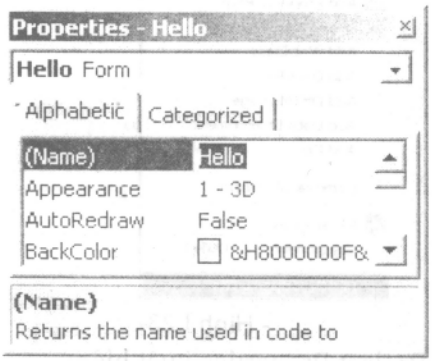
VI.5. Lưu và chạy Form

Tương tự như Project, ta cũng có thể đổi tên hiển thị của Form, bằng cách thay đổi thuộc tính Name trong cửa sổ Properties của Form như hình:



- Hình I.22 -

Ví dụ ở đây ta đổi tên hiển thị của Form là Hello (tên cũ là BEGIN), lúc này tên của cửa sổ Properties của Form cũng trở thành Hello.

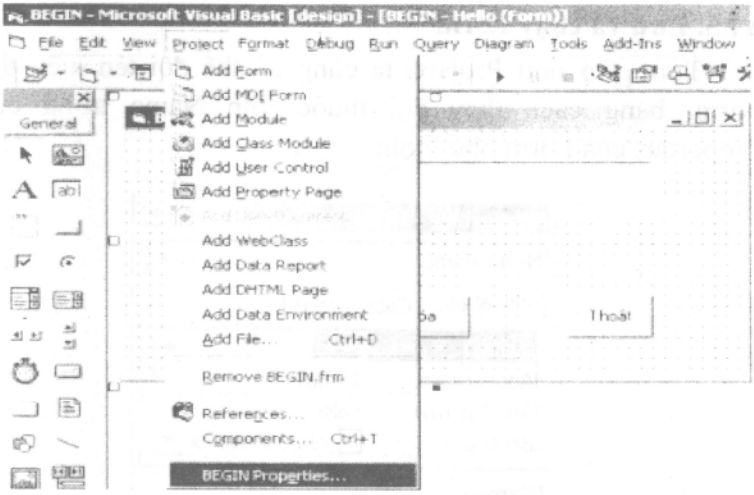
Sau khi tạo xong Form, ta nên lưu Form trước khi chạy. Cách lưu đã trình bày ở trên.

Để xem kết quả của Form. Ta tiến hành biên dịch và cho thực thi chương trình. Các bước được tiến hành như sau:

- Chọn Form muốn chạy:

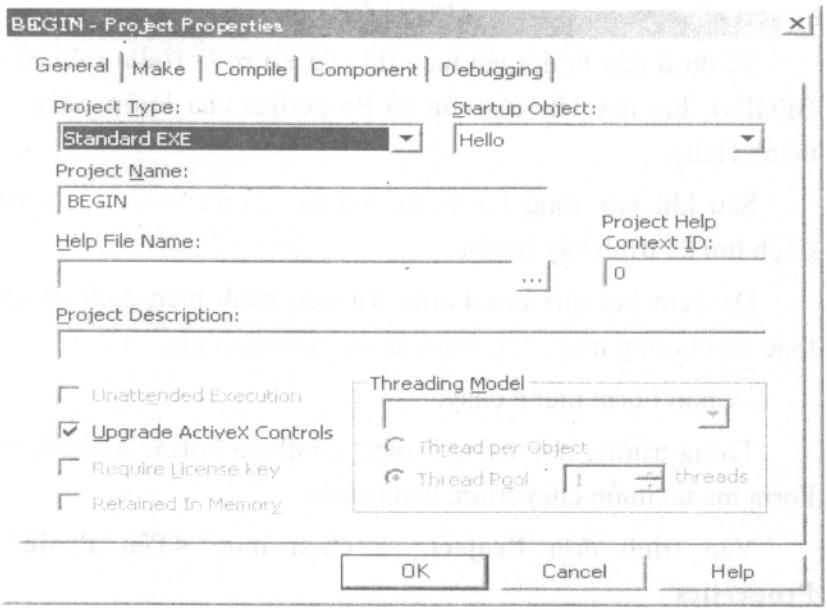
Trong trường hợp trong Project có nhiều Form, ta phải chọn Form mà ta muốn chạy trước bằng cách:

