Treeview - Updown – Slider – Progressbar**

I. TREE VIEW .....	139
1. Các thuộc tính .....	139
2. Các thuộc tính của đối tượng Node.....	140
3. Phương thức .....	141
4. Sự kiện .....	143
II. UPDOWN .....	144
1. Các thuộc tính .....	144
2. Sự kiện .....	144
III. SLIDER.....	145
1. Thuộc tính .....	145
2. Phương thức .....	146
3. Sự kiện .....	146
IV. PROGRESSBAR.....	147

### **Chương 15: RichTextBox – Form MDI**

I. RICHTEXTBOX .....	149
1. Các thuộc tính .....	149
2. Các phương thức .....	150
II. SỬ DỤNG RICHTEXTBOX.....	151
1. Chọn dáng vẽ Font chữ bằng nút lệnh trên Toolbar .....	151
2. Chọn Font chữ bằng lệnh trên menu và hộp thoại Font.....	152
3. Sự kiện SelChange .....	152
III. SỬ DỤNG CLIPBOARD .....	153
1. Sao chép vào Clipboard .....	153
2. Chép dữ liệu từ Clipboard vào văn bản .....	153
3. Cắt dữ liệu vào Clipboard .....	153
IV. SỬ DỤNG COMBOBOX CHỌN FONT VÀ CỖ CHỮ TRÊN TOOLBAR.....	154
V. MDI FORM.....	154



1. Đặc điểm.....	154
2. Form con MDI.....	155
3. Các thuộc tính và phương thức bổ sung so với form thường.....	155
4. Nạp cửa sổ con trong form MDI.....	156
5. Tạo ứng dụng MDI bằng Form Wizard .....	156

## **Chương 16: Lập trình Drag-and-Drop**

I. TỔNG QUAN.....	159
1. Kéo thả tự động .....	159
2. Kéo thả điều khiển bằng chương trình .....	160
II. MỘT CHƯƠNG TRÌNH VÍ DỤ.....	161
1. Khởi tạo hoạt động kéo-thả .....	161
2. Chuẩn bị cho thao tác thả trên đối tượng nguồn .....	162
3. Thả trên đối tượng đích.....	164
4. Nạp dữ liệu theo yêu cầu.....	165
5. Kéo thả File .....	165

# Chương 1

## Giới thiệu

### I. CÁC MÔI TRƯỜNG LẬP TRÌNH

Lập trình: Viết chương trình

Chương trình phải được chạy trên nền một Hệ điều hành. Trên máy PC có hai loại môi trường hệ điều hành : đó là môi trường DOS và môi trường Windows

Đặc điểm của môi trường DOS

- Hệ điều hành đơn chương : mỗi lúc chỉ có 1 chương trình làm việc. Lệnh trong chương trình sẽ qui định hoạt động kế tiếp mà người dùng sẽ tác động vào chương trình.
- Về mặt giao diện: Mỗi lúc chỉ có một chương trình hoạt động. Khi hoạt động giao diện của chương trình sẽ chiếm toàn bộ màn hình. Chỉ khi chương trình này kết thúc thì chương trình khác mới có thể hoạt động được.
- Về nguyên tắc lập trình: Lập trình thủ tục (Procedural Programming)
- Các công cụ lập trình trên DOS thông dụng: BASIC, TURBO PASCAL, TURBO C...

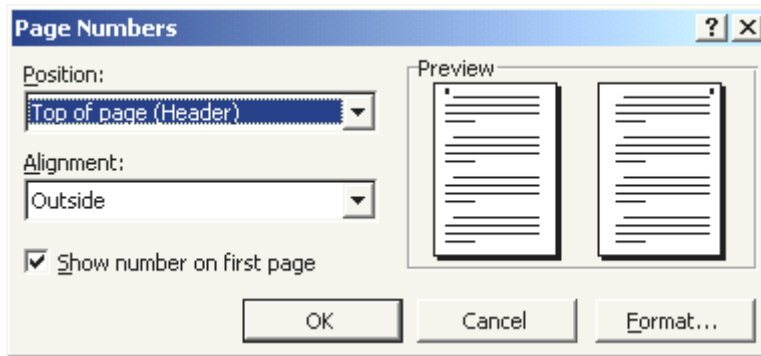
Đặc điểm của môi trường Windows:

- Hệ điều hành đa chương: mỗi lúc có thể có nhiều chương trình hoạt động đồng thời. Mỗi chương trình sẽ không biết trước hoạt động kế tiếp mà người dùng sẽ tác động vào chương trình
- Giao diện đồ họa (GUI-Graphics User Interface): Mỗi chương trình khi hoạt động được trình bày trong 1 cửa sổ
- Các chương trình Windows hoạt động theo nguyên tắc nhận và xử lý thông điệp (Message) đó là các tác động từ người dùng như các sự kiện bấm phím từ bàn phím, sự kiện từ mouse (di chuyển, bấm phím, nhả phím...) . Các tác động này sẽ được chuyển thành các thông điệp chứa trong hàng đợi (Message Queue) của hệ thống. Windows sẽ lần lượt xem xét các thông điệp và chuyển chúng đến các ứng dụng tương ứng. Chương trình đang hoạt động nhận thông điệp sẽ phản ứng theo cách của nó tùy theo ý nghĩa của từng loại thông điệp. Ví dụ: Sự kiện bấm phím trái chuột trên nút Minimize sẽ làm cho cửa sổ phóng lớn, Sự kiện nhấp đúp phím trái chuột trên thanh tiêu đề sẽ làm cho cửa sổ phóng lớn (Maximize) hoặc hoàn nguyên (Restore)...
- Các công cụ lập trình thông dụng trên Windows:  
BPW (Borland Pascal for Windows), BCW (Borland C for Windows),  
Delphi, Visual C++, Visual Basic.

## II. CÁC ĐỐI TƯỢNG ĐIỀU KHIỂN CHUẨN TRÊN WINDOWS

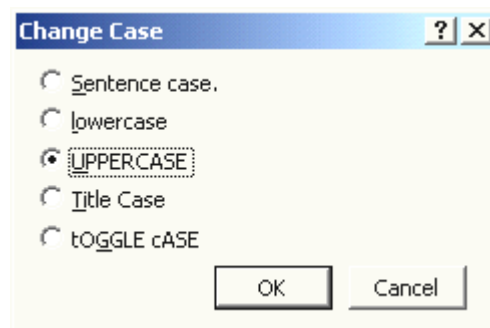
Chương trình trên windows sử dụng giao diện đồ họa để giao tiếp với người sử dụng: Trên mỗi cửa sổ của 1 chương trình Windows sử dụng một số đối tượng điều khiển để người dùng ra lệnh. Có nhiều loại đối tượng với các chức năng khác nhau. Để có thể viết chương trình trên Windows, cần làm quen với các đối tượng chuẩn.

1. Command Button (Nút lệnh): được sử dụng để ra lệnh. Trên các hộp thoại, thường thấy các nút lệnh như OK để chấp nhận hoặc Cancel để hủy một yêu cầu.
2. Checkbox: được sử dụng để chọn hoặc không chọn một yếu tố nào đó



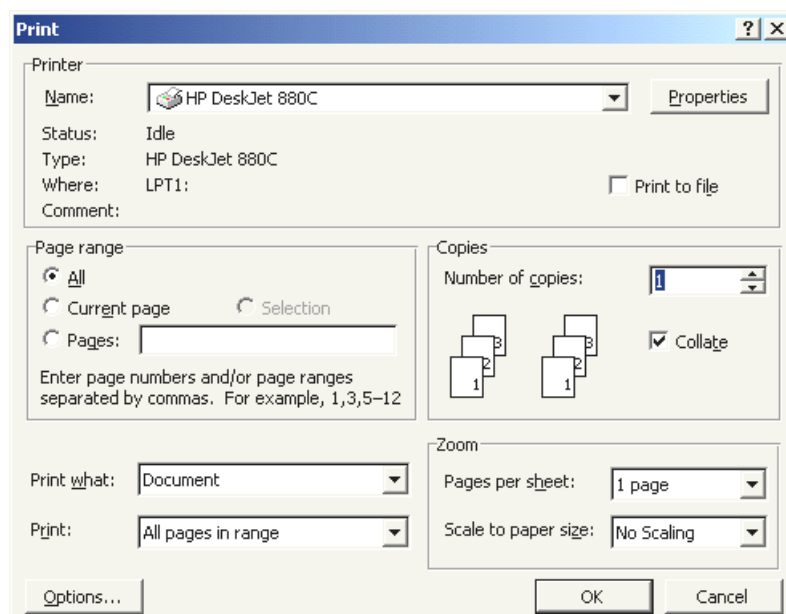
Hình 1.1: CheckBox chọn chế độ đánh số trang cho trang đầu tiên trong chức năng đánh số trang của ứng dụng WORD

3. Option Button (Nút chọn): Thường hoạt động theo nhóm được sử dụng để chọn một trong nhiều yếu tố



Hình 1.2: Chọn cách chuyển dạng chữ trong WORD

4. Textbox: Hộp nhập dữ liệu cho chương trình, có thể nhập 1 dòng hay nhiều dòng.
5. Label (Nhãn): đối tượng điều khiển được sử dụng trình bày một nội dung.
6. List box: Hộp danh sách được sử dụng trình bày một danh sách giá trị .
7. Combobox: Hộp danh sách hoạt động giống ListBox và TextBox, người dùng có thể nhập giá trị hoặc chọn một giá trị trong một danh sách cho trước.

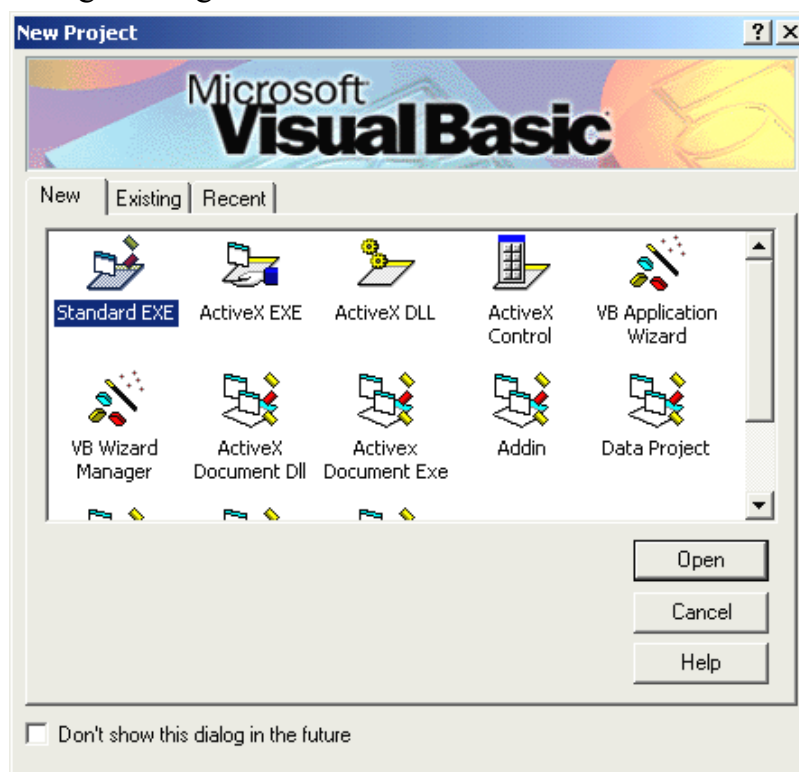


Hình 1.3: Hộp thoại Print với các đối tượng TextBox, Label, ComboBox

### III. GIỚI THIỆU VISUAL BASIC

#### 1. Khởi động - Cửa sổ khởi động

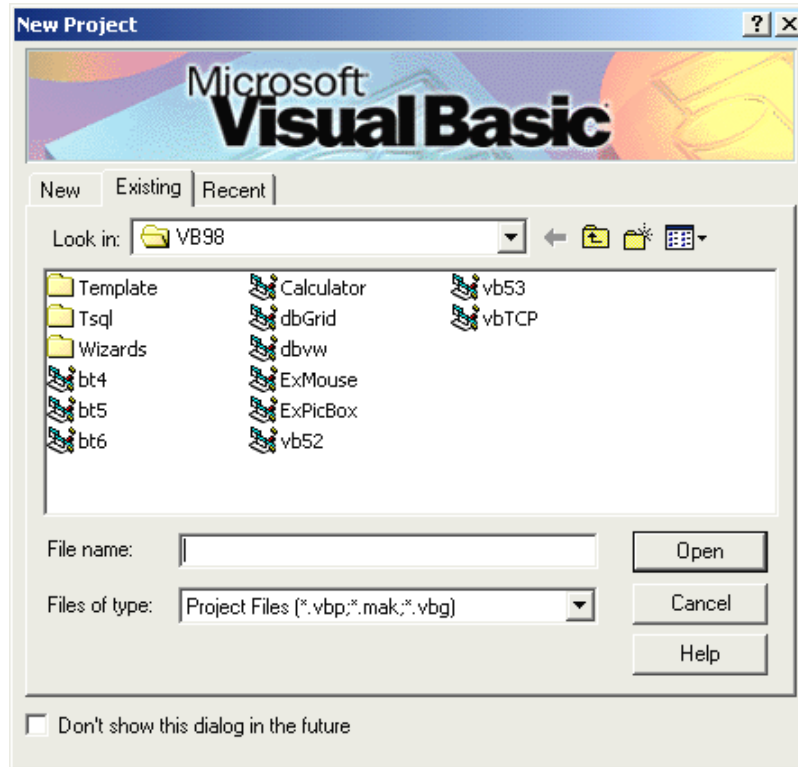
Màn hình khởi động có dạng



Hình 1.4: Cửa sổ khởi động-Thẻ New

**New:** Sử dụng thẻ này để tạo ứng dụng mới, thường chọn biểu tượng đầu tiên (Standard EXE) cho các ứng dụng bình thường chạy trên Windows.

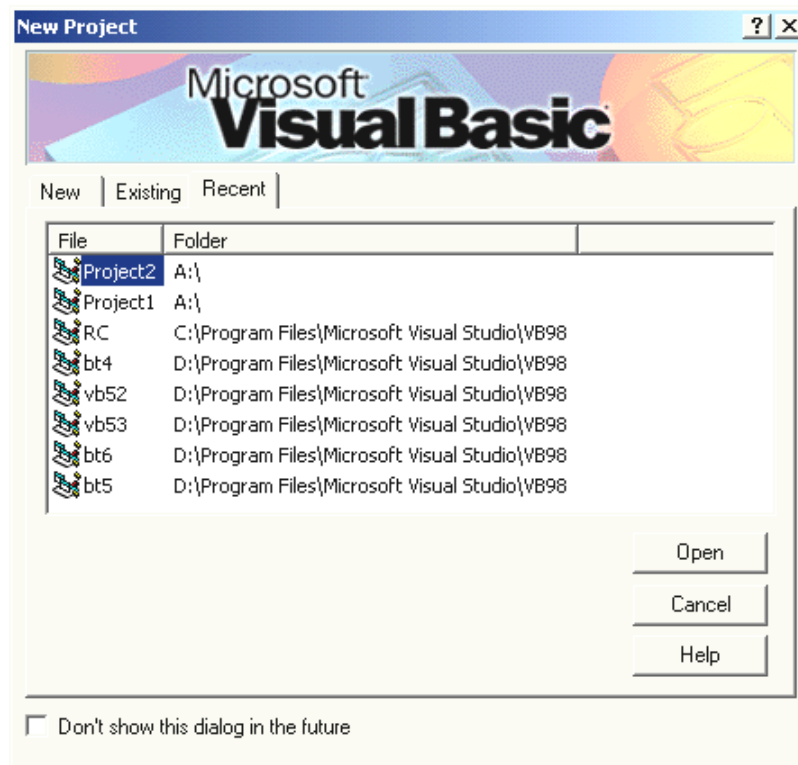
**Existing:** Sử dụng thẻ này để mở một ứng dụng đang có trên đĩa.



Hình 1.5: Cửa sổ khởi động-Thẻ Existing

Đề ý là ứng dụng viết trên VB được gọi là project. Tập tin này có phần mở rộng VBP (Visual Basic Project) , VBG (Visual Basic Group) hoặc MAK - Phần mở rộng loại này chỉ được sử dụng cho các project viết trên VB3.0.

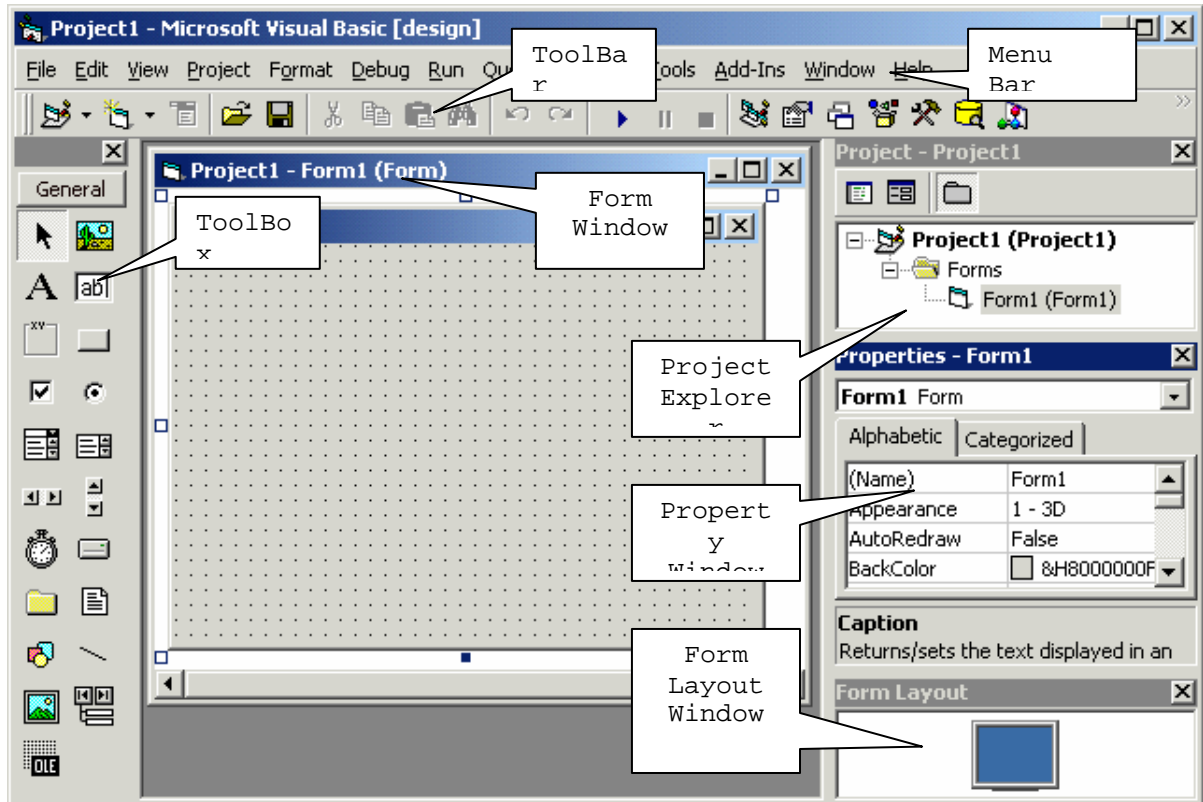
**Recent:** Sử dụng thẻ này để nạp nhanh ứng dụng đã làm việc trước đó



Hình 1.6: Cửa sổ khởi động-Thẻ Recent

## 2. Màn hình làm việc

Màn hình Visual Basic có dạng



Hình 1.7: Màn hình làm việc Visual Basic

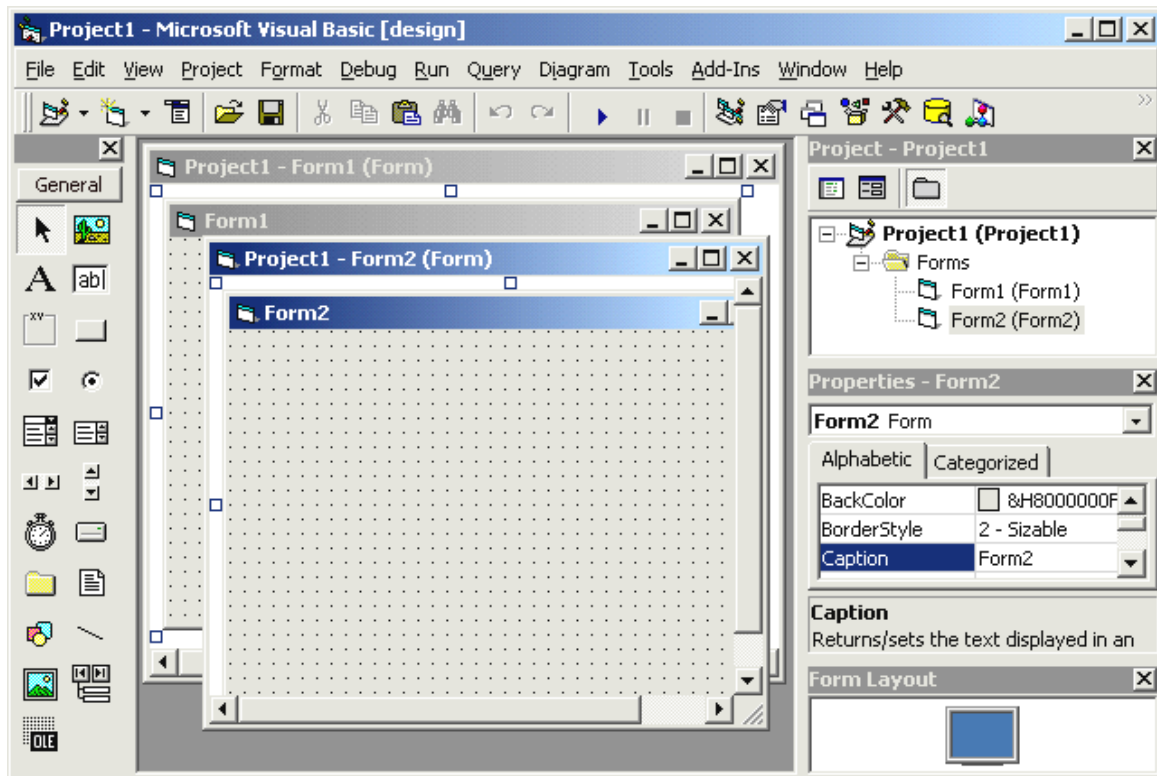
- Thanh menu (Menu Bar) : Menu chính của chương trình. Cùng với các menu kéo xuống (Pulldown) - hệ thống menu trình bày tất cả các chức năng của màn hình VB.
- Thanh công cụ (Toolbar): Giống các ứng dụng khác chạy trên Windows, thanh công cụ trình bày tất cả các chức năng thường sử dụng trong màn hình VB.
- Hộp công cụ (Toolbox): Chứa các đối tượng điều khiển được sử dụng trong thiết kế giao diện của chương trình. Mỗi một biểu tượng trên Toolbox đại diện cho một đối tượng muốn sử dụng trong giao diện của chương trình. Khi di chuyển mouse trên các biểu tượng, lời nhắc chức năng của nút sẽ tự động xuất hiện. Đối tượng được chọn bằng cách click vào hình ảnh biểu tượng trên Toolbox. Đối tượng nào được chọn thì hình ảnh của nó sẽ được vẽ lờm xuống.

ToolBox được đóng lại bằng nút close trên thanh tiêu đề. Làm toolbox xuất hiện trở lại bằng cách chọn View/Toolbox trên menu hoặc bấm nút



Nút Toolbox





Hình 1.8: Chương trình có 2 form

- Cửa sổ Form: Quản lý các cửa sổ được sử dụng trong chương trình. Mỗi cửa sổ trong chương trình gọi là form. Chương trình có bao nhiêu form sẽ có bấy nhiêu cửa sổ form. Hình trên trình bày một ứng dụng có 2 form
- Cửa sổ Project (Project Explorer): Giúp người lập trình dễ dàng theo dõi và quản lý các tập tin trong chương trình đang viết. Mỗi chương trình VB có thể bao gồm nhiều loại tập tin đại diện cho các thành phần sử dụng trong chương trình.

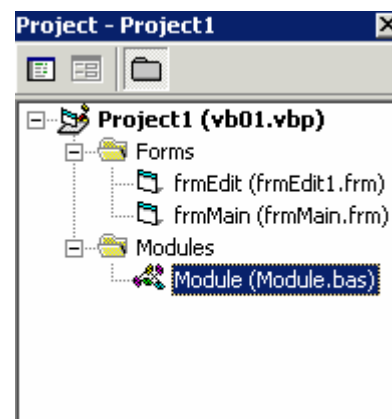
Các loại tập tin trong một chương trình VB có thể gồm:

\*.FRM: Tập tin form. Chương trình có bao nhiêu form sẽ có bấy nhiêu tập tin FRM

\*.BAS: Tập tin Module chứa các khai báo chung sử dụng trong chương trình

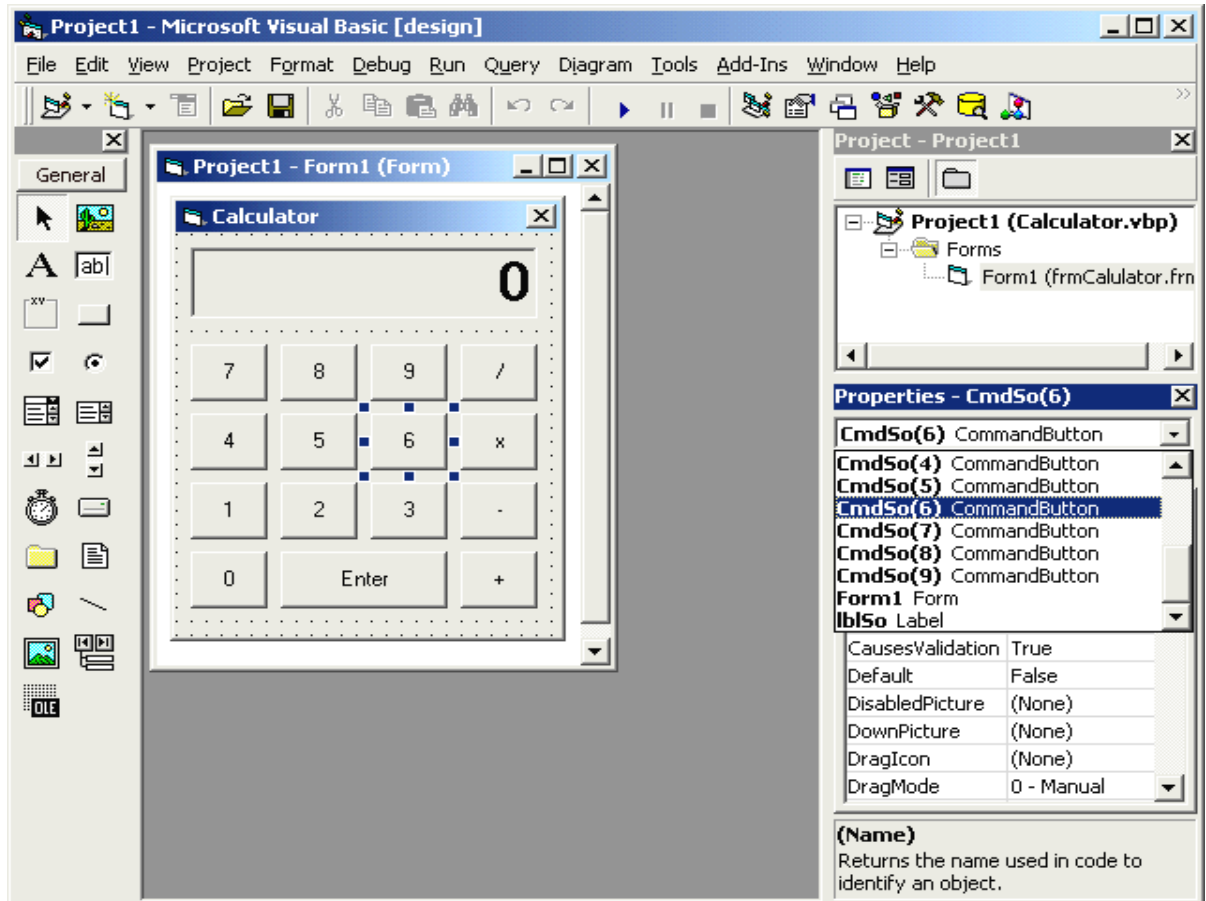
Hình trên là cửa sổ Project của một chương trình có 2 form và 1 module

Project Explorer được đóng lại bằng nút close trên thanh tiêu đề của cửa sổ .  
 Làm xuất hiện trở lại bằng cách chọn View/Project Explorer trên menu hoặc  
 bấm tổ hợp phím CTRL+R hoặc bấm nút Project Explorer trên thanh công cụ



- Cửa sổ thuộc tính (Properties Window): Mỗi đối tượng điều khiển trong chương trình có nhiều đặc điểm để mô tả tính chất của đối tượng như vị trí trên form, màu chữ ... Các đặc điểm thường sử dụng được mô tả trong cửa sổ thuộc tính. Thông qua cửa sổ này, người lập trình sẽ điều chỉnh các thuộc tính của đối tượng theo ý muốn trong quá trình thiết kế giao diện cho chương trình.

Các thành phần của cửa sổ thuộc tính:



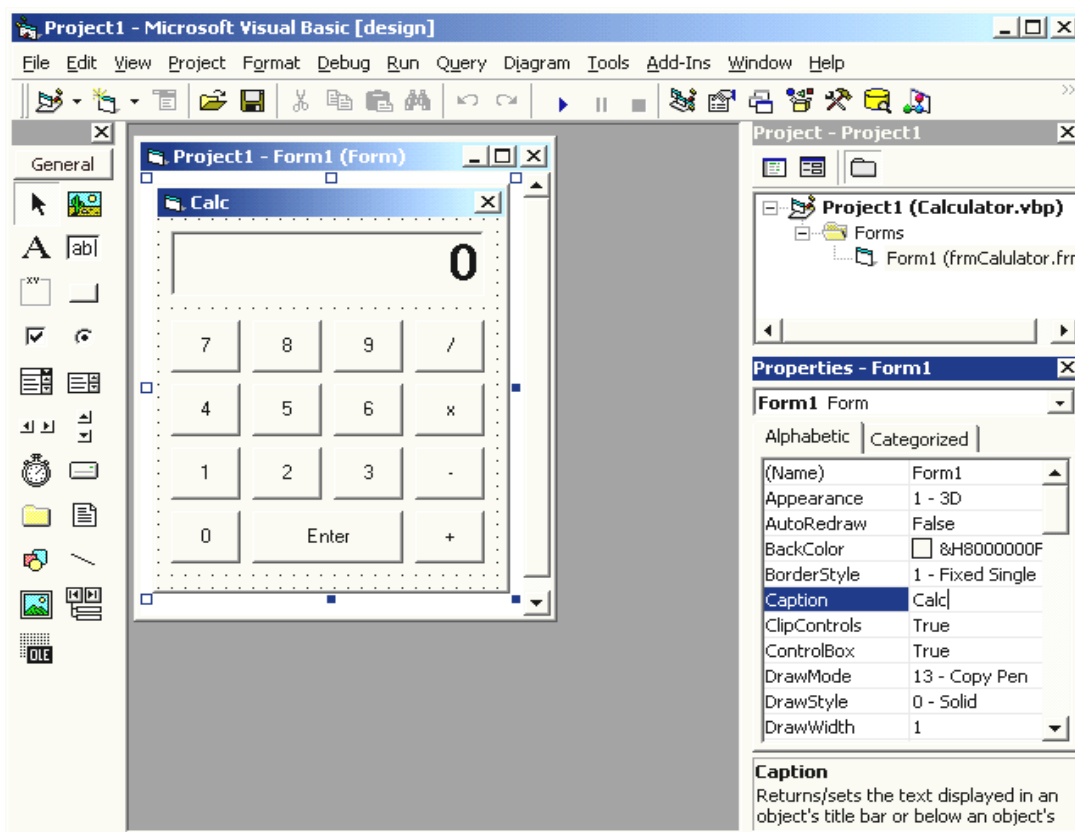
Hình 1.9: Cửa sổ thuộc tính

- Hộp chọn đối tượng: ComboBox phía trên chứa danh sách các đối tượng trên form đang thiết kế. Người lập trình có thể click trên form để chọn đối tượng cần điều chỉnh thuộc tính hoặc click chọn tên đối tượng trong danh sách này.
- Thẻ Alphabetic trình bày các thuộc tính theo thứ tự a, b, c của tên thuộc tính...
- Thẻ Categorize trình bày các thuộc tính theo nhóm chức năng

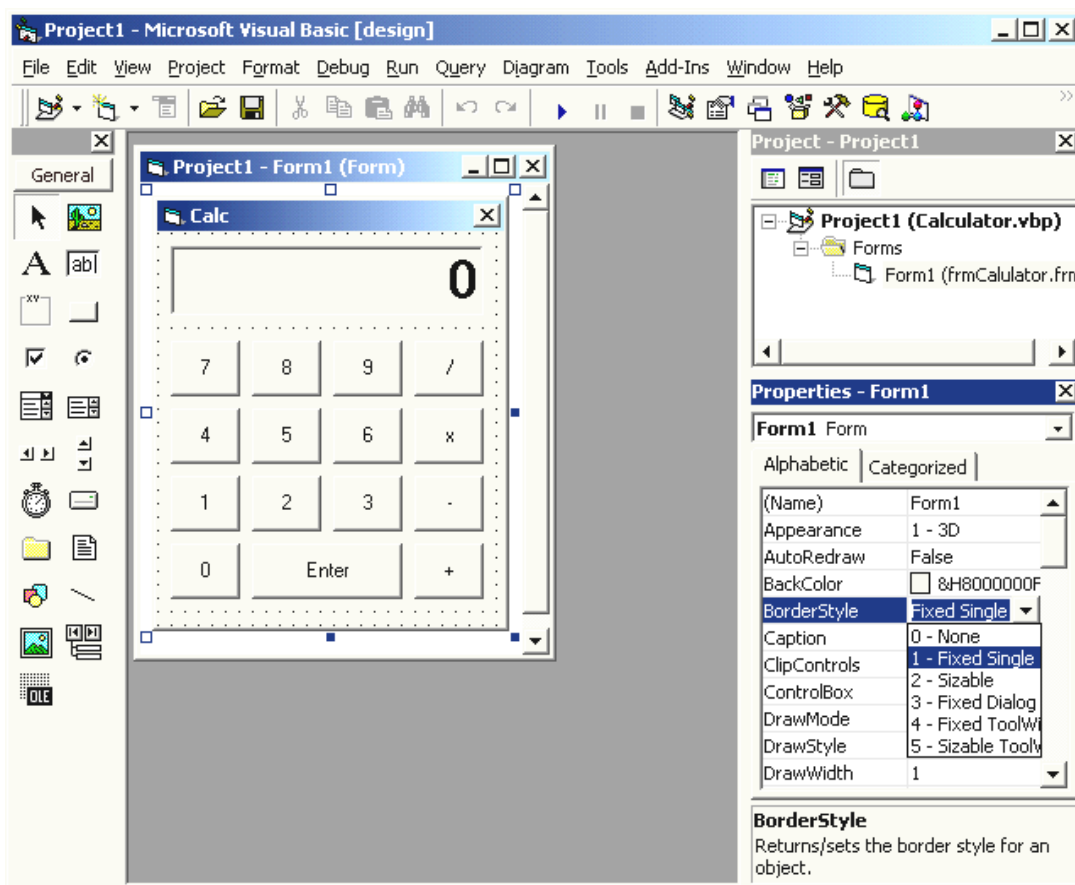
Trong lúc thiết kế, thường xem thuộc tính theo thứ tự alphabetic

Các thuộc tính đối tượng được trình bày thành 2 cột : cột bên trái là tên thuộc tính, cột bên phải là giá trị của thuộc tính. Cách điều chỉnh giá trị thuộc tính phụ thuộc vào thuộc tính cần điều chỉnh - có loại thuộc tính được điều chỉnh giá trị bằng cách nhập giá trị mới tại cột giá trị, có loại thuộc tính chỉ có thể điều chỉnh giá trị bằng cách chọn 1 trong danh sách giá trị đã được qui định trước





Hình 1.10: Nhập giá trị cho thuộc tính tiêu đề của form.



Hình 1.11: Chọn thuộc tính cho viền của form, danh sách trị có sẵn

Properties Window được đóng lại bằng nút close trên thanh tiêu đề. Làm xuất hiện trở lại bằng cách chọn View/Properties Window trên menu hoặc bấm phím F4 hoặc bấm nút Properties Window trên menu

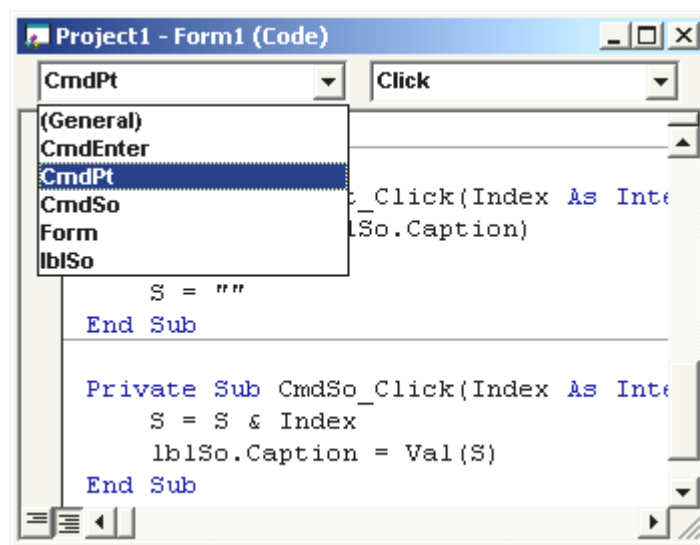


- Cửa sổ Form Layout: Dùng để điều chỉnh vị trí form khi xuất hiện lúc chạy chương trình. Trỏ chuột vào hình chữ nhật vẽ bên trong màn hình của cửa sổ và di chuyển để điều chỉnh vị trí.

Thông thường vị trí của các form khi chạy sẽ được thực hiện bằng lệnh trong chương trình. Để thuận tiện cho việc thao tác trên cửa sổ thuộc tính thường đóng cửa sổ này trong lúc thiết kế. Muốn làm xuất hiện cửa sổ này, bấm nút Form Layout trên thanh công cụ

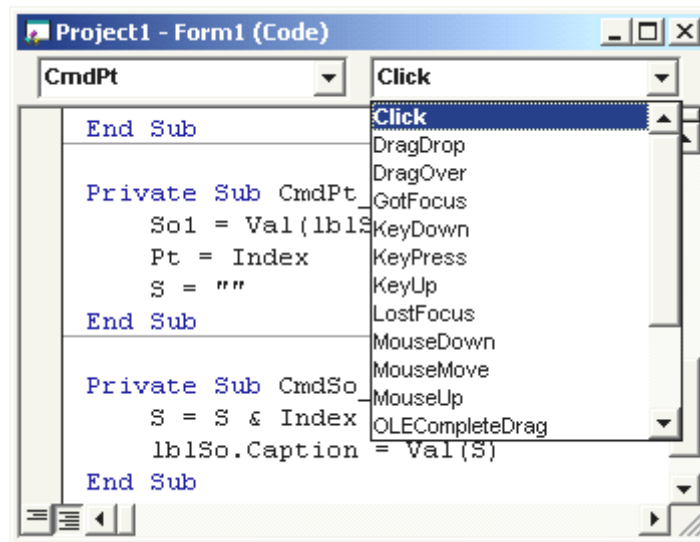


- Cửa sổ lệnh: Cửa sổ dùng để viết lệnh cho các đối tượng trên 1 form. Mỗi Form có một cửa sổ lệnh, cửa sổ lệnh chỉ xuất hiện khi nhấp đúp lên đối tượng muốn viết lệnh. Cửa sổ lệnh gồm các thành phần sau:
  - Hộp chọn đối tượng (Combo box phía trên bên trái) - Click để chọn đối tượng muốn viết lệnh



Hình 1.12: Chọn đối tượng viết lệnh

- Hộp chọn loại sự kiện (ComboBox phía trên bên phải) - Click để chọn sự kiện muốn viết lệnh.



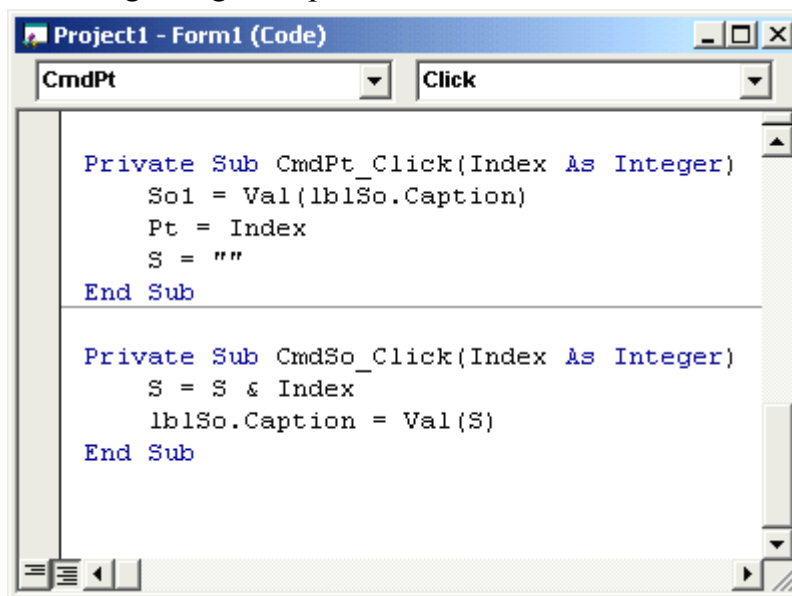
Hình 1.13: Chọn sự kiện viết lệnh

Khi một sự kiện được chọn, dòng khai báo của thủ tục tương ứng xuất hiện trong cửa sổ phía dưới.

Ví dụ: Chọn đối tượng nút bấm CmdPt, sự kiện Click. Dòng khai báo thủ tục có dạng  
*Private sub CmdPt\_Click(Index As Integer)*

*End sub*

Phần lệnh bên trong do người lập trình viết



Hình 1.14: Viết lệnh bên trong các khai báo thủ tục

#### IV. CÁC THAO TÁC CƠ BẢN VỚI ĐỐI TƯỢNG TRÊN FORM

##### 1. Đưa một đối tượng lên form

- Nhấp đúp tại nút đối tượng trên Toolbox, đối tượng sẽ xuất hiện ngay giữa form.

Hoặc

- Click đối tượng trên Toolbox, con trỏ chuyển thành dạng + trên form,
- Click tại vị trí cần đặt đối tượng trên form,
- Kéo lê để định kích thước đối tượng trên form,
- Nhả .

Lưu ý: Trong khi kéo để qui định kích thước đối tượng, có thể quan sát kích thước trên Toolbar hoặc dừng lại đủ lâu trên form, ô kích thước sẽ xuất hiện.

## 2. Chọn đối tượng trên form


*Chọn một đối tượng*

- Click tại đối tượng cần chọn

*Chọn nhiều đối tượng cùng lúc*

- Bấm Shift và Click để chọn nhiều đối tượng

Hoặc

- Sử dụng biểu tượng  chọn trên Toolbox để xác định vùng hình chữ nhật bao quanh các đối tượng cần chọn

## 3. Di chuyển

- Chọn một hoặc nhiều đối tượng cần di chuyển
- Kéo đến vị trí mới
- Nhả

## 4. Hiệu chỉnh kích thước một đối tượng

- Chọn đối tượng
- Trỏ mouse vào 1 trong 8 nút điều khiển quanh đối tượng chọn, kéo để điều chỉnh kích thước

## 5. Xoá

- Chọn một hoặc nhiều đối tượng muốn xoá
- Bấm DEL

## V. GHI NẠP MỘT VB PROJECT

Phần này trình bày các thao tác thường sử dụng đối với màn hình làm việc VB để quản lý các form, module... trong một Project. Đó là cách ghi một project sau khi thiết kế hoặc nạp một Project có sẵn trên đĩa.

