MỤC LỤC

Chương Một - Hoan nghênh đến với VB6

Chào mừng bạn đến với Visual Basic 6

Dùng VB6 là cách nhanh và tốt nhất để lập trình cho Microsoft Windows. Cho dù bạn là chuyên nghiệp hay mới mẻ đối với chương trình Windows, VB6 sẽ cung cấp cho bạn một bộ công cụ hoàn chỉnh để đơn giản hóa việc triển khai lập trình ứng dụng cho MSWindows.

Visual Basic là gì? Phần **"Visual"** đề cập đến phương pháp được sử dụng để tạo giao diện đồ họa người dùng (Graphical User Interface hay viết tắc là GUI) . Có sẵn những bộ phận hình ảnh, gọi là controls, bạn tha hồ sắp đặt vị trí và quyết định các đặc tính của chúng trên một khung màn hình, gọi là form. Nếu bạn đã từng sử dụng chương trình vẽ chẳng hạn như Paint, bạn đã có sẵn các kỹ năng cần thiết để tạo một GUI cho VB6. Phần **"Basic"** đề cập đến ngôn ngữ BASIC (Beginners All-Purpose Symbolic Instruction Code), một ngôn ngữ lập trình đơn giản, dễ học, được chế ra cho các khoa học gia (những người không có thì giờ để học lập trình điện toán) dùng.

Visual Basic đã được ra từ MSBasic, do Bill Gates viết từ thời dùng cho máy tính 8 bits 8080 hay Z80. Hiện nay nó chứa đến hàng trăm câu lệnh (commands), hàm (functions) và từ khóa (keywords). Rất nhiều commands, functions liên hệ trực tiếp đến MSWindows GUI. Những người mới bắt đầu có thể viết chương trình bằng cách học chỉ một vài commands, functions và keywords. Khả năng của ngôn ngữ này cho phép những người chuyên nghiệp hoàn thành bất kỳ điều gì nhờ sử dụng ngôn ngữ lập trình MSWindows nào khác.

Người mang lại phần "Visual" cho VB là ông Alan Cooper. Ông đã gói môi trường hoạt động của Basic trong một phạm vi dễ hiểu, dễ dùng, không cần phải chú ý đến sự tinh xảo của MSWindows, nhưng vẫn dùng các chức năng của MSWindows một cách hiệu quả. Do đó, nhiều người xem ông Alan Cooper là cha già của Visual Basic.

Visual Basic còn có hai dạng khác: **Visual Basic for Application (VBA)** và **VBScript**. VBA là ngôn ngữ nằm phía sau các chương trình Word, Excel, MSAccess, MSProject, .v.v.. còn gọi là Macros. Dùng VBA trong

MSOffice, ta có thể làm tăng chức năng bằng cách tự động hóa các chương trình.

VBScript được dùng cho Internet và chính Operating System. Dù cho mục đích của bạn là tạo một tiện ích nhỏ cho riêng bạn, trong một nhóm làm việc của bạn, trong một công ty lớn, hay cần phân bố chương trình ứng dụng rộng rãi trên thế giới qua Internet, VB6 cũng sẽ có các công cụ lập trình mà bạn cần thiết.

Các ấn bản Visual Basic 6

Có ba ấn bản VB6: Learning, Professional và Enterprise. Chúng ta hãy gát qua ấn bản Learning. Bạn có thể dùng ấn bản Professional hay Enterprise.

Ấn bản Professional cung cấp đầy đủ những gì bạn cần để học và triển khai một chương trình VB6, nhất là các control ActiveX, những bộ phận lập trình tiền chế và rất hữu dụng cho các chương trình ứng dụng (application programs) của bạn trong tương lai. Ngoài đĩa compact chính cho VB6, tài liệu đính kèm gồm có sách Visual Studio Professional Features và hai đĩa CD Microsoft Developer Network (MSDN).

Ấn bản Enterprise là ấn bản Professional cộng thêm các công cụ Back Office chẳng hạn như SQL Server, Microsoft Transaction Server, Internet Information Server.

Cài đặt VB6

Để cài đặt VB6, máy tính của bạn cần phải có một ổ đĩa CD-ROM (CD drive) . Bạn cần ít nhất 32 MB RAM, 2 GB hard disk và CPU Pentium II. Khi bỏ VB6 CD vào CD drive, nó sẽ tự khởi động để display menu cho bạn chọn những thứ gì cần Setup, hãy click **Install Visual Basic 6.0** để cài VB6. Ngoại trừ các file hệ điều hành (Operating System) trong thư mục (folder) **\Os**, các file trong đĩa compact đều không bị nén. Vì thế, bạn có thể sử dụng chúng trực tiếp từ đĩa. Ví dụ, có nhiều công cụ và thành phần trong folder **\Tools** vốn có thể được cài đặt trực tiếp từ CD-ROM.