Vào trình đơn **Project** → chọn mục **<Tên Project> Properties**



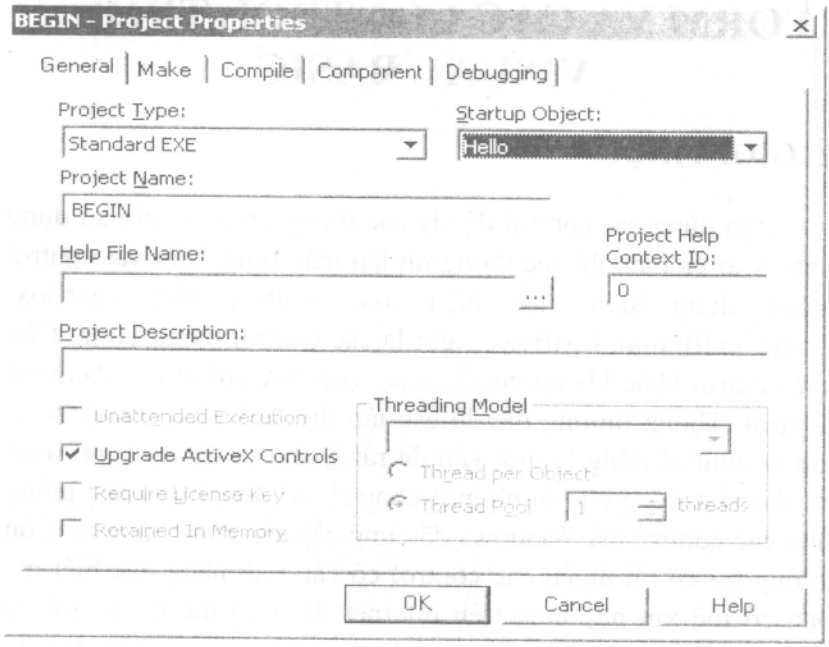
- Hình I.23-

Hộp thoại Project Properties xuất hiện:



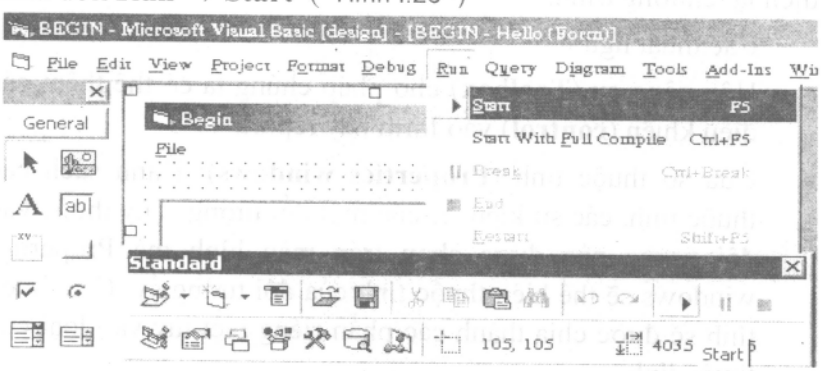
- Hình I.24 -

Trong mục **Startup Object** ta chọn Form muốn chạy. Trong ví dụ này, ta chọn Form có tên là Hello



- Hình I.25 –

- Cho thực thi Form vừa chọn bằng cách dùng Menu: chọn trình đơn **Run → Start** (- Hình I.26 -)



Chương II

FORM VÀ CÁC CONTROL TRONG VISUAL BASIC

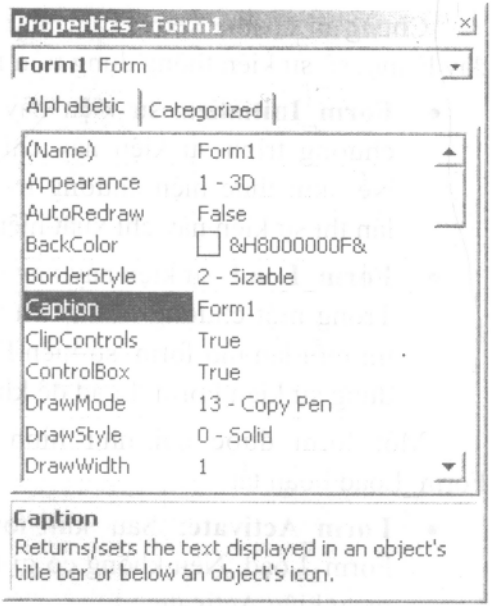
I. GIỚI THIỆU

Bạn dùng các control để lấy các thông tin do người sử dụng nhập vào và hiển thị các thông tin lên màn hình. Một vài control thông dụng luôn xuất hiện trên toolbox như TextBox, CommandButton, ListBox ... gọi là các control thông dụng. Còn các control khác không có sẵn trên Toolbox gọi là các ActiveX control. Thông thường đây là các tập tin ActiveX (*.ocx). Việc tạo ra control riêng là một vấn đề rất lý thú trong lập trình. Bạn có thể viết riêng cho mình một control có các chức năng giống như các control mà Windows đã cung cấp sẵn cho chúng ta. Còn không thì có rất nhiều các control có các tính năng đặc biệt mà bạn có thể xin hay mua trên internet. Bạn có thể tải về rồi sử dụng trong chương trình của mình như các control khác. Do đó bạn có thể update chương trình của mình mà không cần phải biên dịch lại chương trình.