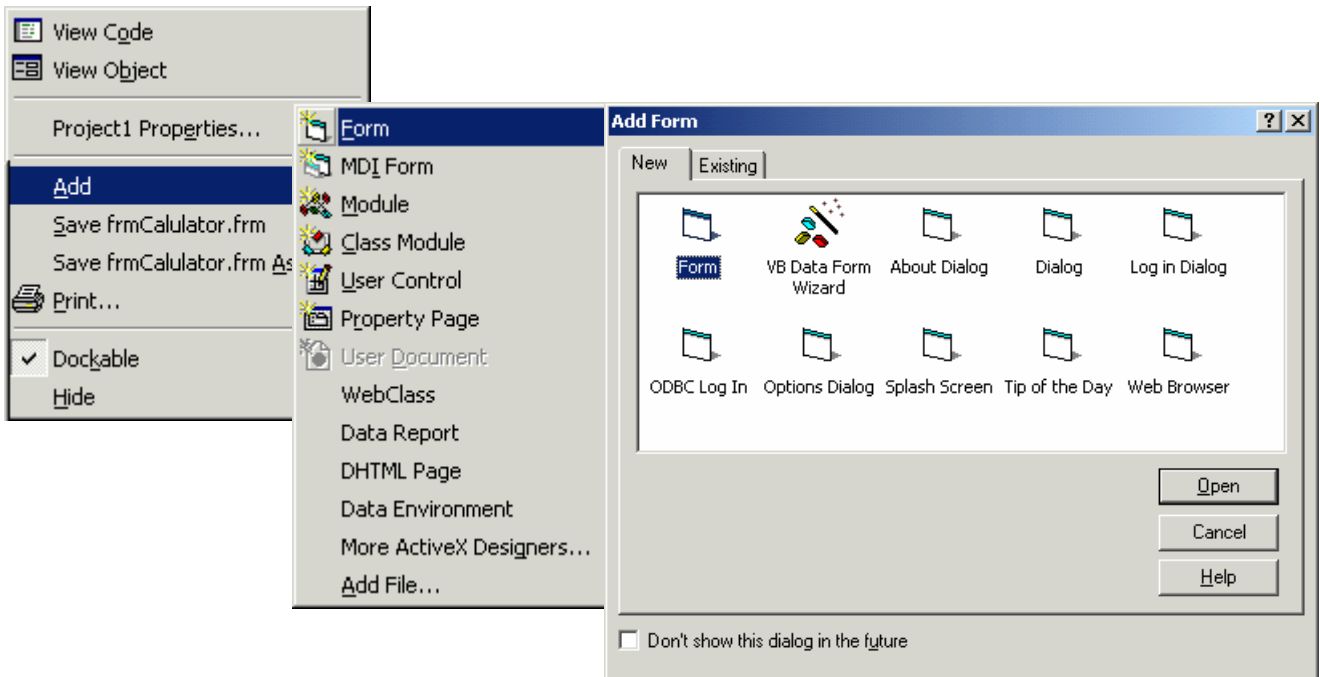
### 1. Thêm một form mới vào chương trình

*Trường hợp thêm form mới*

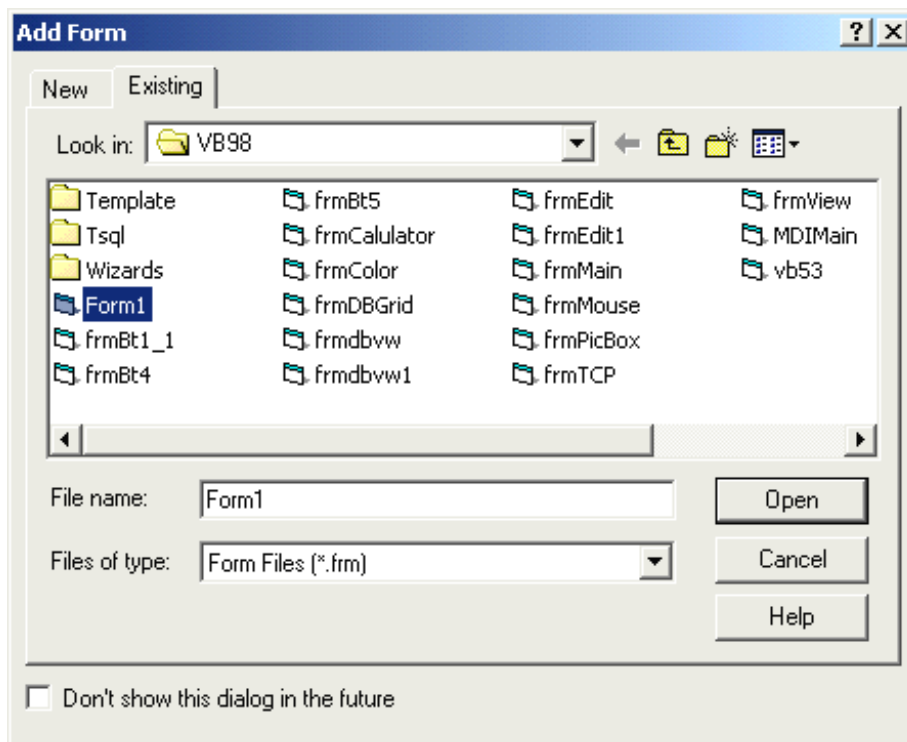
- Right-click trong Project Explorer/Add/Form,
- Nhấp đúp biểu tượng Form trong hộp thoại Add Form, form mới sẽ được thêm vào project (hình 1.15).

*Trường hợp thêm form có sẵn trên đĩa (từ Project khác)*

- Right-click trong Project Explorer/Add/Form ,
- Chọn thẻ Existing, nhấp đúp form cần thêm vào (hình 1.16) .



Hình 1.15: Thêm một form mới vào chương trình



Hình 1.16: Thêm một form có sẵn vào chương trình

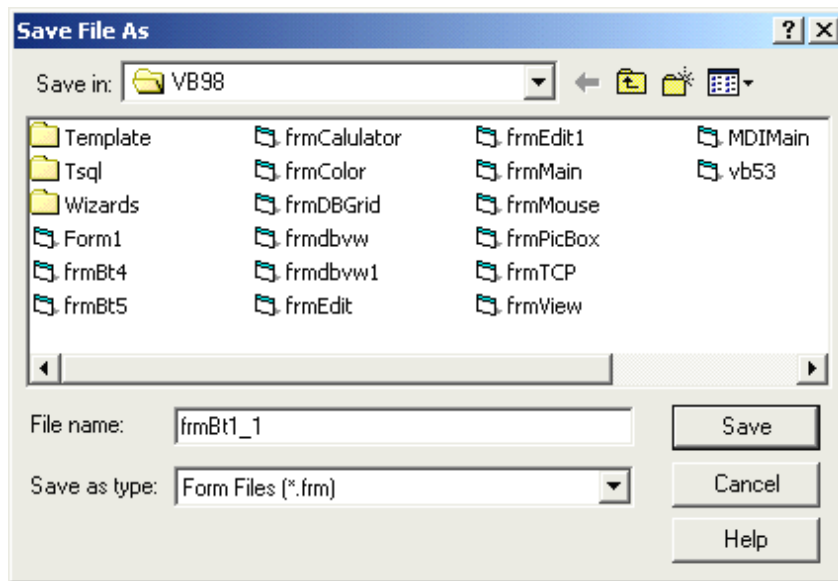
## 2. Xoá một form

- Right-Click form muốn xoá trong Project Explorer,
- Chọn Remove form trên menu.

### 3. Ghi Project lên đĩa

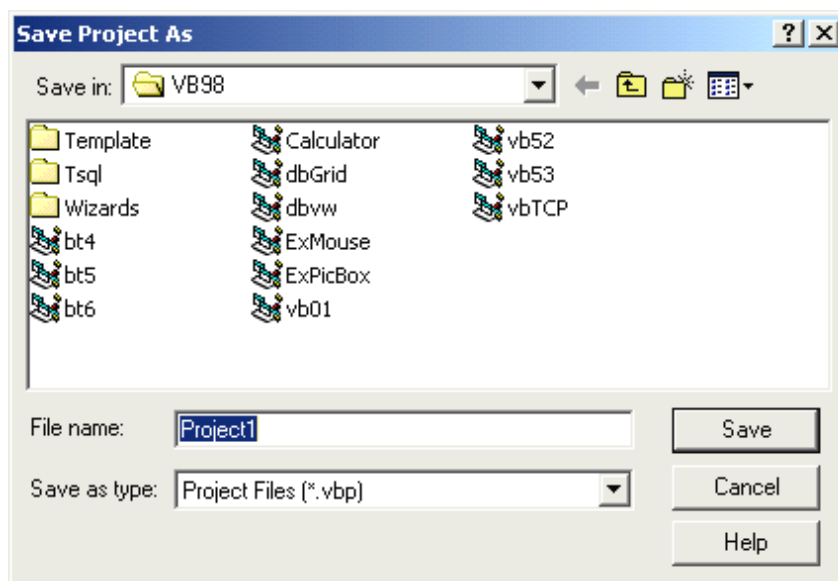
Cần nhắc lại là Project trong VB gồm nhiều thành phần như form, Module... mỗi form hoặc Module sẽ được ghi thành một tập tin. Như vậy phải đặt tên cho các form và module khi ghi Project.

- Bấm nút Save trên Toolbar hoặc chọn lệnh File/Save Project,
- VB sẽ lần lượt nhắc đặt tên cho các form. Hộp thoại đặt tên form có dạng hình 1.17



Hình 1.17: Hộp thoại đặt tên form khi ghi

- Nhập tên form vào hộp File name,
- Bấm nút Save và lặp lại bước này cho tất cả các form trong chương trình,
- Sau khi ghi tất cả các form. Xuất hiện hộp thoại đặt tên Project. Tập tin Project có phần mở rộng VBP.

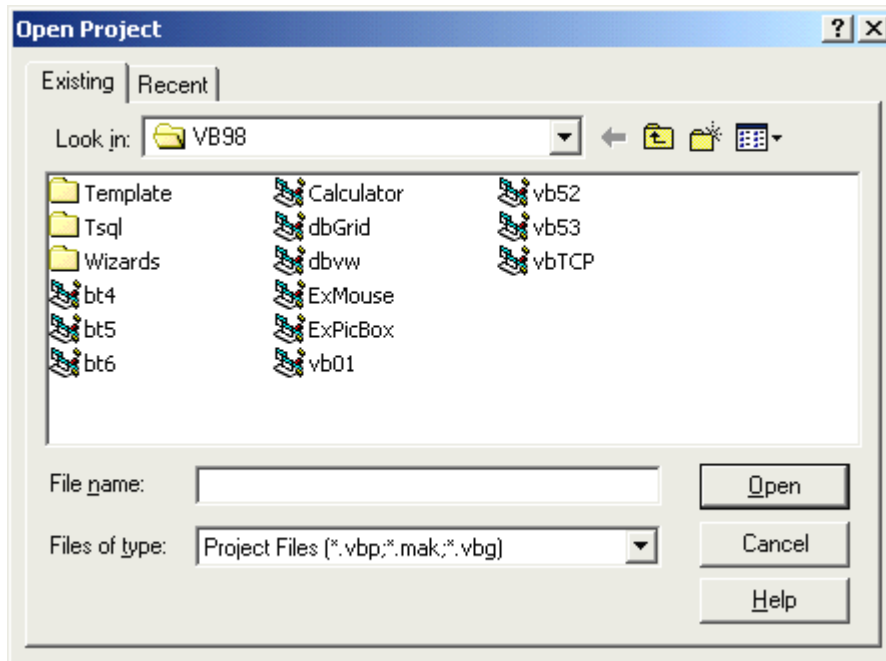


Hình 1.18: Hộp thoại đặt tên chương trình

Lưu ý: VB chỉ nhắc đặt tên khi ghi Project lần đầu tiên

#### 4. Nạp Project từ đĩa

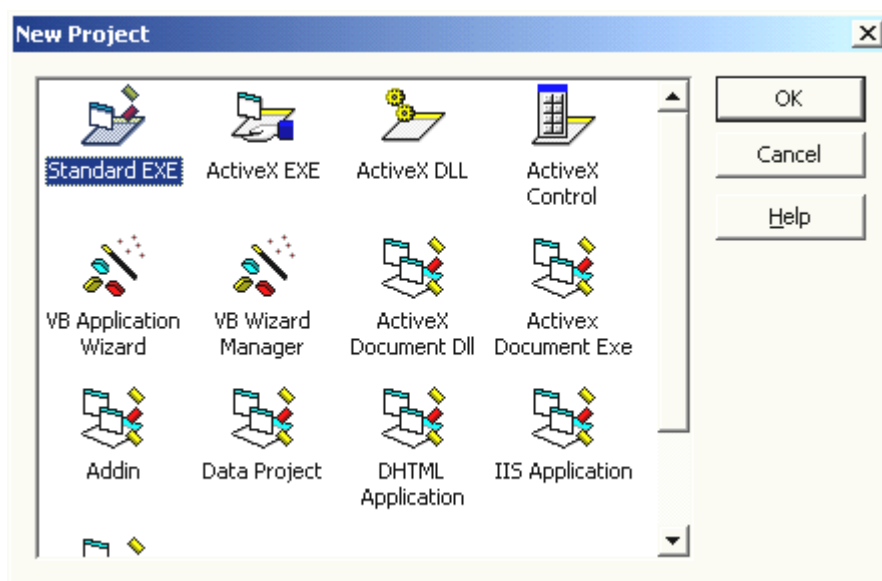
- Bấm nút Open trên Toolbar hoặc chọn lệnh File/Open Project. Hộp thoại Open project xuất hiện,
- Nhấp đúp tên project cần mở để nạp vào màn hình VB.



Hình 1.19: Nạp Project từ đĩa

#### 5. Tạo Project mới

- Chọn lệnh File/New Project. Hộp thoại New project xuất hiện,
- Nhấp đúp biểu tượng Standar EXE để tạo project mới.



Hình 1.20: Tạo một Project mới

## VI. MỘT CHƯƠNG TRÌNH VÍ DỤ

Thiết kế chương trình nhập hai số, tính tổng và in kết quả

Giao diện chương trình có dạng như sau:



Hình 1.21: Giao diện của chương trình ví dụ

- Nhập số thứ nhất vào Textbox phía trên.
- Nhập số thứ hai vào Textbox phía dưới.
- Bấm nút “Tinh”. Kết quả phép cộng 2 số xuất hiện trong ô dưới cùng

Các bước thực hiện như sau:

1. Khởi động Visual Basic , chọn New/ Standard EXE

2. Điều chỉnh các thuộc tính của form1 theo như bảng sau:

Thuộc tính (Property)	Giá trị (Value)
Name	frmTinh
Caption	Cong hai so
Height	2500
Width	2800

3. Nhấp đúp Label trên Toolbox để đặt Label lên giữa form. Di chuyển và đặt các thuộc tính theo như bảng sau:

Thuộc tính (Property)	Giá trị (Value)
Name	Label1
Caption	Nhap so thu 1

4. Làm tương tự như bước 3 với các thuộc tính theo như bảng sau:

Thuộc tính (Property)	Giá trị (Value)
Name	Label2
Caption	Nhap so thu 2

5. Nhấp đúp Textbox trên Toolbox. Textbox thứ nhất sẽ xuất hiện chính giữa form.

Di chuyển và điều chỉnh các thuộc tính theo bảng sau:

Thuộc tính (Property)	Giá trị (Value)
Alignment	1 - Right justify
Name	txtSo1
Height	315
Width	735



6. Nhấp đúp Textbox trên Toolbox. Textbox thứ hai sẽ xuất hiện chính giữa form. Di chuyển và điều chỉnh các thuộc tính theo bảng sau:

Thuộc tính (Property)	Giá trị (Value)
Alignment	1 - Right justify
Name	txtSo2
Height	315
Width	735

7. Nhấp đúp Textbox trên Toolbox. Textbox thứ ba sẽ xuất hiện chính giữa form. Di chuyển và điều chỉnh các thuộc tính theo bảng sau:

Thuộc tính (Property)	Giá trị (Value)
Alignment	1 - Right justify
Name	txtTong
Locked	True
Height	315
Width	735

8. Nhấp đúp CommandButton trên Toolbox. Button sẽ xuất hiện chính giữa form. Di chuyển và điều chỉnh các thuộc tính theo bảng sau:

Thuộc tính (Property)	Giá trị (Value)
Name	CmdTinh
Caption	Tinh
Height	330
Width	1335

9. Nhấp đúp CommandButton để viết mã lệnh cho chức năng tính toán khi người dùng bấm vào nút này. Cửa sổ mã lệnh sẽ xuất hiện với phần khai báo thủ tục khi bấm nút được định nghĩa sẵn:

```
Private Sub CmdTinh_Click()
```

```
End Sub
```

Nhập lệnh để cuối cùng ta có

```
Private Sub CmdTinh_Click()
```

```
txtTong.Text = Val(txtSo1.Text) + Val(txtSo2.Text)
```

```
End Sub
```

10. Bấm nút Run trên Toolbar hoặc F5 để chạy chương trình. Nhập 2 số vào 2 textbox. Bấm nút tính. Kết quả cộng 2 số xuất hiện trong ô thứ 3.

11. Chọn File/Save Project để save các tập tin của Project như sau:

FrmTinh.FRM :Tập tin định nghĩa form frmTinh

Vd1.VBP :Tập tin định nghĩa Project

## Chương 2

# Đối Tượng và cách sử dụng Đối Tượng

### I. ĐỐI TƯỢNG

#### 1. Khái niệm

Visual Basic là ngôn ngữ lập trình kiểu đối tượng, chương trình Basic gồm các đối tượng. Làm việc với VB chính là làm việc với các đối tượng.

#### 2. Các đặc điểm của đối tượng

##### a. Tên

Mỗi đối tượng được đặt tên. Tên đối tượng được viết theo qui tắc sau:

- Có chiều dài tối đa 40 ký tự
- Không được bắt đầu bằng số
- Không có khoảng trắng

Để phân biệt đối tượng này với đối tượng khác, tên đối tượng được viết kèm với tiền tố (prefix) chỉ loại đối tượng. Các tiền tố được qui định như sau:

Loại đối tượng	Tiền tố	Loại đối tượng	Tiền tố
CheckBox	chk	Horizontal ScrollBar	hsb
ComboBox	cbo	Image	img
Command Button	cmd	Label	lbl
Common Dialog	cdl	Line	lin
Data Control	dat	ListBox	lst
Data Bound ComboBox	Bound dbc	Menu	mnu
Data Bound Grid	dbg	OLE Container	ole
Data Bound ListBox	dbl	Option Button	opt
Directory ListBox	dir	Picture Box	pic
Drive ListBox	drv	Shape	shp
File ListBox	fil	TextBox	txt
<u>Form</u>	<u>frm</u>	<u>Timer</u>	<u>tmr</u>

##### b. Thuộc tính (property)

Mỗi đối tượng có một số thuộc tính dùng mô tả đối tượng như vị trí, kích thước, trạng thái... Các thuộc tính của đối tượng trình bày trong cửa sổ thuộc tính.

##### c. Phương thức (method)

Là các hành vi của mỗi đối tượng như di chuyển (move), phóng lớn cửa sổ (maximize), thu nhỏ cửa sổ (minimize)...

#### d. Sự kiện (Event)

Là các tác động lên đối tượng, mỗi đối tượng sẽ phản ứng lại theo cách của nó tùy theo biến cố tác động vào. Người lập trình sẽ định nghĩa các lệnh để chương trình đáp ứng lại các biến cố tác động lên các đối tượng

Khi người lập trình tạo ra một đối tượng, cần:

- Đặt tên (điều chỉnh thuộc tính Name)
- Qui định thuộc tính (trong cửa sổ thuộc tính)
- Định nghĩa các hoạt động của đối tượng tùy theo biến cố tác động vào (chọn loại biến cố trong code view window)

#### 3. Truy xuất đối tượng

Truy xuất đối tượng bao gồm:

- Đọc hoặc đặt giá trị cho một thuộc tính
- Gọi một phương thức

Để truy xuất một đối tượng, sử dụng cách viết

<Tên đối tượng>.<tên property hoặc method>

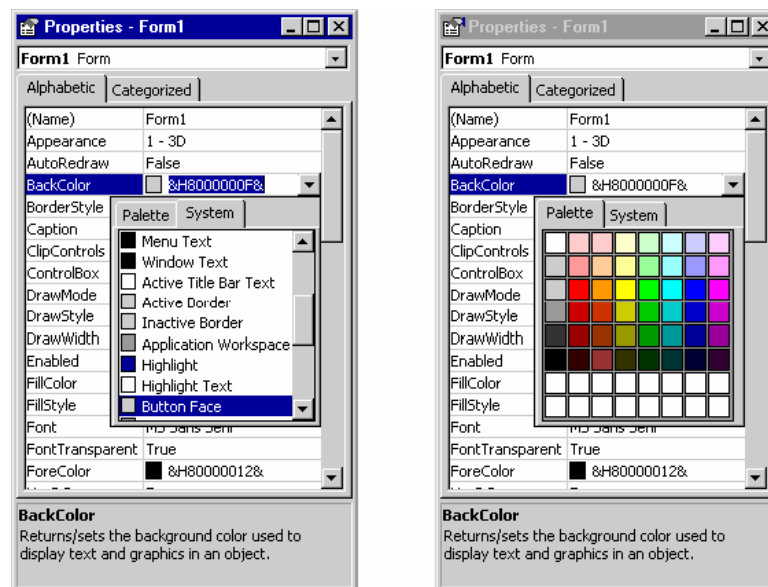
Ví dụ:

`adoRS.MoveNext`

`CmdPrint.Enabled = True`

#### 4. Các thuộc tính chung

- **Left, Top:** Tọa độ góc trên bên trái.
- **Height, Weight:** Chiều cao, độ rộng đối tượng.
- **ForeColor, BackColor:** Màu chữ, màu nền đối tượng. Có thể chọn màu theo bộ màu chuẩn của windows hoặc chọn màu tùy ý trên các thẻ tương ứng tại thuộc tính này trong cửa sổ thuộc tính (Hình 2.1).



Hình 2.1: Các thẻ chọn màu

Bảng sau trình bày một số hằng khai báo giá trị màu hệ thống

Hằng	Giá trị (Hex)	Ý nghĩa
vbActiveBorder	&H8000000A	Màu viền cửa sổ hoạt động
vbActiveTitleBar	&H80000002	Màu thanh tiêu đề cửa sổ hoạt động
vbActiveTitleBarText	&H80000009	Màu chữ tiêu đề cửa sổ hoạt động
vbApplicationWorkspace	&H8000000C	Màu nền cửa sổ ứng dụng giao diện đa tài liệu (MDI)
vbButtonFace	&H8000000F	Màu nút lệnh
vbButtonShadow	&H80000010	Màu bóng viền nút lệnh
vbButtonText	&H80000012	Màu chữ trên nút
vbDesktop	&H80000001	Màu desktop
vbGrayText	&H80000011	Màu chữ trên đối tượng không hoạt động
vbHighlight	&H8000000D	Màu nền phần được chọn
vbHighlightText	&H8000000E	Màu chữ phần được chọn
vbInactiveBorder	&H8000000B	Màu viền cửa sổ không hoạt động
vbInactiveCaptionText	&H80000013	Màu chữ tiêu đề cửa sổ không hoạt động
vbInactiveTitleBar	&H80000003	Màu thanh tiêu đề cửa sổ không hoạt động
vbInactiveTitleBarText	&H80000013	Màu chữ tiêu đề cửa sổ không hoạt động
vbInfoBackground	&H80000018	Màu nền lời nhắc (ToolTips)
vbInfoText	&H80000017	Màu chữ lời nhắc
vbMenuBar	&H80000004	Màu nền menu
vbMenuText	&H80000007	Màu chữ menu
vbScrollBars	&H80000000	Màu thanh cuộn
vbWindowBackground	&H80000005	Màu nền cửa sổ
vbWindowFrame	&H80000006	Màu khung cửa sổ
vbWindowText	&H80000008	Màu chữ trong cửa sổ

**Enabled:** Thuộc tính cho phép đối tượng hoạt động (True, False).

**Font:** Thuộc tính chọn Font chữ.

**Visible:** Thuộc tính cho phép xuất hiện đối tượng (True, False).

**Index:** Chỉ số mảng (mảng đối tượng).

**ToolTipText:** Chuỗi lời nhắc khi trỏ chuột trên đối tượng.

## 5. Các sự kiện chung

Sự kiện	Xảy ra khi
<b>Click</b>	Người dùng click trên đối tượng
<b>DbClick</b>	Người dùng nhấp đúp trên đối tượng
<b>DragDrop</b>	Người dùng kéo thả một đối tượng
<b>DragOver</b>	Người dùng kéo một đối tượng qua một đối tượng khác
<b>Gotfocus</b>	Đối tượng nhận focus

<b>KeyDown</b>	Người dùng nhấn một phím trong khi đối tượng đang nhận focus
<b>KeyPress</b>	Người dùng nhấn và thả một phím trong khi đối tượng đang nhận focus
<b>KeyUp</b>	Người dùng thả phím trong khi đối tượng đang nhận focus
<b>LostFocus</b>	Đối tượng không nhận focus nữa
<b>MouseDown</b>	Người dùng bấm một phím bất kỳ trên mouse trong khi mouse pointer đang ở vị trí đối tượng
<b>MouseMove</b>	Người dùng di chuyển mouse trên đối tượng
<b>MouseUp</b>	Người dùng thả phím mouse trong khi mouse pointer đang ở vị trí đối tượng

## II. ĐỐI TƯỢNG FORM

### 1. Thuộc tính

Thuộc tính	Ý nghĩa
<b>Caption</b>	Đặt tiêu đề cho form. Giá trị mặc định là tên form
<b>BorderStyle</b>	Quy định kiểu khung cho form
<b>Appearance</b>	Quy định cách thể hiện form (Flat/ 3D)
<b>ControlBox</b>	Có hoặc không có Control Menu Box (True/False)
<b>MaxButton</b>	Làm mờ nút phóng lớn (True/False)
<b>MinButton</b>	Làm mờ nút thu nhỏ (True/False)
<b>Icon</b>	Quy định Icon đại diện cho form
<b>Picture</b>	Đặt hình làm nền cho form
<b>Moveable</b>	Di chuyển/ Không di chuyển được (True/False)
<b>ShownInTaskbar</b>	Có nút đại diện chương trình trên taskbar (True/False)
<b>WindowState</b>	Trạng thái form (Normal/Minimized/Maximized)

### 2. Phương thức

<b>Show</b>	Xuất hiện form
<b>Hide</b>	Che dấu form

Ví dụ:

frmMain.Show ‘ Làm xuất hiện form

Hoặc

FrmMain.Hide ‘ Che dấu form

Lưu ý:

Phương thức Show nạp form vào bộ nhớ và làm xuất hiện nó trên màn hình. Nếu form đã được nạp vào trước đó thì nó chỉ làm xuất hiện form trên .

Phương thức Hide làm form không xuất hiện trên màn hình, nó vẫn còn được nạp vào bộ nhớ, để giải phóng form khỏi bộ nhớ, sử dụng phương thức **Unload <Đối tượng>**

### 3. Xử lý sự kiện (Handling Event)

Sau đây là một số sự kiện quan trọng đối với một form

Sự kiện	Xảy ra khi
<b>Load</b>	Form được nạp vào bộ nhớ
<b>Activate</b>	Form xuất hiện lần đầu tiên hoặc khi chuyển trở lại form từ một form khác
<b>Deactivate</b>	Người dùng chuyển sang form khác hoặc form thực hiện phương thức hide
<b>Unload</b>	Form được giải phóng khỏi bộ nhớ
<b>Initialize</b>	Form được tạo ra ban đầu trong bộ nhớ

Ví dụ 1 - Kiểm tra các sự kiện Initialize, Load, Unload:

1. Khởi động Visual Basic/Standard EXE
2. Nhấp đúp vào form1 để mở cửa sổ mã lệnh (code window), viết lệnh cho sự kiện

**Load** như sau:

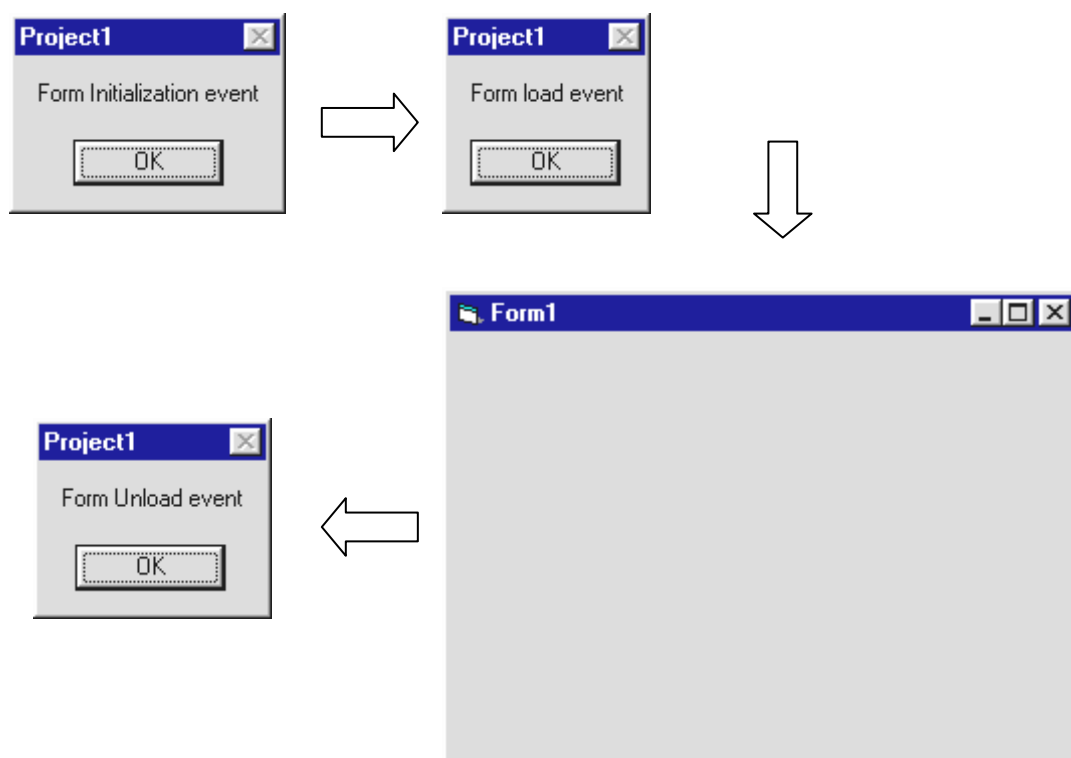
```
Private Sub Form_Load()
    MsgBox "Form Load Event"
End Sub
```

3. Lặp lại bước 3 để định nghĩa mã lệnh cho các sự kiện Initialize và Unload

```
Private Sub Form_Initialize()
    MsgBox "Form Initialization Event"
End Sub

Private Sub Form_Unload(Cancel As Integer)
    MsgBox "Form Unload Event"
End Sub
```

4. Bấm F5 để chạy chương trình, để ý các Message Box sẽ xuất hiện theo thứ tự do trình tự Initialize → Load → Unload



**Hình 2.2:** Kiểm tra các sự kiện Load, Unload, Initialize

### Ví dụ 2 - Kiểm tra các phương thức Show, Hide, Unload

Chương trình khi chạy sẽ xuất hiện 1 form như hình. Bấm nút “Show second form”, form thứ 2 sẽ xuất hiện. Bấm nút “Close this form”, form thứ 2 sẽ đóng lại.

Các bước thiết kế như sau:

1. New/Standard EXE
2. Đặt thuộc tính Caption của Form 1 thành **Vi dụ 2 - Form 1**
3. Nhấp đúp CommandButton trên Toolbox, Button xuất hiện trên Form1. Điều chỉnh thuộc tính Caption thành “Show second form”
4. Bấm nút Add form/form để thêm form2.
5. Đặt thuộc tính Caption của Form 2 thành **Vi dụ 2 - Form 2**
6. Nhấp đúp CommandButton trên Toolbox, Button xuất hiện trên Form2. Điều chỉnh thuộc tính Caption thành “Close this form”
7. Nhấp đúp Button trên form 2, định nghĩa mã lệnh như sau:

```
Private Sub Command1_Click()
    Unload Me
End Sub
```

8. Nhấp đúp Button trên form 1, định nghĩa mã lệnh như sau:

```
Private Sub Command1_Click()
    Form2.Show
End Sub
```

Bấm F5 chạy chương trình để kiểm tra kết quả

### III. LABEL

Trình bày một nội dung trên form

#### 1. Thuộc tính

Thuộc tính	Ý nghĩa
<b>Caption</b>	Qui định nội dung trình bày
<b>Alignment</b>	Quy định kiểu canh lề trong Label (0-Left 1- Right 2- Center)
<b>BackStyle</b>	Kiểu nền Label (0 - Transparent 1 - Opaque)
<b>AutoSize</b>	Tự động co giãn kích thước Label để thể hiện đầy đủ nội dung (True/False)
<b>Wordwrap</b>	Tự động cuộn chữ (True/False)

Và các thuộc tính chung

#### 2. Xử lý sự kiện

Gồm các sự kiện chung

### IV. TEXTBOX

Cho phép người dùng nhập một nội dung

## 1. Thuộc tính

Thuộc tính	Ý nghĩa
<b>Text</b>	Chứa nội dung nhập vào
<b>Alignment</b>	Quy định kiểu canh lề trong TextBox (0-Left 1- Right 2-Center)
<b>Locked</b>	Cho phép thay đổi nội dung textbox (True/False)
<b>MaxLength</b>	Qui định chiều dài tối đa cho phép nhập
<b>Multiline</b>	Cho phép nhập nội dung nhiều dòng (True/False)

Và các thuộc tính chung

## 2. Xử lý sự kiện

Gồm các sự kiện chung

### v. COMMANDBUTTON

Đối tượng được sử dụng để ra lệnh

#### 1. Thuộc tính

**Caption** Nội dung thể hiện trên nút bấm

Và các thuộc tính chung.

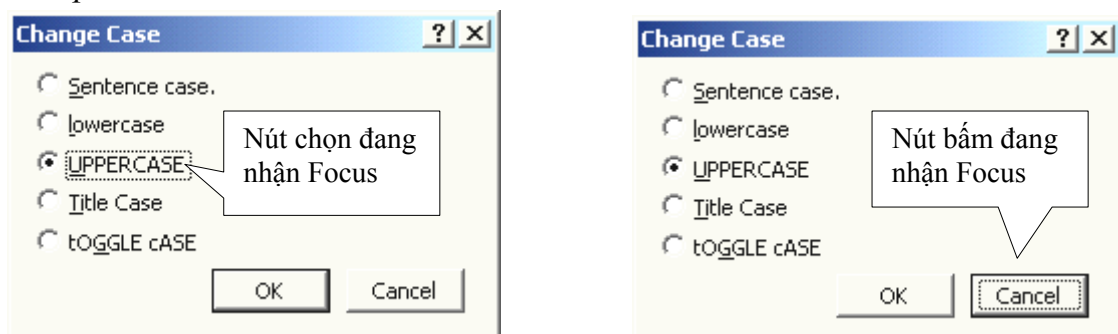
#### 2. Xử lý sự kiện

Gồm các sự kiện chung

### vi. FOCUS VÀ THỨ TỰ TAB

#### 1. Focus

Trên màn hình Windows, mỗi một đối tượng điều khiển khi được chọn để hoạt động (Active) sẽ nhận focus. Khi một cửa sổ hoặc form đang nhận focus thanh tiêu đề (Title Bar) sẽ có màu đậm. Khi một đối tượng điều khiển trên form nhận focus sẽ có đường viền bao quanh đối tượng hoặc cursor xuất hiện bên trong đối tượng (Textbox). Người dùng có thể thay đổi focus của đối tượng trên form bằng cách sử dụng phím Tab hoặc Shift+Tab. *Đối tượng nhận focus sẽ phản ứng với các sự kiện bấm phím*



Hình 2.3: Đối tượng nhận Focus



## 2. Thứ tự Tab (Tab Order)

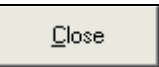
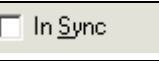
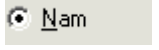
Người dùng có thể chọn đối tượng nhận focus trên form bằng cách bấm phím *Tab* hoặc *Shift+Tab* theo thứ tự các đối tượng được đặt lên form. Có thể qui định thứ tự này trong lúc thiết kế giao diện chương trình bằng cách điều chỉnh thuộc tính *TabIndex*. Đối tượng nhận focus đầu tiên trên form sẽ có *TabIndex = 0*. Để chọn đối tượng nhận focus trên form bằng chương trình, sử dụng phương thức **SetFocus**.

## 3. Phím nóng (HotKey)

Là tổ hợp phím kết hợp giữa phím *Alt* và một phím khác. Hotkey được sử dụng để chọn nhanh một đối tượng trên form bằng bàn phím mà không cần bấm phím *TAB* để chọn đối tượng theo thứ tự *Tab*.

Hotkey được định nghĩa trên thuộc tính *Caption* của đối tượng bằng cách nhập ký tự "&" phía trước ký tự muốn định nghĩa Hotkey

Ví dụ:

Muốn Đối tượng có Hotkey	Giá trị HotKey	Giá trị của thuộc tính Caption
	Alt+C	&Close
	Alt+S	In &Sync
	Alt+S	&Nam

Riêng *TextBox* thì Hotkey được định nghĩa trên thuộc tính *Caption* của *Label* đi kèm với *TextBox*. *Label* được gọi là đi kèm với *TextBox* nếu *TabIndex* của nó có giá trị kế trước (nhỏ hơn 1 đơn vị) giá trị *TabIndex* của *TextBox*

## 4. Ví dụ

Phần sau trình bày ví dụ về các định nghĩa Hotkey và thứ tự nhận focus cho chương trình ví dụ đã trình bày ở chương 1

Mở lại project *vd1.prj* đã làm ở chương 1, điều chỉnh lại thuộc tính của các đối tượng theo như bảng sau:

Form1		TextBox1	
Thuộc tính (Property)	Giá trị (Value)	Thuộc tính (Property)	Giá trị (Value)
Name	FrmTinh	Name	txtSo1
Caption	Cong hai so	Height	315
Height	2500	Width	735
Width	2800	TabIndex	1
Label1		Label2	
Thuộc tính (Property)	Giá trị (Value)	Thuộc tính (Property)	Giá trị (Value)
Name	Label1	Name	Label2
Caption	Nhap so thu &1	Caption	Nhap so thu &2
TabIndex	0	TabIndex	2

TextBox2		Width	1335
Thuộc tính (Property)	Giá trị (Value)	TabIndex	4
Name	txtSo2	TextBox3	
Height	315	Thuộc tính (Property)	Giá trị (Value)
Width	735	Name	txtTong
TabIndex	3	Locked	True
CommandButton		Height	315
Thuộc tính (Property)	Giá trị (Value)	Width	735
Name	CmdTinh	TabIndex	5
Caption	&Tinh		
Height	330		

Bấm F5 chạy chương trình. Để ý thứ tự nhận focus là TextBox1, TextBox2 và CommandButton. Các Hotkey Alt+1, Alt+2, Alt+T cũng có tác dụng tương tự.

Muốn con trỏ tự động chuyển sang TextBox dưới để nhập số thứ hai sau khi nhập số thứ nhất và bấm Enter, viết lệnh cho sự kiện bấm phím trên có TextBox như sau: thêm khả năng chuyển focus bằng cách bấm Enter sau khi nhập số tại các Textbox, có thể định nghĩa thêm các thủ tục xử lý sự kiện bấm phím Enter cho các Textbox1 và 2 như sau:

```
Private Sub txtSo1_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 then txtSo2.Setfocus
End Sub
Private Sub txtSo2_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 then CmdTinh.Setfocus
End Sub
```

# Chương 3

## Kiểu dữ liệu – Hằng – Biến

### I. BIẾN (Variable)

#### 1. Định nghĩa

Biến là ô nhớ chứa dữ liệu, giá trị của biến có thể thay đổi trong chương trình.

#### 2. Khai báo

Dạng

**Dim** <Tên> **As** <Kiểu> [, <Tên> **As** <Kiểu>]

Hoặc

**Dim** <Tên>

Trường hợp đầu, kiểu biến được khai báo rõ ràng, trường hợp sau kiểu của biến sẽ được xác định khi có lệnh gán giá trị cho biến.

Ví dụ:

*Dim X As Integer*

*Dim Ht As String*

#### 3. Quy tắc đặt tên biến

- Có chiều dài tối đa 255
- Không được bắt đầu bằng số
- Không sử dụng khoảng trắng
- Không dùng các ký hiệu toán tử
- Không trùng từ khoá
- Không phân biệt chữ thường và chữ in

Ví dụ

#### Các biến đặt tên đúng

*MyNum&*

*i%*

*iNumOne*

*strInputValue*

#### Các biến đặt tên sai

*1Week*

*Ho ten*

*Giai.thua*

#### 4. Truy xuất biến

Biến được truy xuất bằng cách viết tên.

Ví dụ

*Dim X As Integer*

*Dim Y As Integer*

*X = 5*

*Y = 7*

*X = Y+2*     ‘ *Trị của biến X được gán bằng trị của biến Y cộng thêm 2*

*X = X+1*     ‘ *Tăng giá trị của biến X*

### Lưu ý

Biến sử dụng có thể không cần khai báo. Điều này có thể gây ra lỗi, ví dụ:

*Dim Songay*

*Dim X*

*Songay = 1*

*X = 5*

*SoNgau = X+1*     ‘ *Visual Basic xem Songau là biến mới*

Để buộc Visual Basic không tự động tạo biến khi chưa khai báo có thể thực hiện 1 trong 2 cách sau:

- Viết phát biểu **Option Explicit** trong phần **General** của cửa sổ lệnh.
- Qui định bằng tùy chọn **Require variable Declaration** trong **Tools/Options/Editor**

Giá trị ban đầu của các loại biến sau khi khai báo như sau:

Kiểu dữ liệu	Giá trị đầu
<b>Integer</b>	0
<b>Long</b>	0
<b>Single</b>	0
<b>Double</b>	0
<b>String</b>	"" (blank)
<b>Boolean</b>	False
<b>Variant</b>	EMPTY
<b>Date</b>	0
<b>Currency</b>	0

## 5. Phạm vi sử dụng biến

Một biến được khai báo chỉ tồn tại trong phạm vi khai báo, ngoài phạm vi đó mà sử dụng lại Visual Basic sẽ xem như biến mới.

- Biến khai báo trong chương trình con chỉ có ý nghĩa trong chương trình con đó. Trong ví dụ sau, các biến X, Y, Z chỉ có ý nghĩa trong thủ tục xử lý sự kiện cmdTinh.

*Private sub cmdTinh()*

*Dim X As Integer*

*Dim Y As Integer*

*Dim Z As Integer*

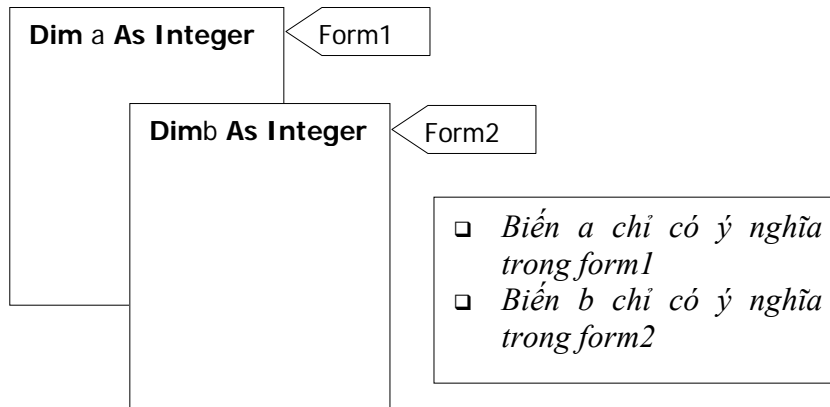
```
X = CInt(txtSo1.Text)
```

```
Y = CInt(txtSo2.Text)
```

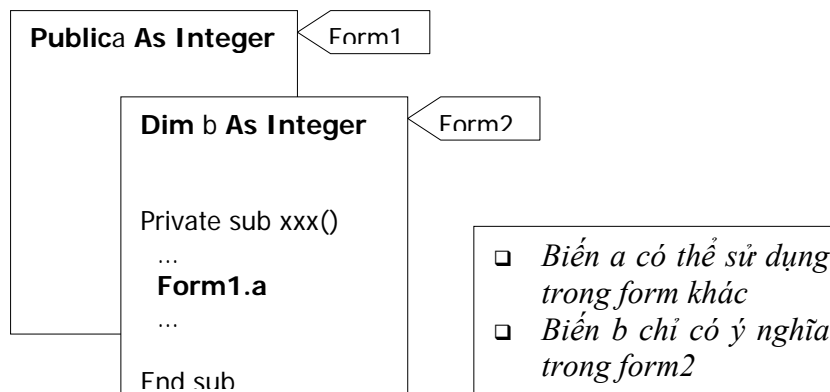
```
TxtTong.Text = X+Y
```

End sub

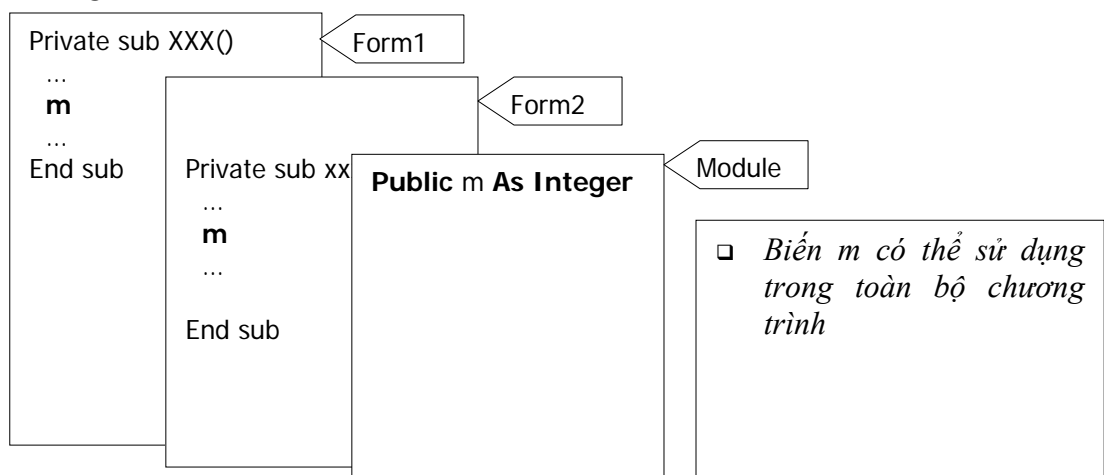
- Biến khai báo với từ khoá *Dim* trong phần *General* của form có ý nghĩa toàn cục trong form, tất cả các chương trình con định nghĩa trong form đều có thể sử dụng biến này.



- Biến khai báo với từ khoá *Public* trong phần *General* của form có ý nghĩa trong tất cả các chương trình con định nghĩa trong form đó và có thể sử dụng trong form khác bằng cách viết *<Tên form>.<Tên biến>*



- Biến khai báo với từ khoá *Public* trong Module có ý nghĩa trong toàn bộ chương trình.