Ngoài ra, bạn có thể chạy Setup khi nào cần thiết. Ví dụ, bạn có thể chạy

Setup để cài đặt lại Visual Basic trong folder khác, hoặc để cài đặt thêm bớt các phần của VB6.

Nếu vì lý do gì hệ thống không install các đĩa compact MSDN (bạn sẽ khám phá ra điều nầy khi thấy Help không có mặt lúc chạy VB6), bạn có thể cài đặt chúng trực tiếp từ đĩa số 1 của bộ MSDN.

Để bổ xung và xóa các thành phần VB:

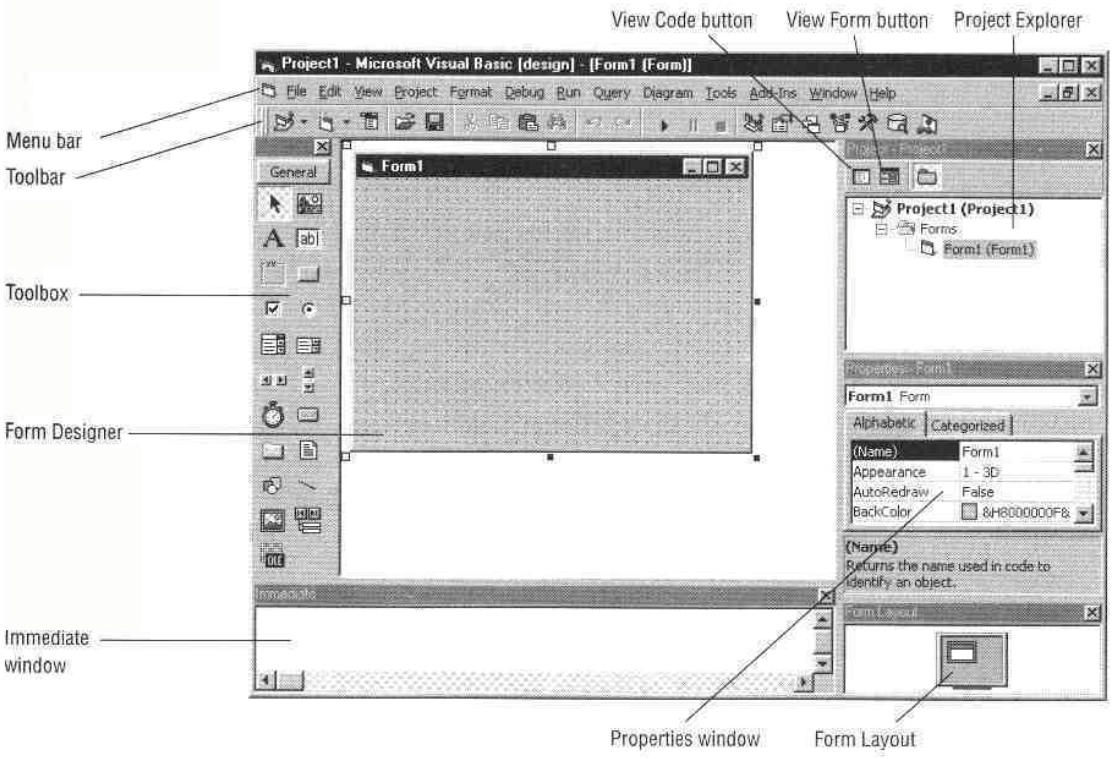
1. Bỏ đĩa compact vào CD drive.
2. Nếu menu không tự động hiện lên thì chạy chương trình Setup có sẵn tong folder gốc trên đĩa compact.
3. Chọn nút **Custom** trong hộp thoại (dialog) **Microsoft Visual Basic 6.0 Setup.**
4. Chọn hay xóa các thành phần bằng cách check hay uncheck các hộp danh sách **Options** của dialog **Custom.**
5. Thực hiện các chỉ dẫn Setup trên màn hình.

Ghi chú: Trong lúc cài VB6, nhớ chọn Graphics nếu không bạn sẽ thiếu một số hình ảnh như icons, bitmaps v.v... Đáng lẽ Microsoft cho tự động cài đặt Graphics, tức là Default (không có nói gì) thì cài đặt Graphics.

Integrated Development Environment (IDE) của VB6

Khi khởi động VB6 bạn sẽ thấy mở ra nhiều cửa sổ (windows), scrollbars, v.v.. và nằm chồng lên là **New Project** dialog. Ở đây VB6 cho bạn chọn một trong nhiều loại công trình.