Các thuật ngữ

- Hộp công cụ (**Toolbox**) cho phép chúng ta có thể thêm các điều khiển (**control**) vào form hay report.
- Cửa sổ thuộc tính (**Properties windows**): Danh sách các thuộc tính, các sự kiện ... của một đối tượng. Tùy thuộc vào đối tượng nào được chọn trên màn hình mà Properties windows sẽ thể hiện thuộc tính của đối tượng ấy. Các thuộc tính sẽ được chia thành các phần riêng biệt tùy vào loại của thuộc tính.

Hình II. 1: Hộp Toolbox chứa các control



Hình II. 2: Properties windows

Chúng ta có thể tác động đến một control thông qua các đặc tính sau:

- Thuộc tính (Properties): tập hợp các thuộc tính của control mà ta có thể thiết lập khi thiết kế hay khi chạy chương trình.
- Phương thức (Method): Những gì control thực hiện được.
- Sự kiện (Event): những sự kiện mà control sẽ thông báo cho chương trình biết khi nó xảy ra với control. Khi một sự kiện xảy ra, chương trình sẽ tự động xử lý một hàm sự kiện (Event_Handler). Ví dụ : khi bạn click vào một CommandButton có tên Command1 thì chương trình tự động thực hiện các hành động trong hàm Command1_Click()

Chúng ta có thể xem Form như một control đặc biệt. Sau đây là một số sự kiện thông dụng trên form.

- **Form_Initialize**: sự kiện này xảy ra trước nhất. Trong chương trình sự kiện này chỉ xảy ra duy nhất một lần. Nếu khi thực hiện chương trình ta đóng mở form nhiều lần thì sự kiện này chỉ xuất hiện ở lần mở form đầu tiên.
- **Form_Load**: sự kiện này xảy ra mỗi khi ta khởi mở form. Trong một chương trình, nếu ta đóng mở form nhiều lần thì mỗi lần mở form, sự kiện Form_Load sẽ xuất hiện. Ta dùng sự kiện Form_Load để khởi tạo biến, control v.v..

Một form được coi như hình thành xong khi sự kiện Form_Load hoàn tất.

- **Form_Activate**: Sau khi form xử lý xong sự kiện Form_Load. Nếu không có gì thay đổi, form sẽ phát sinh ra sự kiện Activate.
- **Form_QueryUnload**: Khi người sử dụng nhấn vào nút X phía trên bên phải để đóng form thì nó tạo ra sự kiện QueryUnload như sau:

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)