## 6. Biến tĩnh

Là biến được khai báo với từ khoá static trong 1 chương trình con. Giá trị của biến tĩnh được sử dụng lại cho các lần gọi sau của chương trình con

Ví dụ: Thông báo số lần bấm nút, biến *iNumOfClicks* được khai báo tĩnh.

```
Private Sub MyButton_Click()
    Static iNumOfClicks as Integer
    iNumOfClicks = iNumOfClicks + 1
    MsgBox "Number of Clicks: " & CStr(iNumOfClicks)
End Sub
```

## II. KIỂU DỮ LIỆU

Kiểu	Kích thước	Phạm vi chứa
<b>Byte</b>	1 byte	0 .. 255
<b>Integer</b>	2 bytes	-32,768 .. 32,767
<b>Long</b>	4 bytes	Khoảng +/- 2.1E9
<b>Single</b>	4 bytes	-3.402823E38 .. -1.401298E-45 (giá trị âm) 1.401298E-45 .. 3.402823E38 (Giá trị dương)
<b>Double</b>	8 bytes	-1.79769313486232E308..-4.94065645841247E-324 (giá trị âm) 4.94065645841247E-324 ..1.79769313486232E308 (giá trị dương)
<b>Currency</b>	8 bytes	922,337,203,685,477.5808 .. 922,337,203,685,477.5807
<b>String</b>	1 byte cho mỗi ký tự	65,000 đối với chuỗi có kích thước cố định 2 tỷ đối với chuỗi động
<b>Boolean</b>	2 bytes	True , False
<b>Date</b>	8 bytes	Jan 1st 100 .. December 31st 9999
<b>Variant</b>	16 bytes + 1 byte cho mỗi ký tự	

## III. HẰNG

Hằng là đại lượng có giá trị không thay đổi trong chương trình. Hằng được khai báo trong phần General. Quy tắc đặt tên hằng cũng như biến. Hằng thường được khai báo bằng ký tự chữ in hoa. Khai báo hằng được viết như sau:

**Const** <Tên> [ **As** <Kiểu> ] = <Giá trị>

Ví dụ:

*Const* *METER\_TO\_FEET* = 3.3

#### IV. TOÁN TỬ

Toán tử	Ý nghĩa
^	Mũ
-	Đảo dấu
*, /	Nhân chia
\	Chia nguyên
Mod	Lấy phần dư phép chia số nguyên
+, -	Cộng, trừ
&	Ghép chuỗi
=, <>, <, >, <=, >=	So sánh
Not, And, Or	Luận lý

#### V. MỘT SỐ HÀM CHUẨN

##### 1. Hàm đại số

Hàm	Ý nghĩa	Ví dụ
<b>Abs(n)</b>	$ x $	Abs(-5) = 5
<b>Sqr(x)</b>	Căn bậc 2	Sqr(4)=2
<b>Exp(x)</b>	$e^x$	Exp(1)= 2.718282
<b>Log(x)</b>	Logx	Tính $\log_n(x)=\text{Log}(x)/\log(n)$
<b>Int(x)</b>	Số nguyên $\leq x$	Int(8.9) = 8, Int(-8.9)= -9
<b>Fix(x)</b>	Số nguyên $\leq x$	Fix(8.4) = 8, Fix(-8.9)= -8
<b>Round(x[,n])</b>	Làm tròn đến n chữ số phần thập phân	Round(4.5)=6, Round(34.673,2)=34.67
<b>Sin(x)</b>	$\sin x$	Sin(pi/2)=1
<b>Cos(x)</b>	$\cos x$	Cos(pi/3)=0.5
<b>Tan(x)</b>	$\tan x$	
<b>Atn(x)</b>	$\arctan x$	

##### 2. Hàm thời gian

Hàm	Ý nghĩa	Ví dụ
<b>Date</b>	Ngày hệ thống	Dim dt As Date Dt = Date
<b>Day(d)</b>	Ngày trong tháng (1-31)	Day(#12/2/00#)=2
<b>Month(d)</b>	Tháng (1-12)	Month(#12/2/00#)=12
<b>Year(d)</b>	Năm	Year((#12/2/00#)=2000
<b>Weekday(d)</b>	Ngày trong tuần (1-Chủ nhật, 2-Thứ hai, 7-Thứ bảy)	Weekday(Date)

### 3. Hàm chuyển đổi

Hàm	Ý nghĩa	Ví dụ
<b>Asc(n)</b>	Mã Ascii của ký tự n	Asc('a')=97, asc('A')=65
<b>Chr(n)</b>	Ký tự có mã n	Chr(65)='A'
<b>Ucase(s)</b>	Đổi chuỗi chữ thường thành chữ in	Ucase("abcd")="ABCD"
<b>Val(s)</b>	Đổi chuỗi thành số	Val("1234")=1234
<b>Str(n)</b>	Đổi số thành chuỗi	Str(12.45)=" 12.45" Str(-4.56) = "-4.56"

### 4. Hàm kiểm tra kiểu dữ liệu

Hàm	Ý nghĩa
<b>IsNumeric(n)</b>	Kiểm tra n có phải là số hợp lệ
<b>IsDate(n)</b>	Kiểm tra n có phải là giá trị ngày hợp lệ

## VI. HỘP THÔNG BÁO (MESSAGE BOX)

Là một lớp cửa sổ windows đã định nghĩa sẵn. Hộp thông báo được để trình bày các thông điệp nhắc nhở người dùng từ chương trình hoặc yêu cầu người dùng xác nhận một điều gì đó. Hộp thông báo thực chất là một form với các thành phần sau:

- Nội dung thông báo
- Icon bên trái dùng mô tả tính chất loại thông báo
- Nút bấm để trả lời, gồm các loại OK, Cancel, Yes, No, Abort, Retry, Ignore

Dạng hàm

**MsgBox(<Thông báo>,<Các nút>,<Tiêu đề>)**

Trong đó:

**<Thông báo>**

Chuỗi ký tự thông báo. Thông báo có chiều dài tối đa 1024 ký tự. Muốn thông báo hiện trên nhiều dòng, sử dụng ký tự chr(13)

**<Các nút>**

Qui định loại nút bấm và icon được sử dụng trong hộp thông báo, gồm các hằng sau:

	Hằng	Giá trị	Ý nghĩa
<b>Button</b>	vbOKOnly	0	Chỉ có nút OK
	vbOKCancel	1	Nút OK và Cancel
	vbAbortRetryIgnore	2	Nút Abort, Retry và Ignore
	vbYesNoCancel	3	Nút Yes, No, Cancel
	vbYesNo	4	Nút Yes, No
	vbRetryCancel	5	Nút Retry và Cancel
<b>Icon</b>	vbCritical	16	Icon
	vbQuestion	32	Icon



	vbExclamation	48	Icon
	vbInformation	64	Icon
<b>Focus</b>	vbDefaultButton1	0	Nút đầu tiên mặc định có focus
	vbDefaultButton2	256	Nút thứ hai mặc định có focus
	vbDefaultButton3	512	Nút thứ ba mặc định có focus
<b>Modal</b>	vbApplicationModal	0	Người dùng phải trả lời rồi mới có thể tiếp tục sử dụng chương trình, có thể chuyển sang các chương trình khác
	vbSystemModal	4096	Người dùng phải trả lời rồi mới có thể tiếp tục sử dụng chương trình, không thể chuyển sang các chương trình khác

Tham số <các button> được lấy giá trị bằng tổng các hằng trong mỗi nhóm trên.

#### Ví dụ

Giá trị **vbYesNo+vbQuestion+vbDefaultButton1** làm cho hộp thông báo có 2 nút Yes-No, Icon hiển thị là Question, nút đầu tiên có focus.

Nếu bỏ qua tham số này, hộp thông báo chỉ có nút OK

#### <Tiêu đề>

Qui định tiêu đề hộp thông báo, nếu không có tham số này, tiêu đề sẽ là tên của chương trình.

- Khi muốn sử dụng MsgBox với mục đích thông báo, thường chỉ cần ghi tham số thứ nhất. Ví dụ:

```
If Not IsNumeric(Text1.Text) then
```

```
MsgBox "Dữ liệu nhập không hợp lệ"
```

```
End if
```

- Khi chương trình muốn người sử dụng xác nhận một điều gì đó thì phải sử dụng MsgBox dưới dạng hàm. Hàm MsgBox khi đó sẽ trả về giá trị tùy theo nút mà người sử dụng bấm, các giá trị trả về có thể là:

Hằng	Giá trị	Nút đã bấm
vbOK	1	OK
vbCancel	2	Cancel
vbAbort	3	Abort
vbRetry	4	Retry
vbIgnore	5	Ignore
VbYes	6	Yes
VbNo	7	No

#### Ví dụ:

```
Ans = MsgBox("Do you want to save ?", _  
vbYesNoCancel + vbApplicationModal, "Warning")
```

```
if Ans = vbYes then  
    SaveDocument  
elseif Ans = vbNo then  
    Quit  
else  
    Continue  
End if
```

# Chương 4

## Các cấu trúc điều khiển

### I. LỆNH ĐIỀU KIỆN IF

#### Dạng 1:

*If <Điều kiện> then <lệnh>*

Chỉ có một <Lệnh> viết sau then

#### Ví dụ:

*Max = a*

*If Max < b then max = b*

#### Dạng 2:

*If <Điều kiện> then*

*<lệnh>*

*end if*

Dạng này được sử dụng thay cho dạng 1 khi có nhiều lệnh sau then

#### Dạng 3:

*If <Điều kiện 1> then*

*<lệnh 1>*

*elseif <Điều kiện 2 > then*

*<lệnh 2>*

...

*elseif <Điều kiện n > then*

*<lệnh n>*

*else*

*<lệnh n+1>*

*end if*

#### Ví dụ:

*If a > b then*

*Max = a*

*Else*

*Max = b*

*End if*

### II. LỆNH CHỌN LỰA CASE

Chọn lựa lệnh thực hiện theo giá trị

*Select Case <Biểu thức>*

...

*Case <Danh sách trị n>*

*<lệnh n>*

...

```

Case else
    <lệnh n+1>
End select

```

Ví dụ

```

Select Case Round(Diem)
Case 0 to 4
    Label1.Caption = "Kem"
Case 5,6
    Label1.Caption = "Trung binh"
Case 7,8
    Label1.Caption = "Khá"
Case 9,10
    Label1.Caption = "Giỏi"
Case else
    Label1.Caption = "Không hợp lệ"
End select

```

**III. LỆNH LẶP FOR .. NEXT**Dạng

```

For <Biến> = <Trị đầu> to <Trị cuối> [ Step <Bước tăng> ]
    <lệnh>
Next <Biến>

```

<lệnh> được thực hiện từ <trị đầu> đến <trị cuối>, giá trị của <biến> được thay đổi theo <bước tăng>

Ví dụ

Tính tổng các số nguyên từ 1 đến 10

```

S = 0
For i = 1 to 10
    s = s + i
Next i

```

Tính tổng các số chẵn

```

For i = 0 to 10 step 2
    s = s + i
Next i

```

Tạo ra một chuỗi có 10 chữ a

```

strS = ""
For i = 1 to 10
    strS = strS & "a"
Next i

```

Ví dụ: Vòng lặp sau tạo ra 10 chuỗi với cùng nội dung

```

Dim Words, Chars, MyString
For Words = 10 To 1 Step -1
    For Chars = 0 To 9
        MyString = MyString & Chars
    Next Chars

```

---

```
MyString = MyString & " "
```

*Next Words*

#### IV. LỆNH LẶP DO .. LOOP

##### Dạng 1

**Do while** <Điều kiện>

<Lệnh>

**Loop**

Các lệnh trong vòng lặp bắt đầu được thực hiện nếu điều kiện đúng và lặp lại cho đến khi nào điều kiện sai

##### Ví dụ

```
Dim I As Integer
```

```
Dim strS As String
```

```
i = 1
```

```
Do while i <= 10
```

```
    StrS = strS & "a"
```

```
    i = i + 1
```

```
Loop
```

##### Dạng 2

**Do**

<Lệnh>

**Loop Until** <điều kiện>

Các lệnh trong vòng lặp được thực hiện cho đến khi nào điều kiện đúng

##### Ví dụ

```
Dim I As Integer
```

```
Dim strS As String
```

```
i = 1
```

```
Do
```

```
    StrS = strS & "a"
```

```
    i = i + 1
```

```
Loop Until I > 10
```

##### Ví dụ:

Đếm số chữ số của 1 số nguyên dương

```
Dim Dem, So As Integer
```

```
So = Text1.Text
```

```
Dem = 0
```

```
Do
```

```
    So = So \ 10
```

```
    Dem = Dem + 1
```

```
Loop Until So = 0
```

##### Ví dụ:

Tìm ước số chung lớn nhất của 2 số nguyên dương x,y

```
Dim x,y As Integer
```

```
x = Text1.Text
```

```
y = Text2.Text
```

```
Do while x <> y
```

```
    If x > y then
```

```
        x = x - y
```

```
    else
```

```

        y = y-x
    end if

```

Loop

Ví dụ: Nhập tuổi từ bàn phím, giá trị tuổi nhập phải trong phạm vi từ 10 đến 99

```
Dim strAge As String
```

```
Dim intAge As Integer
```

```
Dim intPress As Integer
```

```
Do
```

```
    strAge = InputBox("How old are you?", "Age Ask")
```

```
    ' Check for the Cancel command button
```

```
    If (strAge = "") Then
```

```
        End ` Terminate program
```

```
    End If
```

```
    intAge = Val(strAge)
```

```
    If ((intAge < 10) Or (intAge > 99)) Then
```

```
        ' The user's age is out of range
```

```
        intPress = MsgBox("Your age must be between 10 and 99", vbExclamation,
"Error!")
```

```
    End If
```

```
Loop While ((intAge < 10) Or (intAge > 99))
```

## v. CHƯƠNG TRÌNH CON

Khi viết một chương trình lớn, để tránh viết lại nhiều lần các đoạn chương trình giống nhau, người ta định nghĩa các đoạn chương trình giống nhau, được dùng nhiều lần trong chương trình thành các module chương trình, còn được gọi là chương trình con. Các chương trình con này sẽ được định nghĩa ở một nơi nào đó trong chương trình bằng 1 tên, mỗi khi có yêu cầu sử dụng, nó sẽ được gọi bằng tên đã định nghĩa.

Ví dụ:

```

Private Sub ChangeSignal()
    If imgGreen.Visible = True Then
        imgGreen.Visible = False
        imgYellow.Visible = True
    ElseIf imgYellow.Visible = True Then
        imgYellow.Visible = False
        imgRed.Visible = True
    Else
        imgRed.Visible = False
        imgGreen.Visible = True
    End If
End Sub
Private Sub cmdChange_Click()

```

```

    ChangeSignal    ' Gọi thủ tục ChangeSignal.
End Sub

```

```

Private Sub imgGreen_Click()
    ChangeSignal    ' Gọi thủ tục ChangeSignal
End Sub

```

```

Private Sub imgRed_Click()
    ChangeSignal    ' Gọi thủ tục ChangeSignal
End Sub

```

```

Private Sub imgYellow_Click()
    ChangeSignal    ' Gọi thủ tục ChangeSignal
End Sub

```

Có 2 loại chương trình con là thủ tục (Sub) và hàm (Function)

### 1. Sub

Loại chương trình con thực hiện một tác vụ nào đó khi được gọi. Có 2 loại thủ tục là thủ tục tổng quát (General procedure) và thủ tục xử lý sự kiện (Event procedure).

- Thủ tục tổng quát được kích hoạt bằng lệnh gọi trong chương trình.
- Thủ tục xử lý sự kiện được kích hoạt khi có một sự kiện tác động lên form hoặc đối tượng điều khiển trên form. Thủ tục xử lý sự kiện thường có tên là <tên đối tượng>\_<tên sự kiện>. Ví dụ Form\_Load hoặc Commad1\_Click...

Khai báo thủ tục:

```

Private/Public Sub <Tên thủ tục>[(<Danh sách tham số>)]
    <Lệnh>
End sub

```

Thủ tục được khai báo với từ khoá Private chỉ được sử dụng trong form chứa nó (Form level).

Thủ tục được khai báo với từ khoá Public có thể sử dụng trong các form khác.

### 2. Hàm

Loại chương trình con luôn luôn trả về giá trị thông qua tên hàm

Khai báo hàm:

```

Private/Public Function <Tên thủ tục>[(<Danh sách tham số>)] [As
    <Kiểu>]
    <Lệnh>
End sub

```

Ví dụ: Định nghĩa hàm tính chiều dài cạnh huyền của tam giác vuông

```

Function Hypotenuse (A As Integer, B As Integer) As double
    Hypotenuse = Sqr(A ^ 2 + B ^ 2)
End Function

```

Gọi hàm

```

Dim x As double
x = Hypotenuse(Text1.Text, Text2.Text)
TxtTinh.text = str(x,2)

```

#### Ví dụ:

Tính ngày việt nam: Hàm Weekday cho giá trị là số thứ tự chỉ ngày trong tuần. Định nghĩa hàm vnDay cho giá trị là chuỗi ngày Việt nam

```
Public Function vnDay(nDay As Date) As String
```

```
    Select Case Weekday(nDay)
```

```
        Case 1
```

```
            VnDay = "Chủ nhật"
```

```
        Case 2
```

```
            VnDay = "Thứ hai"
```

```
        Case 3
```

```
            VnDay = "Thứ ba"
```

```
        Case 4
```

```
            VnDay = "Thứ tư"
```

```
        Case 5
```

```
            VnDay = "Thứ năm"
```

```
        Case 6
```

```
            VnDay = "Thứ sáu"
```

```
        Case 7
```

```
            VnDay = "Thứ bảy"
```

```
    End select
```

```
End function
```

```
Private Sub Command1_Click()
```

```
    Text1.text = "Hôm nay là " & vnDay(Date)
```

```
End sub
```

### **3. Khai báo**

- Chương trình con khai báo với từ khoá *Private* chỉ có ý nghĩa trong phạm vi khai báo
- Chương trình con khai báo với từ khoá *Public* trong form có thể sử dụng trong form đó và trong các form khác
- Chương trình con khai báo với từ khoá *Public* trong module có thể sử dụng trong toàn bộ chương trình



# Chương 5

## Mảng – Chuỗi – Collection

### I. MẢNG

#### 1. Định nghĩa:

Mảng là tập hợp các phần tử cùng kiểu dữ liệu được đánh thứ tự. Số thứ tự của mỗi phần tử được gọi là *chỉ số*.

#### 2. Khai báo:

**Dim/Public/Static** <Tên>(<Số phần tử>) **As** <Kiểu>

Ví dụ:

*Dim A(10) As Integer*      ‘ Mảng 10 số nguyên

*Dim Hoten(50) As String*      ‘ Mảng 50 chuỗi

Chỉ số đầu tiên mặc định là 0. Có 2 cách để khai báo một mảng bắt đầu từ chỉ số tùy ý:

- **Sử dụng phát biểu Option Base trong phần General**

Ví dụ:

*Option Base 1*      ‘ Khai báo mảng bắt đầu từ 1

- Khai báo phạm vi chỉ số:

**Dim/Public/Static** <Tên>(<Chỉ số đầu> to <Chỉ số cuối>) **As** <Kiểu>

Ví dụ:

*Dim A(1 to 10) As Integer*

Mảng được truy xuất bằng cách viết <tên>(chỉ số)

Ví dụ: Đổi năm dương lịch sang năm âm lịch:

*Dim Can(10) As String*

*Dim Chi(12) As String*

*Can(0) = "Canh"*

*Can(1) = "Tân"*

*Can(2) = "Nhâm"*

*Can(3) = "Quý"*

*Can(4) = "Giáp"*

*Can(5) = "Ất"*

*Can(6) = "Bính"*

*Can(7) = "Đinh"*

*Can(8) = "Mậu"*

*Can(9) = "Kỷ"*

*Chi(0) = "Thân"*

*Chi(1) = "Dậu"*

*Chi(2) = "Tuất"*

*Chi(3) = "Hợi"*

*Chi(4) = "Tý"*

*Chi(5) = "Sửu"*

*Chi(6) = "Dần"*

*Chi(7) = "Mão"*

*Chi(8) = "Thìn"*

*Chi(9) = "Ty"*

*Chi(10) = "Ngọ"*

*Chi(11) = "Mùi"*

*NDL = CInt(txtNDL.Text)*

*LblNAL.Caption = Can(NDL mod 10) & " " & Chi(NDL mod 12)*

Mảng trong ví dụ trên có thể vừa khai báo vừa gán giá trị ban đầu như sau:

*Can = Array("Giáp", "Át", "Bính", "Đinh", "Mậu", "Kỷ", "Canh", "Tân",  
"Nhâm", "Quý")*

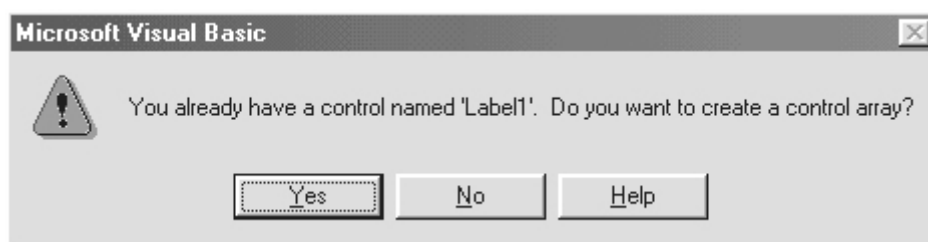
*Chi = Array("Thân", "Dậu", "Tuất", "Hợi", "Tý", "Sửu", "Dần", "Mão", "Thìn",  
"Ty", "Ngọ", "Mùi")*

### 3. Mảng đối tượng điều khiển

Với các đối tượng điều khiển cùng loại, có thể sử dụng mảng để không phải đặt quá nhiều tên và định nghĩa nhiều thủ tục xử lý sự kiện

#### a. Định nghĩa mảng đối tượng điều khiển

- Đặt 1 đối tượng trong nhóm muốn định nghĩa mảng lên form, đặt tên (sẽ dùng làm tên mảng) và qui định giá trị các thuộc tính cần thiết (thuộc tính về kích thước và màu sắc của các phần tử của mảng thường giống nhau, trừ thuộc tính caption),
- Right-Click trên đối tượng, chọn lệnh Copy,
- Right-Click trên form, chọn lệnh Paste. VB sẽ yêu cầu xác nhận muốn định nghĩa mảng vì nhận thấy đối tượng mới được sao chép có cùng tên với đối tượng trước đó trên form,



Hình 5.1: Hộp thông báo xác nhận có định nghĩa mảng đối tượng

- Trả lời Yes để định nghĩa mảng và lặp lại thao tác Paste cho các phần tử kế tiếp. Để ý là thuộc tính Index của các phần tử mảng có thứ tự tăng dần theo

đúng thứ tự được sao chép trên form. Đó cũng chính là chỉ số của đối tượng trong mảng.

### b. Viết lệnh cho mảng đối tượng điều khiển

- Nhấp đúp lên một trong các đối tượng thuộc mảng. Thủ tục xử lý sự kiện có dạng:

```
Private sub <Tên>_<Sự kiện>(Index As Integer)
```

```
End sub
```

Thay vì

```
Private sub <Tên>_<Sự kiện>()
```

```
End sub
```

- Thủ tục xử lý sự kiện được viết chung cho nhóm đối tượng định nghĩa là mảng, tham số Index được dùng để phân biệt phần tử nhận sự kiện đó.

### c. Duyệt mảng đối tượng điều khiển

Để duyệt mảng đối tượng điều khiển trên form, có thể sử dụng vòng lặp như ví dụ sau:

```
For i = txtFields.LBound To txtFields.UBound
```

```
txtFields(i).Text = ""
```

```
Next
```

Tuy nhiên nếu các phần tử mảng được tạo ra không liên tiếp do có một đối tượng thuộc mảng đã bị xóa thì hệ thống sẽ thông báo lỗi. Vì vậy cách tốt hơn là sử dụng lệnh lặp For Each như sau:

```
Dim txt As TextBox
```

```
For Each txt In txtFields
```

```
txt.Text = ""
```

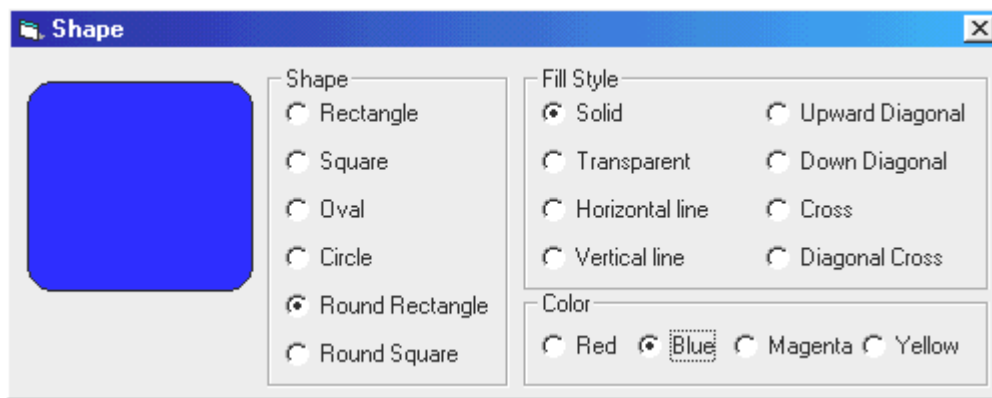
```
Next
```

## 4. Ví dụ

Thiết kế form chọn màu tô (FillColor), mẫu tô (FillStyle) và loại hình vẽ của đối tượng Shape. Form thiết kế có dạng sau:

**Bước 1** Thiết kế giao diện (Hình 5.2)

- Định nghĩa mảng các OptionButton cho nhóm Shape với tên opShape
- Định nghĩa mảng các OptionButton cho nhóm FillStyle với tên opFillStyle
- Định nghĩa mảng các OptionButton cho nhóm Color với tên opColor.



Hình 5.2: Giao diện chương trình ví dụ

### Bước 2 Viết lệnh

- Double-Click OptionButton trong nhóm Shape, viết lệnh :  

```
Private Sub opShape_Click(Index As Integer)
    Shape1.Shape = Index
End Sub
```
- Double-Click OptionButton trong nhóm FillStyle, viết lệnh :  

```
Private Sub opFillStyle_Click(Index As Integer)
    Shape1.FillStyle = Index
End Sub
```
- Double-Click OptionButton trong nhóm Color, viết lệnh :  

```
Private Sub opColor_Click(Index As Integer)
    Select Case Index
        Case 0
            Shape1.FillColor = vbRed
        Case 1
            Shape1.FillColor = vbBlue
        Case 2
            Shape1.FillColor = vbMagenta
        Case 3
            Shape1.FillColor = vbYellow
    End Select
End Sub
```

## 5. Mảng động và mảng tĩnh

### a. Mảng tĩnh

Là mảng được khai báo với từ khóa Dim như đã trình bày ở các phần trên. Mảng tĩnh luôn được khai báo với số phần tử xác định trước để chương trình dịch có thể dành vùng nhớ phù hợp.

**b.Mảng động**

Là mảng có số phần tử có thể thay đổi tùy ý trong lúc chạy chương trình. Điều này phù hợp hơn vì trong thực tế người lập trình không thể tiên liệu trước số phần tử thực tế.

Sử dụng mảng động gồm 2 bước:

- Khai báo hiện diện với từ khóa Dim nhưng số phần tử để rỗng.
- Tạo mảng thực sự khi cần thiết bằng phát biểu ReDim.

Ví dụ:

```
Dim Customers() As String
...
Sub Main()
    ReDim Customers(1000) As String
End Sub
```

Mảng động có thể được tạo lại nhiều lần khi cần thiết:

```
Sub Printeport()
    ReDim Customers(100) As String
    ...
    ReDim Customers(500) As String
    ...
End Sub
```

Tuy nhiên lệnh cấp phát mới sẽ xóa rỗng nội dung (chuỗi) hoặc gán bằng 0 (số) mọi phần tử đã có giá trị trước đó. Để bảo toàn giá trị các phần tử, sử dụng phát biểu ReDim Preserve.

Ví dụ:

```
ReDim Preserve Customers(500) As String
```

Lệnh cấp phát động cũng có thể áp dụng cho mảng nhiều chiều, tuy nhiên chỉ có thể làm thay đổi chiều cuối cùng.

Ví dụ:

```
ReDim Cells(1 To 100, 10) As Integer
...
ReDim Preserve Cells(1 To 100, 20) As Integer ' Đúng
ReDim Preserve Cells(1 To 200, 20) As Integer ' Sai
```

Có thể hủy một mảng bằng lệnh Erase. Đối với mảng động, Visual Basic giải phóng vùng nhớ đã cấp phát cho mảng; đối với mảng tĩnh, mọi phần tử được gán giá trị rỗng (chuỗi) hoặc có giá trị 0 (số).

### c. Các hàm Lbound, Ubound

Hàm được dùng để xác định chỉ số thấp nhất và cao nhất của một mảng. Nếu mảng có nhiều chiều, phải sử dụng thêm tham số thứ hai khi sử dụng hàm. Ví dụ đối với mảng Cells đã khai báo ở ví dụ trên, để lấy chỉ số thấp nhất, cao nhất của mỗi chiều, có thể thực hiện như sau:

```
Print LBound(Cells, 1)      ' In chỉ số thấp nhất của chiều đầu tiên
Print LBound(Cells)       ' Giống như trên
Print UBound(Cells, 2)    ' In chỉ số cao nhất của chiều thứ hai
' Tính số phần tử mảng
Num = (UBound(Cells) - LBound(Cells) + 1) * _
      (UBound(Cells, 2) - LBound(Cells, 2) + 1)
```

## 6. Một số vấn đề khác

### a. Mảng và biến variant

Visual Basic cho phép chứa mảng trong các biến variant rồi truy xuất các phần tử mảng thông qua biến này.

Ví dụ:

```
ReDim Names(100) As String, var As Variant
' Khởi động giá trị cho mảng Names
var = Names()      ' Sao chép mảng vào biến variant
Print var(1)      ' Truy xuất mảng qua biến variant
```

Một cách tương tự, có thể truyền một mảng cho chương trình con với tham số hình thức khai báo là variant rồi truy xuất các phần tử mảng trong chương trình con thông qua tham số đó.

Ví dụ: Hàm tính tổng các phần tử mảng

```
Function ArraySum(arr As Variant) As Variant
    Dim i As Long, result As Variant
    For i = LBound(arr) To UBound(arr)
        result = result + arr(i)
    Next
    ArraySum = result
End Function
```

Cũng có thể áp dụng cách trên để truyền mảng 2 chiều thông qua tham số hình thức có kiểu variant.

Ví dụ:

```
Private Sub Form_Load()
```

```

    Dim Fact(4, 4) As Integer
    abc Fact
End Sub
Private Sub abc(x As Variant)
    For i = 0 To 4
        For j = 0 To 4
            s = s + x(i, j)
        Next
    Next
End Sub

```

Để xác định kiểu của mảng khi truyền cho tham số variant, sử dụng hàm VarType để xác định kiểu như ví dụ sau:

```

If VarType(arr) = (vbArray + vbInteger) Then
    ' Mảng số nguyên
ElseIf VarType(arr) = (vbArray + vbLong) Then
    ' Mảng kiểu long
...
End if

```

### b. Gán mảng và trả về giá trị kiểu mảng

Điểm mới của mảng trong Visual Basic 6.0 so với các phiên bản trước đó là gán mảng và viết chương trình con trả về giá trị kiểu mảng.

Ví dụ: Gán mảng

```

ReDim a(10, 10) As Integer
Dim b() As Integer
...
b() = a()

```

Ví dụ: Hàm trả về giá trị kiểu mảng

```

Function InitArray(first As Long, Last As Long) As Long()
    ReDim result(first To Last) As Long
    Dim i As Long
    For i = first To Last
        result(i) = i
    Next
    InitArray = result
End Function

```

### c. Mảng Byte

Trong Visual Basic, mảng kiểu byte có tính chất đặc biệt, đó là có thể gán trực tiếp chuỗi cho một mảng byte. Khi đó, mảng được gán được cấp phát động để chứa đủ các ký tự được gán. Vì mỗi ký tự trong chuỗi của Visual Basic 5.0 và 6.0 có chiều dài 2 byte (unicode) nên số phần tử mảng được cấp phát sẽ gấp đôi số ký tự trong chuỗi.

Ví dụ:

```

Dim b() As Byte, Text As String
Text = "123"
b() = Text
For i = LBound(b) To UBound(b)
    Debug.Print b(i)      'Giá trị in ra sẽ là 49 0 50 0 51 0
Next

```

**II. CHUỖI KÝ TỰ****1. Khai báo:**

Dim <Biến> As String

Hoặc

Dim <Biến> As String\* Chiều dài

- Khai báo String: Khai báo chuỗi động có chiều dài tối đa 2 tỷ ký tự.
- Khai báo String\* Chiều dài: Khai báo chuỗi có chiều dài cố định, chiều dài tối đa 65535.

**2. Các hàm xử lý chuỗi**

**Len(s):** Lấy chiều dài chuỗi

Ví dụ: Len("abcd")=4

**Ucase(s):** Đổi chuỗi chữ thường thành chuỗi chữ in

Ví dụ: Ucase("abcd")="ABCD"

**Lcase(s):** Đổi chuỗi chữ in thành chuỗi chữ thường

Ví dụ: Lcase("ABCD")="abcd"

**Ltrim(s):** Cắt khoảng trắng bên trái chuỗi

Ví dụ: Ltrim(" Anh")="Anh"

**Rtrim(s):** Cắt khoảng trắng bên phải chuỗi

Ví dụ: Rtrim("Anh ")="Anh"

**Trim(s):** Cắt khoảng trắng 2 bên chuỗi

Ví dụ: Trim(" Anh ")="Anh"

**Left(s,n):** Trả về n ký tự đầu tiên bên trái chuỗi

Ví dụ: Left("Visual Basic",6)="Visual"

**Right(s,n):** Trả về n ký tự đầu tiên bên phải chuỗi

Ví dụ: Right("Visual Basic",5)="Basic"

**Mid(s,i,n):** Trả về n ký tự trong chuỗi bắt đầu từ vị trí i.

Ví dụ: Mid("Visual Basic",8,3)="Bas"

**Space(n):** Trả về chuỗi có n khoảng trắng.

Ví dụ: Space(5)=" "

**String(n,c):** Trả về chuỗi có n ký tự c.

Ví dụ: String(4,"x")="xxxx"



**Instr([i],[s1,s2],[n]):** Cho vị trí xuất hiện của chuỗi s2 trong s1.

Trong đó:

i Vị trí bắt đầu xét (tuỳ chọn)

S1: Chuỗi cần dò tìm

s2: Chuỗi tìm

n : Cách so sánh (0- So từng ký tự, 1-Không phân biệt chữ thường, chữ hoa)

St = "Visual Basic"

Ví dụ:

Instr(St,"a")=5

Instr(6,St,"a")=9

Instr(10,St,"a")=0

**Replace(s,s1,s2[ i, n]):** Tìm và thay thế s1 trong s bởi s2.

Trong đó:

i: vị trí bắt đầu thay thế, giá trị mặc định là 1 (thay thế từ đầu)

n : số lần thay thế, giá trị mặc định là 1 (thay thế tất cả)

Ví dụ:

St = "tôi đi học với bạn tôi"

Replace(St,"tôi","anh") = "anh đi học với bạn anh"

**Format(s,format):** Định dạng chuỗi s theo chuỗi định dạng format.

Ký tự thường sử dụng trong chuỗi định dạng:

@ : Thay thế cho một ký tự hoặc khoảng trắng

&: Thay thế cho một ký tự hoặc không có ký tự nào

Mặc định chuỗi kết quả sẽ được điền đầy theo chuỗi định dạng từ phải sang trái. Ký tự ! phía trước chuỗi định dạng có ý nghĩa điền kết quả từ trái sang phải. Ký tự > (<) phía trước chuỗi định dạng buộc chuyển kết quả thành chữ thường (chữ hoa)

Ví dụ:

Format("abcde","@@@@@") = " abcde"

Format(Format(1234.567, "Currency"), "@@@@@@@@@@@@@") = " \$1,234.57"

Format("abcde", "!@@@@@") = "abcde "

Format("abcde", ">& & & &") = "A B C D E"

Format("6152127865", "&&&-&&&-&&&") = "615-212-7865"

### III. COLLECTION

#### 1. Giới thiệu

Là danh sách nhóm phần tử có quan hệ với nhau. Đối tượng Collection khác với mảng ở những điểm sau:

- Không cần khai báo trước số phần tử, có thể thêm, bớt phần tử bất kỳ lúc nào.
- Việc thêm bớt phần tử được thực hiện một cách tự động, người lập trình không phải quan tâm đến việc cấp phát vùng nhớ cho phần tử muốn thêm hoặc giải phóng vùng nhớ của phần tử bị xóa.

- Dữ liệu của mỗi phần tử chứa trong đối tượng Collection có thể tùy ý trong khi mảng chỉ có thể chứa các phần tử cùng kiểu dữ liệu.
- Ngoài giá trị chứa trong Collection, mỗi phần tử còn chứa kèm một giá trị khóa giúp truy tìm phần tử nhanh chóng ngoài chỉ số và tên.
- Khi một phần tử được thêm vào collection, người lập trình chỉ có thể đọc chứ không thể thay đổi giá trị của phần tử. Muốn thay đổi giá trị phần tử, phải xóa giá trị cũ rồi thêm giá trị mới.

Những đặc điểm trên cho thấy Collection có nhiều ưu điểm hơn so với mảng, tuy nhiên nó vẫn không thể thay thế được mảng trong Visual Basic vì tốc độ truy xuất trên Collection chậm hơn so với mảng. Một ví dụ điển hình như điền một mảng 10000 số nguyên kiểu long nhanh hơn 100 lần so với collection. Vì vậy việc lựa chọn mảng hay collection tùy thuộc vào yêu cầu của chương trình.

## 2. Các thao tác trên Collection

### a. Tạo Collection

Collection là một đối tượng. Khai báo collection như sau:

```
Dim <Tên> As Collection
Set <Tên> = New Collection
Hoặc
Dim <Tên> As New Collection
```

Ví dụ:

```
Dim EmployeeNames As Collection
Set EmployeeNames = New Collection
Hoặc
Dim EmployeeNames As New Collection
```

### b. Thêm giá trị vào Collection

Để thêm một giá trị, sử dụng phương thức Add, dạng như sau:

**Add** Item [, key][, before][, after]

Trong đó

Item	Giá trị thêm.
key	Chuỗi duy nhất đi kèm với mỗi giá trị.
Before, after	Vị trí mốc thêm.

Lưu ý:

- Thứ tự các phần tử trong Collection đánh bắt đầu từ 1,
- Khi mốc thêm là số, vị trí thêm được xác định theo chỉ số,
- Khi mốc thêm là chuỗi, vị trí thêm được xác định dựa theo thuộc tính key,
- Các tham số Before và After là tùy chọn nhưng không thể xuất hiện đồng thời.

Ví dụ 1: Thêm liên tiếp 2 giá trị vào danh sách nhân viên

```
Dim EmployeeNames As New Collection
EmployeeNames.Add "John Smith", "Marketing"
```

```
EmployeeNames.Add "Anne Lipton", "Sales"
```

Ví dụ 2: Thêm giá trị vào trước Anne Lipton trong danh sách trên

```
EmployeeNames.Add "Aves Lipton", "Excecutive", "Sales"
```

### c. Truy xuất giá trị trong Collection

Giá trị phần tử trong Collection được truy xuất thông qua thuộc tính Item bằng chỉ số hoặc key của phần tử.

Ví dụ: Truy xuất phần tử đầu tiên trong danh sách trên

```
Debug.Print EmployeeNames.Item("Sales")
```

```
Debug.Print EmployeeNames.Item(1)
```

Hàm ItemExists sau đây có thể được dùng để kiểm tra một giá trị có tồn tại trong collection hay không dựa theo khóa.

```
Function ItemExists(col As Collection, Key As String) As Boolean
```

```
Dim dummy As Variant
```

```
On Error Resume Next
```

```
dummy = col.Item(Key)
```

```
ItemExists = (Err <> 5)
```

```
End Function
```

### d. Xóa một giá trị trong Collection

Xóa một giá trị trong Collection bằng phương thức Remove :

**Remove** <Index>

Trong đó Index là vị trí xóa, có thể là giá trị số hoặc chuỗi

Ví dụ 1: Xóa phần tử đầu tiên

```
EmployeeNames.Remove 1
```

Ví dụ 2: Xóa phần tử có key là Sales

```
EmployeeNames.Remove "Sales"
```

Ví dụ 3: Xóa toàn bộ danh sách

```
Sub RemoveAllItems(col As Collection)
```

```
Do While col.Count
```

```
col.Remove 1
```

```
Loop
```

```
End Sub
```

Lưu ý:

Có thể xóa nhanh một danh sách bằng cách thực hiện như sau:

```
Set EmployeeNames = Nothing
```

Hoặc

```
Set EmployeeNames = New Collection
```

### e. Thay đổi giá trị một phần tử trong Collection

Không thể thay đổi giá trị phần tử trong collection, cách duy nhất có thể thực hiện là xóa nó rồi thêm giá trị cần sửa đổi. Chương trình con ReplaceItem sau cho phép thực hiện điều này.

*' INDEX có thể có giá trị số hoặc chuỗi.*

*Sub ReplaceItem(col As Collection, index As Variant, newValue As Variant)*

*' Xóa phần tử*

*col.Remove index*

*' Rồi thêm mới*

*If VarType(index) = vbString Then*

*col.Add newValue, index*

*Else*

*col.Add newValue, , index*

*End If*

*End Sub*

### f. Lặp trên Collection

- Lặp thông qua chỉ số phần tử:

Ví dụ: Nạp danh sách giá trị trong Collection vào ListBox

*Dim i As Long*

*For i = 1 To EmployeeNames.Count*

*List1.AddItem EmployeeNames(i)*

*Next*

- Lặp bằng phát biểu For Each...Next

Ví dụ: In danh sách giá trị trong Collection

*Dim var As Variant*

*For Each var in EmployeeNames*

*List1.AddItem var*

*Next*

Hoặc

*Dim cust As Customer*

*For Each cust In Customers*

*List1.AddItem cust.Name*

*Next*

Mảng đối tượng trên form thực chất được quản lý trong một Collection. Có thể sử dụng lệnh lặp For Each trên để thao tác nhanh trên mảng đối tượng.

Ví dụ:

Làm cho tất cả các TextBox (mảng txtFlds) trên Form ở trạng thái disable

*Sub Full\_Disable()*

*Dim oText as TextBox*

```

    For Each oText in txtFlds
        oText.Enabled = False
    Next
End Sub

```

### 3. Ví dụ khác

Giá trị chứa trong Collection có thể có độ phức tạp bất kỳ chứ không chỉ đơn giản là chứa giá trị như các ví dụ trên.

Ví dụ :

Tạo danh sách Collection trong đó mỗi phần tử chứa bao gồm Tên, Đơn vị và mức lương

```

Dim Employees As New Collection
' Mỗi phần tử gồm tên, đơn vị và mức lương
Employees.Add Array("John", "Marketing", 80000), "John"
Employees.Add Array("Anne", "Sales", 75000), "Anne"
Employees.Add Array("Robert", "Administration", 70000), "Robert"
...

```

Để in danh sách nhân viên, có thể viết như sau:

```

Dim var As Variant
For Each var in Employees
    Debug.Print var(0)
Next

```

In đơn vị của Anne

```
Debug.Print Employees("Anne")(1)
```

In mức lương của Robert

```
Debug.Print Employees("Robert")(2)
```

## Chương 6

# TextBox - ListBox-ComboBox

### I. TEXTBOX

Là đối tượng điều khiển dùng nhập một nội dung dạng text. Nội dung nhập có thể một dòng hoặc nhiều dòng.

#### 1. Các thuộc tính bổ sung

<b>Scrollbars</b>	Qui định các loại thanh cuộn được sử dụng trong textbox, chỉ dùng khi <b>Multiline=True</b> , có các giá trị: 0 - None. Không có thanh cuộn 1 - Horizontal. Chỉ có thanh cuộn ngang 2 - Vertical. Chỉ có thanh cuộn dọc 3 - Both. Có cả hai loại thanh cuộn
<b>SelStart</b>	Đọc hoặc đặt vị trí con trỏ (hoặc vị trí bắt đầu chọn) trong textbox
<b>SelLength</b>	Đọc hoặc đặt số ký tự được chọn trong textbox. Khi không chọn, SelLength=0
<b>SelText</b>	Lấy nội dung đang được chọn. Nếu không chọn, SelText="". Gán SelText sẽ chèn chuỗi mới vào textbox tại con trỏ Nếu có chọn. Gán SelText sẽ thay thế chuỗi chọn bởi nội dung mới
<b>HideSelection</b>	= <b>False</b> : Phần nội dung chọn vẫn được highlight khi textbox mất focus = <b>True</b> : Phần nội dung chọn không được highlight khi textbox mất focus
<b>PasswordChar</b>	Được sử dụng khi nhập mật khẩu. Các ký tự nhập vào luôn được trình bày bằng ký tự định nghĩa trong thuộc tính này.

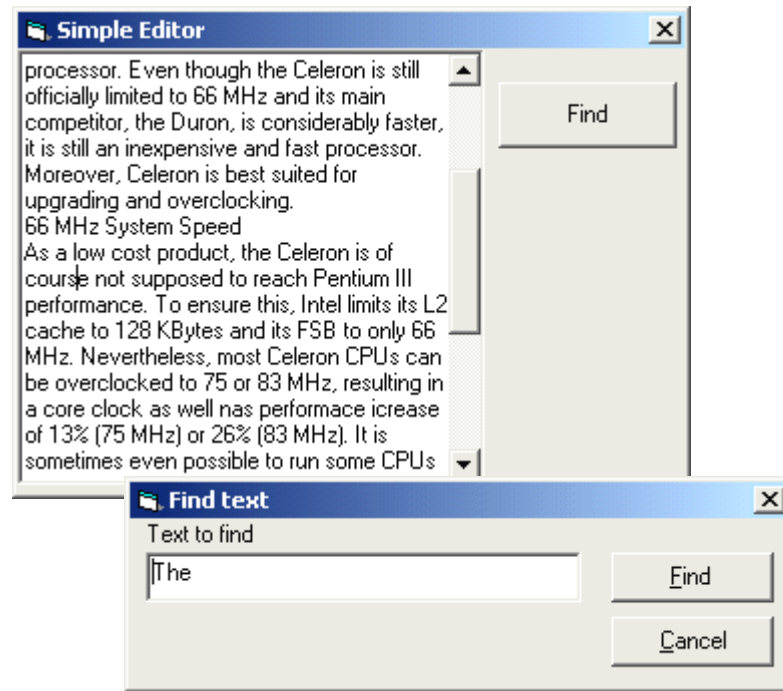
#### 2. Sự kiện

Sự kiện thường hay sử dụng đối với textbox là Change, sự kiện này xảy ra mỗi khi nội dung textbox thay đổi do người dùng nhập hay hiệu chỉnh thuộc tính Text

#### 3. Ví dụ

Thiết kế chương trình có dạng một Editor (trình soạn thảo văn bản) đơn giản. Định nghĩa chức năng tìm kiếm một nội dung trên Editor.