End Sub
```

Trong đó tham số UnloadMode cho biết ai, hay tác vụ nào đóng form.

Ta có thể bắt sự kiện này để hỏi lại người sử dụng có chắc chắn đóng form hay không? Nếu người sử dụng chưa muốn đóng form thì ta thiết lập tham số Cancel bằng 1.

II. CÁC THAO TÁC CƠ BẢN TRÊN FORM

II.1. Thêm một control vào form

Muốn thêm một control vào form, chúng ta chỉ việc chọn biểu tượng của control tương ứng trên Toolbox, sau đó di chuyển

chuột đến vị trí mong muốn trên form và kéo chuột vẽ một hình chữ nhật bằng đúng kích thước của control sẽ được tạo.

II.2. Chọn nhiều control

Khi muốn di chuyển, xóa, thay đổi thuộc tính của nhiều control một lúc, đầu tiên chúng ta phải chọn nhiều control. Sau đó thực hiện các tác vụ trên. Bạn có thể chọn nhiều control một lúc bằng một trong các cách sau:

- Chọn từng đối tượng bằng cách click chuột vào các control trong khi nhấn giữ phím Shift.
- Sử dụng control Pointer trên ToolBox, sau đó kéo chuột tạo thành một hình chữ nhật, khi đo chúng ta sẽ chọn được các control trong phạm vi hình chữ nhật.

II.3. Thay đổi kích thước, di chuyển, xóa ... control

Bạn có thể thay đổi kích thước, di chuyển, xóa ... một control bằng một trong các cách sau:

- Nhấp chọn control và sử dụng chuột để thay đổi kích thước, di chuyển, xóa ... các control bằng cách đưa chuột vào các cạnh của các control và kéo chuột (thay đổi kích thước, di chuyển) hay nhấn phím Del (xóa).
- Chọn control sau đó thay đổi các thuộc tính Left, Top, Width, Height... trong cửa sổ Properties.

II.4. Thiết lập lại thuộc tính cho các control

Bạn có thể chỉnh lại font chữ, kích thước chữ, canh lề, màu sắc, bóng mờ, dòng chữ hiển thị ... của các control thông qua cửa sổ thuộc tính (Properties). Bạn có thể ẩn hay hiện cửa sổ này bằng menu View/ Properties Window hay phím F4.

II.5. Thứ tự Tab của các control

Khi nhập dữ liệu vào form, bạn có thể di chuyển focus giữa các control bằng phím Tab. Muốn thiết lập thứ tự này bằng cách sử dụng thuộc tính Tab Order của control.

III. CÁC THUỘC TÍNH CHUNG

III.1. Thuộc tính Name

Mỗi control trong form có một thuộc tính Name (tên) để phân biệt với các control khác. Thuộc tính này còn được dùng để truy xuất đến control đó. Cách đặt tên cho control bạn nên tuân theo qui tắc đặt tên của Visual Basic (ví dụ: trên form có một control textbox cho người dùng nhập họ lót thì bạn đặt thuộc tính name cho textbox này là "txtHoLot").

III.2. Thuộc tính định dạng

Thuộc tính	Giải thích
Alignment	Canh lề cho dòng chữ (Text, Caption) trong control
Appearance	Xác định control là 3D hay không?
ForeColor	Màu chữ trên control
BackColor	Màu nền trên control
Font	Font của control
Enable	Nếu thuộc tính này có giá trị FALSE, control sẽ không sử dụng được, dù có thể nhìn thấy.
Visible	Ẩn hay hiện control.
Top	Qui định góc trái trên của control
Left	
Width	Qui định kích thước control
Height	
ToolTextTip	Ghi chú nội dung của control. Dòng chữ này sẽ hiện ra khi ta dừng chuột ngay trên control.

III.3. Thuộc tính giá trị (Value)

Khi thao tác với các control, bạn muốn lấy hay gán giá trị cho một control. Tùy theo loại control mà chúng ta có thể lấy các giá trị của control qua các thuộc tính khác nhau. Bảng dưới đây mô tả thuộc tính giá trị cho mỗi control.

Control	Thuộc tính giá trị
Label	Caption
TextBox	Text
OptionButton	Value
CheckBox	Value
Frame	Caption
ListBox	Text
ComboBox	Text
HScrollBar	Value
VScrollBar	Value
DriveListBox	Path
FileListBox	FileName
Line	Visiable
Shape	Shape
PictureBox	Picture
Image	Picture

Tuy nhiên các thuộc tính này được sử dụng thường xuyên nên Visual Basic cho phép chúng ta có thể dùng tên của control để truy cập đến thuộc tính giá trị.

Ví dụ:

txtHoLot.Text = "Nguyễn Thùy" có thể được viết như sau

txtHoLot = "Nguyễn Thùy"

IV. GIỚI THIỆU CÁC CONTROL CƠ BẢN TRONG TOOLBOX

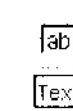
IV.1. Pointer (Đối tượng lựa chọn)

Đối tượng này cho phép chúng ta có thể chọn một hay nhiều đối tượng trên màn hình thiết kế. Sau khi chọn đối tượng xong , chúng ta có thể cho các đối tượng đó di chuyển, thay đổi các thuộc tính v..v... Chúng ta có thể chọn nhiều đối tượng bằng cách kéo chuột từ góc trái trên đến góc phải dưới, khi đó, các đối tượng thuộc phạm vi hình chữ nhật đó xem như được chọn. Ngoài ra chúng ta có thể chọn lần lượt các control bằng cách giữ phím SHIFT.

IV.2. Label

Mục đích của Label là hiển thị một đoạn văn bản lên form (Label.Caption=đoạn-văn-bản). Label khác với textbox ở chỗ nó không cho người sử dụng nhập một đoạn văn bản vào như là textbox.

IV.3. TextBox



TextBox được sử dụng để nhập vào một đoạn văn bản hay để hiển thị một đoạn văn bản.

IV.3.1. Thuộc tính Text

Thuộc tính này chứa nội dung của TextBox. Thông thường bạn có thể nhập vào tối đa 2048 ký tự vào TextBox. Nếu bạn

thiết lập thuộc tính MultiLine bằng True, bạn có thể nhập vào 32K ký tự.

IV.3.2. Thuộc tính MultiLine

Khi thuộc tính MultiLine bằng True, Visual Basic sẽ tự động xuống hàng khi phần nội dung của TextBox quá dài.

IV.3.3. Thuộc tính Alignment

Thuộc tính này sẽ điều chỉnh chế độ canh lề của chuỗi là canh lề trái (0 - Left Justify), Giữa (1 – Center), hay Phải (2 - Right Justify).

IV.3.4. Chọn khối trong TextBox

Bạn có thể xác định và xử lý khối văn bản được chọn trong TextBox bằng các thuộc tính sau:

- **SelLength** — trả về hay thiết lập số ký tự trong khối văn bản được chọn.
- **SelStart** — trả về hay thiết lập vị trí bắt đầu của khối văn bản được chọn; xác định vị trí của điểm chèn nếu không có văn bản nào được chọn.
- **SelText** — trả về hay thiết lập nội dung của đoạn văn bản được chọn. Nếu không có ký tự nào được chọn, giá trị của thuộc tính này là chuỗi rỗng("").

Các thuộc tính này không có giá trị khi form đang ở chế độ thiết kế.

Đôi khi 3 thuộc tính này dùng với đối tượng Clipboard của Windows. Đối tượng Clipboard có các thuộc tính và phương thức sau:

- **SetText**: gán giá trị vào Clipboard.
- **GetText**: có giá trị là chuỗi ký tự đang có trong Clipboard.
- **Clear**: xóa nội dung có trong Clipboard.

Ví dụ:

```
Clipboard.SetText  Text1.SelText
Text1.SelText = Clipboard.SetText
Clipboard.Clear
```

IV.3.5. Tạo một TextBox để nhập password

Một TextBox cho phép người sử dụng đánh password vào, các ký tự được nhập vào sẽ được thay thế bằng một ký tự nào đó, ký tự * chẳng hạn.

Để thực hiện điều này bạn set thuộc tính PasswordChar trong cửa sổ Properties bằng một ký tự (ví dụ ký tự *) (ký tự này sẽ hiển thị lên TextBox khi người dùng nhập password). Thuộc tính MaxLength, xác định số ký tự nhiều nhất mà bạn có thể nhập vào ô TextBox, nếu bạn nhập nhiều hơn chiều dài qui định, Visual Basic sẽ phát ra tiếng Beep và không nhận thêm ký tự nào nữa.

IV.3.6. Lọc phím trong TextBox

Bạn có thể dùng sự kiện KeyPress để điều khiển dữ liệu nhập vào trong TextBox. Thủ tục xử lý sự kiện KeyPress dùng 1 tham số là **KeyAscii**. Tham số này là một số nguyên tượng trưng cho một mã số ASCII của ký tự được đánh vào.

Ví dụ: Bạn muốn người sử dụng chỉ được nhập số vào trong TextBox. Khi đó bạn sử dụng phương pháp như sau: nếu ký tự được nhập không thuộc phạm vi số thì gán tham số KeyAscii là 0. TextBox trong ví dụ này có thuộc tính name là txtEnterNums.

```
Private Sub txtEnterNums_KeyPress (KeyAscii As Integer)
        If KeyAscii < Asc("0") Or KeyAscii > Asc("9") Then
            KeyAscii = 0   ' Hủy bỏ phím được nhập vào.
            Beep           ' Phát âm thanh beep báo lỗi.
        End If
End Sub
```

Tuy nhiên sự kiện KeyPress chỉ phát sinh khi có các ký tự được đánh vào, còn các ký tự điều khiển thì sự kiện này không phát sinh. Muốn bắt được sự kiện nhấn các ký tự điều khiển bạn sử dụng sự kiện KeyDown, KeyUp.

Thủ tục xử lý sự kiện KeyDown và KeyUp có hai tham số là KeyCode và Shift. Tham số KeyCode sẽ xác định phím nào được nhấn, còn tham số Shift sẽ có giá trị như sau:

Hằng	Giá trị	Mô tả
vbShiftMask	1	Mặt nạ bit phím SHIFT
VbCtrlMask	2	Mặt nạ bit phím CTRL
VbAltMask	4	Mặt nạ bit phím ALT

Ví dụ:

Khi người sử dụng nhấn {F1} thì Key Code =111 và Shift=0

Khi người sử dụng nhấn {Ctrl+F1} thì Key Code = 111 và Shift = 2

Khi người sử dụng nhấn {Ctrl+Shift+F1} thì Key Code = 111 và Shift = 3 (1+2=3)

IV.3.7. Tạo một TextBox chỉ hiển thị

Khi bạn muốn người sử dụng không thể nhập hay thay đổi dữ liệu trong TextBox, bạn thiết lập thuộc tính Locked bằng True. Với thuộc tính Locked bằng True, lệnh Copy vẫn hoạt động trong khi lệnh Cắt (Cut)và Dán (Paste) không hoạt động với TextBox. Thuộc tính chỉ có hiệu quả về mặt giao diện người dùng (*user interaction*) khi chạy chương trình. Bạn vẫn có thể thay đổi nội dung của thuộc tính Text của the TextBox bằng lệnh.

IV.3.8. In dấu nháy kép (") trong TextBox

Thỉnh thoảng bạn có thể đưa lên TextBox một dòng có dấu nháy kép như sau.

She said, "You deserve a treat!"

Bởi vì trong Visual Basic chuỗi được xác định là các ký tự được bao bởi dấu nháy kép (") nên chúng ta khó hiển thị dấu nháy kép (") vào trong TextBox với phương pháp bình thường. Muốn hiển thị dấu nháy kép trong chuỗi thì ta sử dụng 2 dấu nháy kép liên tục. Ví dụ, muốn gán câu trên cho một TextBox ta viết như sau:

```
Text1.Text = "She said, ""You deserve a treat!"" "
```

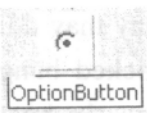
Hay bạn có thể dùng ký tự thứ 34 trong bảng mã ASCII để thể hiện dấu nháy kép:

```
Text1.Text = "She said, " & Chr(34) + "You deserve a treat!" & Chr(34)
```

IV.4. CommandButton

CommandButton là một nút ta có thể nhấn vào (sự kiện Click). Khi đó nó sẽ thực hiện một hành động nào đó kéo theo.

IV.5. OptionButton



OptionButton còn gọi là RadioButton. Các OptionButton thường được nhóm thành một nhóm. Điều này bởi vì trong một nhóm, chỉ có duy nhất một OptionButton được chọn. Khi một OptionButton được chọn thì các OptionButton còn lại trong nhóm sẽ mất chọn. Nhóm này thường được bao bởi một Frame. Muốn đặt OptionButton (RadioButton) lên một Frame ta chỉ việc cắt (cut) một đối tượng OptionButton (RadioButton) và dán (paste) lên Frame trên. Hay ta có thể chọn Frame, sau đó vẽ một OptionButton lên đó.

Muốn biết một OptionButton có nằm trong Frame nào đó hay không ta chỉ việc di chuyển Frame, nếu OptionButton nào bị kéo theo thì OptionButton thuộc Frame đó.

Giá trị trị trả về của một OptionButton có thể được lấy thông qua thuộc tính Value của nó. Nếu OptionButton được chọn (Checked) thì thuộc tính Value sẽ có giá trị là True. Ngược lại nếu OptionButton không được chọn (Uncheck) thì thuộc tính Value sẽ có giá trị là False.

Thuộc tính Value có thể được sử dụng để gán giá trị cho OptionButton.

IV.6. CheckBox

Các thuộc tính và cách sử dụng của CheckBox cũng giống như OptionButton. Các giá trị trả về cũng giống như OptionButton. Tuy nhiên chúng ta chú ý một số điểm khác nhau giữa CheckBox và OptionButton như sau:

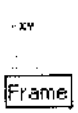
CheckBox là control cho phép chúng ta có thể chọn hay không chọn nhiều CheckBox. Đặc điểm của CheckBox này khác với OptionButton.

Điểm khác biệt thứ hai so với Option là thuộc tính Value của CheckBox có đến 3 giá trị trạng thái có thể có là:

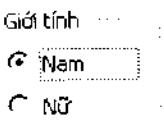
Trạng thái	Giá trị	Hằng
Unchecked	0	vbUnchecked
Checked	1	vbChecked
Unavailable	2	vbGrayed

Trạng thái Grayed của CheckBox được dùng theo chủ ý riêng của người lập trình.

IV.7. Frame



Control này cho phép chúng ta thể hiện một nhóm các lựa chọn. Control được dùng để gom nhóm một số control khác. Thông thường Frame dùng để gom các OptionButton lại thành một nhóm. Khi đó chúng ta chỉ có thể chọn duy nhất một OptionButton trong nhóm mà thôi. Nếu chúng ta đã chọn một OptionButton, sau đó chọn OptionButton khác thì OptionButton đã được chọn trước đó sẽ không được chọn nữa. Ví dụ một cách dùng của Frame:



IV.8. ListBox

ListBox được sử dụng để hiển thị một loạt các thông tin dưới dạng các hàng, mỗi hàng được gọi là một Item. Người sử dụng chỉ có thể chọn mà không được phép nhập giá trị vào trong ListBox. Chúng ta có thể chọn nhiều item trong listbox.

Sau đây là các thuộc tính, phương thức và cách sử dụng quan trọng của ListBox:

IV.8.1. Thuộc tính Style

Thuộc tính Style có 2 giá trị tương ứng với 2 loại ListBox như sau:

- **Standard** (Style = 0): Đây là giá trị mặc định cho ListBox.
- **Check Box** (Style = 1): Loại này có thêm một check box bên trái mỗi item trong danh sách (list).

IV.8.2. Thuộc tính List

Đây là tập hợp các Item trong ListBox. Đơn giản hơn có thể xem đây là mảng các chuỗi hiển thị trong ListBox. Chỉ số của mảng này bắt đầu từ 0. Để truy xuất giá trị của Item thứ index ta sử dụng cú pháp sau:

<Tên ListBox>.List(<index>)

Ví dụ: bạn muốn lấy giá trị của Item thứ ba (index=2) trong ListBox và gán cho TextBox, bạn viết lệnh như sau:

```
Text1.Text = List1.List(2)
```

IV.8.3. Thuộc tính ListIndex

Thuộc tính ListIndex cho biết vị trí item được chọn trên list. Thuộc tính này dùng để lấy hay gán chỉ số (index) của item đang được chọn của ListBox. Thuộc tính này chỉ có tác dụng khi chương trình đang chạy (run time). Khi chúng ta thiết lập thuộc tính ListIndex cho ListBox, Visual Basic cũng đồng thời tạo ra sự kiện Click cho ListBox.

Giá trị của thuộc tính này bằng 0 nếu item đầu tiên được chọn, 1 nếu item kế tiếp được chọn, ... ListIndex bằng – 1 nếu không có item nào được chọn.

Chú ý Thuộc tính NewIndex cho phép chúng ta có thể biết được item cuối cùng nào được thêm vào ListBox. Thuộc tính này rất hữu ích khi chúng ta chèn một item vào một list có thứ tự (sorted list).

IV.8.4. Thuộc tính ListCount

Thuộc tính này xác định số lượng các Item trong ListBox. Ví dụ đoạn lệnh sau sẽ cho chúng ta biết số lượng của các item trong ListBox thông qua TextBox:

```
Text1.Text = "Bạn có " & List1.ListCount & " Item trong ListBox"
```

IV.8.5. Thuộc tính Text

Thuộc tính Text sẽ cho biết giá trị một Item mà người sử dụng đang chọn. Thông thường, cách dễ nhất để lấy giá trị của item đang được chọn trong ListBox bằng thuộc tính Text.

IV.8.6. Thuộc tính ItemData

Thuộc tính ItemData là một mảng số long. Số phần tử của mảng ItemData bằng số phần tử trong List. Thuộc tính này dùng để thêm các dữ liệu phụ bên trong ListBox và ComboBox.

IV.8.7. Sự kiên Click và Double-Click

Sự kiện Click sẽ chọn một Item trên ListBox.

Chúng ta giả sử có một ListBox và một CommandButton dùng với ListBox trên cùng trên một Form. Thủ tục xử lý sự kiện Click của CommandButton sẽ làm xuất hiện một MessageBox thông báo Item nào trong ListBox được chọn.

Bạn muốn khi Double-clicking trên list có tác dụng tương đương với việc chọn một Item và click vào CommandButton. Để có thể thực hiện được công việc trên bạn viết sự kiện Click của ListBox như sau:

```
Private Sub List1_DblClick ()
    Command1_Click
End Sub
```

IV.8.8. Thêm một Item vào ListBox

Khi thiết kế form, chúng ta có thể đặt các item vào trong ListBox bằng cách sử dụng thuộc tính List. Ta nhập từng item vào ô ListBox, mỗi item cách nhau bởi một hàng. Để xuống hàng, ta sử dụng Ctrl+Enter.

Để thêm một Item vào một ListBox khi chương trình đang chạy (run-time), chúng ta dùng phương thức AddItem:

<Tên ListBox>.AddItem *item*[, *chỉ số*]

Tham số	Mô tả
Item	Chuỗi cần thêm vào list.
Chỉ số	Chỉ định vị trí item mới sẽ được thêm vào list. Nếu ta không chỉ định tham số này, Item sẽ được thêm vào cuối (hay bị trí thích hợp nếu list được sắp xếp).

Đoạn lệnh sau sẽ thêm tên bốn quốc gia "Germany," "India," "France," và "USA" vào ListBox có tên List1 khi form chứa ListBox được khởi tạo:

```
Private Sub Form_Load ()
  List1.AddItem "Germany"
  List1.AddItem "India"
  List1.AddItem "France"
  List1.AddItem "USA"
End Sub
```

IV.8.9. Thuộc tính Sorted

Bạn có thể chỉ định cho các Item đã có trong ListBox được sắp xếp theo thứ tự alphabet bằng cách thiết lập thuộc tính Sorted thành True. Việc sắp thứ tự này sẽ không phân biệt chữ hoa và chữ thường, ví dụ, từ "japan" và "Japan" là như nhau.

IV.8.10. Xóa một Item trong ListBox

Bạn có thể dùng phương thức RemoveItem để xóa một item trong ListBox. RemoveItem cần một tham số là *index* (vị trí nào sẽ bị xóa):

<Tên ListBox>.RemoveItem *index*

Các tham số này cũng như trong phần AddItem

Ví dụ: để xóa Item đầu tiên trong ListBox ta có đoạn lệnh sau:

```
List1.RemoveItem 0
```

Để xóa tất cả các Item trong ListBox chúng ta sử dụng phương thức Clear.

Ví dụ: List1.Clear

Khi xóa một Item trong ListBox, chỉ số index của các Item sẽ thay đổi. Ví dụ: Nếu chúng ta xóa Item có index bằng 0, thì Item có index bằng 1 sẽ chuyển sang index mới là 0...

IV.8.11. Tạo ListBox có nhiều cột và ListBox có nhiều sự lựa chọn (Multiple-Selection)

Thuộc tính Columns cho phép bạn xác định số cột trong một ListBox. Thuộc tính này có thể có các giá trị sau.

Giá trị	Giải thích
0	ListBox một cột với thanh cuộn dọc.
1	ListBox một cột với thanh cuộn ngang.
>1	ListBox nhiều cột với thanh cuộn dọc.

Bạn có thể cho phép người sử dụng có thể chọn nhiều item trong ListBox bằng cách thiết lập thuộc tính MultiSelect mang một trong các giá trị sau.

Giá trị	Hằng chuỗi	Giải thích
0	None	ListBox bình thường
1	Simple multiple selection	Click hay nhấn SPACEBAR để chọn hay không chọn một item trong list.
2	Extended multiple selection	Dùng SHIFT+ click hay SHIFT+ phím mũi tên để chọn các Item ở giữa vị trí chọn trước đó và vị trí chọn hiện thời. CTRL+ click để chọn hay không chọn một item trong list.

IV.8.12. Xác định mục được chọn trong một MultiSelect ListBox

Bạn có thể sử dụng thuộc tính Selected của ListBox để kiểm tra xem một mục nào đó có được chọn hay không? Nếu mục nào được chọn thì giá trị **<Tên ListBox>.Selected (index)** có giá trị là True, ngược lại sẽ có giá trị là False.

IV.9. ComboBox

ComboBox được sử dụng để hiển thị một loạt các thông tin dưới dạng các hàng, mỗi hàng được gọi là một Item. Tuy nhiên ComboBox chỉ hiện ra một dòng, bạn muốn xem các dòng khác thì click vào nút mũi tên ở bên cạnh. Khi đó sẽ có một **list** hiện ra bên dưới. ComboBox cho phép người sử dụng nhập giá trị vào.
Một số thuộc tính và phương thức đáng chú ý:

IV.9.1. Thuộc tính Style

Thuộc tính Style có 3 giá trị tương ứng với 3 loại ComboBox như sau:

- **Drop-down ComboBox** (Style = 0): Đây là giá trị mặc định cho ComboBox, một drop-down ComboBox là một ComboBox bình thường. Người sử dụng có thể hoặc đánh chuỗi văn bản trực tiếp vào (giống như textbox) hay chọn vào mũi tên chỉ xuống ở bên phải của ComboBox để chọn từ danh sách.
- **Simple ComboBox** (Style = 1): Loại này giống Drop-Down ComboBox, tuy nhiên có thêm một ListBox bên dưới. Tuy nhiên ListBox này luôn hiện ra, không cần người sử dụng phải click vào dấu mũi tên. Người sử dụng có thể nhập chuỗi trực tiếp hay chọn từ ListBox bên dưới. Cũng giống như Drop-Down ComboBox, Simple

ComboBox có thể cho người sử dụng nhập vào một giá trị không có trong ListBox.

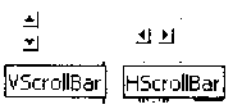
- **Drop-down List** (Style = 2) loại này giống như một ListBox ở chỗ chỉ cho phép người sử dụng chọn các giá trị có trong List. Tuy nhiên, List trong Drop-down ListBox chỉ hiện ra khi người sử dụng nhấn vào mũi tên ở bên phải ComboBox.

IV.9.2. Sự kiện Change

Sự kiện này xảy ra khi người sử dụng nhập một giá trị vào ComboBox. Chú ý: khi người sử dụng chọn một item trong phần list thì nội dung của ComboBox sẽ thay đổi nhưng sự kiện Change sẽ không phát sinh.

Các thuộc tính và sự kiện khác giống như ListBox.

IV.10. VScrollBar và HScrollBar



Đối tượng này hơi khác với các đối tượng mà chúng ta đã khảo sát ở trên. Các đối tượng này được tự động thêm vào TextBox, ComboBox, MDI Form... khi kích thước dữ liệu lớn hơn kích thước định sẵn trên màn hình. Hai control này được gọi tên chung là ScrollBar.

Trong ScrollBar có các thành phần như sau:

ScrollBar có các sự kiện như sau:

Event	Description
Change	Xảy ra sau khi scrollbox được di chuyển
Scroll	Xảy ra khi scrollbox được di chuyển. sự kiện này không xảy ra khi hai mũi trên (scroll arrow) hay thanh trống (scroll bar) bị nhấn.

IV.11. Timer

Control dùng để hiển thị tự động thực hiện một công việc nào đó sau một khoảng thời gian nhất định.

IV.12. DriveListBox

Control dùng để hiển thị và chọn các ổ đĩa của máy.

IV.13. FileListBox

Control dùng để hiển thị và chọn các tập tin trong một thư mục.

IV.14. Line

Control này dùng để vẽ một đường thẳng trên report hay form. Mục đích của control này là để trang trí.

IV.15. Shape

Control này dùng để vẽ một đa giác trên report hay form. Mục đích của control này là để trang trí.