Form chính gồm 1 textbox làm editor và nút bấm Find để thực hiện chức năng tìm kiếm (Hình 6.1)



Hình 6.1: Các Form của chương trình soạn thảo văn bản

Trên form chính (đặt tên frmMain), có các đối tượng sau:

#### TextBox

Thuộc tính	Giá trị
Name	TxtEditor
Multiline	True
Scrollbar	None

#### CommandButton

Thuộc tính	Giá trị
Name	CmdFind
Caption	&Find
Enabled	False

Trên form tìm kiếm (đặt tên frmFind), có các đối tượng sau:

#### Label

Thuộc tính	Giá trị
Name	Label1
Caption	Find text

#### CommandButton

Thuộc tính	Giá trị
Name	CmdFind
Caption	&Find

#### CommandButton

Thuộc tính	Giá trị
Name	CmdCancel
Caption	&Cancel

Người chạy chương trình sẽ nhập nội dung vào textbox như một Editor. Khi chưa có nội dung, chức năng tìm kiếm sẽ không có tác dụng. Vì vậy định nghĩa thủ tục xử lý sự kiện Change: Khi textbox có thay đổi và nội dung khác rỗng thì nút lệnh sẽ được kích hoạt (Enabled)

```
Private Sub txtEditor_Change()
    If txtEditor.Text <> "" Then
        cmdFind.Enabled = True
    Else
        cmdFind.Enabled = False
    End If
End Sub
```

*End Sub*

Khi bấm nút Find, form thứ hai sẽ xuất hiện. Định nghĩa sự kiện Click cho nút bấm này:

```
Private Sub cmdFind_Click()
    frmFind.Show
End Sub
```

Trên form tìm kiếm, nếu người dùng nhập một nội dung tìm kiếm và bấm nút Find, chương trình sẽ thực hiện chức năng tìm kiếm. Vì vậy định nghĩa lệnh cho sự kiện Click

```
Private Sub cmdFind_Click()
    Dim p As Integer
    If txtFind.Text <> "" Then
        txtText = frmMain.txtEditor.Text
        p = InStr(1, txtText, txtFind.Text)           'Tìm vị trí xuất hiện đầu tiên
        If p <> 0 Then
            frmMain.txtEditor.SelStart = p - 1       'Giữ vị trí đầu tiên
            frmMain.txtEditor.SelLength = Len(txtFind.Text)
        Else
            MsgBox "Search text not found"
        End If
    End If
End Sub
```

Vì chuỗi cần tìm có thể xuất hiện nhiều lần trong editor. Định nghĩa lại Caption của Nút Find thành Find Next để mỗi lần bấm nút này thì chương trình sẽ tìm tiếp từ sau vị trí vừa tìm thấy và viết lại lệnh như sau:

```
Private Sub cmdFind_Click()
    Static p As Integer
    If txtFind.Text <> "" Then
        txtText = frmMain.txtEditor.Text
        p = InStr(p+1, txtText, txtFind.Text)       'Tìm từ sau vị trí vừa tìm thấy
        If p <> 0 Then
            frmMain.txtEditor.SelStart = p - 1       'Giữ vị trí đầu tiên
            frmMain.txtEditor.SelLength = Len(txtFind.Text) 'Highlight chuỗi tìm thấy
        Else
            MsgBox "Search text not found"
        End If
    End If
End Sub
```

Biến p được định nghĩa lại thành biến static để giữ lại vị trí vừa tìm thấy và chương trình sẽ tiếp tục tìm từ vị trí kế sau đó.

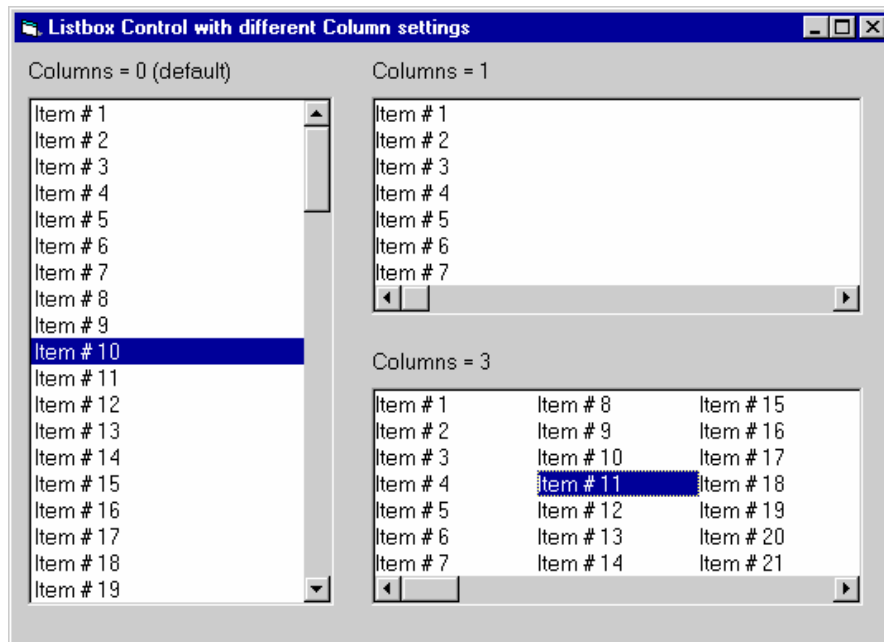
## 11. LISTBOX

Dùng liệt kê danh sách nhiều đối tượng và cho phép người dùng chọn lựa.



## 1. Các thuộc tính

**Columns** Số cột được dùng để thể hiện nội dung của listbox, mặc định =0 (Hình 6.2).



Hình 6.2: ListBox với thuộc tính Columns

**IntegralHeight** ListBox tự động điều chỉnh kích thước sao cho luôn thể hiện đầy đủ nội dung, có giá trị True/False.

**ListCount** Thuộc tính này đọc - Số phần tử có trong listbox.

**List** Danh sách nội dung (chuỗi) trình bày trong listbox

**ItemData** Danh sách chứa nội dung đi kèm với các phần tử chứa trong thuộc tính List. Danh sách này không được trình bày, chỉ phục vụ cho chương trình.

**ListIndex** Chỉ số của phần tử được chọn trong listbox - Phần tử đầu tiên có chỉ số 0. Khi không có phần tử nào được chọn, thuộc tính này có giá trị -1.

**Text** Chuỗi chứa nội dung mục được chọn trong danh sách. Tương đương với cách viết List(Listindex)

**Multiselect** Qui định chế độ chọn các phần tử trong ListBox

0 - None: Mỗi lúc chỉ chọn được một phần tử.

1 - Simple: Cho phép chọn nhiều phần tử bằng cách click.

2 - Extended: Chọn nhiều phần tử theo kiểu chọn trong Windows Explorer.

**SelCount** Cho biết số phần tử đang được chọn trong danh sách List

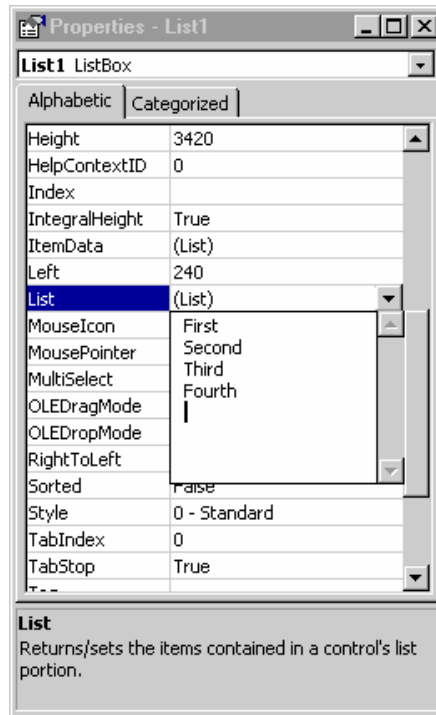
**Selected** Mảng tương ứng với danh sách list, xác định phần tử có được chọn trong listbox (True/False)

**Topindex** Chỉ số phần tử xuất hiện đầu tiên trong ListBox.

- Sorted** Qui định các phần tử trong danh sách được sắp thứ tự (True/false)
- NewIndex** Chỉ số phần tử vừa mới được thêm vào Listbox.

Lưu ý:

- Có thể nhập trực tiếp nội dung danh sách trong lúc thiết kế tại cửa sổ thuộc tính, xuống dòng bằng Ctrl+Enter (Hình 6.2)



Hình 6.2: Nhập giá trị cho thuộc tính List trong cửa sổ thuộc tính

- Để lấy nội dung của đối tượng đang được chọn. Sử dụng cách viết  
 $Dim St As String$   
 $St = Listbox.List(ListBox.ListIndex)$   
 Hoặc  
 $St = Listbox.Text$
- Để duyệt danh sách chứa trong danh sách list, sử dụng vòng lặp for  
 $For i = 0 to ListBox.ListCount - 1$   
 $Debug.Print ListBox.List(i)$   
 $Next i$

## 2. Các phương thức

**AddItem** <Item>[,<Index>]: thêm một phần tử vào danh sách

Thêm phần tử Item vào danh sách tại vị trí Index. Nếu không có tham số index, phần tử sẽ được thêm vào cuối danh sách.

Ví dụ 1:

```
ListBox.AddItem "Hoa"
ListBox.AddItem "Ngoc"
ListBox.AddItem "Hai"
ListBox.AddItem "Tuan"
```

Thêm 4 phần tử vào cuối danh sách .

Ví dụ 2:

Thêm 100 phần tử vào danh sách

```
For i = 0 to 99
    ListBox.AddItem "Item " & i
Next i
```

Phương thức AddItem thường được viết trong form\_load để khởi động giá trị cho ListBox.

Dữ liệu khởi động có thể chứa trong một mảng như ví dụ sau:

```
For i = LBound(MyData) To UBound(MyData)
    List1.AddItem MyData(i)
Next
```

Hoặc có thể khởi động một danh sách nhưng không cần định nghĩa mảng bằng cách sử dụng hàm choose như sau:

```
For i = 1 To 5
    List1.AddItem Choose(i, "America", "Europe", "Asia", "Africa",
        "Australia")
Next
```

**RemoveItem** <Index>: xóa một phần tử khỏi danh sách với Index là số thứ tự của phần tử cần xóa.

Ví dụ :

Xoá 50 phần tử đầu tiên của danh sách

```
For i = 0 to 49
    ListBox.RemoveItem i
Next i
```

**Clear:** Xóa toàn bộ danh sách List

Ví dụ :

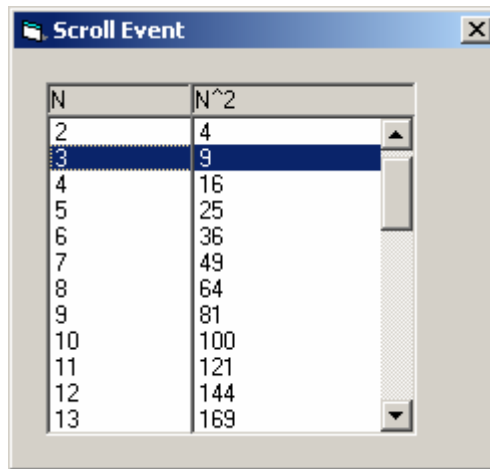
Xoá tất cả các phần tử trong danh sách

```
ListBox.Clear
```

### 3. Sự kiện

Sự kiện thường hay sử dụng đối với Listbox là Click, sự kiện này xảy ra mỗi khi click một phần tử trên Listbox hoặc thay đổi phần tử chọn bằng bàn phím (di chuyển vật sáng).

Một sự kiện cũng đôi khi được sử dụng đó là sự kiện scroll, sự kiện này xảy ra khi nội dung listbox cuộn. Ví dụ sau minh họa cách sử dụng sự kiện scroll kết hợp với thuộc tính TopIndex để đồng bộ hoạt động cuộn của 2 listbox.



Hình 6.3: Đồng bộ hoạt động của 2 listbox

Lệnh viết trên các Listbox như sau:

```
Private Sub lstN_Click()
```

```
    lstSquare.TopIndex = lstN.TopIndex
```

```
    lstSquare.ListIndex = lstN.ListIndex
```

```
End Sub
```

```
Private Sub lstSquare_Click()
```

```
    lstN.TopIndex = lstSquare.TopIndex
```

```
    lstN.ListIndex = lstSquare.ListIndex
```

```
End Sub
```

```
Private Sub lstN_MouseDown(Button As Integer, Shift As Integer, _
    X As Single, Y As Single)
```

```
    Call lstN_Click
```

```
End Sub
```

```
Private Sub lstSquare_MouseDown(Button As Integer, _
    Shift As Integer, X As Single, Y As Single)
```

```
    Call lstSquare_Click
```

```
End Sub
```

```
Private Sub lstN_MouseMove(Button As Integer, Shift As Integer, _
    X As Single, Y As Single)
```

```
    Call lstN_Click
```

```
End Sub
```

```
Private Sub lstSquare_MouseMove(Button As Integer, _
    Shift As Integer, X As Single, Y As Single)
```

```
    Call lstSquare_Click
```

```
End Sub
```

```
Private Sub lstN_Scroll()
```

```
    lstSquare.TopIndex = lstN.TopIndex
```

```
End Sub
```

```
Private Sub lstSquare_Scroll()
```

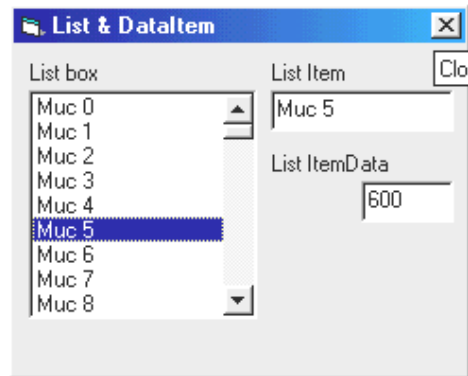
```
    lstN.TopIndex = lstSquare.TopIndex
```

```
End Sub
```

#### 4. Một số ví dụ

Ví dụ 1: Sử dụng thuộc tính ItemData .

Thiết kế chương trình với Form gồm Listbox, 2 Textbox trình bày phần tử được chọn và Itemdata của phần tử được chọn (Hình 6.4).



Hình 6.4: Form chương trình của ví dụ 1

- Tại thuộc tính List của listbox, nhập các nội dung “Muc 1”, “Muc 2”...”Muc 7” . Sử dụng Ctrl+Enter để xuống dòng.
- Tại thuộc tính ItemData của listbox, nhập các nội dung 100, 200...700 . Sử dụng Ctrl+Enter để xuống dòng.

Mỗi khi người dùng click một phần tử trên listbox. Nội dung của phần tử tương ứng xuất hiện trong textbox. Định nghĩa lệnh xử lý sự kiện Click như sau:

```
Private Sub List1_Click()
    Dim p As Integer
    Text1.Text = List1.Text
    p = List1.ListIndex
    Text2.Text = List1.ItemData(p)
End Sub
```

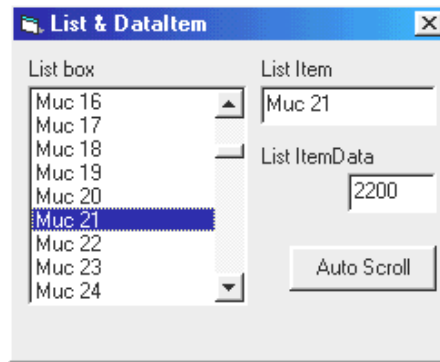
Nội dung listbox được khởi động trong formLoad như sau:

```
Private Sub Form_Load()
    For i = 0 To 99
        List1.AddItem "Muc " & i
        List1.ItemData(i) = (i + 1) * 100
    Next
End Sub
```

Ví dụ 2: Sử dụng thuộc tính TopIndex để tự động cuộn ListBox.

Sử dụng form của ví dụ 1, thêm nút “Auto Scroll” và viết lệnh như sau:

```
Private Sub Command1_Click()
    For i = 0 To 99
        List1.TopIndex = i
    Next
End Sub
```

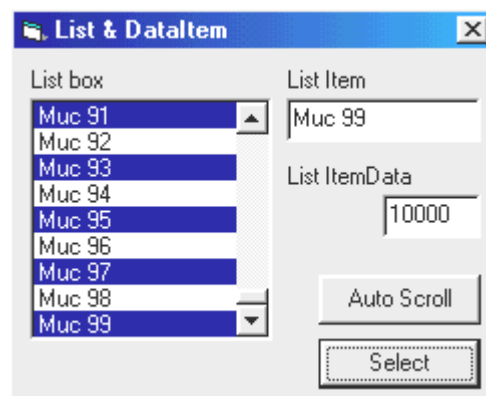


Hình 6.5: Form chương trình của ví dụ 2

**Ví dụ 3:** Sử dụng thuộc tính Selected để tự động chọn xen kẽ các phần tử trong listbox. (Hình 6.6).

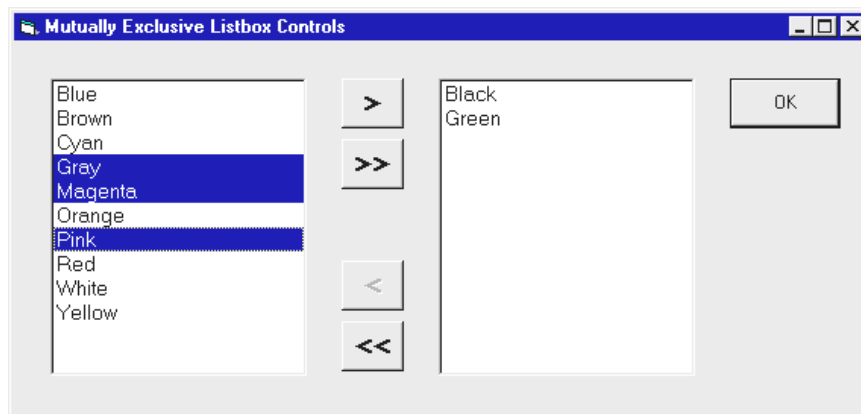
Sử dụng form của ví dụ 2, thêm nút “Select” và viết lệnh như sau:

```
Private Sub Command2_Click()
    For i = 0 To 99
        If i Mod 2 = 0 Then List1.Selected(i) = True
    Next
End Sub
```



Hình 6.6: Form của chương trình ví dụ 3

**Ví dụ 4:** Thiết kế form gồm 2 ListBox, dữ liệu được khởi động trước cho một listbox, viết lệnh trên các nút để di chuyển các phần tử chọn (có thể chọn nhiều) từ listbox này sang listbox khác. Việc di chuyển cũng có thể thực hiện khi nhấp đúp trong listbox. Giao diện của chương trình như hình 6.7.



Hình 6.7: Form chương trình ví dụ 4

Lệnh cho các nút viết như sau:

```

Private Sub cmdMove_Click()
    ' Di chuyển phần tử đang chọn từ listbox trái sang listbox phải
    If lstLeft.ListIndex >= 0 Then
        lstRight.AddItem lstLeft.Text
        lstLeft.RemoveItem lstLeft.ListIndex
    End If
End Sub

Private Sub cmdMoveAll_Click()
    ' Di chuyển mọi phần tử từ listbox trái sang listbox phải
    Do While lstLeft.ListCount
        lstRight.AddItem lstLeft.List(0)
        lstLeft.RemoveItem 0
    Loop
End Sub

Private Sub cmdBack_Click()
    ' Di chuyển phần tử đang chọn từ listbox phải sang listbox trái
    If lstRight.ListIndex >= 0 Then
        lstLeft.AddItem lstRight.Text
        lstRight.RemoveItem lstRight.ListIndex
    End If
End Sub

Private Sub cmdBackAll_Click()
    ' Di chuyển mọi phần tử từ listbox phải sang listbox trái
    Do While lstRight.ListCount
        lstLeft.AddItem lstRight.List(0)
        lstRight.RemoveItem 0
    Loop
End Sub

Private Sub lstLeft_DblClick()
    ' Mô phỏng tác động bấm phím
    cmdMove.Value = True
End Sub

Private Sub lstRight_DblClick()
    ' Mô phỏng tác động bấm phím
    cmdBack.Value = True
End Sub

```

### III. COMBOBOX

Là đối tượng điều khiển kết hợp giữa textbox và listbox. Trong combobox, người dùng có thể chọn một đối tượng có trước hoặc nhập mới một nội dung trong textbox phía trên.

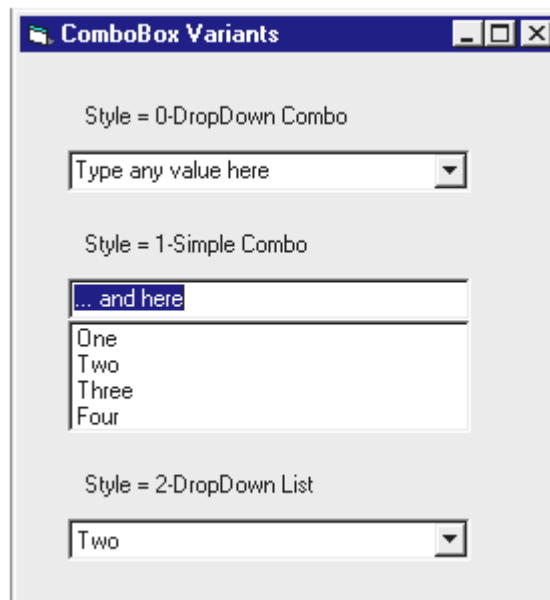
Combo box cũng có các thuộc tính của ListBox (trừ MultiSelect, Selected) , ngoài ra còn có thêm thuộc tính Style để quy định kiểu của combobox

**0 - Dropdown Combo** Combo chuẩn. Chỉ trình bày listbox khi click nút mũi tên bên phải.

**1 - Simple** Luôn thể hiện listbox bên dưới textbox. Khi nội dung nhập trong textbox giống với một nội dung trong listbox, nó sẽ tự động cuộn để thể hiện đầy đủ phần tử đó

**2 - Dropdown List** Không có Textbox. Chỉ trình bày listbox khi click nút mũi tên bên phải

Hình 6.8 trình bày các dạng ComboBox với các giá trị của thuộc tính Style.



**Hình 6.8:** Các dạng ComboBox

ComboBox cũng có các sự kiện giống như ListBox. Sự kiện thường viết lệnh là sự kiện Click.



#### IV. DRIVELISTBOX, DIRLISTBOX VÀ FILELISTBOX

Là các đối tượng điều khiển được xây dựng dựa trên ListBox và ComboBox, thường sử dụng kết hợp với nhau cho phép người sử dụng dễ dàng chọn lựa một ổ đĩa, một thư mục hoặc một tập tin trong máy.

##### 1. DriveListBox

ComboBox cho phép chọn một ổ đĩa trong các ổ đĩa của máy. Thuộc tính thường sử dụng là thuộc tính Drive cho biết ổ đĩa đang được chọn. Khi chọn một ổ đĩa mới, sự kiện Change xảy ra trên DriveListBox.

##### 2. DirListBox

Đối tượng trình bày cây thư mục của một ổ đĩa và cho phép người sử dụng chọn lựa. Thuộc tính thường sử dụng là Path cho biết đường dẫn thư mục đang chọn. Khi chọn một thư mục mới, sự kiện Change xảy ra trên DirListBox.

##### 3. FileListBox

Đối tượng trình bày các tập tin trong một thư mục và cho phép người sử dụng chọn lựa. Các thuộc tính thường sử dụng là :

Path	Đường dẫn thư mục
FileName	Tên đầy đủ tập tin đang chọn trong FileListBox
Pattern	Loại tập tin được xuất hiện trong FileListBox *.* - Mọi tập tin chứa trong thư mục *.txt; *.doc; *.rtf – 3 loại tập tin được xuất hiện

Khi chọn một tập tin, sự kiện Click xảy ra trên FileListBox.

DriveListBox, DirListBox và FileListBox thường sử dụng chung với nhau để chọn một tập tin trên đĩa. Khi đặt chúng lên form cần viết lệnh để đồng bộ hoạt động như sau:

Lệnh viết trên DriveListBox

```
Private Sub Drive1_Change()  
    ' Khi chọn ổ đĩa trên DriveListBox, gán cho thuộc tính Path  
    ' của DirListBox để làm thay đổi cây thư mục  
    Dir1.Path = Left$(Drive1.Drive, 1) & ":\"  
End Sub
```

Lệnh viết trên DirListBox

```
Private Sub Dir1_Change()  
    ' Gán đường dẫn chọn cho đối tượng FileListBox để làm thay  
    ' Đối nội dung FileListBox  
    File1.Path = Dir1.Path  
End Sub
```

Cuối cùng khi người sử dụng click tại tên một tập tin trong FileListBox, tên tập tin sẽ được xử lý được xác định như sau:

---

```
Filename = File1.Path  
If Right$(Filename, 1) <> "\" Then Filename = Filename & "\"  
Filename = Filename & File1.FileName
```

# Chương 7

## Scrollbar – Image – Timer

### I. SCROLLBAR

Là một đối tượng điều khiển dùng chọn một giá trị trong một khoảng cố định cho trước một cách trực quan.

Có hai loại thanh cuộn: thanh cuộn dọc (VScrollbar) và thanh cuộn ngang (HScrollbar)

#### 1. Các thuộc tính

**Min** Qui định giá trị cực tiểu của thanh cuộn  
**Max** Qui định giá trị cực đại của thanh cuộn  
**Value** Giá trị đang được chọn của thanh cuộn, phụ thuộc vào vị trí của con chạy trên thanh cuộn. Giá trị này có thể đọc từ vị trí của con chạy hoặc gán trong chương trình.

**SmallChange** Qui định khoảng tăng/giảm của giá trị chọn trên thanh cuộn mỗi khi bấm nút mũi tên ở hai đầu (default=1)

**LargeChange** Qui định khoảng tăng/giảm của giá trị chọn trên thanh cuộn mỗi khi click trên vùng chạy của con chạy (default=1)

Khi một thanh cuộn được tạo ra trên form, luôn luôn cần định nghĩa các giá trị min và max

#### 2. Sự kiện

**Change** Sự kiện xảy ra sau khi con chạy thay đổi vị trí hoặc thuộc tính value thay đổi

**Scroll** Sự kiện xảy ra khi con chạy thay đổi vị trí hoặc thuộc tính value thay đổi

#### 3. Ví dụ:

Thiết kế form chọn màu bằng cách phối hợp 3 màu cơ bản RGB. Giá trị của các thành phần màu sẽ được chọn bằng thanh cuộn. Màu chọn được thể hiện bằng đối tượng Shape.



Hình 7.1: Thiết kế form chọn màu

Các đối tượng trên form được chọn như sau:

Đối tượng Shape		Các Textbox (Mảng)	
Thuộc tính	Giá trị	Thuộc tính	Giá trị
Name	Shape1	Name	txtColor
FillStyle	0 - Solid	Đối tượng Hscrollbar (mảng)	
Shape	0 - Rectangle	Thuộc tính	Value
		Name	hsbColor
		Min	0
		Max	255
		Value	100

Viết lệnh cho sự kiện Form\_Load như sau:

```
Private Sub Form_Load()
    Shape1.FillColor = RGB(hsbColor(0).Value, hsbColor(1).Value, hsbColor(2).Value)
End sub
```

Viết lệnh cho sự kiện Change của các thanh cuộn như sau:

```
Private Sub hsbColor_Change(Index As Integer)
    TxtColor(Index).Text = hsbColor(Index).Value
    ChangeFillColor
End Sub
```

Định nghĩa sub ChangeFillColor trong phần General như sau:

```
Sub ChangeFillColor()
    Shape1.FillColor = RGB(hsbColor(0).Value, hsbColor(1).Value, hsbColor(2).Value)
End sub
```

Sửa định nghĩa Form\_Load thành

```
Private Sub Form_Load()
    ChangeFillColor
End sub
```

## II. IMAGE

Sử dụng để đặt một hình ảnh lên form.

### Các thuộc tính

**Picture** Giữ hình cần trình bày, thường nhận giá trị trả về từ hàm LoadPicture  
**BorderStyle** Kiểu khung (0-None, 1-Fixed Single)  
**Stretch** Hình tự động co giãn để nằm gọn trong khung đã qui định (True/False)  
Hàm LoadPicture(PathName) nạp các tập tin ảnh và chứa vào thuộc tính Picture của đối tượng Image. Các loại tập tin ảnh có thể nạp là : .BMP, .GIF, .JPG, .WMF, .CUR, .ICO

Ví dụ: Nạp tập tin ảnh từ đĩa

```
Image1.Picture = LoadPicture("C:\WINDOWS\SETUP.BMP")
```

### III. TIMER

Đối tượng dùng xử lý các sự kiện thời gian. Lệnh viết trong đối tượng timer sẽ tự động thực hiện sau một khoảng thời gian xác định.

#### 1. Thuộc tính

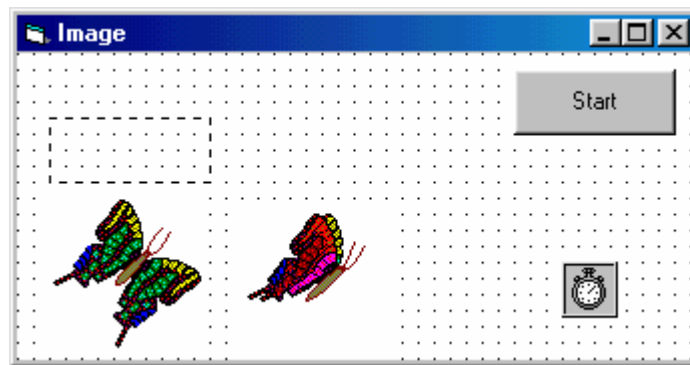
**Interval** Qui định khoảng thời gian xảy ra sự kiện. Tính bằng ms. Giá trị 0 làm Timer ngưng hoạt động)

#### 2. Sự kiện

Sự kiện được dùng để định nghĩa mã lệnh là sự kiện Timer. Sự kiện này xảy ra mỗi khi timer đếm đủ khoảng thời gian qui định trong thuộc tính Interval

#### 3. Ví dụ

Sử dụng đối tượng Image và Timer để tạo đối tượng hoạt động trong chương trình



Hình 7.1: Form chương trình tạo đối tượng chuyển động

Image1		Image3	
Thuộc tính	Giá trị	Thuộc tính	Giá trị
Name	Image1	Name	Image3
Picture	Buttfly1		
Visible	False		
Image2		Timer1	
Thuộc tính	Giá trị	Thuộc tính	Giá trị
Name	Image2	Name	Timer1
Picture	Buttfly2	Interval	150
Visible	False	Enabled	False
		CommandButton	
		Thuộc tính	Giá trị
		Name	CmdSw
		Caption	Start

Viết lệnh cho sự kiện form\_load như sau:

```
Private Sub Form_Load()
    Image3.Picture = Image1.Picture
End Sub
```

Viết lệnh cho sự kiện thời gian của Timer:

```
Private Sub Timer1_Timer()  
    Static T As Integer  
    If T = 0 Then  
        Image3.Picture = Image1.Picture  
        T = 1  
    Else  
        Image3.Picture = Image2.Picture  
        T = 0  
    End If  
End Sub
```

Viết lệnh cho CommandButton:

```
Private Sub CmdSw_Click()  
    If CmdSw.Caption="Start" Then  
        Timer1.Interval=150  
        CmdSw.Caption = "Stop"  
    Else  
        Timer1.Interval=0  
        CmdSw.Caption = "Start"  
    End If  
End Sub
```

# Chương 8

## Truy xuất dữ liệu

### I. TRUY XUẤT DỮ LIỆU BẰNG ĐỐI TƯỢNG ĐK CÓ KẾT NỐI CSDL

#### 1. DataControl

Data control là đối tượng điều khiển cho phép tự động hoá quá trình kết nối và truy xuất dữ liệu từ các tập tin cơ sở dữ liệu Access, Foxpro, Excel, Text...

Data control cho phép duyệt, thao tác trên các vùng của cơ sở dữ liệu thông qua các đối tượng điều khiển kết nối cơ sở dữ liệu (Bound-controls) mà không cần viết lệnh.

#### 2. Các thuộc tính

<b>Connect</b>	Loại cơ sở dữ liệu kết nối (Access, Dbase, Excel).
<b>DatabaseName</b>	Chuỗi đường dẫn tên tập tin cơ sở dữ liệu.
<b>Recordsource</b>	Tên tập tin dữ liệu (Tên bảng nếu là cơ sở dữ liệu Access).
<b>Recordset</b>	Thuộc tính dùng truy xuất các mẫu tin trong cơ sở dữ liệu đã được kết nối bằng Datacontrol.
<b>Recordsettype</b>	Loại recordset, có các giá trị sau: <ul style="list-style-type: none"> <li>– dbOpenTable: Sử dụng khi mở 1 table. Có thể thêm, xoá, cập nhật các mẫu tin.</li> <li>– dbOpenDynaset: Sử dụng khi mở 1 table hay 1 query, có thể gồm nhiều vùng từ nhiều tập tin. Cho phép thể thêm, xoá, cập nhật các mẫu tin.</li> <li>– DbOpenSnapshot: Sử dụng khi mở 1 table hay 1 query, có thể gồm nhiều vùng từ nhiều tập tin, được dùng để duyệt hay tạo report, không thể thay đổi.</li> </ul>
<b>ReadOnly</b>	(True/False) Cơ sở dữ liệu có thể cập nhật được hay không
<b>BOFAction</b>	Thuộc tính định nghĩa hoạt động của Datacontrol khi di chuyển đến mẫu tin đầu tiên, có các giá trị sau: <ul style="list-style-type: none"> <li>- 0: MoveFirst - Di chuyển về mẫu tin đầu tiên,</li> <li>- 1: BOF - Ở vị trí đầu tiên</li> </ul>
<b>EOFAction</b>	Thuộc tính định nghĩa hoạt động của Datacontrol khi di chuyển đến mẫu tin cuối cùng, có các giá trị sau: <ul style="list-style-type: none"> <li>– vbEOFActionMoveLast = 0: Khi di chuyển đến hết tập tin trên recordset tự động nhảy đến phần tử cuối cùng,</li> <li>– vbEOFActionEOF = 1: Khi di chuyển đến hết tập tin trên recordset, disable nút MoveNext trên Datacontrol,</li> <li>– vbEOFActionAddnew = 2: Khi di chuyển đến hết tập tin trên recordset, tự động kiểm tra dữ liệu (Validate) và thêm mẫu tin mới vào Recordset</li> </ul>

#### 3. Các đối tượng điều khiển có kết nối cơ sở dữ liệu (Bound-controls)

Đối tượng điều khiển có kết nối cơ sở dữ liệu là các đối tượng điều khiển có thêm các thuộc tính (Datasource, DataField) cho phép kết nối với một field của bảng dữ liệu để trình bày hoặc cập nhật nội dung của field.

Trong Visual Basic, các đối tượng điều khiển có kết nối cơ sở dữ liệu gồm:

#### Các đối tượng chuẩn (Intrinsic)

- Check box
- Image
- Label
- Picture box
- Text box
- List box
- Combo box

#### Các đối tượng mở rộng (Extended)

- Data-bound list box
- Data-bound combo box
- Data-Bound Grid (DBGrid)

Các thuộc tính được sử dụng khi truy xuất cơ sở dữ liệu

Thuộc tính	Ý nghĩa
DataChanged	True/False: Cho biết nội dung field có thay đổi hay không
DataField	Tên vùng dữ liệu.
Datasource	Tên nguồn dữ liệu, thường là tên của Datacontrol.

Ví dụ:

Thiết kế form duyệt bảng Employee chứa trong cơ sở dữ liệu Access NWIND.mdb.

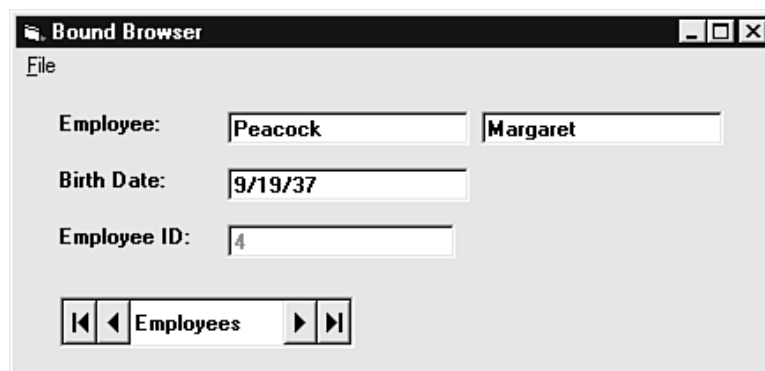
Các Field muốn trình bày dữ liệu gồm:

Họ tên : FirstName + LastName

Ngày sinh: Birth Date

Mã số nhân viên: Employee ID

Hình 8.1: Duyệt bảng Employees trong cơ sở dữ liệu NWIND.MDB





Đối tượng, thuộc tính và giá trị thuộc tính của các đối tượng được tóm tắt trong bảng sau

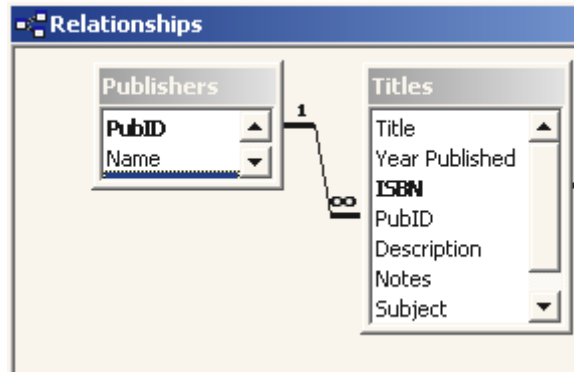
Đối tượng	Thuộc tính	Giá trị	Ý nghĩa
Form	Name	FrmData	
	Caption	Bound Browser	
Data	Name	datEmployees	Tên DataControl
	Caption	Employees	
	DatabaseName	C:\Program Files\VB6\NWIND.MDB	Cơ sở dữ liệu NWIND.MDB
	RecordSource	Employees	Bảng Employees
TextBox	Name	TxtLastName	
	DataField	LastName	Vùng họ (LastName)
	DataSource	DatEmployees	
TextBox	Name	TxtFirstName	
	DataField	FirstName	Vùng tên (FirstName)
	DataSource	DatEmployees	
TextBox	Name	TxtBirthDate	
	DataField	BirthDate	Vùng ngày sinh (BirthDate)
	DataSource	DatEmployees	
TextBox	Name	TxtEmployeeId	
	DataField	EmployeeID	Vùng mã nhân viên
	DataSource	DatEmployees	
	Enabled	False	
Label	Name	Label1	
	Caption	Employee:	
Label	Name	Label2	
	Caption	Birth Date:	
Label	Name	Label3	
	Caption	Employee ID:	

Bấm phím F5 để chạy chương trình, sử dụng các phím mũi tên trên DataControl để duyệt xem các mẫu tin.

#### 4. Sử dụng data-bound listbox (DBList) và combobox (DBCombo)

Giả sử có 2 bảng dữ liệu cho như sau:

- **Publishers** chứa thông tin về các nhà xuất bản sách, thông tin chứa trong bảng gồm các field PubID (Mã NXB) và Name (Tên NXB),
- **Titles** chứa thông tin về mỗi quyển sách, thông tin chứa trong bảng gồm các field PubID (Mã NXB), AU\_ID (Mã tác giả), Title (Tựa sách), Year\_Published (Năm XB) và ISBN.



Hình 8.2: Quan hệ giữa các bảng Publishers và Titles

Người ta muốn xem các thông tin về một quyển sách gồm tựa (Title), năm XB (Year\_Published), ISBN và tên NXB (Name) với vùng Name có thể chọn để cập nhật từ danh sách các nhà xuất bản. Form dữ liệu có dạng như hình 8.3.

Hình 8.3: Xem thông tin sách có thể điều chỉnh tên nhà xuất bản

Databound ListBox (DBList) và Databound ComboBox (DBCombo) là các đối tượng được thiết kế để thực hiện chức năng này, chúng là các ListBox và ComboBox có thêm một số thuộc tính đặc biệt để kết nối với cơ sở dữ liệu. Hai đối tượng điều khiển này chứa trong Microsoft Databound List Control. Để sử dụng, nạp lên Toolbox bằng cách bấm tổ hợp phím CTRL+T và chọn trong hộp thoại Components.

### Các thuộc tính kết nối cơ sở dữ liệu của DBList và DBCombo

- Datasource** Tên nguồn dữ liệu, thường là tên của Datacontrol
- DataField** Tên vùng dữ liệu chứa trong Recordset đã mở bằng Datacontrol. Khi giá trị mới được chọn, nội dung vùng tương ứng trong CSDL sẽ được tự động cập nhật khi di chuyển sang mẫu tin mới.

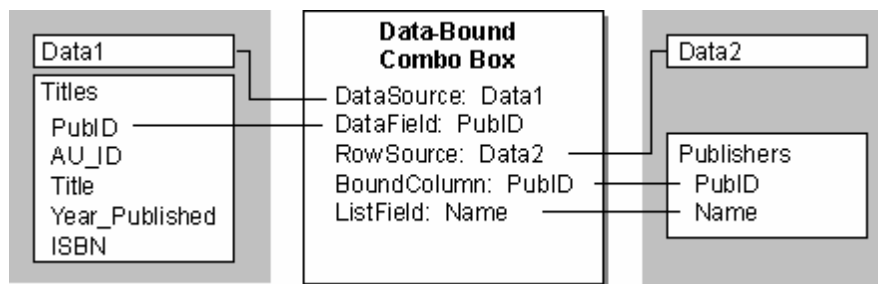
<b>Rowsource</b>	Tên Datacontrol kết nối với table có nội dung được sử dụng để điền vào DBList/DBCombo
<b>BoundColumn</b>	Tên vùng thuộc recordset được chỉ định bởi Rowsource được sử dụng để cập nhật Recordset trong Datasource
<b>ListField</b>	Tên vùng thuộc Recordset trong Rowsource được sử dụng để điền vào DBList/DBCombo

Lưu ý:

- Tên các vùng DataField và BoundColumn nên giống nhau trong 2 cơ sở dữ liệu. Thường đó là các vùng có thiết lập quan hệ (Relationship).
- Trong trường hợp chỉ sử dụng 1 Datacontrol các thuộc tính Datasource và Rowsource sẽ có cùng giá trị. Khi đó BoundColumn và DataField cùng có giá trị là tên vùng cần cập nhật trong cơ sở dữ liệu.

Ví dụ

Muốn tạo form duyệt bảng Titles với mã NXB (PubID) và Name được lấy từ bảng Publishers. Giá trị các vùng gán được tóm tắt theo hình 8.4 .



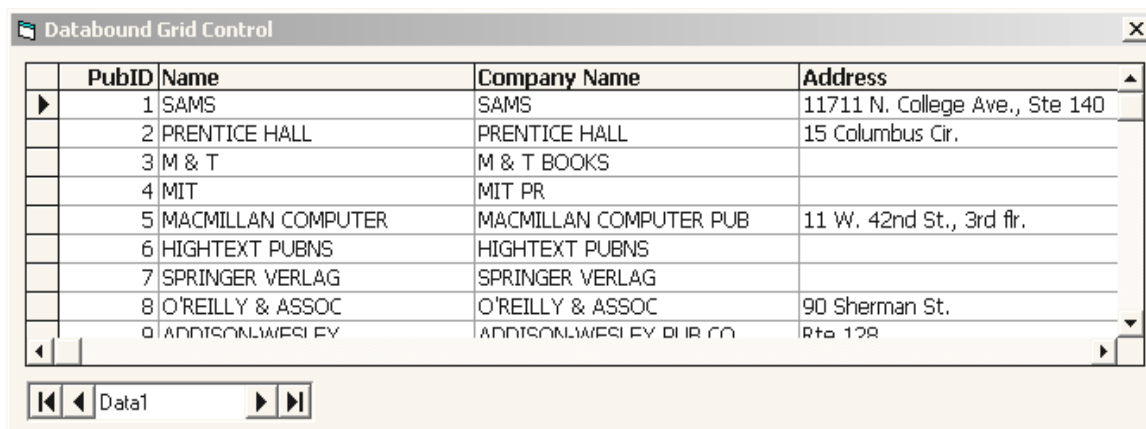
**Hình 8.4:** Sử dụng DBCombo và DBList để duyệt bảng Titles và Publishers  
Các đối tượng điều khiển sử dụng và giá trị thuộc tính được tóm tắt theo bảng sau:

Đối tượng	Thuộc tính	Giá trị
Data1	Connect	Access
	DatabaseName	Biblio.mdb
	Recordsettype	0-dbOpenTable
	RecordSource	Titles
Data2	Connect	Access
	DatabaseName	Biblio.mdb
	Recordsettype	2-dbOpenSnapshot
	RecordSource	SELECT PubID, Name FROM Publishers ORDER BY PubID
TextBox	Name	TxtTitle
	Datasource	Data1
	DataField	Title
TextBox	Name	TxtYear
	Datasource	Data1

	DataField	Year Published
TextBox	Name	TxtISBN
	Datasource	Data1
DBCombo	Name	DBCombo1
	Datasource	Data1
	DataField	PubID
	Rowsource	Data2
	BoundColumn	PubID
	ListField	Name

### 5. Sử dụng Databound Grid Control (DBGrid)

Databound Grid control là đối tượng điều khiển giúp trình bày nội dung bảng dữ liệu dưới dạng bảng (dòng, cột), nó có các thuộc tính kiểm soát thao tác trên bảng dữ liệu như sửa chữa (edit), thêm mới (addnew) hoặc xóa một mẫu tin trong bảng. Hình dưới trình bày một form sử dụng DBGrid để thao tác trên bảng Publishers (Nhà xuất bản) của cơ sở dữ liệu BIBLIO.MDB.



Hình 8.5: Trình bày dữ liệu bằng DBGrid

Phần này trình bày các bước sử dụng DBGrid để thao tác trên bảng dữ liệu kết hợp với Data control mà không cần viết lệnh.

**Bước 1:** Đặt Datacontrol lên form, điều chỉnh các thuộc tính:

Database Name: Tên cơ sở dữ liệu,

Recordsource: Tên bảng dữ liệu ,

**Bước 2:** Nạp đối DBGrid lên Toolbox

Bấm tổ hợp phím Ctrl+T để mở hộp thoại Components,

Check tại đối tượng Microsoft Databound Grid Control 5.0

Bấm nút OK

Biểu tượng DBGrid  xuất hiện trên Toolbox.

**Bước 3:** Đặt DBGrid lên form

Click tại biểu tượng DBGrid trên Toolbox,

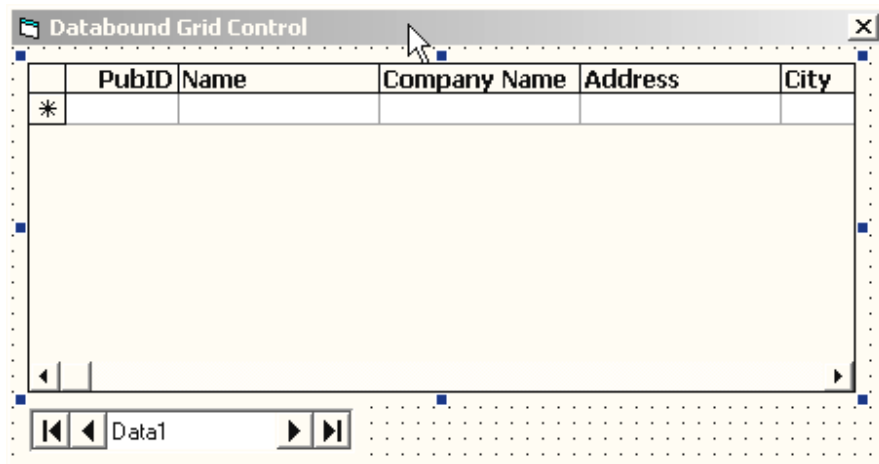
Click , kéo để xác định kích thước DBGrid trên form.

**Bước 4:** Liên kết DBGrid với DataControl

Điều chỉnh thuộc tính Datasource của DBGrid thành tên của Datacontrol,

Bấm phím phải trên DBGrid rồi chọn Retrieve fields

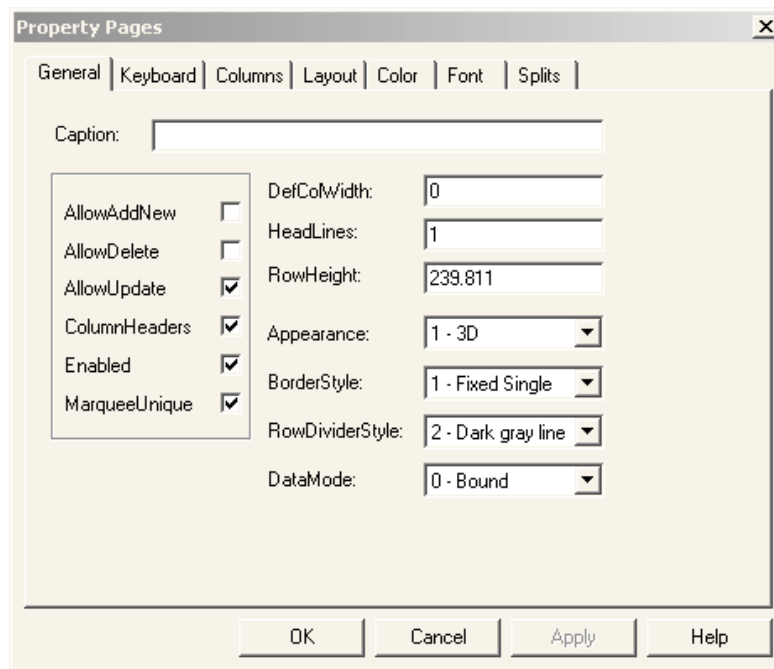
Tên các vùng của bảng dữ liệu sẽ xuất hiện trên dòng tiêu đề của DbGrid.



**Hình 8.6:** DBGrid sau khi nạp các vùng của bảng dữ liệu

**Bước 5:** Điều chỉnh các thuộc tính của DBGrid

Bấm phím phải trên DBGrid, chọn properties để mở Property Pages,



**Hình 8.7:** Thẻ General

**Thẻ General**

Ý nghĩa các tùy chọn được tóm tắt trong bảng sau:

**AllowAddnew** Cho phép nhập thêm mẫu tin mới trên DBGrid

**AllowDelete** Cho phép xóa mẫu tin ngay trên DBGrid bằng cách chọn mẫu tin rồi bấm phím Delete

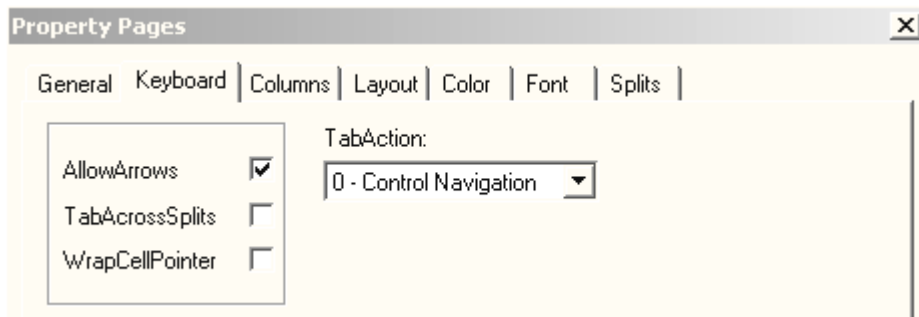
**AllowUpdate** Cho phép thay đổi nội dung các mẫu tin ngay trên DBGrid

**ColumnHeaders** Có tiêu đề hay không

**Headlines** Kích thước dòng tiêu đề tính theo dòng

- RowHeight** Chiều cao dòng  
**RowdividerStyle** Kiểu đường phân cách các mẫu tin trong DBGrid  
**Datamode** DBGrid có kết nối với bảng dữ liệu hay không  
**Thẻ Keyboard**

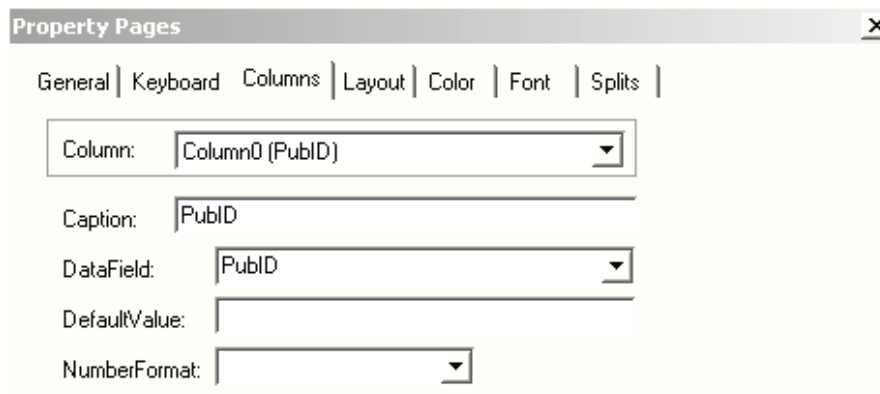
Định nghĩa cách sử dụng phím Tab trong DBGrid



Hình 8.8: thẻ Keyboard

### Thẻ Columns

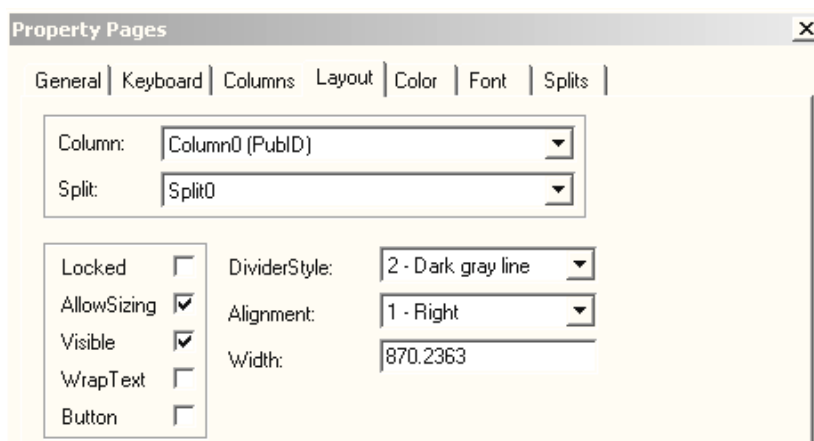
Thứ tự trình bày các Field trên các cột của DBGrid và định dạng giá trị trên mỗi cột.



Hình 8.9: Thẻ Columns

### Thẻ Layout

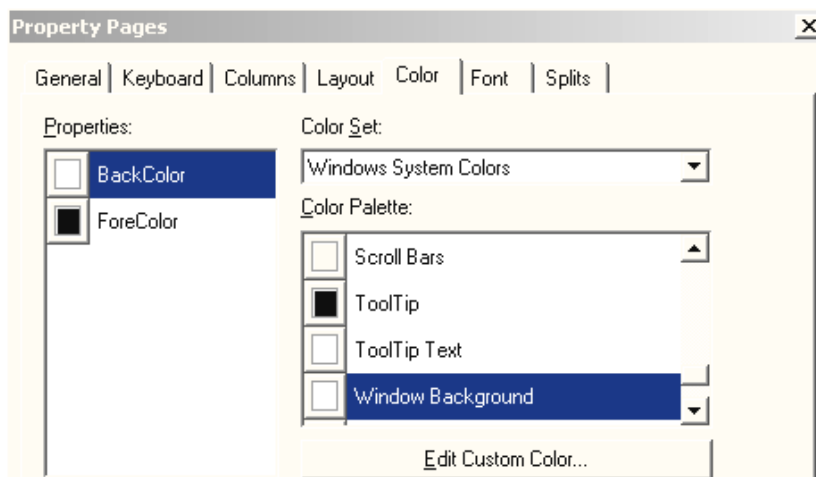
Độ rộng cột và chế độ căn nội dung trong cột của DBGrid, kiểu của vạch phân cách cột



Hình 8.10: Thẻ Layout

## Thẻ Color

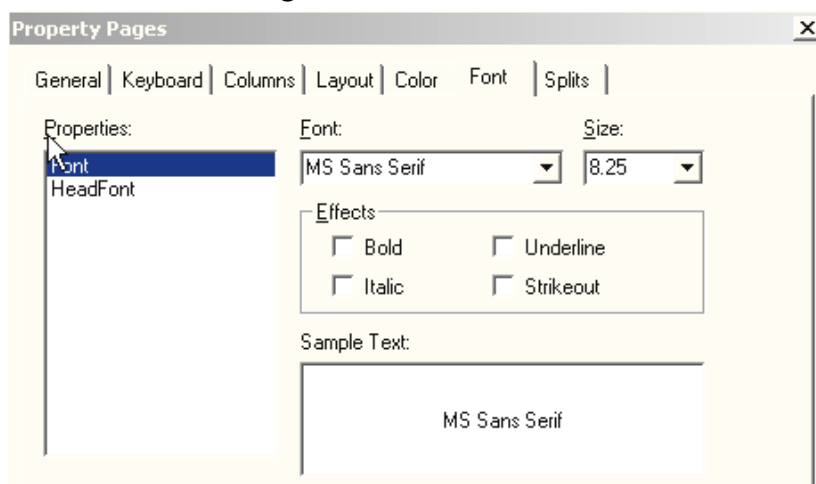
Chọn màu chữ, màu nền trong DBGrid



Hình 8.11: Thẻ Color

## Thẻ Font

Chọn kiểu chữ tiêu đề và nội dung DBGrid



Hình 8.12: Thẻ Font

Ví dụ :

Sử dụng DBGrid để duyệt bảng Title trong cơ sở dữ liệu Biblio.mdb. Các bước thực hiện như sau:

**Bước 1:** Đặt DataControl lên form, điều chỉnh các thuộc tính theo bảng sau:

Thuộc tính	Giá trị
Name	Data1
Connect	Access
DatabaseName	Biblio.mdb
Recordsettype	1-dbOpenDynaset
RecordSource	Titles

**Bước 2:** Đặt DBGrid lên form, điều chỉnh các thuộc tính theo bảng sau:

Thuộc tính	Giá trị
DataSource	Data1

---

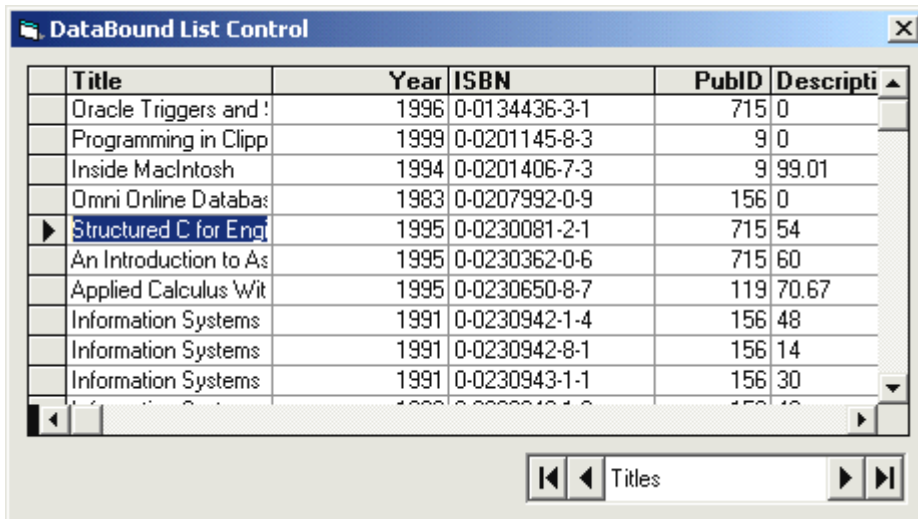
AllowAddNew	False
AllowDelete	False
AllowUpdate	False

---

**Bước 3:** Nạp các vùng của bảng dữ liệu Title lên DBGrid

Right-Click, chọn Retrieve Fields để lấy thông tin của và vùng thành các cột trong DBGrid.

Bấm F5 chạy chương trình. Di chuyển qua các mẫu tin trong DBGrid bằng các phím mũi tên hoặc sử dụng Datacontrol để di chuyển. Chương trình có dạng như hình 8.12.



Hình 8.13: Sử dụng DBGrid duyệt bảng Title

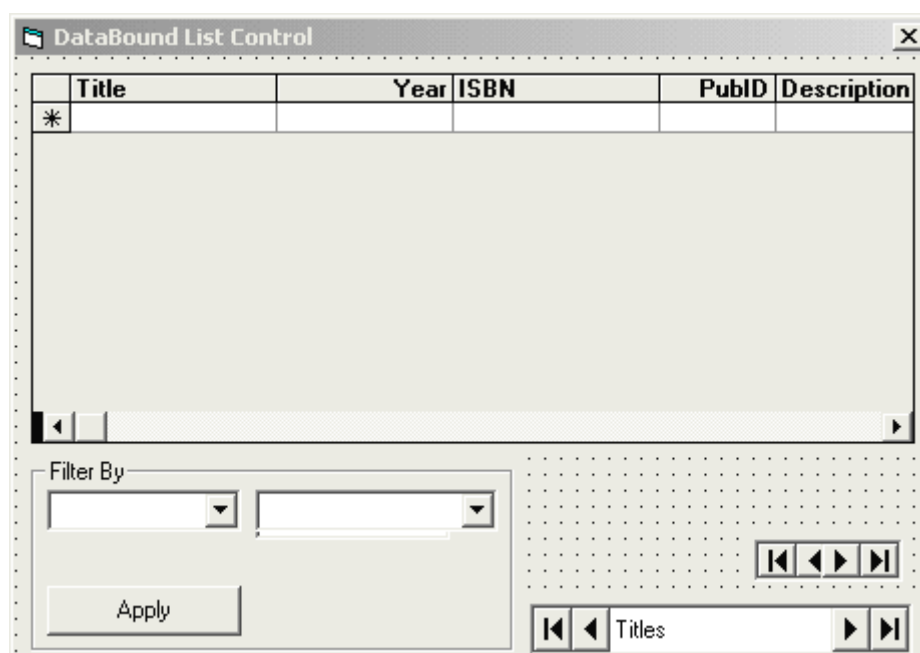
Thêm DBCombo chọn nhà xuất bản (Publisher) để người dùng có thể chọn chỉ xem các tựa sách theo một nhà xuất bản. Thêm TextBox chọn năm xuất bản để người dùng có thể chọn xem các tựa sách xuất bản trong một năm nào đó.

Đặt thêm các đối tượng lên form và qui định các thuộc tính như cho trong bảng sau:

Đối tượng	Thuộc tính	Giá trị
Data2	Connect	Access
	DatabaseName	Biblio.mdb
	Recordsettype	2-dbOpenSnapshot
	RecordSource	SELECT PubID, Name FROM Publishers ORDER BY PubID
TextBox	Name	TxtFlt
ComboBox	Name	cbFlt
	List	Year Published Publisher
DBCombo	Name	cbPublisher
	Datasource	Data2
	DataField	PubID
	Rowsource	Data2
	BoundColumn	PubID



	ListField	Name
Button	Name	CmdApply
	Caption	Apply



Hình 8.14: Thêm đối tượng để thực hiện chức năng lọc

Viết lệnh cho sự kiện click trên ComboBox

```
Private Sub CbFlt_Click()
    Select Case CbFlt.ListIndex
        Case 0
            FltMode = 0
            txtFlt.Visible = True
            cbPublisher.Visible = False
        Case 1
            FltMode = 1
            txtFlt.Visible = False
            cbPublisher.Visible = True
    End Select
End Sub
```

Trong đó biến FltMode là biến chung được dùng để chọn chế độ xem theo năm xuất bản hay nhà xuất bản

Viết lệnh cho nút lệnh Apply

```
Private Sub CmdApply_Click()
    QryStr = "SELECT * FROM Titles WHERE "
    Select Case FltMode
        Case 0
            QryStr = QryStr & "[Year Published]= " & txtFlt.Text
        Case 1
```

```

    QryStr = QryStr & "PubID=" & cbPublisher.BoundText
End Select
Data1.RecordSource = QryStr
Data1.Refresh
End Sub

```

Viết lệnh cho form\_load qui định phần tử đầu tiên được chọn trong ComboBox

```

Private Sub Form_Load()
    CbFlt.ListIndex = 0
End Sub

```

Bấm F5 chạy chương trình, thay đổi giá trị chọn năm xuất bản rồi bấm nút Apply để xem tác dụng.

## II. TRUY XUẤT DỮ LIỆU THÔNG QUA DATA ACCESS OBJECT

Một cách khác để truy xuất dữ liệu là sử dụng Data Access Object (DAO). Đối tượng này phép truy xuất dữ liệu bằng chương trình.

### 1. Các thao tác cơ bản

#### a. Mở Cơ sở dữ liệu

Lệnh *Set Db = OpenDatabase(DbName, Options, read-only)*

Trong đó

Db	Biến kiểu Database đại diện cho cơ sở dữ liệu cần truy xuất
DbName	Chuỗi tên tập tin dữ liệu
Options	Tùy chọn khi mở cơ sở dữ liệu. Có giá trị True khi mở ở chế độ Exclusive (chỉ có một người truy xuất), có giá trị False khi mở ở chế độ Shared (nhiều người truy xuất)
read-only	(True/False) Chế độ mở cơ sở dữ liệu

Ví dụ: Mở cơ sở dữ liệu BIBLIO.MDB

```

Private Sub Form_Load()
    AppFolder = App.Path
    Dim db As Database
    Set db = OpenDatabase(AppFolder & "\BIBLIO.MDB")
    ...
End sub

```

#### b. Mở Recordset

Lệnh *Set rs = db.OpenRecordset(Source)*

Trong đó

rs	Biến kiểu Recordset
Source	Tên bảng dữ liệu hoặc chuỗi câu lệnh SQL

Ví dụ:

```

Dim Db As Database, rs As Recordset
Set Db = OpenDatabase("BIBLIO.MDB")
Set rs = Db.OpenRecordset("AUTHORS")

```

Hoặc mở Recordset từ chuỗi SQL

```
Set Db = OpenDatabase("BIBLIO.MDB")
```

```
Set Rs = Db.OpenRecordset("SELECT * FROM AUTHORS")
```

Có thể liên kết một recordset đã mở với Datacontrol như sau:

```
Set Data1.Recordset = Rs
```

## 2. Các thuộc tính của Recordset

<b>AbsolutePosition</b>	Vị trí tuyệt đối của mẫu tin trong Recordset, mẫu tin đầu tiên có thứ tự 0
<b>Bookmark</b>	Vị trí của mẫu tin hiện thời (kiểu Variant)
<b>EOF</b> <b>(True/False)</b>	Thuộc tính cho biết mẫu tin hiện hành có ở sau mẫu tin cuối cùng hay không, cho giá trị True khi không có mẫu tin nào.
<b>BOF</b> <b>(True/False)</b>	Thuộc tính cho biết mẫu tin hiện hành có ở trước mẫu tin đầu tiên không, cho giá trị True khi không có mẫu tin nào.
<b>NoMatch</b> <b>(True/False)</b>	Cho kết quả tìm kiếm sau lệnh Seek hoặc Find
<b>RecordCount</b>	Số mẫu tin trong Recordset
<b>Fields</b>	Mảng các vùng trong Recordset
<b>Index</b>	Chọn tên vùng Index
<b>Sort</b>	Chọn tên vùng sort

## 3. Các thao tác trên Recordset

### a. Lấy giá trị của 1 vùng :

Có 3 cách viết

Cách 1: *Recordset.Fields(<Chỉ số vùng>)*

Ví dụ:

```
mYear = rs.Fields(5)
```

Cách 2: *Recordset.Fields(<Chuỗi tên vùng>)*

Ví dụ:

```
mYear = rs.Fields("Year Published")
```

Cách 3: *Recordset![Chuỗi tên vùng]*

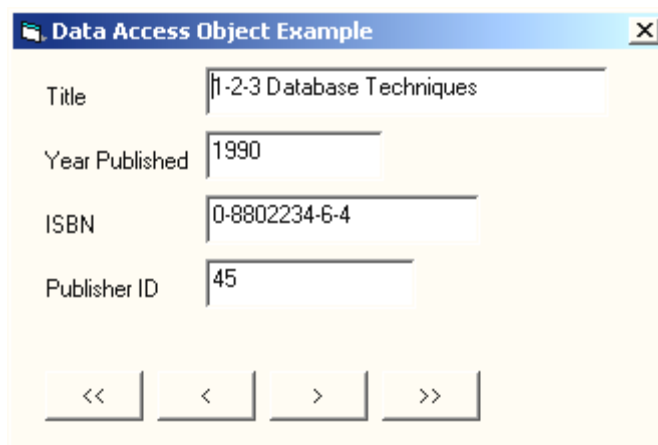
Ví dụ:

```
mYear = rs![Year Published]
```

### b. Duyệt

Phương thức	Công dụng	Ví dụ
<b>MoveNext</b>	Chuyển sang mẫu tin kế tiếp	rs.MoveNext
<b>MoveLast</b>	Chuyển đến mẫu tin cuối cùng	rs.MoveLast
<b>MovePrevious</b>	Chuyển về mẫu tin trước	rs.MovePrevious
<b>MoveFirst</b>	Chuyển về mẫu tin đầu tiên	rs.MoveFirst

Ví dụ: Thiết kế form duyệt bảng Titles của cơ sở dữ liệu BIBLIO.MDB không sử dụng DataControl. Giao diện chương trình có dạng như hình dưới 8.14



**Hình 8.14:** Truy xuất dữ liệu bằng DAO

Khai báo biến toàn cục như sau:

Dim myDb As Database

Dim myRS As Recordset

Mở cơ sở dữ liệu khi nạp form và làm xuất hiện mẫu tin đầu tiên

*Private Sub Form\_Load()*

*AppFolder = App.Path ' Lấy đường dẫn hướng trình*

*If Right(AppFolder, 1) <> "\" Then AppFolder = AppFolder & "\"*

*' Mở cơ sở dữ liệu BIBLIO.MDB*

*Set myDb = OpenDatabase(AppFolder & "BIBLIO.MDB")*

*' Mở Recordset*

*Set myRS = myDb.OpenRecordset("Select \* from Titles ORDER BY Title")*

*' Hiện mẫu tin đầu tiên*

*If myRS.RecordCount > 0 Then*

*myRS.MoveFirst ' di chuyển về mẫu tin đầu tiên*

*Displayrecord ' Hiện nội dung mẫu tin*

*End If*

*End Sub*

*Private Sub Displayrecord()*

*' Làm xuất hiện nội dung mẫu tin bằng cách gán giá trị vào các textbox*

*With myRS*

*txtTitle.Text = .Fields("Title")*

*txtYearPublished.Text = .Fields("[Year Published]")*

*txtISBN.Text = .Fields("ISBN")*

*txtPubID.Text = .Fields("PubID")*

*End With*

*End Sub*

Viết lệnh cho các nút di chuyển

```
Private Sub CmdNext_Click()
    myRS.MoveNext      ' Di chuyển sang mẫu tin kế tiếp
    If Not myRS.EOF Then
        Displayrecord
    Else
        myRS.MoveLast
    End If
End Sub
```

```
Private Sub CmdPrevious_Click()
    myRS.MovePrevious  ' Di chuyển về mẫu tin trước đó
    If Not myRS.BOF Then
        Displayrecord
    Else
        myRS.MoveFirst
    End If
End Sub
```

```
Private Sub CmdFirst_Click()
    myRS.MoveFirst     ' Di chuyển về mẫu tin đầu tiên
    Displayrecord
End Sub
```

```
Private Sub CmdLast_Click()
    myRS.MoveLast      ' Di chuyển về mẫu tin cuối cùng
    Displayrecord
End Sub
```

### c. Di chuyển nhanh bằng Bookmark

```
Dim Lastmark As Variant
Lastmark = rs.Bookmark      ' Đánh dấu trước khi di chuyển
rs.MoveLast                 ' Di chuyển đến mẫu tin cuối
...
rs.Bookmark = Lastmark     ' Quay trở lại mẫu tin trước khi dc
```

### c. Tìm kiếm bằng Find

Phương thức	Hoạt động	Ví dụ
<b>FindNext</b>	Tìm mẫu tin kế tiếp thỏa mãn điều kiện	rs.FindNext "PubID=5"
<b>FindLast</b>	Tìm mẫu tin cuối cùng thỏa mãn điều kiện	rs.FindLast "PubID=5"

---

<b>FindPrevious</b>	Tìm mẫu tin phía trước thỏa mãn điều kiện	rs.FindPrevious "PubID=5"
<b>FindFirst</b>	Tìm mẫu tin đầu tiên thỏa mãn điều kiện	rs.FindFirst "PubID=5"

---

Ví dụ:

```
FindStr = "PubID = " & mID
```

```
With rs.Recordset
```

```
    Bm = .Bookmark
```

```
    .MoveFirst
```

```
    .FindFirst FindStr
```

```
    if .Nomatch then
```

```
        MsgBox "Not found"
```

```
        .Bookmark = bm
```

```
    End if
```

```
End with
```

**d. Tìm kiếm bằng Seek**Ví dụ:

```
Dim MyTable As Recordset
```

```
Set MyTable = Data1.Recordset
```

```
MyTable.Index = "Supplier ID"
```

```
With MyTable
```

```
    Bm = .Bookmark
```

```
    .Seek "=", mID
```

```
    if .Nomatch then
```

```
        MsgBox "Not found"
```

```
        .Bookmark = bm
```

```
    End if
```

```
End with
```

**e. Cập nhật cơ sở dữ liệu**

- Thay đổi giá trị 1 Field

Gọi phương thức **Edit**, gán giá trị mới cho vùng cần thay đổi, sau đó sử dụng phương thức **Update** để cập nhật thay đổi.

Ví dụ:

```
With rs
```

```
    .Edit
```

```
    .Fields![PubID] = "12345"
```

```
    .Update
```

```
End With
```

- **Thêm mẫu tin mới**

Sử dụng phương thức **AddNew** để thêm mới một mẫu tin trống, gán giá trị rồi sử dụng phương thức **Update** để cập nhật.

Ví dụ:

*With rs*

*' Thêm mẫu tin trống*

*.AddNew*

*' Gán giá trị mới*

*.rs.Fields("Title") = "The Data Control"*

*.rs.Fields("Year Published") = "1993"*

*.rs.Fields("AU\_ID") = 37*

*.rs.Fields("ISBN") = "2344456533"*

*.rs.Fields("PubID") = 43*

*' Cập nhật*

*.rs.Update*

*End With*

**f. Xoá 1 mẫu tin**

Sử dụng phương thức **Delete** để xoá mẫu tin hiện hành

*rs.Delete*

## Chương 9

# PictureBox – Xử lý mouse

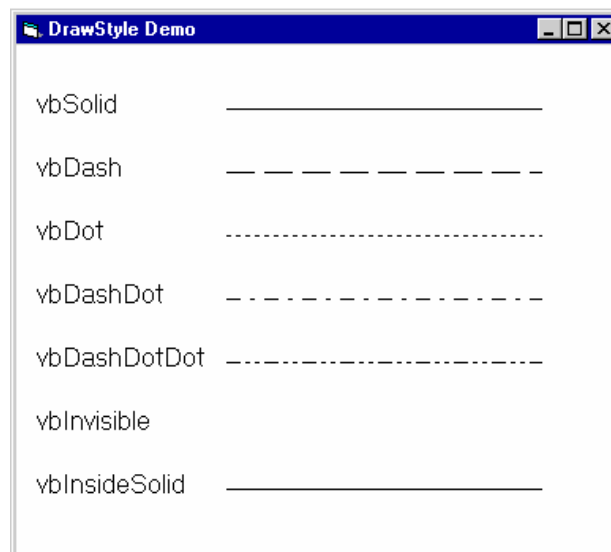
### I. PICTUREBOX

Là đối tượng điều khiển dùng để trình bày các hình ảnh. Picture Box khác với Image ở chỗ Image chỉ trình bày ảnh, không thể xử lý trên ảnh. PictureBox ngoài chức năng trình bày hình ảnh, còn có các phương thức đồ họa cho phép xử lý trên ảnh như xoá ảnh, vẽ thêm ...

PictureBox còn dùng làm đối tượng chứa để chứa các đối tượng khác.

#### 1. Thuộc tính:

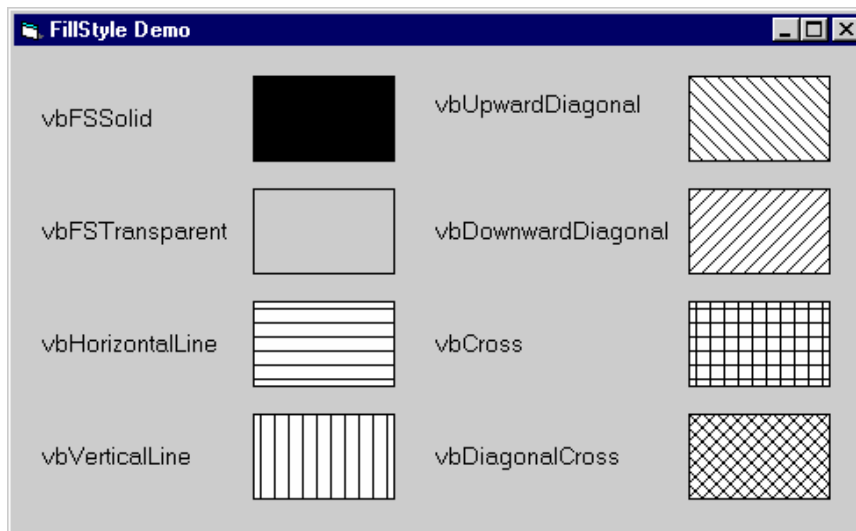
<b>Picture</b>	Giữ hình ảnh cần trình bày, sử dụng LoadPicture để nạp tập tin ảnh
<b>BorderStyle</b>	Kiểu khung 0-None , 1- Fixed Single
<b>Align</b>	Vị trí đặc biệt trên form 0-None,1-Align Top, 2-Align Bottom, 3-Align Left, 4-Align Right
<b>AutoSize</b>	True/False: Thuộc tính tự động điều chỉnh kích thước PictureBox để thể hiện đầy đủ đối tượng chứa trong nó (giống thuộc tính stretch của Image)
<b>FillColor</b>	Màu tô
<b>FillStyle</b>	Mẫu tô, có các giá trị: 0-Solid, 1-Transparent, 2- Horizontal Line, 3-Vertical Line, 4-Upward Diagonal, 5-Downward Diagonal, 6-Cross, 7-Diagonal Cross



Hình 9.1: Các mẫu nét vẽ



**DrawStyle** Kiểu của nét vẽ, có các giá trị:  
0-vbSolid, 1-vbDash, 2-vbDot, 3-vbDashDot, 4- vbDashDotDot, 5-  
vbInvisible, 6-vbInsideSolid



Hình 9.2: Mẫu tô

**DrawWidth** Bề dày nét vẽ tính theo điểm. Có giá trị 0-32767  
**CurrentX,** Tọa độ vẽ hiện tại trong PictureBox. Được cập nhật sau mỗi lệnh vẽ  
**CurrentY**  
**AutoRedraw** True/False: Nội dung vẽ trong PictureBox được tự động vẽ lại mỗi khi thay đổi kích thước

## 2. Các phương thức đồ họa

**Pset [Step] (x,y), color**

Chấm 1 điểm trên Picture Box với

(x,y) Tọa độ điểm. Khi có từ khóa Step, tọa độ điểm có ý nghĩa là độ dời điểm so với tọa độ hiện tại.

Color Màu điểm chấm, nét chấm phục thuộc vào thuộc tính DrawWidth

Ví dụ: Chấm 1000 với màu và độ dày nét ngẫu nhiên trên Picture Box

```
For i = 1 To 1000
```

```
    Picture1.DrawWidth = Rnd * 10 + 1
```

```
    Picture1.PSet (Rnd * ScaleWidth, Rnd * ScaleHeight), _
```

```
        RGB(Rnd * 255, Rnd * 255, Rnd * 255)
```

```
Next
```

```
Picture1.DrawWidth = 1
```

**Print** <Giá trị>

In <Giá trị> lên PictureBox tại tọa độ hiện tại và cập nhật CurrentX = 0 , CurrentY = tọa độ dòng kế tiếp.

Ví dụ :

Picture1.Print "Hello"

**Cls**

Xoá PictureBox

**Line** [Step] (x1,y1) – [Step] (x2,y2), Color, BF

Vẽ đoạn thẳng hoặc hình chữ nhật

Trong đó:

Step (x1,y1) Tọa độ đầu.

**Step** là tùy chọn, khi đó (x1,y1) có ý nghĩa là độ dời so với tọa độ vẽ hiện tại.

(x1,y1) Tọa độ bắt đầu vẽ, nếu không có thì vẽ từ tọa độ hiện tại.

Step (x2,y2) Tọa độ cuối

**Step** là tùy chọn, khi đó (x2,y2) có ý nghĩa là độ dời so với tọa độ vẽ hiện tại.

(x2,y2) Tọa độ cuối, bắt buộc phải có.

Color Màu nét vẽ .

B Vẽ hình chữ nhật

F Tô hình chữ nhật, chỉ được dùng với F

Lưu ý:

- Màu nét vẽ cũng có thể xác định bằng thuộc tính ForeColor.
- Kiểu , cỡ nét vẽ xác định bằng thuộc tính DrawStyle, DrawWidth

Ví dụ 1:

*Picture1.Line (0,0) - (100,100) ' Vẽ đoạn thẳng từ (0,0) - (100,100)*

*Picture1.Line - Step (50,50) ' Vẽ tiếp từ (100,100) đến (150,150)*

*Picture1.Line (0,0) - (100,100), vbRed , B 'Vẽ hình chữ nhật với nét vẽ màu đỏ*

*Picture1.Line (0,0) - (100,100), vbRed , BF 'Vẽ hình chữ nhật tô màu đỏ*

Ví dụ 2: Vẽ tam giác

*Line (1000, 2000)- Step (1000, 0) ' Vẽ đường ngang*

*Line -Step (0, 1000) ' Vẽ đường dọc*

*Line -(1000, 2000) ' Khép kín tam giác*

Ví dụ 3: Vẽ đồ thị hàm số  $y = \sin(x)$  trong đoạn  $-\pi$  đến  $+\pi$

*Const pi = 3.141593*

*Xc = Picture1.ScaleWidth/2*

*Yc = Picture1.ScaleHeight/2*

*kx = Picture1.Width / (2 \* pi) ' Hệ số giãn đồ thị theo trục x*

*ky = Picture1.Height / 2 ' Hệ số giãn đồ thị theo trục y*

*For i = -pi To pi Step 0.2*

```

    Picture1.Line -(Xc + kx * i, Yc - ky * Sin(i))
Next I

```

Ví dụ 4: Vẽ đồ thị hàm số theo cận trái, cận phải và số mẫu vẽ

```

Dim Count As Integer, xLeft As Integer, xRight As Integer
Dim nStep As Single
pi = 3.141593
Picture1.Cls
Count = Val(txtCnt.Text)
xLeft = Val(txtFrom.Text)
xRight = Val(txtTo.Text)
nStep = (xRight - xLeft) / Count
With Picture1
    kx = .ScaleWidth / (xRight - xLeft)
    ky = .ScaleHeight / 2
    .CurrentX = kx * xLeft
    .CurrentY = -ky * Sin(xLeft)
End With
For i = xLeft To xRight Step nStep
    Picture1.Line -(kx * i, -ky * Sin(i)), vbRed
Next i

```

**Circle** [Step] (x, y), radius, color, start, end [,aspect]

Vẽ đường tròn, ellipse hoặc một cung tròn, ellipse

Trong đó:

(x,y)	Tọa độ tâm, khi có Step thì có ý nghĩa là độ dời so với tọa độ hiện tại
radius	Bán kính
Color	Màu nét vẽ. Cỡ nét vẽ qui định bởi thuộc tính DrawStyle
Start	Góc đầu (radian)
End	Góc cuối (radian)
Aspect	Số dương, thực hoặc nguyên dùng qui định tỷ lệ vẽ 2 trục. Aspect > 1: Ellipse kéo dài theo trục X Aspect < 1: Ellipse kéo dài theo trục Y

Ví dụ :

- Vẽ hình tròn với nét vẽ có cỡ 3 pixel, viền màu xanh, tô màu vàng

```

With Pict1
    .DrawWidth = 3
    .FillStyle = vbFSSolid

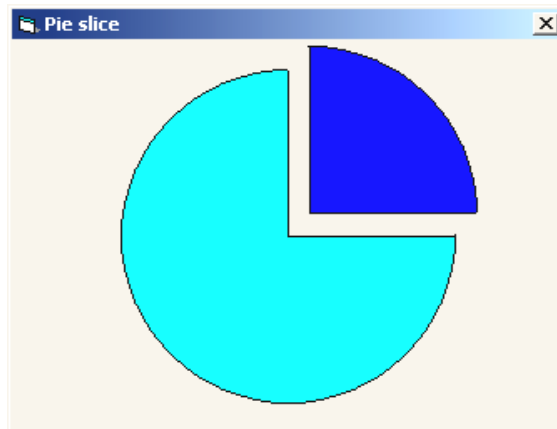
```

```

    .FillColor = vbYellow
    .Circle (1200, 1200), 1000, vbGreen
End With
- Vẽ Ellipse đặc, bán kính trục Y là 500, trục X là 250
  With Pict1
    .FillStyle = 0
    .Circle (1000,1000), 500, , , , 2
  End With
- Vẽ Ellipse đặc, bán kính trục Y là 250, trục X là 500
  With Pict1
    .FillStyle = 0
    .Circle (1000,1000), 500, , , , 1/2
  End With
Để đơn giản hoá lệnh vẽ Ellipse có thể định nghĩa thủ tục vẽ Ellipse như sau:
Sub Ellipse(X As Single, Y As Single, RadiusX As Single, RadiusY As Single)
  Dim ratio As Single, radius As Single
  ratio = RadiusY / RadiusX
  If ratio < 1 Then
    radius = RadiusX
  Else
    radius = RadiusY
  End If
  Circle (X, Y), radius, , , , ratio
End Sub
- Vẽ một cung 1/4 đường tròn:
  Const PI = 3.141593
  Circle (ScaleWidth / 2, ScaleHeight / 2), 1500, vbBlack, 0, PI / 2
- Vẽ Pie slice:
  Const PI = 3.141593
  FillStyle = vbFSSolid
  FillColor = vbBlue
  Circle (ScaleWidth / 2 + 200, ScaleHeight / 2 - 200), 1500, vbBlack, -(PI * 2), -
  (PI / 2)
  FillColor = vbCyan

```

*Circle (ScaleWidth / 2, ScaleHeight / 2), 1500, vbBlack, -(PI / 2), -(PI \* 2)*



Hình 9.3: Vẽ PieSlice

### 3. Các thuộc tính qui định đơn vị vẽ

- ScaleMode Qui định đơn vị vẽ, có các giá trị
- 0 - User: Đơn vị vẽ do người dùng định nghĩa
  - 1 - Twip: 1440 twips = 1'', 567 twips = 1 cm (Đơn vị mặc định)
  - 2 - Point: 72 = 1''
  - 3 - Pixel: Đơn vị điểm trên màn hình
  - 4 - Character: Tính theo đơn vị ký tự. Theo chiều ngang mỗi đơn vị bằng 120 twip, theo chiều dọc mỗi đơn vị bằng 240 twip
  - 5 - Inch
  - 6 - Milimeter
  - 7 - Centimeter

ScaleHeight, Qui định đơn vị tính theo ScaleMode.

ScaleWidth

ScaleLeft, Qui định lại trục tọa độ trên PictureBox. Giá trị mặc định là (0,0).

ScaleTop Ví dụ:

Muốn dời trục tọa độ đến (50,50). Đặt ScaleLeft = -50, ScaleTop = -50

Ví dụ : Vẽ đồ thị hàm số trên form bằng cách dời hệ trục tọa độ sử dụng các thuộc tính ScaleLeft, ScaleTop.

*' Vùng cần vẽ trên mặt phẳng X-Y*

*Const XMIN = -5, XMAX = 5, YMIN = -100, YMAX = 100*

*Const XSTEP = 0.01*

*Private Sub Form\_Resize()*

*' Tự động điều chỉnh vùng vẽ khi kích thước form thay đổi*

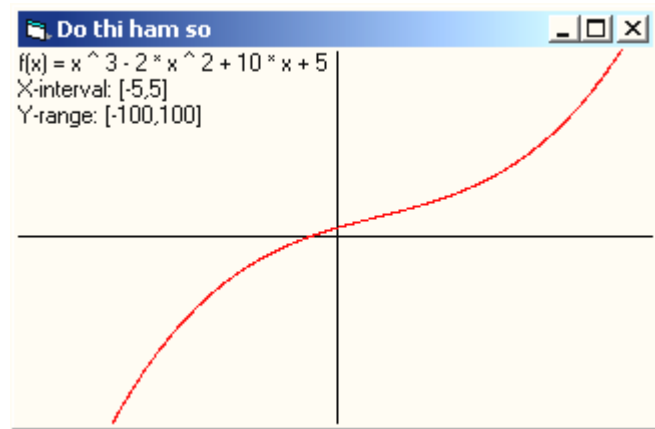
*ScaleLeft = XMIN*

*ScaleTop = YMAX*

```

ScaleWidth = XMAX - XMIN
ScaleHeight = -(YMAX - YMIN)
' Vẽ lại
Refresh
End Sub
Private Sub Form_Paint()
Dim x As Single, y As Single
Cls
ForeColor = vbBlack
CurrentX = ScaleLeft
CurrentY = ScaleTop
Print "f(x) = x ^ 3 - 2 * x ^ 2 + 10 * x + 5"
CurrentX = ScaleLeft
Print "X-interval: [" & XMIN & "," & XMAX & "]"
CurrentX = ScaleLeft
Print "Y-range: [" & YMIN & "," & YMAX & "]"
' Vẽ hệ trục x- y
Line (XMIN, 0)-(XMAX, 0)
Line (0, YMIN)-(0, YMAX)
' Vẽ đồ thị với nét vẽ màu đỏ
ForeColor = vbRed
For x = XMIN To XMAX Step XSTEP
y = x ^ 3 - 2 * x ^ 2 + 10 * x + 5
PSet (x, y)
Next
End Sub

```



#### 4. Các lệnh ghi nạp ảnh

- Hàm LoadPicture nạp tập tin ảnh từ đĩa và trình bày trong PictureBox.

Dạng `Picture1.Picture = LoadPicture(<Tên tập tin>)`

Ví dụ:

```
Picture1.Picture = LoadPicture("C:\Window\setup.bmp")
```

- 🚩 Lệnh SavePicture lưu ảnh trên PictureBox lên đĩa thành tập tin. Ảnh luôn được lưu dạng Bitmap

Dạng `SavePicture Picture1.Image, <Tên tập tin>`

Ví dụ:

```
SavePicture Picture1.Image, "MyPic.bmp"
```

## II. XỬ LÝ MOUSE

Xử lý Mouse bao gồm việc viết lệnh để xử lý cho các sự kiện sau:

- Sự kiện bấm phím mouse

**Private Sub MouseDown**(Button As Integer, Shift As Integer, X As Single, Y As Single)

- Sự kiện di chuyển mouse

**Private Sub MouseMove**(Button As Integer, Shift As Integer, X As Single, Y As Single)

- Sự kiện nhả phím mouse

**Private Sub MouseUp**(Button As Integer, Shift As Integer, X As Single, Y As Single)

Ý nghĩa của các tham số:

Button Cho biết nút trên mouse đang được bấm, có các giá trị sau:

- 1 - vbLeftButton: Nút trái
- 2 - vbRightButton: Nút phải
- 4 - vbMiddleButton: Nút giữa

Shift Giá trị cho biết trạng thái các phím Shift, Ctrl và Alt khi xảy ra sự kiện Mouse. Có các giá trị sau:

- 1 - vbShiftMask: Bấm Shift
- 2 - vbCtrlMask: Bấm Control
- 4 - vbAltMask: Bấm Alternate

X,Y Tọa độ hiện tại trên đối tượng

Lưu ý:

Trạng thái của các phím tổ hợp được tính như sau:

Giá trị nhị phân	Trị thập phân	Hằng	Ý nghĩa
001	1	vbShiftMask	Bấm phím SHIFT
010	2	vbCtrlMask	Bấm phím CTRL
100	4	VbAltMask	Bấm phím ALT
011	3	vbShiftMask + vbCtrlMask	Bấm tổ hợp phím SHIFT+CTRL
101	5	vbShiftMask + vbAltMask	Bấm tổ hợp phím SHIFT+ALT

110	6	vbCtrlMask + vbAltMask	CTRL+ALT
111	7	vbCtrlMask + vbAltMask + vbShiftMask	Bấm tổ hợp phím SHIFT+ CTRL+ALT

Quan hệ giữa các sự kiện MouseDown, MouseUp, MouseMove với Click và DblClick như sau:

- Sự kiện Click xảy ra sau chuỗi sự kiện MouseDown...MouseUp
- Khi người dùng DbClick trên đối tượng, chuỗi sự kiện sau xảy ra: MouseDown, MouseUp, Click, MouseMove, DblClick, MouseUp, MouseMove

Ví dụ 1: Xét sự kiện bấm phím khi di chuyển chuột

*Private Sub Form\_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)*

```
    If Button = 1 Then
        Print "You're pressing the left button."
    ElseIf Button = 2 Then
        Print "You're pressing the right button."
    ElseIf Button = 3 Then
        Print "You're pressing both buttons."
    End If
End Sub
```

Ví dụ 2: Xét phím được bấm kèm với sự kiện bấm phím chuột

*Private Sub Form\_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)*

```
    Select Case Shift
        Case 1 ' or vbShiftMask
            Print "You pressed the SHIFT key."
        Case 2 ' or vbCtrlMask
            Print "You pressed the CTRL key."
        Case 4 ' or vbAltMask
            Print "You pressed the ALT key."
        Case 3
            Print "You pressed both SHIFT and CTRL."
        Case 5
            Print "You pressed both SHIFT and ALT."
        Case 6
            Print "You pressed both CTRL and ALT."
        Case 7
            Print "You pressed SHIFT, CTRL, and ALT."
```



*End Select*

*End Sub*

Thuộc tính **DrawMode** qui định chế độ của bút vẽ, gồm các hằng có ý nghĩa như sau:

Hằng	Giá trị	Hoạt động	Phép toán (S=Screen, P=Pen)
<b>vbBlackness</b>	1	Màu nền = 0 (Đen), màu bút vẽ không có tác dụng	$S = 0$
<b>vbNotMergePen</b>	2	Thực hiện phép OR trên màu bút vẽ và màu nền rồi đảo các bit kết quả bằng phép NOT	$S = \text{Not} (S \text{ Or } P)$
<b>vbMaskNotPen</b>	3	Đảo màu bút vẽ bằng phép NOT rồi AND với màu nền.	$S = S \text{ And Not } P$
<b>vbNotCopyPen</b>	4	Đảo màu bút vẽ	$S = \text{Not } P$
<b>vbMaskPenNot</b>	5	Đảo màu nền bằng phép NOT rồi AND với màu bút vẽ.	$S = \text{Not } S \text{ And } P$
<b>vbInvert</b>	6	Đảo màu nền, màu bút vẽ không có tác dụng	$S = \text{Not } S$
<b>vbXorPen</b>	7	Thực hiện phép XOR trên màu bút vẽ và màu nền	$S = S \text{ Xor } P$
<b>vbNotMaskPen</b>	8	Thực hiện phép AND trên màu bút vẽ và màu nền rồi đảo màu kết quả bằng phép NOT	$S = \text{Not} (S \text{ And } P)$
<b>vbMaskPen</b>	9	Thực hiện phép AND trên màu bút vẽ và màu nền	$S = S \text{ And } P$
<b>vbNotXorPen</b>	10	Thực hiện phép XOR trên màu bút vẽ và màu nền rồi đảo màu kết quả bằng phép NOT	$S = \text{Not} (S \text{ Xor } P)$
<b>vbNop</b>	11	Tắt chế độ vẽ	$S = S$
<b>vbMergeNotPen</b>	12	Đảo màu bút vẽ rồi OR kết quả với màu nền	$S = S \text{ Or Not } P$
<b>vbCopyPen</b>	13	Màu bút vẽ	$S = P$
<b>vbMergePenNot</b>	14	Đảo màu nền rồi OR kết quả với màu bút vẽ	$S = \text{Not } S \text{ Or } P$
<b>vbMergePen</b>	15	Thực hiện phép OR giữa màu bút vẽ và màu nền	$S = S \text{ Or } P$

Ví dụ 1: Viết chương trình vẽ tự do bằng mouse. Chế độ vẽ bắt đầu khi bấm phím trái rồi di chuyển. Kết thúc chế độ vẽ khi nhả phím trái

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    If Button=1 then
        Pict1.CurrentX = X
        Pict1.CurrentY = Y
    End if
```

```
End Sub
```

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    If Button = 1 Then Pict1.Line -(X, Y)
```

```
End Sub
```

Ví dụ 2 : Viết chương trình vẽ tự do bằng mouse trên form, màu nét vẽ được chọn ngẫu nhiên. Cách vẽ là Click để xác định điểm đầu, kéo để xác định kích thước và nhả mouse để xác định hình; nếu click phím trái thì vẽ hình chữ nhật, click phím phải thì vẽ hình chữ nhật có tô màu bên trong

```
' Biến toàn cục của form
```

```
Dim X1 As Single, X2 As Single
```

```
Dim Y1 As Single, Y2 As Single
```

```
' Biến xác định chế độ vẽ, có giá trị True nếu đang vẽ
```

```
Dim dragging As Boolean
```

```
Private Sub Form_Load()
```

```
    ' Xoá nền form thành màu đen
```

```
    BackColor = vbBlack
```

```
End Sub
```

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
    X As Single, Y As Single)
```

```
    If (Button=1) or (Button=2) Then
```

```
        dragging = True
```

```
        ' Ghi lại tọa độ bắt đầu
```

```
        X1 = X: Y1 = Y: X2 = X: Y2 = Y
```

```
        ' Chọn màu vẽ ngẫu nhiên
```

```
        ForeColor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
```

```
        DrawWidth = Rnd * 3 + 1
```

```
        DrawMode = vbXorPen
```

```
        ' Vẽ hình chữ nhật
```

```
        Line (X1, Y1)-(X2, Y2), , B
```

```
        If Button = 2 Then           ' Nếu là phím phải thì tô hình
```

```
        FillStyle = vbFSSolid
        FillColor = ForeColor
    End If
End If
End Sub
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, _
    X As Single, Y As Single)
    If dragging Then
        ' Xoá hình cũ
        Line (X1, Y1)-(X2, Y2), , B
        ' Vẽ hình mới
        X2 = X: Y2 = Y
        Line (X1, Y1)-(X2, Y2), , B
    End If
End Sub
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, _
    X As Single, Y As Single)
    If dragging Then
        dragging = False
        DrawMode = vbCopyPen
        Line (X1, Y1)-(X, Y), , B
        FillStyle = vbFSTransparent
    End If
End Sub
```

# Chương 10


## Menu – Common Dialog

### I. MENU

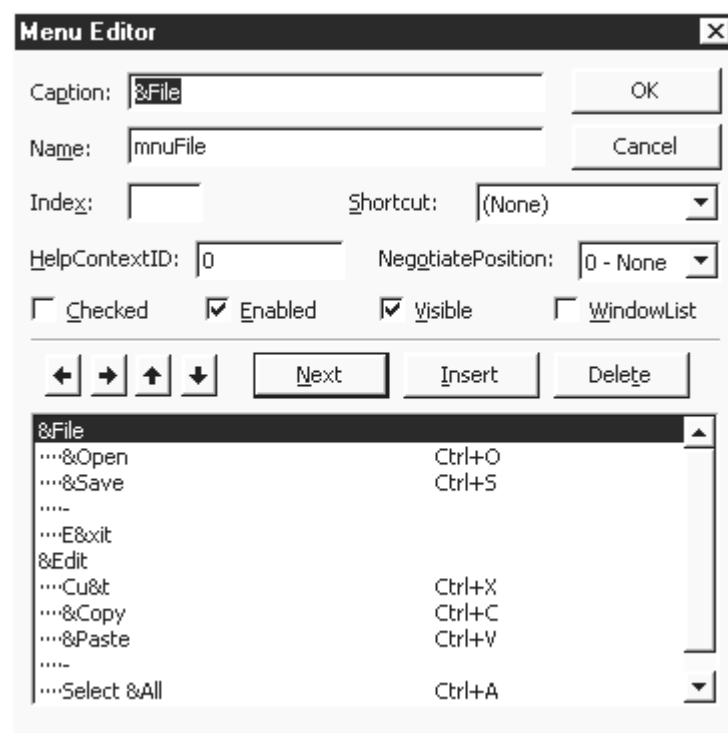
Về cấu trúc, menu là một danh sách mục chọn, mỗi mục chọn là một đối tượng. Hệ thống menu là một danh sách đối tượng và được tổ chức phân cấp. Cấp trên cùng là menu thanh (bar), cấp kế tiếp là menu kéo xuống (PullDown), trong menu kéo xuống lại có thể có những menu con cấp thấp hơn...

#### 1. Định nghĩa Menu

Để định nghĩa menu thực hiện như sau:

- Bấm tổ hợp phím CTRL-E,  
hoặc
- Nút Menu Editor  trên thanh công cụ,

Hiện hộp thoại Menu Editor xuất hiện như hình 10.1.



Hình 10.1: Hộp thoại Menu Editor

Hộp thoại Menu Editor gồm 3 phần:

- Phần trên cùng: Các thuộc tính của một mục chọn. Các thuộc tính này phải được xác định khi định nghĩa mới 1 mục chọn.
- Phần các nút lệnh.
- Phần danh sách các mục chọn đã định nghĩa.

Các thuộc tính của một mục chọn này được tóm tắt trong bảng sau:

Thuộc tính	Ý nghĩa
Caption	Tên mục chọn Menu, có thể định nghĩa Hotkey . Sử dụng ký tự “-“ cho vạch phân cách trên menu
Name	Tên trong chương trình, thường bắt đầu bằng mnu
Index	Đánh chỉ số nếu sử dụng mảng mục chọn
Shortcut	Định nghĩa tổ hợp phím tắt
Checked	Mục chọn thuộc loại chọn, bỏ chọn
Enabled	Cho phép/Không cho phép hoạt động
Visible	Xuất hiện/Không xuất hiện mục tương ứng trên menu
WindowList	Menu có chứa danh sách các form đang mở trong chương trình (ứng dụng MDI)

- Mỗi mục chọn trên menu được định nghĩa bằng cách nhập các thuộc tính Name, Caption, Shortcut... Giá trị Caption xuất hiện trong danh sách mục chọn phía dưới của Menu Editor. Sau khi nhập đầy đủ các thuộc tính, bấm nút **Next** để định nghĩa mục chọn kế tiếp.
- Danh sách mục chọn định nghĩa trình bày theo cột. Mục chọn ở cột ngoài cùng bên trái tương ứng với các mục chọn trên menu bar. Mục chọn ở cột kế tiếp tương ứng với các mục chọn trên menu kéo xuống, cột kế tiếp nữa tương ứng với các mục chọn trên menu cấp thấp hơn ... Sử dụng các nút ← → để chuyển một mục chọn lên (xuống) cấp menu tương ứng. Sử dụng các nút ↓↑ để thay đổi thứ tự các mục chọn trên menu.
- Nút **Insert** chèn thêm một mục chọn.
- Nút **Delete** xoá một mục chọn.

## 2. Viết lệnh

- Click vào mục chọn cần định nghĩa mã lệnh, khai báo của thủ tục xử lý sự kiện tương ứng sẽ xuất hiện trong cửa sổ lệnh

**Private sub** Tênmenu\_click()

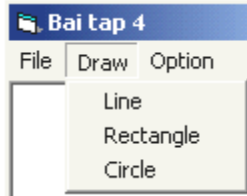
**End sub**

- Nhập lệnh định nghĩa cho mục chọn bên trong thủ tục xử lý sự kiện.

Lưu ý:

- Khi có nhiều mục chọn trên menu cùng cấp, có thể định nghĩa mảng mục chọn để thuận tiện cho việc xử lý lệnh. Khi định nghĩa mảng mục chọn cần lưu ý các mục chọn sẽ được định nghĩa cùng tên, gán thuộc tính Index liên tiếp cho mỗi mục chọn.

Ví dụ: Định nghĩa menu chọn chế độ vẽ trong chương trình vẽ hình, các chế độ vẽ được định nghĩa là mảng mnuDraw với chỉ số liên tiếp (0,1,2) hoặc (1,2,3)



Thủ tục xử lý sự kiện khi đó có dạng:

```
Private Sub mnuDraw_Click(Index As Integer)
    DrMode = Index          ' Chọn chế độ vẽ
End Sub
```

- Khi mảng mục chọn hoạt động theo nhóm (kiểu nút chọn Options), sử dụng thêm thuộc tính checked để ký hiệu giá trị đang chọn và viết thêm lệnh đồng bộ hoạt động của các mục này. Đoạn lệnh sau đồng bộ hoạt động của các mục chọn hoạt động theo nhóm.

```
Private Sub mnuDraw_Click(Index As Integer)
    For i = 1 To mnuDraw.Count
        mnuDraw(i).Checked = False    ' Uncheck tất cả các mục chọn
    Next
    MnuDraw(Index).Checked = True     ' Check mục chọn
    DrMode = Index
End Sub
```

### c. Menu Popup

Là loại menu được kích hoạt khi người sử dụng bấm phím phải chuột trên một đối tượng. Menu popup có thể là một menu độc lập được thiết kế bằng Menu Editor, nó cũng có thể là một menu thành phần trong hệ thống menu đã được thiết kế bằng Menu Editor. Để làm xuất hiện menu Popup, sử dụng phương thức PopupMenu <Menu>

Trong đó tham số <menu> là tên của menu Popup

Ví dụ: Làm xuất hiện menu popup khi bấm phím phải chuột trên listbox

```
Private Sub List1_MouseDown(Button As Integer, Shift As Integer,
    X As Single, Y As Single)
    If Button And vbRightButton Then PopupMenu mnuListPopup
End Sub
```

## II. COMMON DIALOG

Là một lớp hộp đối thoại thường được sử dụng trên các ứng dụng chạy trên windows. Các loại hộp thoại này gồm:

- Hộp thoại thao tác trên tập tin: File Open, File Save,
- Hộp thoại định dạng font chữ : Chọn kiểu chữ, kiểu dáng chữ, cỡ chữ...,
- Hộp thoại chọn màu,
- Hộp thoại in ấn.

Common Dialog được chứa trong hệ thống dưới dạng ActiveX Control, có thể được gọi sử dụng trong các ứng dụng viết trên windows. Để có thể sử dụng đối tượng này, cần nạp lên Toolbox..

Để nạp đối tượng lên Toolbox thực hiện như sau:

- Chọn Project/Components hoặc bấm Ctrl-T, xuất hiện Dialog Components.
- Trong listbox Control, chọn Microsoft Common Dialog Control
- Bấm nút Apply, biểu tượng xuất hiện trên Toolbox

Để đưa vào chương trình:

- Double-Click để đặt đối tượng lên form

Để làm xuất hiện hộp thoại trong chương trình, sử dụng các phương thức tương ứng sau:

Tên phương thức	Ý nghĩa
ShowOpen	Xuất hiện hộp đối thoại open file
ShowSave	Xuất hiện hộp đối thoại Save
ShowColor	Xuất hiện hộp đối thoại chọn màu
ShowFont	Xuất hiện hộp đối thoại chọn font
ShowPrinter	Xuất hiện hộp đối thoại in

### 1. Hộp đối thoại Open, Save

Thuộc tính	Ý nghĩa
DialogTitle	Tiêu đề Dialog.
InitDir	Đường dẫn thư mục đầu tiên xuất hiện trong Dialog
FileName	Gán hoặc lấy tên tập tin được chọn (đầy đủ đường dẫn)
FileTitle	Tên tập tin không có đường dẫn
Filter	Chuỗi chứa các loại tập tin được trình bày trong Dialog. Ví dụ: Text (*.txt) *.txt Pictures (*.bmp;*.ico) *.bmp;*.ico
FilterIndex	Chỉ số qui định loại tập tin được chọn đầu tiên khi trình bày dialog (Loại đầu tiên có Index=1)
DefaultExt	Chuỗi ký tự qui định phần mở rộng mặc định
CancelError	True/False: Gây lỗi hệ thống khi người dùng bấm nút Cancel

flags	Thuộc tính tùy chọn tính chất của hộp thoại, có các giá trị: cdIOFNReadOnly – Đặt tùy chọn chỉ đọc khi mở tập tin cdIOFNAllowMultiselect – Cho phép chọn nhiều tập tin cdIOFNHideReadOnly – Bỏ nút chọn Read-Only trên hộp thoại
-------	---

Ví dụ 1: Viết lệnh xử lý sự kiện click mục chọn Open trên menu

**Private Sub mnuOpen\_Click()**

*On Error GoTo ErrorOpen*

*With CmdlG*

*.InitDir = "C:\"*

*.Filter = "Text (\*.txt)|\*.txt|Pictures (\*.bmp;\*.ico)|\*.bmp;\*.ico"*

*.FilterIndex = 2*

*.CancelError = True*

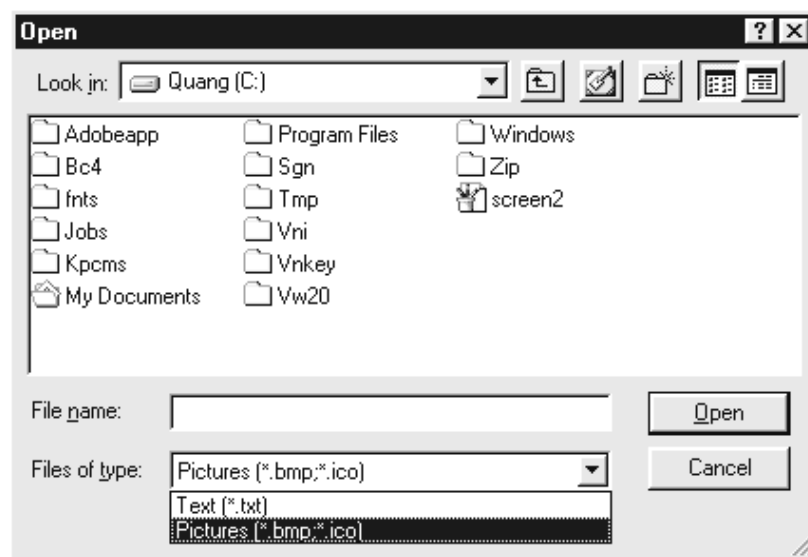
*.ShowOpen*

...

*End With*

*ErrorOpen:*

**End Sub**



Hình 10.2: Hộp thoại Open

Ví dụ 2 : Viết lệnh xử lý sự kiện click mục chọn Save trên menu

**Private Sub mnuSave\_Click()**

*On Error GoTo ErrorOpen*

*With CmdlG*

*.InitDir = "C:\"*

*.Filter = "Text (\*.txt)|\*.txt|Pictures (\*.bmp;\*.ico)|\*.bmp;\*.ico"*

*.FilterIndex = 2*



```

.CancelError = True
.ShowSave
...
End With
ErrorOpen:
End Sub

```

## 2. Hộp thoại chọn màu

Cho phép người sử dụng chọn màu hoặc định nghĩa thêm một màu tùy chọn ngoài các màu có sẵn của hệ thống.

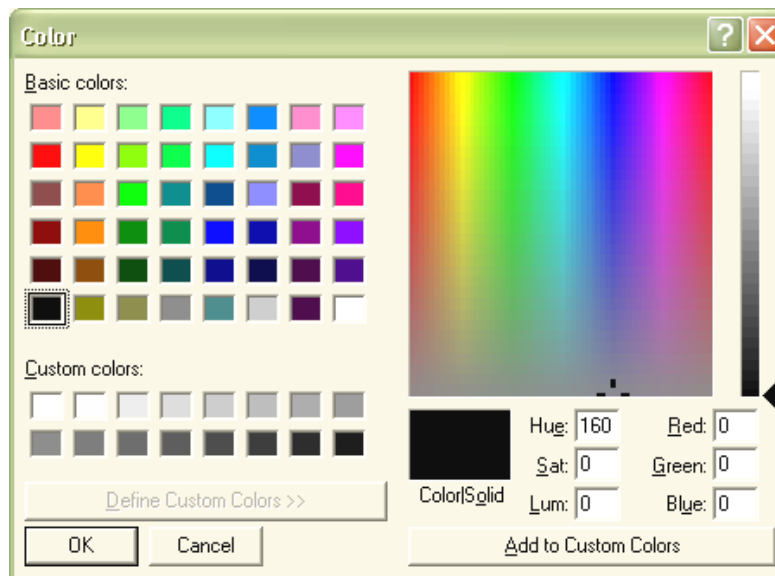
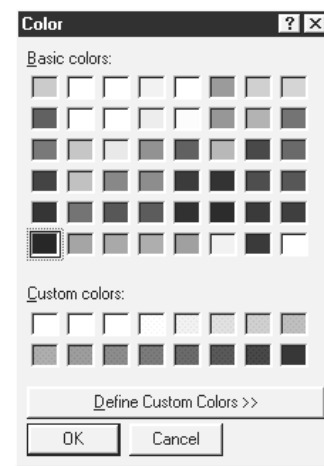
Để mở hộp thoại chọn màu, sử dụng phương thức ShowColor. Hộp thoại chọn màu xuất hiện có dạng như hình bên

Để có thể chọn được nhiều màu hơn hoặc định nghĩa màu tùy ý người sử dụng phải bấm nút Define Custom Colors để mở thêm bảng bên phải (Hình 10.3). Trong chương trình có thể thực hiện điều này bằng cách gán giá trị cdlCCFullOpen cho thuộc tính Flags như sau:

```

With Cmdlg
.Flags = cdlCCFullOpen
.ShowColor
...
End with

```



**Hình 10.3:** Hộp thoại chọn màu đầy đủ

Giá trị màu chọn sau khi người sử dụng bấm nút OK được lấy thông qua thuộc tính color.

# Chương 11

## Kiểu bản ghi – Tập tin

### I. KIỂU BẢN GHI

#### 1. Định nghĩa:

Là kiểu dữ liệu gồm nhiều thành phần gọi là vùng/trường (Fields), mỗi thành phần dùng mô tả một đặc điểm của đối tượng. Bản ghi được sử dụng để lưu trữ các đối tượng mà mô tả về đối tượng đó cần nhiều thông tin.

#### 2. Khai báo:

```
Type <Tên>
    <Vùng 1> As <Kiểu>
    <Vùng 2> As <Kiểu>
    ...
End Type
```

Ví dụ:

```
Type Sinhvien
    Hoten As String*25
    Phai As Byte
    DiemToan As Single
    DiemLy As Single
    DiemHoa As Single
End Type
```

Và khai báo biến:

```
Dim SV As Sinhvien
```

Lưu ý:

Khai báo Type phải được viết trong tập tin module.

### II. TẬP TIN

#### 1. Định nghĩa:

Là đối tượng được sử dụng để lưu trữ dữ liệu trên bộ nhớ ngoài.

#### 2. Phân loại:

Theo cách truy xuất, có 2 loại tập tin:

- Tập tin truy xuất ngẫu nhiên (Random Access File): Là tập tin cho phép đọc hoặc ghi ở vị trí bất kỳ trên file. Dữ liệu ghi trên tập tin truy xuất ngẫu nhiên được tổ chức thành các mẫu tin (Record) có kích thước giống nhau.
- Tập tin truy xuất tuần tự (Sequential Access File): Dữ liệu ghi lên tập tin có kích thước mỗi phần tử không giống nhau, để phân biệt các phần tử với nhau, sử dụng ký hiệu phân cách giữa các phần tử.

### 3. Thủ tục truy xuất dữ liệu trên tập tin:

Việc truy xuất trên tập tin được thực hiện thành 3 bước:

- Mở tập tin
- Truy xuất (Đọc/Ghi)
- Đóng tập tin

### 4. Các lệnh trên tập tin truy xuất ngẫu nhiên

**Lệnh Open** <Đường dẫn> **For Random As #n Len=** <RecLen>

Mở tập tin để đọc hoặc tạo mới.

Trong đó:

<Đường dẫn>	Chuỗi ký tự đường dẫn tên tập tin
n	Số thứ tự tên tập tin mở, giá trị này là số nguyên duy nhất đối với mỗi tập tin, tập tin mở đầu tiên có giá trị là 1. Để lấy số thứ tự của tập tin có thể mở kế tiếp, sử dụng hàm freefile()
<RecLen>	Kích thước mỗi phần tử

Lưu ý:

Để tính kích thước của một kiểu dữ liệu, sử dụng hàm Len(<Tên>). Trong đó <Tên> là tên của một biến.

Ví dụ:

*Type Sinhvien*

*Hoten As String\*25*

*Phai As Byte*

*DiemToan As Single*

*DiemLy As Single*

*DiemHoa As Single*

*End Type*

*Dim sv As SinhVien*

*Dim fnum As Integer*

*fnum = freefile()* ‘Lấy số thứ tự tập tin mở kế tiếp

*Open “Thu.dat” for Random As #fnum Len = Len(sv)*

**Lệnh Put #n, [<Vị trí>], <Biến>**

Ghi giá trị của <Biến> lên tập tin tại <Vị trí>

Ví dụ:

```
Type Record
  ID As Integer
  Name As String * 20
End Type
Dim MyRecord As Record, RecordNumber
Dim fnum As Integer
fnum = freefile()
' Mở file.
Open "TESTFILE" For Random As #fnum Len = Len(MyRecord)
For RecordNumber = 1 To 5
  MyRecord.ID = RecordNumber
  MyRecord.Name = "My Name" & RecordNumber
  Put #1, RecordNumber, MyRecord ' Ghi record lên file
Next RecordNumber
Close #1 ' Đóng file.
```

Lưu ý:

- Vị trí các mẫu tin trên tập tin có thứ tự bắt đầu từ 1.
- Mẫu tin ghi lên tập tin phải có chiều dài đúng bằng chiều dài khai báo khi mở tập tin. Trường hợp ghi mẫu tin có kích thước nhỏ hơn, vb tự động điền cho đủ (với các giá trị ngẫu nhiên). Trường hợp ngược lại sẽ cho thông báo lỗi.
- Không thể ghi đối tượng lên tập tin.
- Tham số vị trí là tùy chọn, khi không có tham số này, dữ liệu sẽ được ghi vào kế sau mẫu tin vừa truy xuất.
- Muốn ghi dữ liệu vào cuối tập tin, cho giá trị của <vị trí> lớn hơn số mẫu tin hiện có trong tập tin. Ví dụ sau mở và ghi dữ liệu vào cuối tập tin bằng cách sử dụng hàm **LOF**

```
Type Record
  ID As Integer
  Name As String * 20
End Type
Dim MyRecord As Record, RecCount
Dim fnum As Integer, fsize As long
Dim recsize As Integer
fnum = freefile()
```

```

recsize = Len(MyRecord)
' Mở file.
Open "TESTFILE" For Random As #fnum Len = recsize
fsize = LOF(fnum)           ' Lấy kích thước tập tin
RecCount = fsize \ recsize   ' Tính số mẫu tin
MyRecord.ID = RecCount+1
MyRecord.Name = "My Name" & RecCount
Put #fnum, RecordNumber, MyRecord
Close #fnum

```

### **Lệnh Get #n, [<Vị trí>], <Biến>**

Đọc từ <vị trí> n vào <Biến> từ tập tin. Lệnh đọc báo lỗi khi <vị trí> lớn hơn số mẫu tin hiện có .

#### Ví dụ 1:

```

Type Record
  ID As Integer
  Name As String * 20
End Type

Dim MyRecord As Record, Position
' Mở file
Open "TESTFILE" For Random As #1 Len = Len(MyRecord)
Position = 3
Get #1, Position, MyRecord ' Đọc mẫu tin thứ 3
Close #1 ' Đóng file.

```

#### Ví dụ 2: Đọc tuần tự từ tập tin

```

Type Record
  ID As Integer
  Name As String * 20
End Type

Dim MyRecord As Record
Open "TESTFILE" For Random As #1 Len = Len(MyRecord)
Do While Not EOF(1)
  Get #1, , MyRecord
  Debug.Print Myrecord.ID, MyRecord.Name
Loop
Close #1

```

### **Sửa chữa một mẫu tin**

Để sửa chữa một mẫu tin, thực hiện các bước sau:

- Đọc mẫu tin cần sửa chữa bằng lệnh Get
- Sửa chữa mẫu tin với giá trị mới
- Ghi lên tập tin tại vị trí cũ bằng lệnh Put

Ví dụ:

```
Get #1, 2, MyRecord
MyRecord.Name = "New Name"
Put #1, 2, MyRecord
```

### Lệnh Close #n

Đóng tập tin

## III. CÁC LỆNH TRÊN TẬP TIN VĂN BẢN

### Lệnh Open <PathName> For <Mode> As #n

Trong đó:

<PathName>: Chuỗi ký tự đường dẫn tên tập tin

<Mode>: Chế độ truy xuất tập tin, gồm:

Output Tạo tập tin mới, nếu tên tập tin đã có trên đĩa, tập tin cũ bị xoá

Input Mở tập tin để đọc

Append Mở tập tin để viết thêm nội dung

n: Số thứ tự tên tập tin mở, mỗi tập tin được mở với 1 số duy nhất. Có giá trị 1-511

Ví dụ: Mở tập tin readme.txt để đọc

```
Dim fnum As Integer
fnum = FreeFile()
Open "readme.txt" For Input As #fnum
```

### Lệnh Print #n,<Danh sách biến>

<Danh sách biến>: Danh sách các biến muốn ghi giá trị, sử dụng dấu ; giữa các biến, mỗi lệnh in danh sách trị trên một dòng.

Ví dụ: Tạo tập tin văn bản có 10 dòng

```
Private Sub Command1_Click()
    Open "F:\Test.txt" For Output As #1
    For i = 1 To 10
        Print #1, "Line " & i
    Next
    Close #1
End Sub
```

Lưu ý:

Dấu “;” cuối danh sách biến sẽ làm cho dòng được in không có ký tự xuống dòng ở cuối dòng

Để đọc tập tin ghi dạng này, sử dụng lệnh **Input**

Ví dụ: Thủ tục ghi Text File với tùy chọn ghép thêm hoặc tạo mới

```
Private Sub WriteTextFileContents(Text As String, filename As String,
    Optional AppendMode As Boolean)
    Dim fnum As Integer
    fnum = FreeFile()
    If AppendMode Then
        Open filename For Append As #fnum
    Else
        Open filename For Output As #fnum
    End If
    Print #fnum, Text
    Close #fnum
End Sub
```

**Lệnh Write #n,<Danh sách biến>**

In giá trị các biến lên tập tin , giá trị được rào bằng dấu nháy kép “” , dấu phẩy là ký hiệu phân cách các giá trị ghi.

Ví dụ lệnh Write #1, Maso, Hoten, Quoctich với Maso, Hoten, Quoctich là các biến chứa giá trị sẽ cho kết quả ghi lên tập tin như sau:

“001”,” Tigana”,”Phap”

Sử dụng lệnh **Input** để đọc tập tin ghi dạng này

**Lệnh input #n,<Biến chuỗi>**

Đọc tập tin văn bản ghi bằng lệnh **Print #n, <Chuỗi>**

Ví dụ:

Tập tin tạo bằng đoạn chương trình

```
Open "F:\Test.txt" For Output As #1
For i = 1 To 10
    Print #1, "Line " & i
Next
Close #1
```

Sẽ được đọc như sau

```
Open "F:\Test.txt" For Input As #1
For i = 1 To 10
```

```

    Input #1, Line
    Debug.Print Line

```

```
Next
```

```
Close #1
```

**Lệnh input #n,<Danh sách biến >**

Đọc tập tin văn bản ghi bằng lệnh **Write #n, <Danh sách biến>**

Ví dụ:

Tập tin tạo bằng đoạn chương trình

```
Open "C:\test.txt" For Output As #1
```

```
...
```

```
Write #1, txtMa.Text, txtHoten.Text, iCQT.Text
```

```
...
```

```
Close #1
```

Sẽ được đọc như sau

```
Open "C:\test.txt" For Input As #1
```

```
Do While Not EOF(1)
```

```
    Input #1, Maso, Hoten, QT
```

```
    Debug.Print Maso, Hoten, QT
```

```
Loop
```

```
Close #1
```

**Lệnh Line input #n,<Biến chuỗi>**

Đọc 1 dòng từ văn bản (không kể ký tự xuống dòng)

Ví dụ: Đọc dữ liệu từ tập tin văn bản

```
Private Sub Command1_Click()
```

```
    Open "F:\Test.txt" For Input As #1
```

```
    Do while not eof(1)
```

```
        Line input #1, Line
```

```
        St = St & Line & vbCRLF
```

```
    Loop
```

```
    Close #1
```

```
    Text1.Text = St
```

```
End Sub
```

**Hàm input (<bytenum>,#n)**

Hàm đọc dữ liệu từ tập tin, kết quả trả về là một chuỗi. Nếu đọc từ tập văn bản, chuỗi trả về gồm tất cả các ký hiệu xuống dòng.

Trong đó:



<bytenum> Số byte muốn đọc

n Số thứ tự tập tin

Ví dụ: Định nghĩa hàm *ReadTextFileContents* đọc tập tin văn bản, dữ liệu đọc chứa vào một chuỗi.

```
Function ReadTextFileContents(filename As String) As String
    Dim fnum As Integer
    ' Lấy số thứ tự tập tin mở kế tiếp
    fnum = FreeFile()
    Open filename For Input As #fnum
    ' Đọc toàn bộ nội dung file bằng một lệnh
    ReadTextFileContents = Input(LOF(fnum), fnum)
    Close #fnum
End Function
```

Nạp tập tin Bootlog.txt vào textbox

```
Text1.Text = ReadTextFileContents("c:\bootlog.txt")
```

Ví dụ: Đọc tập tin văn bản vào listbox

```
Sub TextFileToListbox(lst As ListBox, filename As String)
    Dim items() As String, i As Long
    ' Đọc nội dung file rồi sử dụng hàm split để chuyển các dòng
    ' vào mảng chuỗi
    items() = Split(ReadTextFileContents(filename), vbCrLf)
    ' Nạp các chuỗi khác rỗng vào ListBox.
    For i = LBound(items) To UBound(items)
        If Len(items(i)) > 0 Then lst.AddItem items(i)
    Next
End Sub
```

Lưu ý:

Hàm **Split**(<chuỗi>,<Ký hiệu>[,<số chuỗi con>]) cho giá trị là một mảng chuỗi con được trích ra từ <chuỗi> với ký hiệu phân cách được cho trong tham số <ký hiệu> , tham số thứ ba qui định số chuỗi con muốn trích ra.

## Chương 12

# Microsoft Windows Common Controls

## Imagelist - Listview - Imagecombo

Windows Common Controls là tên gọi chung của các loại đối tượng điều khiển chỉ có trong Windows 9x. Các đối tượng này chứa trong thư viện Microsoft Windows Common Controls. Sử dụng phương pháp đã mô tả ở chương trước để nạp đối tượng lên Toolbox .

### I. IMAGELIST

Đối tượng được sử dụng để quản lý một mảng hình ảnh hay danh sách hình ảnh. Danh sách hình ảnh được sử dụng trong các ứng dụng cần tạo các hiệu ứng hình ảnh động hoặc sử dụng kết hợp với các đối tượng điều khiển khác có sử dụng hình ảnh như ListView, ImageCombo...

#### Sử dụng ImageList

- Nhấp đúp biểu tượng ImageList trên ToolBox để đặt ImageList lên form
- Nhấp phím phải trên biểu tượng ImageList trên form
- Chọn Properties trên menu xuất hiện hộp thoại Property Pages, chọn thẻ Images
- Bấm nút Insert Picture để chọn các hình (\*.BMP, \*.ICO) đưa vào danh sách hình ảnh.

Thuộc tính Index chỉ thứ tự của hình trong danh sách, giá trị này được tự động gán cho mỗi hình

Thuộc tính ImageCount cho biết tổng số hình hiện có trong danh sách, giá trị này được tự động tăng lên khi có một hình mới được chèn thêm vào danh sách

Bấm nút Remove Picture để xoá 1 hình trong danh sách.

### II. LISTVIEW

Là đối tượng điều khiển được sử dụng để trình bày danh sách đối tượng. Các đối tượng trong ListView có thể trình bày theo nhiều kiểu khác nhau.



### 1. Các thuộc tính

**View:** Thay đổi cách trình bày các đối tượng trong listview, có các giá trị như sau:

Hằng	Giá trị	Ý nghĩa
lvwIcon	0	Trình bày đối tượng bằng icon lớn với nhãn ở phía dưới
lvwSmallIcon	1	Các đối tượng được trình bày bằng icon nhỏ, nhãn ở phía bên phải, các đối tượng được liệt kê theo chiều ngang
lvwList	2	Các đối tượng được trình bày bằng icon nhỏ, nhãn ở phía bên phải, các đối tượng được liệt kê theo chiều dọc
lvwReport	3	Các đối tượng được trình bày bằng icon nhỏ với nhãn ở cột đầu tiên, các thông tin khác về đối tượng được trình bày trong các cột kế tiếp

Thay đổi chế độ trình bày của Listview bằng lệnh trên menu

```
Private Sub mnuLarge_Click()
```

```
    ListView1.View = lvwIcon
```

```
End Sub
```

```
Private Sub mnuList_Click()
```

```
    ListView1.View = lvwList
```

```
End Sub
```

```

Private Sub mnuRpt_Click()
    ListView1.View = lvwReport
End Sub
Private Sub mnuSmall_Click()
    ListView1.View = lvwSmallIcon
End Sub

```

**GridLine** (True/False): Kẻ đường lưới trong chế độ ReportView

**FullRowSelect** (True/False): Phần tử chọn được highlight cả dòng.

**MultiSelect** (True/False): Qui định thuộc tính cho phép chọn nhiều

**CheckBoxes**: (True/False): Có/ không có checkbox

**Text**: Nhãn/giá trị cột đầu tiên trong listview khi ở chế độ Report

**LabelEdit** :(0-lvwAutomatic, 1-lvwmanual) Qui định nhãn đối tượng (text) có thể sửa chữa trực tiếp trên listview khi người dùng Click trên nhãn

Các thuộc tính mô tả trên có thể chọn trực tiếp trong Property Pages/General của Listview (bấm phím phải mouse)

**ListItems**: Thuộc tính quan trọng nhất, chứa danh sách các phần tử được trình bày trong listview. Mỗi phần tử là một đối tượng có kiểu ListItem. Cách truy xuất các phần tử trong listItems cũng giống như mảng

Ví dụ:

```

With lvw
    For i = 1 to .ListItems.Count
        Debug.Print .ListItems(i).Text
    Next

```

End with

**SelectedItem**: Cho giá trị là đối tượng ListItem đang được chọn trong Listview hoặc dùng chọn một phần tử trong Listview

- Chọn phần tử đầu tiên trong listview  
`Set lvw.SelectedItem = lvw.ListItems(1)`
- Lấy giá trị phần tử đang được chọn  
`Dim Item as ListItem`  
`Set Item = lvw.SelectedItem`

**ImageList**: Tham chiếu đến đối tượng ImageList quản lý danh sách hình sử dụng trong Listview

Để định nghĩa danh sách hình cho Listview, sử dụng thẻ Image Lists trong Property Pages của Listview

Mỗi Listview có thể liên kết với 3 loại danh sách hình:

- Normal : danh sách hình để xem listview ở dạng Large Icon - Kích thước mặc định của biểu tượng là 32x32
- Small : danh sách hình để xem listview ở dạng Small Icon.- Kích thước mặc định của biểu tượng là 16x16
- Column Header: danh sách hình sử dụng cho dòng tiêu đề cột - Kích thước mặc định của hình là 16x16

Như vậy khi viết ứng dụng có sử dụng Listview thì các danh sách hình phải được định nghĩa trước rồi mới được liên kết với Listview bằng thẻ Image Lists như hình trên.

## 2. Các thuộc tính của đối tượng ListItem

Thuộc tính	Ý nghĩa
Text	Chuỗi ký tự mô tả đối tượng, là nhãn đi kèm với biểu tượng trong listview, cũng chính là nội dung cột đầu tiên trong chế độ Report
Index	Chỉ số của phần tử trong mảng ListItem
Key	Chuỗi ký tự duy nhất xác định phần tử trong danh sách ListItem
Icon, SmallIcons	Gán hoặc lấy chỉ số của hình (index) tương ứng trong danh sách hình (Imagelist) liên kết với listView. Ví dụ: <code>lvw.ListItems(5).SmallIcons=1</code>
Selected	(True/False): Cho biết phần tử có được chọn trong Listview hay không
SubItems	Mảng chuỗi chứa các thông tin khác của đối tượng, các thông tin này được trình bày trong chế độ Report, số phần tử của mảng phải tương ứng với số cột trong listview
Ghosted	(True/False) Làm mờ icon của phần tử trong listview

**ColumnHeaders:** Thuộc tính quản lý danh sách cột trong Listview khi sử dụng ở chế độ Report view. Mỗi cột là một đối tượng có kiểu ColumnHeader. Cách truy xuất các cột trong ColumnHeaders cũng giống như truy xuất các phần tử trong ListItem. Một số phương thức cũng được áp dụng chung cho cả hai đối tượng.

Ví dụ:

In danh sách các cột trong Listview ở chế độ report

*With lvw*

*For i = 1 to .ColumnHeaders.Count*

*Debug.Print .ColumnHeaders(i).Text*

*Next*

*End with*

## 3. Các phương thức

**Add <Danh sách tham số>**

Thêm 1 phần tử vào danh sách ListItems hoặc ColumnHeaders.

Dạng áp dụng cho ListItems:

**Add** [Index][, Key][, Text][, Icon][, SmallIcon]

Trong đó:

Index        Vị trí thêm, nếu không có : thêm vào cuối danh sách

Key         Khoá của phần tử thêm

Text        Giá trị thêm

Icon        Biểu tượng lớn

SmallIcon  Biểu tượng nhỏ

Ví dụ 1: Thêm 5 giá trị vào ListItems không sử dụng Icon

*For i = 1 to 5*

*Listview1.ListItems.Add , , "Item No. " & i*

*Next*

Ví dụ 2: Thêm các giá trị vào listview sử dụng cả Icons và SmallIcons

*With ListView1.ListItems*

*.Add , , "Brazil", 1, 1*

*.Add , , "Italia", 2, 2*

*.Add , , "Japan", 3, 3*

*.Add , , "Usa", 4, 4*

*End with*

Dạng áp dụng cho ColumnHeaders:

**Add** [Index][, Key][, Text][, Width][, Alignment]

Trong đó:

Index        Vị trí thêm, nếu không có : thêm vào cuối danh sách

Key         Khoá của cột thêm

Text        Tiêu đề cột

Width       Bề rộng cột

Alignment  Chế độ canh lề cột

Ví dụ 1: Thêm 2 cột Name, Phone vào Listview với các độ rộng 1400, 1500

*With ListView1.ColumnHeaders*

*.Add , , "Name", 1400*

*.Add , , "Phone", 1500*

*End with*

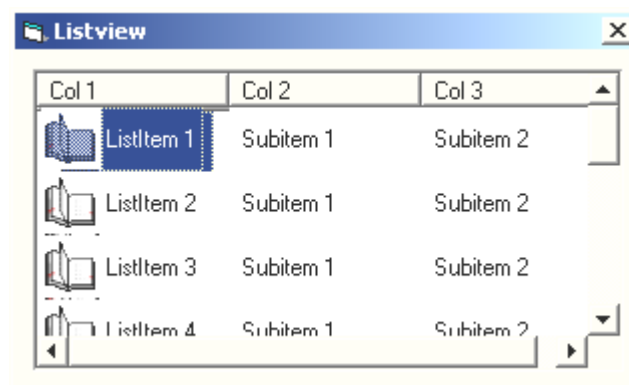
Ví dụ 2: Tạo một listview ở chế độ Report có 3 cột với tiêu đề Col 1, Col 2, Col 3 và thêm vào đó 10 phần tử

```

Private Sub Form_Load()
    Dim clm As ColumnHeader
    Dim itm As ListItem
    Dim i As Integer
    For i = 1 To 3
        Set clm = ListView1.ColumnHeaders.Add()
        clm.Text = "Col " & i
    Next i
    For i = 1 To 10
        Set itm = ListView1.ListItems.Add()
        itm.SmallIcon = 1
        itm.Text = "ListItem " & i
        itm.SubItems(1) = "Subitem 1"
        itm.SubItems(2) = "Subitem 2"
    Next i
End Sub

```

Chương trình khi chạy sẽ có dạng như hình 12.1



Hình 12.1: Form chương trình ví dụ 2

### FindItem(String, Value, Index, Match)

Tìm kiếm 1 giá trị trong danh sách ListItems, có thể qui định việc tìm kiếm được thực hiện trên thuộc tính text, subitems . Phương thức cho giá trị là tham chiếu đến đối tượng ListItem tìm thấy. Trong đó:

String	Giá trị cần tìm
Value	Qui định loại thuộc tính tìm kiếm lvwText - 0 : Tìm trên thuộc tính Text (*) lvwSubItem - 1: Tìm trên SubItem
Index	Có thể là 1 chuỗi hoặc 1 số. Khi là chuỗi, nó có ý nghĩa là Key, khi là số, nó có ý nghĩa là Index. Dùng để qui định vị trí bắt đầu tìm kiếm

Match	Qui định phương thức tìm kiếm lvwWholeWord - 0: Tìm từ toàn vẹn (*) lvwPartial - 1: Tìm 1 phần
-------	--

Ví dụ:

Tìm người có họ là Nguyễn Thị trong danh sách:

*Set It = lvw.FindItem("Nguyen Thi", , ,lvwPartial)*

### **GetFirstVisible**

Hàm cho giá trị là tham chiếu đến đối tượng đầu tiên xuất hiện trong Listview

### **Remove Index**

Xoá một phần tử tại vị trí Index

### **Clear**

Xoá tất cả các phần tử trong danh sách ListItems

## **4. Sự kiện**

### **Private Sub object\_ItemClick(ByVal Item As ListItem)**

Sự kiện xảy ra khi click trên biểu tượng hoặc hình ảnh đại diện cho đối tượng.

Tham số của thủ tục là đối tượng ListItem mà sự kiện xảy ra trên đó

Ví dụ :

```
Private Sub lvw_ItemClick(ByVal Item As ListItem)
    Item.Ghosted = Abs(Item.Ghosted) - 1
End Sub
```

Lưu ý: Trên listview cũng có sự kiện click nhưng sự kiện này xảy ra khi người dùng click tại một vùng bất kỳ trên listview.

### **Private Sub object\_ColumnClick(ByVal columnHeader As ColumnHeader)**

Sự kiện xảy ra khi click trên dòng tiêu đề của listview. Tham số của thủ tục là đối tượng ColumnHeader mà sự kiện xảy ra trên đó.

Ví dụ: Sử dụng thuộc tính Sorted và Sortkey để sắp xếp nội dung Listview theo cột

```
Private Sub lvw_ColumnClick(ByVal columnHeader As MSComctlLib.ColumnHeader)
    lvw.Sorted = True
    lvw.SortKey = columnHeader.Index - 1
End Sub
```

*End Sub*

## **III. IMAGECOMBO**

Là đối tượng điều khiển giống ComboBox nhưng có thêm hình ảnh đi kèm.

### **1. Các thuộc tính**

**ComboItems:** Thuộc tính quan trọng nhất, chứa danh sách các phần tử được trình bày trong ComboItems. Mỗi





phần tử là một đối tượng có kiểu `ComboItem`. Thuộc tính này cũng giống như `ListItems` và `ListItem` trong `Listview`. Cách truy xuất các phần tử trong `ComboItems` cũng giống như mảng

Ví dụ:

```
With ImageCombo1
```


```
    For i = 1 to .ComboItems.Count
```

```
        Debug.Print .ComboItems(i).Text
```


```
    Next
```

```
End with
```

**SelectedItem:** Cho giá trị là tham chiếu (reference) đến đối tượng `ComboItem` đang được chọn trong `ImageCombo` hoặc dùng chọn một phần tử trong `ImageCombo`

-  Chọn phần tử đầu tiên trong `ImageCombo`

```
Set ImageCombo1.SelectedItem = ImageCombo1.ComboItems(1)
```

-  Lấy giá trị phần tử đang được chọn

```
Dim icItem as ComboItem
```

```
Set icItem = ImageCombo1.SelectedItem
```

**ImageList:** Tham chiếu đến đối tượng `ImageList` quản lý danh sách hình sử dụng trong `ImageCombo`

Danh sách hình cho `ImageCombo` được định nghĩa bằng `Property Pages/General` của `ImageCombo` (Bấm phím phải rồi chọn properties)

**Locked:** (True, False) Thuộc tính qui định các giá trị xuất hiện chỉ ở trạng thái chỉ đọc

## 2. Các thuộc tính của đối tượng `ComboItem`

Thuộc tính	Ý nghĩa
Text	Chuỗi ký tự giá trị xuất hiện trong <code>ImageCombo</code>
Index	Chỉ số của phần tử trong mảng <code>ComboItems</code>
Key	Chuỗi ký tự duy nhất xác định phần tử trong danh sách <code>ComboItems</code>
SellImage	Chỉ số của hình (index) tương ứng trong danh sách hình ( <code>Imagelist</code> ) liên kết với <code>ImageCombo</code> khi phần tử được chọn
Image	Chỉ số của hình (index) tương ứng trong danh sách hình
Selected	(True/False): Cho biết phần tử có được chọn trong <code>ImageCombo</code> hay không
Indentation	Số nguyên qui định khoảng canh lề của đối tượng so với lề trái của <code>ImageCombo</code>

### 3. Các phương thức



**Add** [Index][, key][, Text][, Image][, SellImage][, Indentation]

Thêm một phần tử vào danh sách ComboItems của ImageCombo

Trong đó:

---

Index	Chỉ số của phần tử trong mảng ComboItems
Key	Chuỗi ký tự duy nhất xác định phần tử trong danh sách ComboItems
Text	Chuỗi ký tự giá trị xuất hiện trong ImageCombo
Image	Chỉ số của hình (index) tương ứng trong danh sách hình
SellImage	Chỉ số của hình (index) tương ứng trong danh sách hình (Imagelist) liên kết với ImageCombo khi phần tử được chọn
Indentation	Số nguyên qui định khoảng canh lề của đối tượng so với lề trái của ImageCombo

---

Ví dụ:

Sử dụng Add để khởi động ImageCombo trong sự kiện Form\_Load.

```
Private Sub Form_Load()
```

```
    With ImageCombo1.ComboItems
```

```
        .Add , , "Brazil", 1, , 1
```

```
        .Add , , "Italia", 2, , 2
```

```
        .Add , , "Spain", 3, , 3
```

```
        .Add , , "Usa", 4, , 4
```

```
    End With
```

```
    Set ImageCombo1.SelectedItem = ImageCombo1.ComboItems(1)
```

```
End Sub
```

#### **GetFirstVisible**

Hàm cho giá trị là tham chiếu đến đối tượng đầu tiên xuất hiện trong ImageCombo

#### **Remove Index**

Xoá một phần tử tại vị trí Index

#### **Clear**

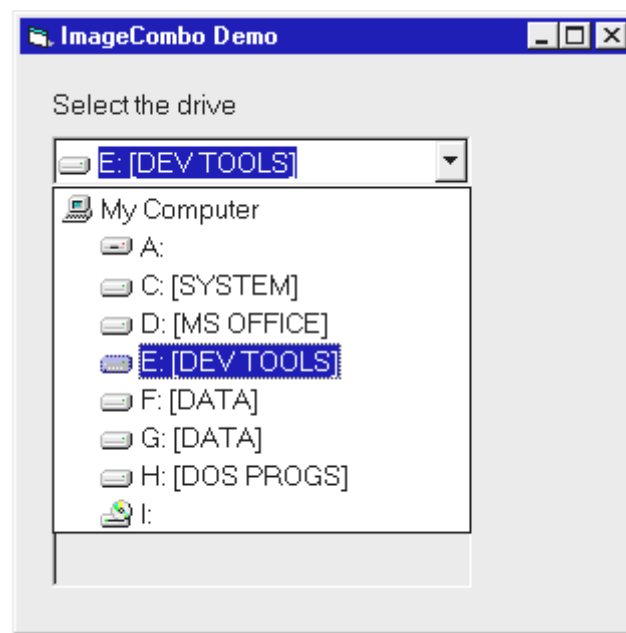
Xoá tất cả các phần tử trong danh sách ComboItems.

Ví dụ sau minh họa cách nạp các ổ đĩa (và nhãn đĩa) vào ImageCombo.

```

Sub LoadDrivesIntoImageCombo(ImgCombo As ImageCombo)
    Dim fso As New Scripting.FileSystemObject, dr As Scripting.Drive
    Dim drLabel As String, drImage As String
    ' ImageCombo phải được liên kết với một danh sách hình đã
    ' có sẵn biểu tượng các loại ổ đĩa
    ImgCombo.ComboItems.Add , , "My Computer", "MyComputer"
    For Each dr In fso.Drives
        Select Case dr.DriveType
            Case Removable: drImage = "FloppyDrive"
            Case CDRom:     drImage = "CDDrive"
            Case Else:     drImage = "HardDrive"
        End Select
        drLabel = dr.DriveLetter & ": "
        If dr.IsReady Then
            If Len(dr.VolumeName) Then drLabel = drLabel & "[" & _
                dr.VolumeName & "]"
        End If
        ImgCombo.ComboItems.Add , dr.DriveLetter, drLabel, drImage, , 2
    Next
    ' Chọn ổ đĩa hiện tại.
    Set ImgCombo.SelectedItem = ImgCombo.ComboItems(Left$(CurDir$, 1))
End Sub

```



Hình 12.2: Giao diện chương trình ví dụ

# Chương 13

## Microsoft Windows Common Controls

### Toolbar - Statusbar - DTPicker

#### I. TOOLBAR

Toolbar là thanh công cụ, được sử dụng để trình bày các chức năng thường sử dụng trong chương trình. Trên Toolbar có thể gồm các loại đối tượng sau:

- Nút bấm thường
- Nút bấm dạng Check
- Nút bấm hoạt động theo nhóm (Option Buttons)
- ComboBox hoặc TextBox

Nội dung trình bày trên nút bấm của Toolbar có thể là text hoặc hình ảnh. Hình ảnh xuất hiện trên các nút bấm của Toolbar được quản lý thông qua ImageList.

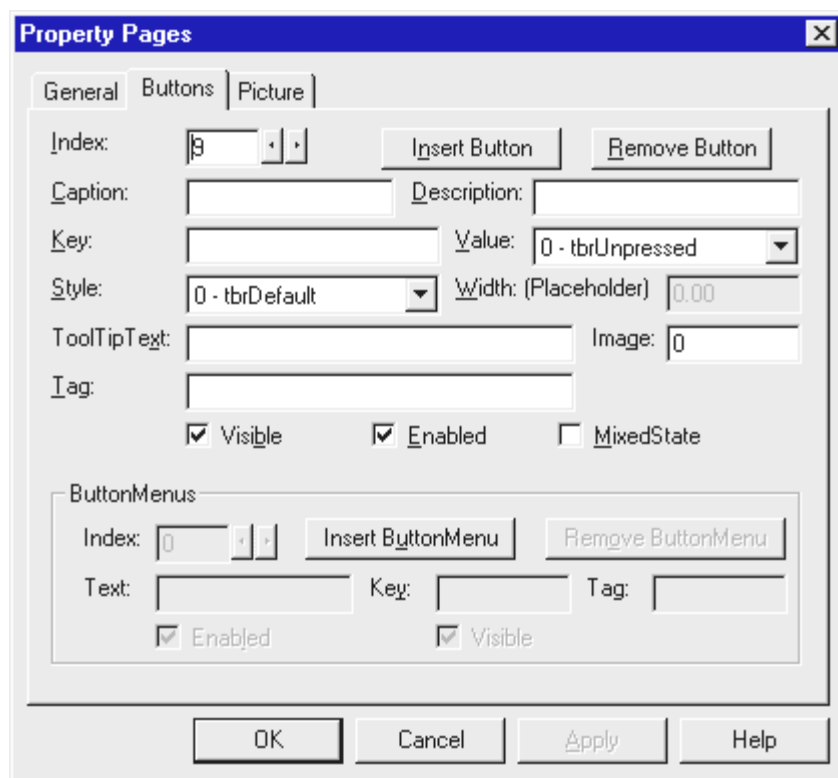
#### 1. Sử dụng Toolbar

- Nhấp đúp biểu tượng Toolbar trên ToolBox
- Nhấp phím phải trên biểu tượng Toolbar trên form
- Chọn Properties, xuất hiện hộp thoại Property Pages

#### a. Thẻ General

Qui định các thuộc tính cơ bản nhất của một Toolbar, gồm :

Thuộc tính	Ý nghĩa
<b>MousePointer</b>	Chọn dạng con trỏ .
<b>ImageList</b>	Tên Imagelist quản lý danh sách hình .
<b>BorderStyle</b>	Kiểu viền (0 - ccNone, 1 - ccFixedSingle)
<b>Appearance</b>	Dạng Toolbar (0-ccFlat, 1-cc3D)
<b>ButtonHeight</b>	Chiều cao nút bấm.
<b>ButtonWidth</b>	Chiều rộng nút bấm.
<b>AllowCustomize</b>	Cho phép thay đổi các nút trên Toolbar khi chạy chương trình
<b>Wrappable</b>	Cho phép cuộn toolbar thành nhiều hàng nút
<b>ShowTips</b>	Xuất hiện lời nhắc chức năng của nút

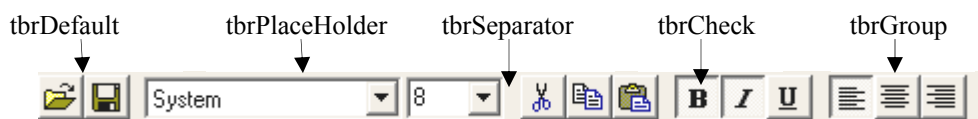


Hình 13.1: Thẻ Button

### b. Thẻ Button

Các nút bấm trên Toolbar được quản lý trong mảng Buttons (là một thuộc tính của Toolbar). Các thuộc tính của mỗi nút bấm được trình bày trong thẻ Buttons:

Thuộc tính	Ý nghĩa
<b>Index</b>	Số thứ tự của nút trên Toolbar
<b>Caption</b>	Nội dung xuất hiện trên nút
<b>Key</b>	Tên nút được sử dụng trong chương trình
<b>Value</b>	Trạng thái của nút (0 - tbrUnpressed, 1 - tbrPressed)
<b>Style</b>	Loại nút: 0 – tbrDefault, 1 – tbrCheck, 2 – tbrButtonGroup, 3 – tbrSeparator, 4 – tbrPlaceholder, 5- Dropdown
<b>ToolTipText</b>	Lời nhắc khi con trỏ di chuyển trên nút
<b>Image</b>	Chỉ số hình trong ImageList
<b>Width</b>	Độ rộng khoảng chứa chỗ trên Toolbar



Hình 13.2: Các loại nút trên Toolbar

## 2. Định nghĩa Toolbar

- Định nghĩa ImageList chứa danh sách hình sẽ sử dụng trên Toolbar,
- Đặt Toolbar lên form,
- Trong hộp thoại Property Pages của Toolbar:

### Thẻ General

- Chọn danh sách hình đã định nghĩa (thuộc tính ImageList)
- Chọn dạng thể hiện (Flat, 3D), viền/không có viền (thuộc tính Appearance)
- Chọn kích thước nút bấm

### Thẻ Buttons

- Bấm nút Insert Button để thêm nút mới. Qui định thuộc tính cho mỗi nút :
  - o Key: Chuỗi tên dùng trong chương trình
  - o Style: Loại nút: bình thường, thanh phân cách, dành chỗ cho combobox...
  - o Value: Giá trị ban đầu cho nút (nếu nút bấm loại check)
  - o Image: Chọn chỉ số hình trong ImageList
- Bấm Apply sau khi định nghĩa xong 1 nút
- Lặp lại nhiều lần để định nghĩa cho các nút khác
- Viết lệnh

Nhấp đúp trên toolbar, xuất hiện khai báo

```
Private Sub Toolbar1_ButtonClick(ByVal Button As ComctlLib.Button)
```

```
End Sub
```

Trong đó tham số Button chứa thông tin về nút được bấm trên Toolbar.

Để xác định nút được bấm, có thể sử dụng thuộc tính Index hoặc Key của Button:

```
Private Sub Toolbar1_ButtonClick(ByVal Button As ComctlLib.Button)
```

```
    Select Case Button.Key  
        Case "FileOpen"  
            Do_Open  
        Case "FileSave"  
            Do_Save
```

```
    ...
```

```
End Select
```

```
End Sub
```

Hoặc

```
Private Sub Toolbar1_ButtonClick(ByVal Button As ComctlLib.Button)
```

```
    Select Case Button.index  
        Case 1
```

```

        Do_Open
    Case 2
        Do_Save
    ...
End Select
End Sub

```

### 3. Định nghĩa nút Toolbar lúc chạy chương trình

Có thể thêm nút Toolbar lúc chạy chương trình bằng phương thức Add với có dạng như sau:

**Add** ([Index], [key], [caption], [Style], [Image]) As Button

Trong đó:

Index	vị trí nút thêm vào.
Key	Chuỗi duy nhất
Caption	Chuỗi xuất hiện trên nút.
Style	Loại nút.
Image	Chỉ số hình trong danh sách hình.

Ví dụ 1: Thêm một nút bấm hoạt động theo kiểu CheckBox lên Toolbar.

```

Dim btn As Button
Set btn = Toolbar1.Buttons.Add( , , tbrCheck, "Lock")
btn.Value = tbrPressed

```

Ví dụ 2: Thêm một nút phân cách trên Toolbar.

```

Toolbar1.Buttons.Add , , tbrSeparator

```

Ví dụ 3: Thêm hai nút hoạt động theo nhóm trên Toolbar.

```

Set btn = Toolbar1.Buttons.Add( , , tbrButtonGroup, "Green")
Set btn = Toolbar1.Buttons.Add( , , tbrButtonGroup, "Red")
btn.Value = tbrPressed

```

Ví dụ 4: Thêm khoảng trống trên Toolbar và đặt ComboBox vào khoảng trống đã tạo.

```

Dim btn As Button
Set btn = Toolbar1.Buttons.Add( , , tbrPlaceholder)
btn.Width = cboFontSizes.Width
Set cboFontSizes.Container = Toolbar1
cboFontSizes.Move btn.Left, btn.Top

```

Trường hợp nút tạo ra có Style = tbrDropDown, có thể định nghĩa các mục chọn khi người sử dụng bấm mũi tên bên phải nút bằng phương thức Add như sau:

Add ([Index], [key], [caption], [Style], [Image]) As ButtonMenu

Ví dụ: Thêm nút bấm loại Drop-down rồi tạo menu có 3 mục chọn.

```
Dim btn As Button
```

```
Set btn = Toolbar1.Buttons.Add( , , tbrDropDown, "New")
```

```
With btn.ButtonMenus
```

```
    .Add , , "File"
```

```
    .Add , , "Document"
```

```
    .Add , , "Image"
```

```
End With
```

Sự kiện ButtonMenuClick xảy ra khi mục chọn trên menu kéo xuống của nút bấm kiểu drop-down được chọn. Ví dụ sau trình bày lệnh xử lý sự kiện khi nút bấm được chọn.

```
Private Sub Toolbar1_ButtonMenuClick(ByVal ButtonMenu As
```

```
    MSComctlLib.ButtonMenu)
```

```
    Select Case ButtonMenu.Key
```

```
        Case "Document"
```

```
            Call mnuFileNewDocument
```

```
        Case "Image"
```

```
            Call mnuFileNewImage
```

```
    End Select
```

```
End Sub
```

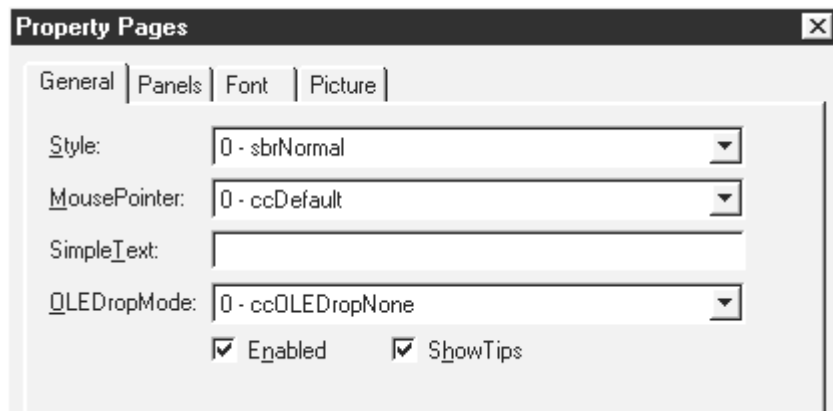
## II. STATUS BAR

Đối tượng điều khiển thường đặt phía dưới form để thông báo tình trạng hoạt động của chương trình hoặc thông báo trạng thái của các nút bấm.

### 1. Sử dụng

Đặt statusbar lên form. Click mục (Custom) tại properties windows, xuất hiện hộp thoại Property Pages.



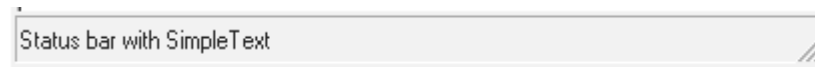


Hình 13.3: Thẻ general

### a. Thẻ General

Style            Loại Status bar (0 - sbrNormal, 1 - sbrSimple)

SimpleText    Chuỗi xuất hiện trên Toolbar khi style = 1



StatusBar dạng Simple

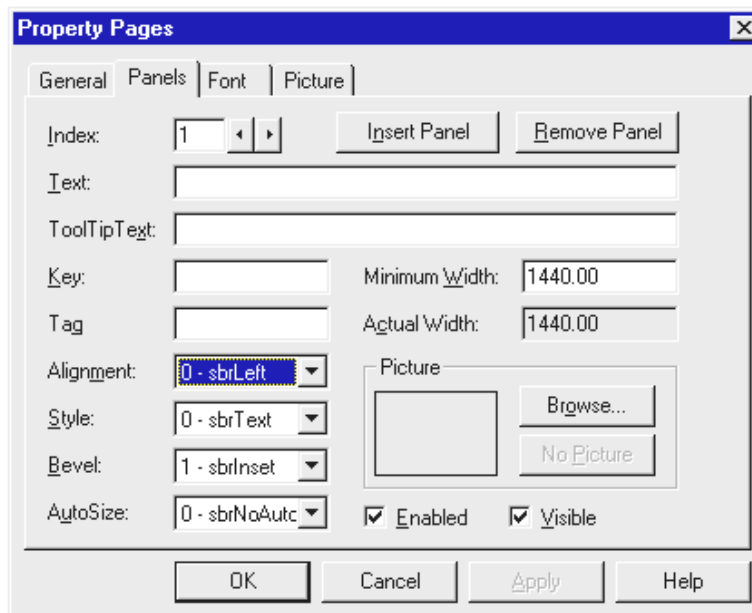


Panels

StatusBar dạng Normal

Hình 13.4: Các Style của StatusBar

### b. Thẻ Panel



Hình 13.5: Thẻ Panel

Index	Chỉ số các panel trên toolbar	
Text	Chuỗi xuất hiện trong Panel	
ToolTipText	Lời nhắc khi mouse di chuyển trên Panel	
Minimum width	Bề rộng tối thiểu của Panel	
Alignment	Dạng canh lề text trong panel (0-sbrLeft,1-Center,2-Right)	
Style	Loại Panel	3-sbrIns
	0-sbrText	4-sbrCtrl
	1-sbrCaps	5-sbrTime
	2-sbrNum	6-sbrDate
Bevel	Kiểu viền của Panel (0-sbrNoBevel,1-sbrInset, 2-sbrRaised)	
AutoSize	Tự động điều chỉnh text xuất hiện trong Panel	
	0 - sbrNoAutosize, 1-sbrSprings, 2-sbrContents	
Picture	Hình xuất hiện trong Panel	


Để thêm Panels cho StatusBar dạng Normal, bấm nút Insert Panel, gán các thuộc tính cần thiết cho Panel.

Để điều chỉnh thuộc tính của một Panel, click nút mũi tên bên phải hộp Index để chọn, điều chỉnh giá trị thuộc tính rồi bấm nút Apply.

Để xóa một Panel, bấm nút Remove Panel.

## 2. Viết lệnh cho StatusBar


Khi viết lệnh cho statusbar, phân biệt hai trường hợp:


 StatusBar có Style = sbrSimple: Sử dụng thuộc tính Simple Text

```
StatusBar1.SimpleText = "StatusBar with Simple text"
```

 StatusBar có Style = sbrNormal: Sử dụng thuộc tính Text của các Panels

```
StatusBar1.Panels(1).Text = "Edit mode"
```

 Các Panel loại sbrCaps, sbrNum, sbrIns, sbrCtrl,sbrTime.sbrDate tự động cập nhật propert text theo thời gian, trạng thái của các phím tương ứng trên bàn phím.

 Với các thông báo dài, có thể tạm thời chuyển Style thành sbrSimple để trình bày thông báo rồi chuyển trở lại Style Normal:

```
StatusBar1.Style = sbrSimple
```

```
StatusBar1.SimpleText = "Saving data to file..."
```

```
' ...
```

```
' Chuyển trở lại sbrSimple
```

```
StatusBar1.Style = sbrText
```

Có thể thêm một Panel trong chương trình bằng phương thức Add, dạng như sau:

**Add** ([Index], [Key], [Text], [Style], [Picture]) As Panel

Ví dụ: Thêm 1 Panel loại Text vào bên trái StatusBar

```
With StatusBar1.Panels.Add(1, "temporary", "Hello World", sbrText)
    .Alignment = sbrCenter
    .Bevel = sbrNoBevel
    .AutoSize = sbrContents
End With
```

Xóa một Panel bằng phương thức **Remove** với tham số là vị trí của Panel.

Ví dụ sau yêu cầu người sử dụng nhập nội dung cho Panel khi người sử dụng nhấp đúp tại Panel.

```
Private Sub StatusBar1_PanelDoubleClick(ByVal Panel As
MSComctlLib.Panel)
    Dim s As String
    If Panel.Style = sbrText Then
        s = InputBox("Enter a new text for this panel")
        If Len(s) Then Panel.Text = s
    End If
End Sub
```

Ví dụ sau tạo hình ảnh một mặt trăng xoay trên Panel. Chương trình sử dụng mảng đối tượng Image để quản lý 8 hình ảnh mặt trăng ở các vị trí khác nhau.

```
Private Sub Timer1_Timer()
    Static n As Integer
    StatusBar1.Panels("moon").Picture = imgMoon(n).Picture
    n = (n + 1) Mod 8
End Sub
```

StatusBar có thể thông báo trạng thái các phím Lock (Caps, Num...) nhưng chỉ có thể thay đổi trạng thái các phím này bằng bàn phím. Ví dụ sau sử dụng các hàm API để thay đổi trạng thái các phím Lock bằng mouse.

```
' Khai báo sử dụng hàm API
Declare Function GetKeyboardState Lib "user32" (KeyState As Byte) As
Long
Declare Function SetKeyboardState Lib "user32" (KeyState As Byte) As
Long
Private Sub StatusBar1_PanelDoubleClick(ByVal Panel As
MSComctlLib.Panel)
    Select Case Panel.Style
```

```

    Case sbrCaps: ToggleKey vbKeyCapital
    Case sbrNum: ToggleKey vbKeyNumlock
    Case sbrScrl: ToggleKey vbKeyScrollLock
    Case sbrIns: ToggleKey vbKeyInsert
End Select
StatusBar1.Refresh
End Sub

```

```

Sub ToggleKey(vKey As KeyCodeConstants)
    Dim keys(255) As Byte
    ' Đọc trạng thái hiện tại từ bàn phím.
    GetKeyboardState keys(0)
    keys(vKey) = keys(vKey) Xor 1      ' Thay đổi trạng thái
    ' Gán giá trị mới
    SetKeyboardState keys(0)
End Sub

```

### III. DTPICKER


Là đối tượng điều khiển có 3 chức năng:

- Thông báo ngày giờ theo định dạng,
- Nhập giá trị ngày giờ theo dạng và phạm vi định trước,
- Tự động kiểm tra giá trị nhập ngày giờ theo định dạng.

#### 1. Thuộc tính

CalendarBackColor,	Các thuộc tính màu nền, màu chữ lịch, màu nền và màu
CalendarForeColor,	chữ tiêu đề
CalendarTitleForeColor,	
CalendarTitleBackColor	
DayOfWeek	Giá trị ngày trong tuần (1-Chủ nhật, 2- Thứ hai,...7-Thứ bảy) của ngày đang chọn
Day, Month, Year	Giá trị ngày (1-31), tháng (1-12), năm đang chọn
MinDate, MaxDate	Các thuộc tính qui định phạm vi chọn ngày tháng
Value	Giá trị ngày đang chọn
Format	Qui định loại định dạng sử dụng. có các giá trị: 0-dtpLongDate . Ví dụ Friday, Nov 14, 1972 1-dtpShortDate. Ví dụ 11/14/1972 2-dtpTime. Ví dụ 5:31:47 PM 3-dtpCustom. Định dạng theo kiểu của người sử dụng

CustomFormat Chuỗi ký tự qui định dạng ngày giờ xuất hiện trong DTPicker. Chỉ có tác dụng khi thuộc tính Format có giá trị dtpCustom

 Các loại ký tự sử dụng trong chuỗi CustomFormat và ý nghĩa:

d	Giá trị ngày 1 hoặc 2 chữ số (1-31)
dd	Giá trị ngày 2 chữ số có chữ số 0 phía trước (01-31)
ddd	Chuỗi 3 ký tự đầu tiên viết tắt tên ngày tiếng anh (Sun, Tue, Wed...)
dddd	Chuỗi tên ngày tiếng anh (Sunday, Tuesday, Wednesday...)
h	Giá trị giờ 1 hoặc 2 chữ số (1-12)
hh	Giá trị giờ 2 chữ số có chữ số 0 phía trước (01-12)
H	Giá trị giờ 1 hoặc 2 chữ số (0-23)
HH	Giá trị giờ 2 chữ số có chữ số 0 phía trước (0-23)
m	Giá trị phút 1 hoặc 2 chữ số (0-59)
mm	Giá trị phút 2 chữ số có chữ số 0 phía trước (01-59)
M	Giá trị tháng 1 hoặc 2 chữ số (1-12)
MM	Giá trị tháng 2 chữ số có chữ số 0 phía trước (01-12)
MMM	Chuỗi 3 ký tự đầu tiên viết tắt tên tháng tiếng anh (Jan, Feb, Mar...)
MMMM	Chuỗi tên tháng tiếng anh (January, February, March...)
s	Giá trị giây 1 hoặc 2 chữ số (0-59)
ss	Giá trị giây 2 chữ số có chữ số 0 phía trước (01-59)
t	AM-PM (1 ký tự)
tt	AM-PM (2 ký tự)
X	Vùng CallBack
y	Giá trị 1 chữ số cuối của năm
yy	Giá trị 2 chữ số cuối của năm
yyy	Giá trị năm đầy đủ (4 chữ số)

Một số ví dụ về cách sử dụng thuộc tính CustomFormat như sau:

dtpDate.Format = dtpCustom	Sẽ cho kết quả: 01/25/1999 08:24:24
dtpCustomFormat = "MM/dd/yyyy hh:mm:ss"	
dtpDate.Format = dtpCustom	Sẽ cho kết quả: January/Monday/1999
dtpCustomFormat = "MMMM/dddd/yyyy"	
dtpDate.Format = dtpCustom	Sẽ cho kết quả: Thursday , February 14,
dtpCustomFormat = "dddd, MMMM dd, yyy"	2002
dtpDate.Format = dtpCustom	Sẽ cho kết quả: Thursday Feb 14, 2002

dtpCustomFormat = "dddd MMM d, yyy"

### **Vùng Callback (callback fields)**

Ngoài các loại ký tự cho trong bảng trên, người lập trình còn có thể định nghĩa thêm các vùng trong chuỗi CustomFormat với nội dung tùy ý, các vùng này được mô tả bằng chuỗi các ký tự X. Số lượng ký tự X liên tiếp xác định các vùng khác nhau trong chuỗi CustomFormat.

Ví dụ:

Chuỗi	Số vùng Callback
MMMM ddXXX yyy	1
MMMM ddXXX yyyy hh:mm:ss XXXX	2

Giá trị của các vùng Callback được xác định bằng cách viết lệnh trong sự kiện Format. Sự kiện này xảy ra khi DTPicker chuẩn bị trình bày giá trị của nó.

## 2. Sự kiện

### **Private Sub object\_Format(CallbackField As String, FormattedString As String)**

Sự kiện xảy ra trước khi DTPicker trình bày giá trị. Giá trị cần trình bày sẽ được xác định và gán cho chuỗi FormattedString

### **Private Sub object\_FormatSize(CallbackField As String, Size As Long)**

Sự kiện xảy ra trước khi Format được sử dụng để DTPicker cấp phát đủ bộ nhớ chứa chuỗi Callback. Thường viết lệnh cho sự kiện này cùng với sự kiện Format

Ví dụ:

Muốn trình bày thêm các chuỗi "st", "nd", "rd" sau giá trị ngày có dạng Thursday, February 2nd, 2002, chuỗi CustomFormat phải có dạng dddd, MMMM dXXX, yyyy. Viết lệnh cho sự kiện Format như sau:

```
Private Sub DTPicker1_Format(ByVal CallbackField As String, FormattedString As String)
```

```
    If CallbackField = "XXX" Then
```

```
        Select Case DTPicker1.Day Mod 10
```

```
            Case 1
```

```
                FormattedString = "st"
```

```
            Case 2
```

```
                FormattedString = "nd"
```

```
            Case 3
```

```
                FormattedString = "rd"
```

```
            Case Else
```

```
        FormattedString = "th"  
    End Select  
End If  
End Sub  
Private Sub DTPicker1_FormatSize(ByVal CallbackField As String, Size As  
Integer)  
    If CallbackField = "XXX" Then Size = 2  
End Sub
```

# Chương 14

## Microsoft Windows Common Controls

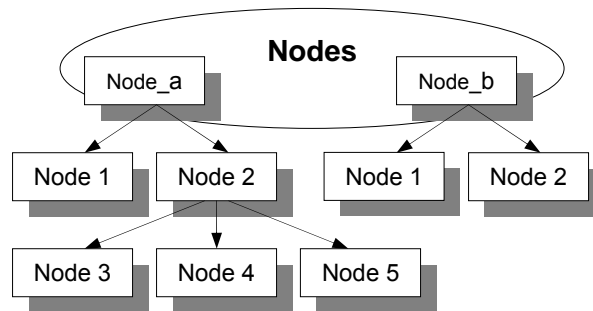
### Treeview - Updown - Slider - Progressbar

#### I. TREEVIEW

Là đối tượng điều khiển thường được sử dụng để trình bày cấu trúc tổ chức của một đối tượng như cấu trúc cây thư mục, tổ chức một cơ quan, một đơn vị

##### 1. Các thuộc tính

**Nodes:** Thuộc tính quan trọng nhất của Treeview, là một danh sách các đối tượng Node. Mỗi Node được xem là một cây con, có các nút con, nút cháu...



Hình 14.1: Đối tượng Nodes

**ImageList:** Tham chiếu đến đối tượng ImageList quản lý danh sách hình liên kết với TreeView. Gán giá trị này trong Property Pages

Hoặc gán bằng lệnh như sau:

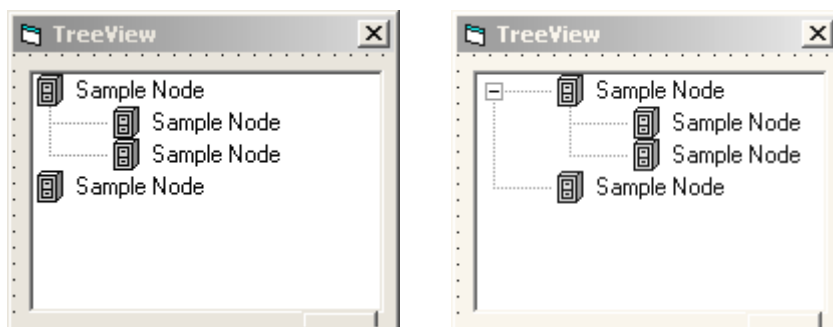
```
Private Sub Form_Load()
    Set TreeView1.ImageList = ImageList1
End Sub
```

**CheckBoxes (True/False):** Làm xuất hiện Checkbox bên trái mỗi nút

**LabelEdit:** Chế độ sửa chữa giá trị nhãn (0-tvwAutomatic, 1-tvwManual)

**LineStyle:** Chế độ vẽ đường nối giữa nút và nút cha, có hai giá trị 0-tvwTreelines, 1- twwRootLines





Hình 14.2: Thuộc tính Linestyle

**SelectedItem:** Cho giá trị là tham chiếu (reference) đến đối tượng Node đang được chọn trong TreeView hoặc dùng chọn một Node trong TreeView

🔧 Chọn nút gốc trong Treeview

```
Set Treeview1.SelectedItem = Treeview1.Nodes("Root")
```

🔧 In giá trị nút đang được chọn

```
Dim nd as Node
```

```
Set nd = Treeview1.SelectedItem
```

```
Debug.Print nd.Text
```

## 2. Các thuộc tính của đối tượng Node

**Child** Tham chiếu đến nút con đầu tiên

**Children** Số nút con

**Expanded** (True/False) Là xuất hiện các nút con của một nút - Tương đương với việc nhấp đúp tại nút hoặc click tại dấu +/- để triển khai/thu gọn một nút

**FirstSibling** Tham chiếu đến nút con đầu tiên ở cùng cấp

**LastSibling** Tham chiếu đến nút con cuối cùng ở cùng cấp

**Next** Tham chiếu đến nút con kế sau ở cùng cấp

**Previous** Tham chiếu đến nút con kế trước ở cùng cấp

**Parent** Tham chiếu đến nút cha của một nút

**Root** Tham chiếu đến nút gốc

**Sorted** Sắp xếp các nút con cùng cấp theo thứ tự alphabet

**Text** Chuỗi xuất hiện bên phải nút

**Index** Số thứ tự của nút trong mảng các nút cùng cấp

Chương trình con sau in nhãn (Text) các nút con của một nút

```
Private Sub ListChildren(pnod As Node)
```

```
Dim pnodCurrent As Node
```

```
Set pnodCurrent = pnod.Child
```

```
For i = 1 To pnod.Children
```

```
Debug.Print pnodCurrent.Text
```

‘ Tham chiếu đến nút con đầu tiên

‘ Lặp qua số nút con

‘ In thuộc tính Text

*Set pnodCurrent = pnodCurrent.Next*      ‘ Trỏ sang nút con kế tiếp

*Next*

*End Sub*

In các nút con của nút gốc

*Private sub Command1\_Click()*

*Dim nd as Node*

*Set nd = Treeview1.Nodes(“Root”)*

*ListChildren nd*

*End sub*

### 3. Phương thức

**Add** [Relative][, Relationship][, Key][, Text][, Image][, SelImage]

Thêm một nút mới vào danh sách Nodes trong Treeview. Trong đó

**Relative**      Key hoặc Index của nút mà nút mới được thêm vào  
Quan hệ của nút mới so với nút cho trong tham số relative, có các giá trị như sau:

0-tvwFirst	Nút được thêm là nút đầu tiên so với các nút cùng cấp với nút cho trong tham số relative
1-tvwLast	Nút được thêm là nút sau cùng so với các nút cùng cấp với nút cho trong tham số relative
2-tvwNext	Nút được thêm vào sau nút cho trong tham số relative
3-tvwPrevious	Nút được thêm vào trước so với nút cho trong tham số relative
4-tvwChild	Nút được thêm vào là nút con của nút cho trong tham số relative

**Key**      Chuỗi mã số duy nhất cho mỗi nút

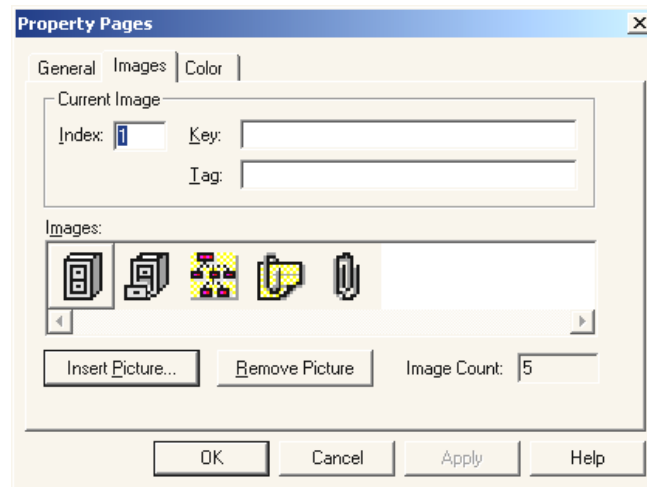
**Text**      Chuỗi xuất hiện bên phải nút

**Image**      Hình ảnh của nút ở trạng thái bình thường (Lấy từ Imagelist)

**SelImage**      Hình ảnh của nút ở trạng thái được chọn (Lấy từ Imagelist)

Ví dụ: Sử dụng đối tượng Treeview trình bày sơ đồ tổ chức của công ty Kova như hình

Sử dụng đối tượng ImageList với các hình ảnh được chọn như sau:



Hình 14.3: Thẻ general

Lệnh cho sự kiện Form\_load

*With tvw.Nodes*

*.Add , , "Root", "Kova Co.ltd", 3*

*.Add "Root", tvwChild, "KD", "P.Kinh doanh", 1, 2*

*.Add "Root", tvwChild, "KT", "P.Ke toan", 1, 2*

*.Add "Root", tvwChild, "VT", "P.Vat tu", 1, 2*

*End With*

- Lệnh *.Add , , "Root", "Kova Co.ltd", 3* tạo nút gốc với Key=Root, Text="Kova Co.ltd" sử dụng hình thứ ba (3) trong Imagelist
- Lệnh *.Add "Root", tvwChild, "KD", "P.Kinh doanh", 1, 2* tạo nút con dưới gốc với Key=KD, Text = "P.Kinh doanh" sử dụng các hình 1, 2 trong ImageList
- Lệnh *.Add "Root", tvwChild, "KT", "P.Ke toan", 1, 2* tạo nút con dưới gốc với Key=KT, Text = "P.Ke toan" sử dụng các hình 1, 2 trong ImageList
- Lệnh *.Add "Root", tvwChild, "VT", "P.Vat tu", 1, 2* tạo nút con dưới gốc với Key=VT, Text = "P.Vat tu" sử dụng các hình 1, 2 trong ImageList

Cách viết khác

*Dim nd As Node*

*Set nd = tvw.Nodes.Add( , , "Root", "Kova Co.ltd", 3)*

*Set nd = tvw.Nodes.Add("Root", tvwChild, "KD", "P.Kinh doanh", 1, 2)*

*Set nd = tvw.Nodes.Add("Root", tvwChild, "KT", "P.Ke toan", 1, 2)*

*Set nd = tvw.Nodes.Add("Root", tvwChild, "VT", "P.Vat tu", 1, 2)*

Muốn thêm 5 nhân viên vào phòng kinh doanh như hình , viết thêm lệnh như sau:

*For i = 1 To 5*

*Set nd = tvw.Nodes.Add("KD", tvwChild, , "Nhan vien #" & i, 4, 5)*

Next

*nd.Expanded = True*

Có thể viết cách khác như sau:

*Private Sub Form\_Load()*

*Dim DptStr*

*DptStr = Array("P.Kinh doanh", "P.Ke toan", "P.Vat tu")*

*Dim nd As Node, nd1 As Node*

*Set nd = tvw.Nodes.Add(, , "Root", "Kova Co.ltd", 3)*

*For i = 0 To 2*

*Set nd = tvw.Nodes.Add("Root", tvwChild, , DptStr(i), 1, 2)*

*nd.Expanded = True*

*For j = 1 To 5*

*Set nd1 = tvw.Nodes.Add(nd.Index, tvwChild, , "Nhan vien #" & j, 4, 5)*

*Next*

*Next*

*End sub*

### Remove Index

Xoá nút tại vị trí Index hoặc nút có Key cho trước

Ví dụ:

Xoá nút đang được chọn trên Treeview khi nhấp đúp trong Treeview

*Private Sub tvw\_DblClick()*

*Dim nd As Node*

*Set nd = tvw.SelectedItem*

*idx = nd.Index*

*tvw.Nodes.Remove idx*

*End Sub*

### Clear

Xoá toàn bộ cây trong Treeview

Ví dụ:

*Private Sub tvw\_DblClick()*

*tvw.Nodes.Clear*

*End Sub*

## 4. Sự kiện

### Private Sub object\_NodeClick(ByVal node As Node)

Sự kiện xảy ra khi người dùng click tại một nút trên Treeview



Ví dụ:

In giá trị Key và Text lên tiêu đề form khi người sử dụng Click tại một nút trên Treeview

```
Private Sub tvw_NodeClick(ByVal Node As Node)
    Me.Caption = "Index = " & Node.Index & " Text:" & Node.Text
End Sub
```

Ví dụ:

In tất cả các nút con khi người sử dụng Click tại một nút bằng cách sử dụng chương trình con ListChildren ở phần trên.

```
Private Sub tvw_NodeClick(ByVal Node As Node)
    ListChildren Node
End Sub
```

**II. UPDOWN**

Đối tượng điều khiển hoạt động theo kiểu thanh cuộn nhưng luôn gắn liền với một đối tượng khác để tăng hoặc giảm giá trị của đối tượng gắn liền với nó

**1. Các thuộc tính**

<b>AutoBuddy</b>	Tự động chọn đối tượng đi kèm với Updown dựa trên TabIndex, khi thuộc tính này được chọn, thuộc tính Buddy Control cũng được tự động chọn (True/False)
<b>BuddyControl</b>	Tên đối tượng đi kèm với Updown. Được tự động chọn khi Autobuddy=True
<b>BuddyProperty</b>	Qui định thuộc tính muốn thay đổi giá trị của đối tượng đi kèm với updown
<b>Min</b>	Qui định giá trị nhỏ nhất trên đối tượng khi bấm nút down
<b>Max</b>	Qui định giá trị lớn nhất trên đối tượng khi bấm nút up
<b>Increment</b>	Qui định độ tăng giá trị khi bấm nút Up/Down (Luôn dương)
<b>Value</b>	Giá trị đang chọn
<b>Wrap</b>	Giá trị chọn trong đối tượng được tự động quay về min/max khi tăng/giảm đến hết phạm vi giá trị (True/False)

**2. Sự kiện****Private Sub object\_Change()**

Sự kiện xảy ra khi Updown thay đổi giá trị

**Private Sub object\_DownClick()**

Sự kiện xảy ra bấm nút Down



```

        .ListIndex = 0
    End With
End Sub
Private Sub combo1_Click()
    Slider1.TickStyle = combo1.ListIndex
End Sub

```

**TickFrequency:** Thuộc tính qui định số khoảng vạch xuất hiện

SelectRange (True/False): Thuộc tính qui định chế độ chọn phạm vi giá trị

**SelStart:** Giá trị bắt đầu phạm vi chọn trên thanh trượt

**SelRange:** Giá trị phạm vi chọn

Thường lập trình chế độ chọn phạm vi được kích hoạt khi người dùng bấm phím SHIFT

## 2. Phương thức

**ClearSel:** Xoá phạm vi chọn trên thanh trượt

**GetNumTicks:** Phương thức cho số vạch giá trị xuất hiện trên thanh trượt

Ví dụ:

```

Private Sub Slider1_Click()
    MsgBox Slider1.GetNumTicks
    Slider1.Max = Slider1.Max + 10
End Sub

```

## 3. Sự kiện

**Private Sub object\_Scroll( )**

Sự kiện xảy ra khi thanh trượt di chuyển. Sự kiện này xảy ra trước sự kiện Click

**Private Sub object\_Change( )**

Sự kiện xảy ra sau khi thuộc tính Value đã thay đổi giá trị

Ví dụ sau minh hoạ hoạt động chọn phạm vi giá trị trên slider bằng cách bấm phím SHIFT khi click trên thanh trượt

```

Private Sub Slider1_MouseDown(Button As Integer, Shift As Integer, x As Single,
y As Single)

```

```

    If Shift = vbShiftMask Then

```

```

        ' Bấm phím SHIFT, chuyển sang chế độ chọn phạm vi.

```

```

        Slider1.SelectRange = True

```

```

        Slider1.SelLength = 0

```

```

        StartSelection = Slider1.Value
    End If
End Sub

```

```



Else
    ' Bỏ chế độ chọn phạm vi.
    Slider1.SelectRange = False
End If
End Sub
Private Sub Slider1_Scroll()
    Slider1.Text = "Value = " & Slider1.Value
    If Slider1.SelectRange Then
        If Slider1.Value > StartSelection Then
            Slider1.SelStart = StartSelection
            Slider1.SelLength = Slider1.Value - StartSelection
        Else
            Slider1.SelStart = Slider1.Value
            Slider1.SelLength = StartSelection - Slider1.Value
        End If
    End If
End Sub

```

#### IV. PROGRESSBAR

Đối tượng điều khiển trình bày tốc độ hoặc thời gian thực hiện của một hoạt động mất một khoảng thời gian khá lâu trên máy như hoạt động mở File, sao chép, truyền dữ liệu trên mạng...

##### Các Thuộc tính

<b>Min</b>	Giá trị nhỏ nhất
<b>Max</b>	Giá trị lớn nhất
<b>Value</b>	Giá trị hiện tại
<b>Scrolling</b>	Thuộc tính qui định tính chất của vạch mô tả tiến trình, có các giá trị:
0-	
ccScrollingStandard	
1-ccScrollingSmooth	

Lập trình trên ProgressBar thường gồm việc qui định các giá trị Min, Max trước khi tiến trình thực hiện. Trong quá trình thực hiện tiến trình (thường là vòng lặp), giá trị Value sẽ được cập nhật

Ví dụ sau minh họa tiến trình gán giá trị cho một mảng có 1000 phần tử

```

Dim A(1000) As Integer
Private Sub Form_Load()

```



```
With ProgressBar1
    .Min = 0
    .Max = 1000
    .Value = 0
End With
End Sub
Private Sub CmdStart_Click()
    For i = 0 To 1000
        A(i) = i
        ProgressBar1.Value = i
    Next
End Sub
```

# Chương 15

## RichTextbox – Form MDI

### I. RICHTEXTBOX

Là một loại Textbox đặc biệt, ngoài các thuộc tính cơ bản của một textbox. RichTextBox còn bao gồm các thuộc tính định dạng văn bản như:

- Định dạng font chữ,
- Định dạng đoạn văn như bullet, canh lề,
- Khả năng nhúng đối tượng (Object Embedding),
- Khả năng lưu văn bản dạng text hoặc dạng có định dạng (RTF - Rich Text Format).

RichTextBox không có sẵn trên ToolBox. Sử dụng hộp thoại Components để nạp RichTextBox lên ToolBox.

#### 1. Các thuộc tính

Các thuộc tính bổ sung so với textbox

<b>SelRTF</b>	Giống thuộc tính SelText của TextBox nhưng có thêm định dạng
<b>SelfontName</b>	Chọn font
<b>Selfontsize</b>	Chọn font size
<b>SelfontColor</b>	Chọn màu
<b>SelBold</b>	(True/False) Chọn chữ đậm
<b>SelItalic</b>	(True/False) Chọn chữ nghiêng
<b>SelUnderline</b>	(True/False) Chọn chữ gạch chân
<b>SelAlignment</b>	Canh lề cho đoạn văn, có các giá trị: Null : Phần văn bản chọn trên nhiều đoạn có trạng thái canh lề khác nhau 0 - rtfleft: Canh trái (default) 1 - rtfRight: Canh phải 2 - rtfCenter: Canh giữa
<b>SelBullet</b>	(True/False) Đánh bullet cho đoạn văn
<b>SelIndent</b>	Qui định lề trái
<b>SelRightIndent</b>	Qui định lề phải
<b>SelHangingIndent</b>	Qui định đoạn thụt vào của dòng đầu tiên trong đoạn
<b>AutoverbMenu</b>	(True/False) Cho phép xuất hiện menu popup khi bấm phím phải trên richtextbox

## 2. Các phương thức

### a. LoadFile Path, Filetype

Nạp tập tin Text hoặc RTF lên RichTextBox. Trong đó:

Path: Đường dẫn

FileType: Loại tập tin nạp (0 - rtfRTF, 1 - rtfText). Giá trị mặc định là rtfRTF

Ví dụ:

```
Private Sub mnuOpen_Click()
    On Error GoTo ErrorOpen
    With CFileDialog
        .InitDir = "C:\My Documents"
        .Filter = "Text (*.txt)|*.txt|RichText format (*.rtf)|*.rtf"
        .FilterIndex = 2
        .CancelError = True
        .ShowOpen
        rtfData.LoadFile .FileName, rtfRTF
    Exit sub
    End With
ErrorOpen:
End Sub
```

### b. SaveFile Path, Filetype

Ghi nội dung RichTextBox lên tập tin. Các tham số và tùy chọn giống LoadFile

Ví dụ:

```
Private Sub mnuSave_Click()
    On Error GoTo ErrorSave
    With CFileDialog
        .InitDir = "C:\My Documents"
        .Filter = "Text (*.txt)|*.txt|RichText format (*.rtf)|*.rtf"
        .FilterIndex = 2
        .CancelError = True
        .ShowSave
        rtfData.SaveFile .FileName, rtfRTF
    Exit sub
    End With
ErrorOpen:
    MsgBox "Not save"
End Sub
```

**c. Find(string, start, end, option)**

Tìm kiếm một chuỗi trong RichTextBox, chuỗi tìm thấy được highlight. Phương thức trả về giá trị là vị trí đầu tiên của chuỗi trong RichTextBox

Các tham số:

String	Chuỗi cần tìm
Start	Vị trí bắt đầu (vị trí đầu tiên = 0)
End	Vị trí kết thúc tìm kiếm
Option	Qui định cách thức tìm: 2 - rtfWholeword: Tìm từ trọn vẹn 4 - rtfMatchCase: Phân biệt chữ thường, chữ in Các tham số trên có thể kết hợp bằng phép OR

Ví dụ:

```
Private Sub mnuFind_Click()  
    Static p As Long  
    p = Form1.RichTextBox.Find(txtFind.Text, p + 1)  
    If p <> -1 Then  
        MsgBox "Find text at position " & p  
    Else  
        MsgBox "Search text not found"  
    End If  
End Sub
```

**d. GetLineFromChar(charpos)**

Cho giá trị là số thứ tự dòng chứa vị trí cần xét

Ví dụ:

```
Private Sub mnuFind_Click()  
    Static p As Long  
    p = Form1.RichTextBox.Find(txtFind.Text, p + 1)  
    If p <> -1 Then  
        MsgBox "Find text at line " & Form1.RichTextBox.GetLineFromChar(p)  
    Else  
        MsgBox "Search text not found"  
    End If  
End Sub
```

**II. SỬ DỤNG RICHTEXTBOX****1. Chọn dáng vẽ Font chữ (Font style) bằng nút lệnh trên Toolbar**

```
Private Sub tbr_ButtonClick(ByVal Button As MSComctlLib.Button)
```

```
Select Case Button.Index
```

```
Case 8
```

```
rtfBox.SelBold = Not rtfBox.SelBold
```

```
Case 9
```

```
rtfBox.SelItalic = Not rtfBox.SelItalic
```

```
Case 10
```

```
rtfBox.SelUnderline = Not rtfBox.SelUnderline
```

```
Case 11
```

```
rtfBox.SelAlignment = rtfLeft
```

```
Case 12
```

```
rtfBox.SelAlignment = rtfCenter
```

```
Case 13
```

```
rtfBox.SelAlignment = rtfRight
```

```
End Select
```

```
End Sub
```

## 2. Chọn Font chữ bằng lệnh trên menu và hộp thoại Font

```
Private Sub mnuFont_Click()
```

```
Cmdlg.Flags = CdlCFBoth
```

```
Cmdlg.ShowFont
```

```
With rtfBox
```

```
.SelFontName = Cmdlg.FontName
```

```
.SelFontSize = Cmdlg.FontSize
```

```
.SelBold = Cmdlg.FontBold
```

```
.SelItalic = Cmdlg.FontItalic
```

```
.SelUnderline = Cmdlg.FontUnderline
```

```
End With
```

```
End Sub
```

## 3. Sự kiện SelChange

Do có thêm tính chất của đoạn văn, font chữ, người lập trình còn viết lệnh trên sự kiện Selchange để thông báo tính chất của font chữ, trạng thái canh lề của đoạn văn... Sự kiện này xảy ra khi con trỏ chèn thay đổi vị trí hoặc thay đổi phần chọn trong RichTextBox

Ví dụ:

Chương trình soạn thảo văn bản có RichTextBox và Toolbar với các nút bấm

```
Private Sub rtfBox_SelChange()
```

```
' Trạng thái Text - Bold, Italic, Underline
```

```

    If rtfBox.SelBold Then
        tbr.Buttons(8).Value = tbrPressed
    Else
        tbr.Buttons(8).Value = tbrUnpressed
    End If
    If rtfBox.SelItalic Then
        tbr.Buttons(9).Value = tbrPressed
    Else
        tbr.Buttons(9).Value = tbrUnpressed
    End If
    If rtfBox.SelUnderline Then
        tbr.Buttons(10).Value = tbrPressed
    Else
        tbr.Buttons(10).Value = tbrUnpressed
    End If
    If rtfBox.SelAlignment = rtfLeft Then
        tbr.Buttons(12).Value = tbrPressed
    ElseIf rtfBox.SelAlignment = rtfCenter Then
        tbr.Buttons(13).Value = tbrPressed
    ElseIf rtfBox.SelAlignment = rtfRight Then
        tbr.Buttons(14).Value = tbrPressed
    Else
        For i = 12 To 14
            tbr.Buttons(i).Value = tbrUnpressed
        Next
    End If
End Sub

```

### III. SỬ DỤNG CLIPBOARD

#### 1. Sao chép vào clipboard (Copy)

```

Private Sub mnuCopy_Click()
    Clipboard.SetText rtfBox.SelRTF
End Sub

```

#### 2. Chèn dữ liệu từ Clipboard vào văn bản (Paste)

```

Private Sub mnuPaste_Click()
    rtfBox.SelRTF = Clipboard.GetText
End Sub

```

#### 3. Cắt dữ liệu vào Clipboard (Cut)

```

Private Sub mnuCut_Click()
    Clipboard.SetText rtfBox.SelRTF
    rtfBox.SelRTF = ""
End Sub

```

#### IV. SỬ DỤNG COMBO BOX CHỌN FONT CHỮ VÀ CỖ CHỮ TRÊN TOOLBAR

Nạp font chữ và cỡ chữ trong sự kiện Form\_load

```

Private Sub Form_Load()
    Dim i As Integer
    With cmbFontName
        For i = 0 to Screen.FontCount - 1
            .AddItem Screen.Fonts(i)
        Next i
        ' Set ListIndex to 0.
        .ListIndex = 0
    End With
    With cmbFontSize
        For i = 8 To 72 Step 2
            .AddItem i
        Next i
        ' Set ListIndex to 0
        .ListIndex = 1 ' size 10.
    End With
End Sub

```

#### V. MDI FORM

Là loại form được sử dụng trong các ứng dụng có giao diện đa tài liệu (Multiple Document Interface) là loại ứng dụng mà mỗi lúc trong nó có thể mở nhiều cửa sổ tài liệu khác nhau.

##### 1. Đặc điểm:

- Luôn là cửa sổ chính của một chương trình
- Mỗi chương trình chỉ có một form dạng MDI
- Cửa sổ MDI chứa các cửa sổ khác bên trong nó, các cửa sổ này chỉ nằm trong vùng làm việc của form MDI
- Khi một cửa sổ con được cực đại, kích thước của nó sẽ bằng kích thước vùng làm việc của cửa sổ MDI và tiêu đề của cửa sổ này chính là tiêu đề của cửa sổ MDI

- Khi một cửa sổ con được cực tiểu, icon của nó sẽ nằm trong MDI chứ không nằm trên taskbar.

## 2. Form con MDI

- Một form MDI có thể chứa một hoặc nhiều form con. Để định nghĩa một form là con của form MDI, đặt thuộc tính MDIChild thành True.
- Form con MDI không thể xuất hiện bên ngoài form MDI. Nếu một form con MDI được chọn làm form bắt đầu (startup) thì form MDI cũng được tự động nạp trước khi nạp form con.
- Menu thanh (nếu có) trên form con MDI sẽ trở thành menu của form MDI khi form con được kích hoạt (Active), chính vì thế thường chỉ định nghĩa Menu cho form MDI.

## 3. Các thuộc tính và phương thức bổ sung so với form thường

Thuộc tính **ActiveForm**: cho biết form con đang nhận focus trong MDIForm.

Ví dụ: Sử dụng thuộc tính ActiveForm để đóng form đang hoạt động khi chọn lệnh File/Close trên menu.

```
Private Sub mnuFileClose_Click()
    If Not (ActiveForm Is Nothing) Then Unload ActiveForm
End Sub
```

Với các ứng dụng MDI có nhiều form khác loại chẳng hạn như ứng dụng vừa soạn văn bản (kiểu Wordpad) vừa vẽ hình (kiểu Paint), phải có các thanh công cụ tương ứng cho chức năng vẽ hình và định dạng văn bản. Chương trình phải có khả năng nhận biết loại form đang hoạt động để hiển thị ToolBar tương ứng:

```
Private Sub mnuFilePrint_Click()
    If TypeOf ActiveForm Is frmDocument Then
        Tbr_Draw.Enabled = True
        Tbr_Doc.Enabled = False
    ElseIf TypeOf ActiveForm Is frmDraw Then
        Tbr_Draw.Enabled = False
        Tbr_Doc.Enabled = True
    End If
End Sub
```

Phương thức **Arrange** <Option>

Sắp xếp các cửa sổ con bên trong cửa sổ MDI

Trong đó Option có các giá trị sau:

Hằng	Giá trị	Ý nghĩa
vbCascade	0	Sắp xếp theo kiểu Cascade



---

vbTileHorizontal	1	Sắp xếp kiểu Tile theo chiều ngang
vbTileVertical	2	Sắp xếp kiểu Tile theo chiều dọc
vbArrangeIcons	3	Sắp xếp các icon đang minimize trong MDI

---

Ví dụ: Xếp sắp các cửa sổ con trong ứng dụng MDI bằng lệnh trên menu

```
Private Sub mnuTileHorizontally_Click()
```

```
    Arrange vbTileHorizontal
```

```
End Sub
```

```
Private Sub mnuTileVertically_Click()
```

```
    Arrange vbTileVertical
```

```
End Sub
```

```
Private Sub mnuCascade_Click()
```

```
    Arrange vbCascade
```

```
End Sub
```

```
Private Sub mnuArrangeIcons_Click()
```

```
    Arrange vbArrangeIcons
```

```
End Sub
```

#### 4. Nạp cửa sổ con trong form MDI

- Khi form đã được thiết kế: <Tên form>.Show
- Nạp nhiều form tương tự như form đã thiết kế

```
Dim <Tên> As New <Tên form đã thiết kế>
```

```
Load <Tên>
```

```
<Tên>.Show
```

Ví dụ:

```
Private sub mnuNew_Click()
```

```
    Dim f As frmEdit
```

```
    Load f
```

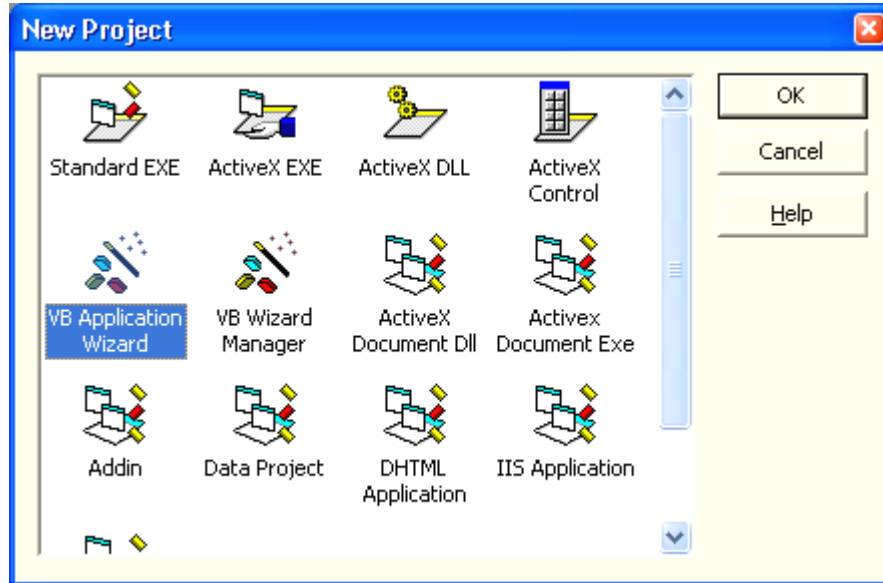
```
    f.Show
```

```
End sub
```

#### 5. Sử dụng Form Wizard tạo ứng dụng MDI

Hộp thoại New Project có chức năng tạo tự động ứng dụng MDI với Menu và Toolbar có dạng chuẩn của Microsoft. Có thể sử dụng chức năng này để tạo nhanh các ứng dụng MDI. Các bước thực hiện như sau:

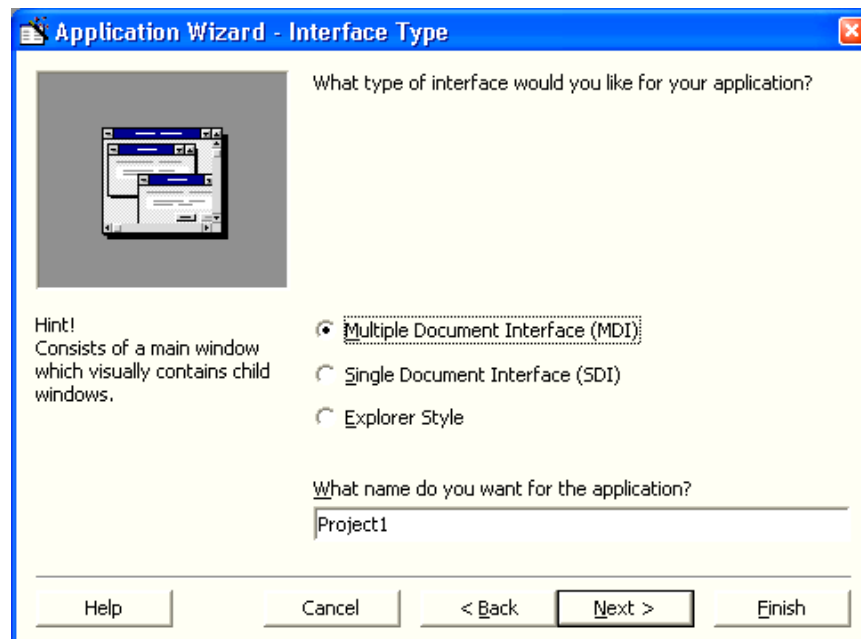
1. Chọn File/New Project xuất hiện cửa sổ New Project (Hình 15.1), chọn VB Application Wizard, bấm OK.



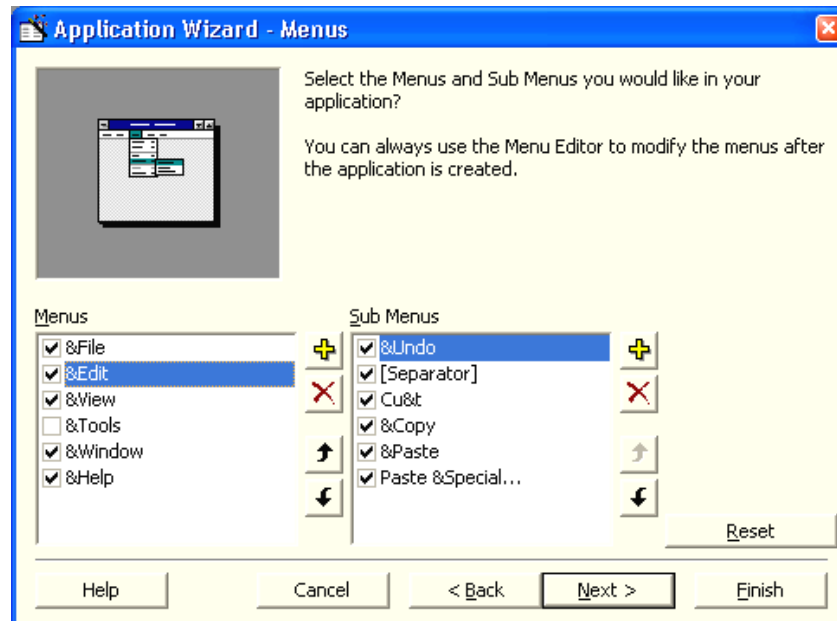
Hình 15.1: Cửa sổ chọn New Project

- Trong hộp thoại Interface Type, click chọn loại giao diện ứng dụng cần tạo là MDI rồi bấm nút Next (Hình 15.2).

Hình 15.2: Chọn loại giao diện MDI

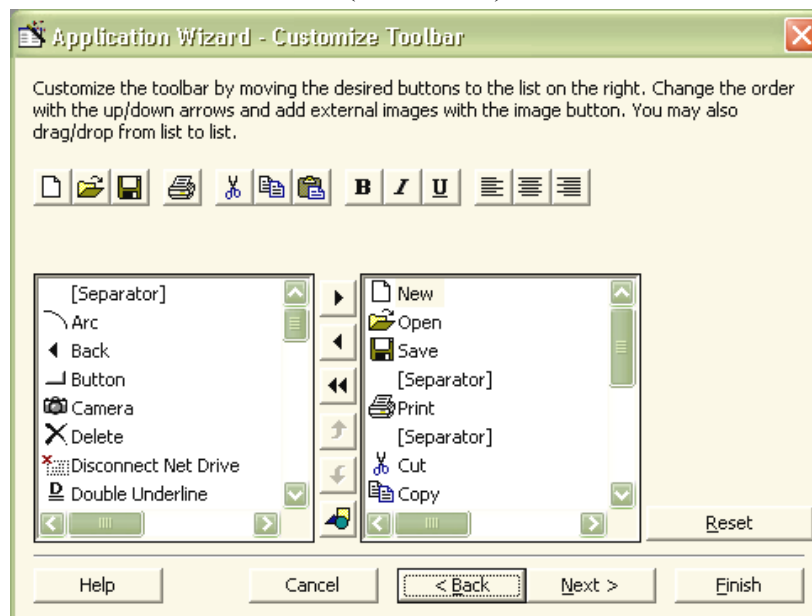


- Chọn loại menu và menu con (submenu) muốn sử dụng trong chương trình, Check để chọn lệnh muốn xuất hiện trên menu, Uncheck để bỏ lệnh xuất hiện, bấm nút ↑ hoặc ↓ để di chuyển thứ tự xuất hiện các mục trên menu rồi bấm nút Next.



Hình 15.3: Chọn menu trong Application Wizard

4. Hộp thoại Customize Toolbar xuất hiện cho phép chọn lựa loại nút lệnh muốn xuất hiện trên Toolbar (Hình 15.4).



Hình 15.4: Lựa chọn nút lệnh trên Toolbar

5. Bấm nút Finish để kết thúc.

# Chương 16

## Lập trình Drag-and-Drop

### I. TỔNG QUAN

Khả năng lập trình Drag-and-Drop (kéo-nhả) đã được hỗ trợ từ những phiên bản đầu tiên của Visual Basic, chương này trình bày kỹ thuật lập trình kéo-nhả trong Visual Basic 6.0.

#### 1. Kéo-nhả tự động

Visual Basic hỗ trợ hai chế độ kéo-nhả: tự động và bằng tay. Trong chế độ tự động, người lập trình chỉ cần gán một thuộc tính trong lúc thiết kế (design-time) hoặc trong lúc chạy chương trình (run-time), Visual Basic sẽ thực hiện mọi việc; ngược lại trong chế độ bằng tay, người lập trình phải viết lệnh trong một số sự kiện xảy ra trong lúc đối tượng đang được kéo. Chế độ lập trình bằng tay cho phép người lập trình tác động lên qui trình kéo-nhả một cách linh hoạt hơn.

Hầu hết các đối tượng điều khiển chuẩn và một số đối tượng điều khiển ActiveX hỗ trợ chế độ kéo-nhả. Một số chỉ có thể là đối tượng đích, một số khác có thể vừa là đối tượng nguồn, vừa là đối tượng đích. Chỉ có một số đối tượng có thể hoạt động ở chế độ kéo-thả tự động.

Để xác định đối tượng là nguồn trong hoạt động kéo-nhả, người lập trình sử dụng thuộc tính `OLEDragMode`. Để xác định đối tượng là đích trong hoạt động kéo-nhả, người lập trình sử dụng thuộc tính `OLEDropMode`. Bảng 16.1 tóm tắt mức độ hỗ trợ của các loại đối tượng trong hoạt động kéo-nhả.

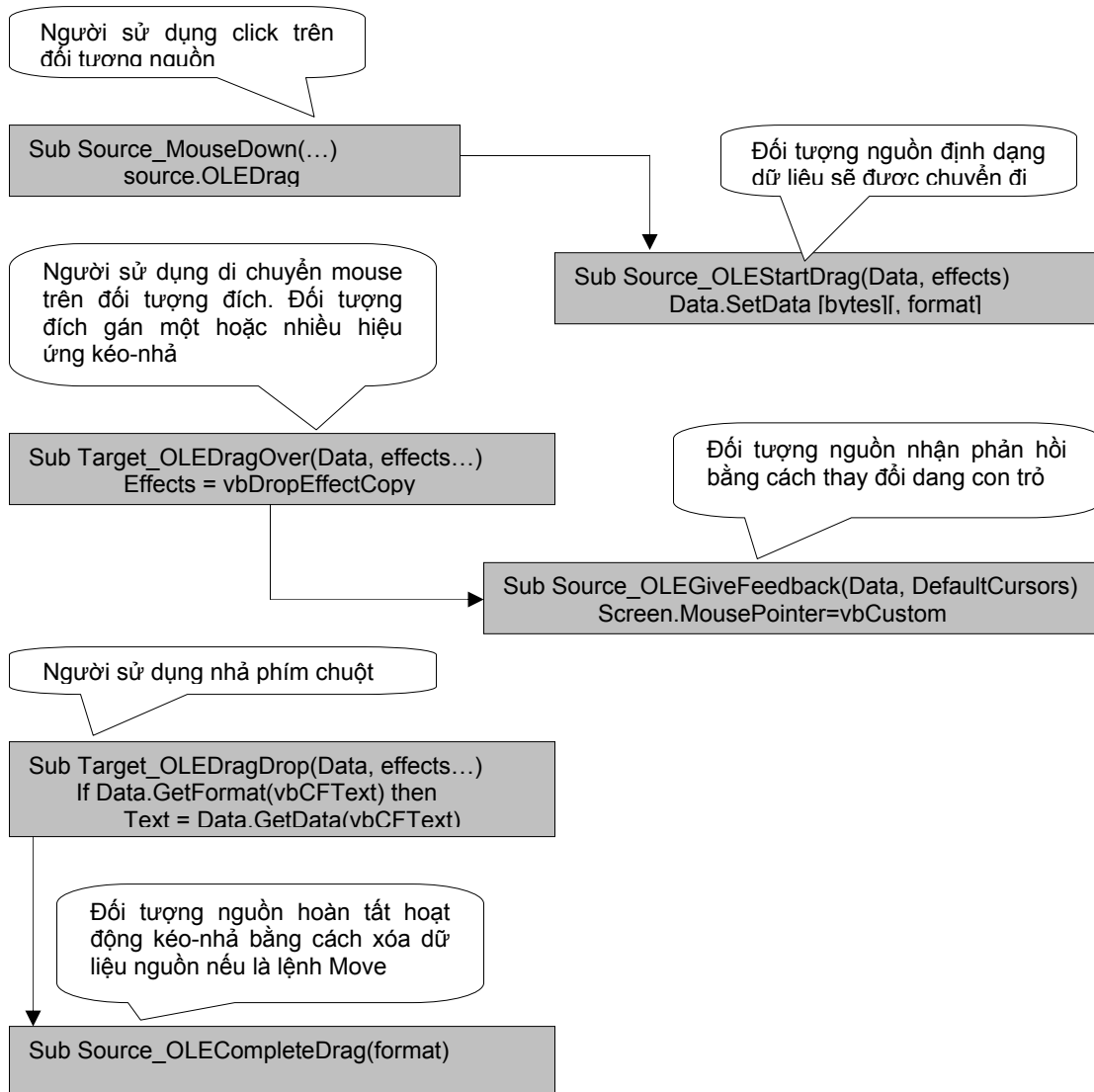
Bảng 16.1

Đối tượng	OLEDragMode	OLEDropMode
TextBox, PictureBox, Image, RichTextBox, MaskedTextBox	vbManual, vbAutomatic	vbNone, vbManual, vbAutomatic
ComboBox, ListBox, DirListBox, FileListBox, DBCombo, DBList, TreeView, ListView, ImageCombo, DataList, DataCombo	vbManual, vbAutomatic	vbNone, vbManual
Form, Label, Frame, CommandButton, DriveListBox, Data, MSFlexGrid, SSTab, TabStrip, Toolbar, StatusBar, ProgressBar, Slider, Animation, UpDown, MonthView, DateTimePicker, CoolBar	Không hỗ trợ	vbNone, vbManual

Đối với đối tượng hỗ trợ chế độ kéo-nhả tự động, để lập trình kéo-nhả, người lập trình chỉ cần gán các thuộc tính `OLEDragMode` và `OLEDropMode` có giá trị `vbAutomatic`. Ví dụ để viết một ứng dụng sử dụng `RichTextBox` cho phép nhận dữ liệu kéo-nhả từ các ứng dụng xử lý văn bản khác như MS Word hoặc WordPad, người lập trình chỉ cần gán giá trị `vbAutomatic` cho các thuộc tính `OLEDragMode` và `OLEDropMode` của `RichTextBox`.

## 2. Kéo-nhả điều khiển bằng chương trình

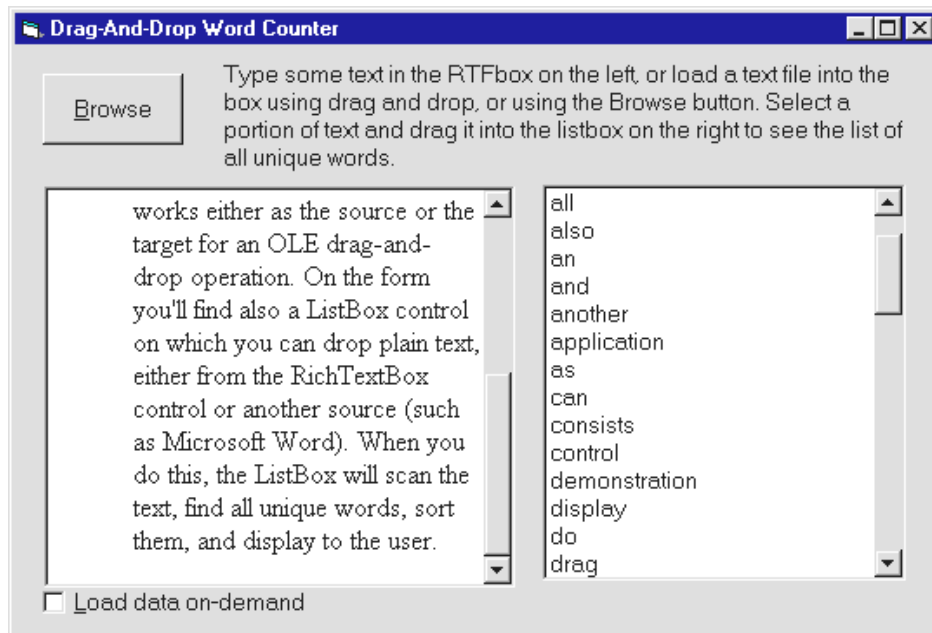
Chế độ kéo-nhả điều khiển bằng chương trình (manual drag-and-drop) bằng tay cho phép người lập trình tác động lên qui trình kéo-nhả một cách linh hoạt hơn. Hình 16.1 mô tả các bước viết lệnh trên đối tượng nguồn và đích khi muốn điều khiển hoạt động kéo-nhả bằng chương trình.



Hình 16.1: Các sự kiện được kích hoạt khi kéo-nhả điều khiển bằng chương trình

## II. MỘT CHƯƠNG TRÌNH VÍ DỤ

Phần này trình bày một chương trình ví dụ có giao diện như hình 16.2



Hình 16.2: Giao diện chương trình ví dụ

Chương trình gồm một RichTextBox được sử dụng vừa làm đối tượng nguồn, vừa làm đối tượng đích cho các hoạt động kéo-nhả. ListBox bên phải được sử dụng làm đối tượng đích cho các thao tác kéo thả. Khi nội dung từ RichTextBox hoặc từ một chương trình soạn thảo văn bản khác được kéo thả vào ListBox, nó sẽ tách từ, sắp xếp rồi đưa vào thuộc tính List như hình 16.2. Lệnh viết cho từng sự kiện theo sơ đồ hình 16.1 được lần lượt trình bày như sau:

### 1. Khởi tạo hoạt động kéo-nhả

Khởi tạo hoạt động kéo-nhả bằng cách đặt thuộc tính OLEDragMode thành vbManual rồi khởi động quá trình kéo bằng cách cho thực hiện phương thức OLEDrag trong sự kiệnMouseDown:

```
Private Sub rtfText_MouseDown(Button As Integer, Shift As Integer, _
    x As Single, y As Single)
    ' Khởi động hoạt động kéo khi phím phải được bấm
    If Button = 2 Then rtfText.OLEDrag
End Sub
```

Khi phương thức OLEDrag được gọi thực thi, sự kiện OLEStartDrag được kích hoạt trên đối tượng nguồn. Sự kiện này có tham số là đối tượng DataObject và tham số AllowedEffects. DataObject là đối tượng chứa dữ liệu chuyển giữa đối tượng nguồn và đối tượng đích. Dữ liệu cũng có thể chứa trong đối tượng này

bằng phương thức setData. Tương tự như cách sử dụng Clipboard, dữ liệu chứa có thể ở nhiều dạng khác nhau như tóm tắt trong bảng 16.2.

**Bảng 16.2:** Khai báo hằng các loại dữ liệu chứa trong Clipboard

Hằng	Giá trị	Ý nghĩa
vbCFText	1	Text
vbCFBitmap	2	Bitmap (BMP)
vbCFMetafile	3	Metafile (WMF)
vbCFEMetafile	14	Enhanced metafile (.emf)
vbCFDIB	8	Device independent bitmap (dib or bmp)
vbCFPalette	9	Color palette
vbCFFiles	15	List of files
vbCFRTF	-16639	Rich Text Format (RTF)

Ví dụ đối với RichTextBox, dữ liệu di chuyển có thể ở dạng RTF hoặc Text không có định dạng:

```
Private Sub rtfText_OLEStartDrag(Data As RichTextLib.DataObject, _
    AllowedEffects As Long)
    If rtfText.SelLength Then
        Data.SetData rtfText.SelRTF, vbCFRTF
        Data.SetData rtfText.SelText, vbCFText
    Else
        Data.SetData rtfText.TextRTF, vbCFRTF
        Data.SetData rtfText.Text, vbCFText
    End If
    AllowedEffects = vbDropEffectMove Or vbDropEffectCopy
End Sub
```

## 2. Chuẩn bị cho thao tác nhả trên đối tượng nguồn

Khi hoạt động kéo đang xảy ra, Visual Basic kích hoạt biến cố OLEDragOver trên mọi đối tượng mà mouse di chuyển ngang qua nó. Biến cố này nhận các tham số là đối tượng DataObject và giá trị Effect đã được chuẩn bị bởi đối tượng nguồn. Căn cứ trên các thông tin này, người lập trình sẽ gán cho tham số Effect giá trị tương ứng với hoạt động sẽ được thực hiện khi người sử dụng nhả chuột trên đối tượng. Giá trị effect có thể có giá trị như cho trong bảng sau:

Hằng	Giá trị
0	vbDropEffectNone
1	vbDropEffectCopy

2	vbDropEffectMove
&H80000000	vbDropEffectScroll

Giá trị cuối cùng có ý nghĩa đối tượng đích sẽ cuộn nội dung bên trong nó ví dụ khi mouse di chuyển trên mouse thanh cuộn của Listbox. Tham số trạng thái (State) chứa giá trị xác định trạng thái mouse đang di chuyển theo hướng vào hay ra khỏi hoặc di chuyển ngang qua đối tượng và có giá trị như sau:

Hằng	Giá trị
0	vbEnter
1	vbLeave
2	vbOver

Ví dụ sau làm thay đổi màu nền Listbox khi kéo mouse ngang qua Listbox

```
Private Sub lstWords_OLEDragOver(Data As DataObject, Effect As Long,
    Button As Integer, Shift As Integer, X As Single, Y As Single, State As Integer)
    If Data.GetFormat(vbCFText) Then
        Effect = Effect And vbDropEffectCopy
    Else
        Effect = vbDropEffectNone
    End If
    ' Làm thay đổi màu nền ListBox khi kéo mouse ngang qua listbox.
    If State = vbLeave Then
        ' Khôi phục màu nền khi di chuyển mouse ra khỏi Listbox
        lstWords.BackColor = vbWindowBackground
    ElseIf Effect <> 0 And State = vbEnter Then
        ' Đổi màu nền thành màu vàng khi di chuyển mouse vào Listbox
        lstWords.BackColor = vbYellow
    End If
End Sub
```

Ngay sau biến cố OLEDragOver xảy ra trên đối tượng nguồn, Visual Basic kích hoạt tiếp biến cố OLEGiveFeedback trên đối tượng này để nhận biết hoạt động kéo tác động như thế nào đối với đối tượng đích để có thể thực hiện thao tác tương ứng ví dụ dạng con trỏ mouse được thay đổi khác nhau với hoạt động sao chép hoặc di chuyển. Ví dụ sau thay đổi dạng con trỏ Custom khi thực hiện thao tác sao chép trên đối tượng đích.

```
Private Sub lstWords_OLEGiveFeedback(Effect As Long, _
    DefaultCursors As Boolean)
    ' effect là Copy, sử dụng dạng con trỏ custom.
```



```

If Effect = vbDropEffectCopy Then
    DefaultCursors = False
    Screen.MousePointer = vbCustom
    'Dạng con trỏ nạp trong đối tượng Image.
    Screen.MouseIcon = imgCopy.Picture
Else
    DefaultCursors = True
End If
End Sub

```

Cần lưu ý là nếu không cần thay đổi dạng con trỏ thì không phải viết lệnh cho biến cố OLEGiveFeedback.

### 3. Nhả trên đối tượng đích

Khi người sử dụng nhả mouse trên đối tượng đích, Visual Basic kích hoạt biến cố OLEDragDrop trên đối tượng đích. Ngoài tham số State, biến cố này nhận các tham số tương tự như OLEDragOver. Trong trường hợp này, tác dụng của tham số Effect hơi khác so với biến cố OLEDragOver vì nó thể hiện hành vi được quyết định trên đối tượng đích.

Thủ tục dưới đây minh họa lệnh viết trong biến cố OLEDragDrop trên ListBox.

```

Private Sub lstWords_OLEDragDrop(Data As DataObject, Effect As Long,
    Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' Khôi phục màu nền ListBox.
    lstWords.BackColor = vbWindowBackground
    ' Xác định giá trị tham số Effect: sao chép hay di chuyển.
    If Effect And vbDropEffectCopy Then
        Effect = vbDropEffectCopy
    ElseIf Effect And vbDropEffectMove Then
        Effect = vbDropEffectMove
    End If
    ' Trong cả hai trường hợp, chỉ nhận dữ liệu dạng Text
    Dim text As String
    text = Data.GetData(vbCFText)
    ' Lệnh xử lý Text và nạp vào ListBox
    ...
End Sub

```

Ngay sau khi biến cố OLEDragDrop xảy ra, Visual Basic kích hoạt biến cố OLECompleteDrag. Người lập trình phải viết lệnh trong biến cố này để hoàn tất

thao tác đã thực hiện trên đối tượng nguồn chẳng hạn như xóa phần text trong đối tượng nguồn nếu là hoạt động di chuyển (effect = vbDropEffectMove) hoặc khôi phục dữ liệu trên đối tượng nguồn nếu là hoạt động sao chép. Thủ tục dưới đây minh họa lệnh viết trong biến cố OLECompleteDrag trên Listbox.

```
Private Sub rtfText_OLECompleteDrag(Effect As Long)
    If Effect = vbDropEffectMove Then
        ' Nếu là di chuyển thì xóa phần Text chọn.
        rtfText.SelText = ""
    Else
        ' Nếu là sao chép thì thôi chọn.
        rtfText.SelLength = 0
    End If
End Sub
```

#### 4. Nạp dữ liệu theo yêu cầu

Khi phương thức GetData của đối tượng DataObject trên đối tượng đích để nhận dữ liệu ở một dạng nào đó, Visual Basic kích hoạt biến cố OLESetData trên đối tượng nguồn. Lệnh viết cho biến cố OLESetData trên RichTextbox của ví dụ trên như sau:

```
Private Sub rtfText_OLESetData(Data As RichTextLib.DataObject, _
    DataFormat As Integer)
    If DataFormat = vbCFText Then
        If rtfText.SelLength Then
            Data.SetData rtfText.SelText, vbCFText
        Else
            Data.SetData rtfText.text, vbCFText
        End If
    ElseIf DataFormat = vbCFRTF Then
        If rtfText.SelLength Then
            Data.SetData rtfText.SelRTF, vbCFRTF
        Else
            Data.SetData rtfText.TextRTF, vbCFRTF
        End If
    End If
End Sub
```

#### 5. Kéo-nhả File

Windows Explorer hỗ trợ kéo thả tập tin và nhiều ứng dụng windows có thể làm đối tượng đích cho hoạt động kéo-thả từ windows explorer. Phần này trình bày cách thực hiện thao tác kéo-thả tập tin.

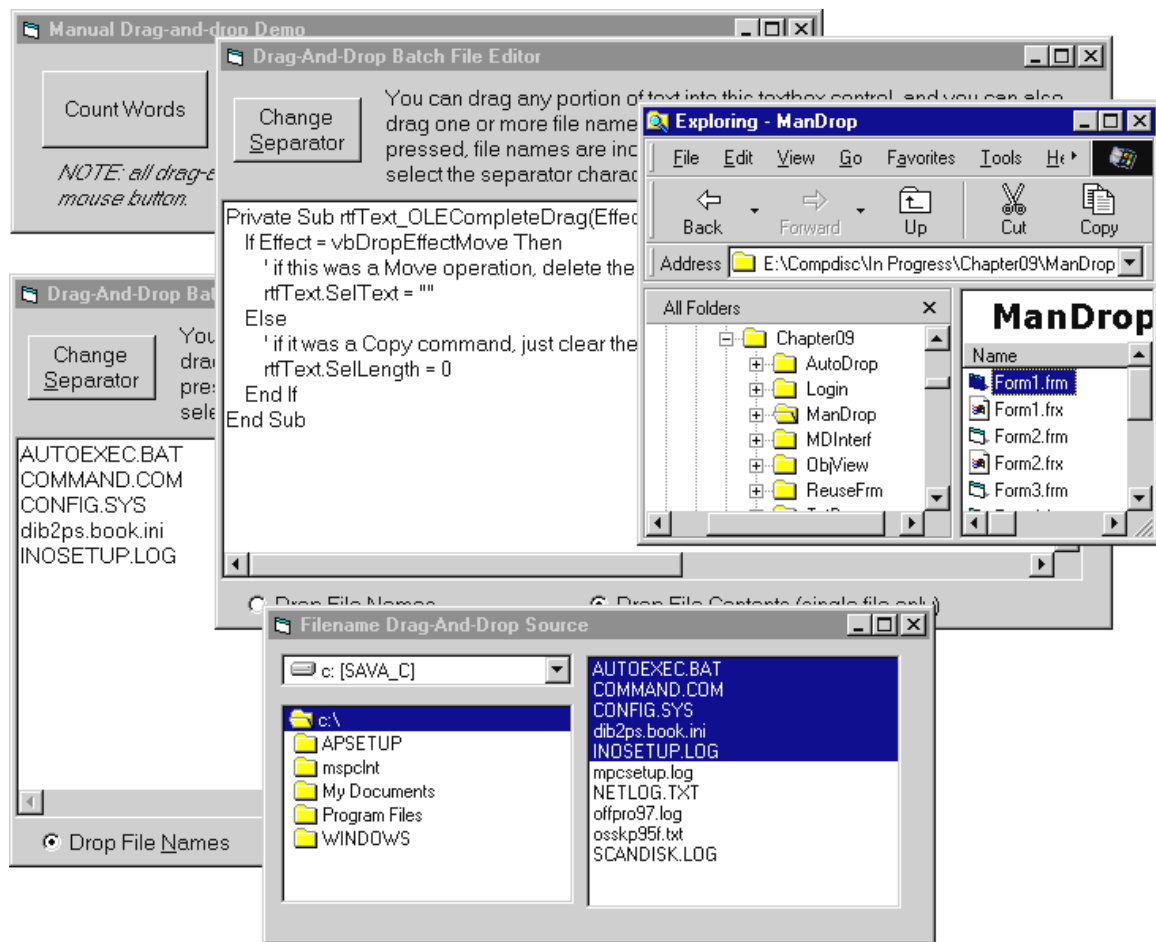
Yếu tố chính của hoạt động này là thuộc tính files của đối tượng DataObject. Nếu người lập trình muốn ứng dụng làm đối tượng đích cho hoạt động kéo-thả file thì phải kiểm tra xem đối tượng DataObject có chứa dữ liệu ở dạng vbCFFile hay không. Ví dụ sau minh họa cách nạp tên tập tin khi người sử dụng thả tập tin trên Listbox :

```
If Data.GetFormat(vbCFFiles) Then
  For i = 1 To Data.Files.Count
    lstFiles.AddItem Data.Files(i)
  Next
End If
```

Sự kiện OLEStartDrag viết sau đây minh họa đối tượng FileListBox làm đối tượng nguồn cho hoạt động kéo thả:

```
Private Sub File1_OLEStartDrag(Data As DataObject, AllowedEffects As
Long)
  Dim i As Integer, path As String
  path = File1.path & IIf(Right$(File1.path, 1) <> "\", "\", "")
  Data.Files.Clear
  For i = 0 To File1.ListCount - 1
    If File1.Selected(i) Then
      Data.Files.Add path & File1.List(i)
    End If
  Next
  If Data.Files.Count Then
    Data.SetData , vbCFFiles
    AllowedEffects = vbDropEffectCopy
  End If
End Sub
```

Hình 16.3 là ví dụ giao diện chương trình thực hiện chức năng kéo thả tập tin. Cửa sổ phía trên trình bày nội dung tập tin AutoDrop.vbp được kéo-thả từ Windows Explorer, còn cửa sổ phía dưới trình bày danh sách tập tin được thả từ hộp thoại File ở bên phải.



Hình 16.3: Kéo nhả file