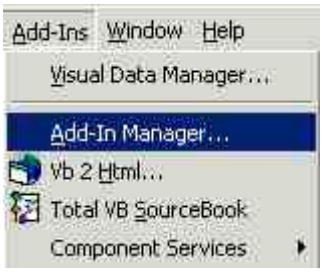
Chọn Standard EXE. Một lát sau trên màn ảnh sẽ hiện ra giao diện của môi trường phát triển tích hợp (Integrated Development Environment - IDE) giống như dưới đây:



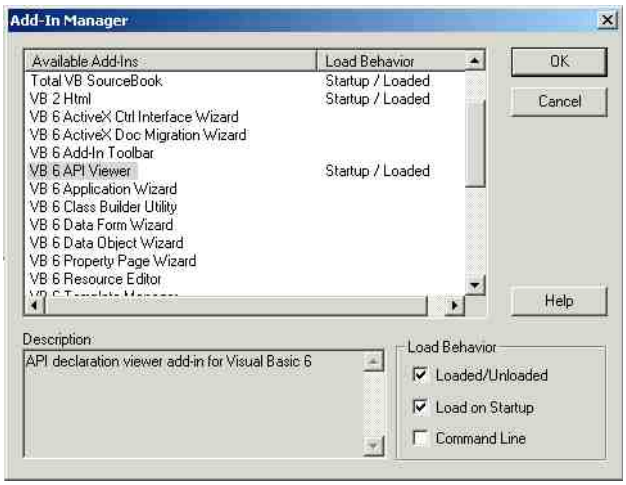
IDE của VB6 bao gồm các yếu tố sau:

Menu Bar

Chứa đầy đủ các commands mà bạn sử dụng để làm việc với VB6, kể cả các menu để truy cập các chức năng đặc biệt dành cho việc lập trình chẳng hạn như Project, Format, hoặc Debug. Trong Menu Add-Ins có Add-Ins Manager cho phép bạn gắn thêm những menu con nhiệm ý để chạy các chương trình lợi ích cho việc lập trình.



Trong Add-Ins Manager dialog bạn chọn một Add-In rồi check một hay nhiều hộp trong khung Load behavior:

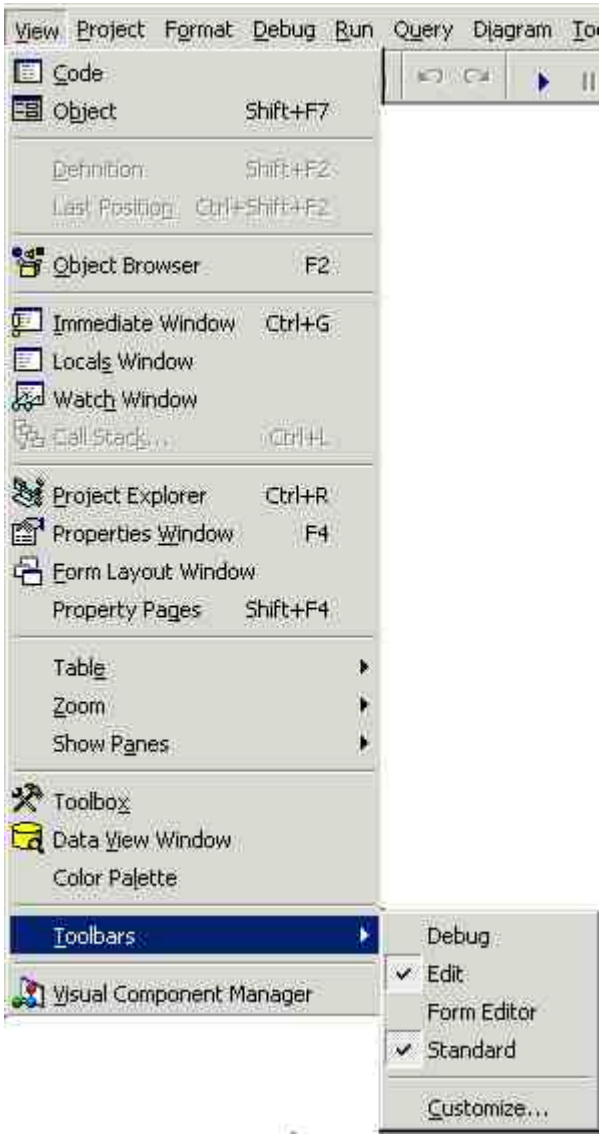


Toolbars (Debug, Edit, form Editor, Standard)

Các toolbars có hình các icons cho phép bạn click để thực hiện công việc tương đương với dùng một menu command, nhưng nhanh và tiện hơn. Bạn dùng menu command **View | Toolbars** (click lên menu command View cho popupmenu hiện ra rồi click command con Toolbars) để làm cho các toolbars hiện ra hay biến mất đi. Bạn có thể thay đổi vị trí một toolbar bằng cách nắm vào hai gạch vertical nằm bên trái toolbar rồi dời toolbar đi chỗ khác (nắm ở đây nghĩa là để pointer của mouse lên chỗ

chấm đỏ trong hình phía dưới rồi bấm xuống và giữ nút bên trái của mouse, trong khi kéo pointer đi nơi khác).

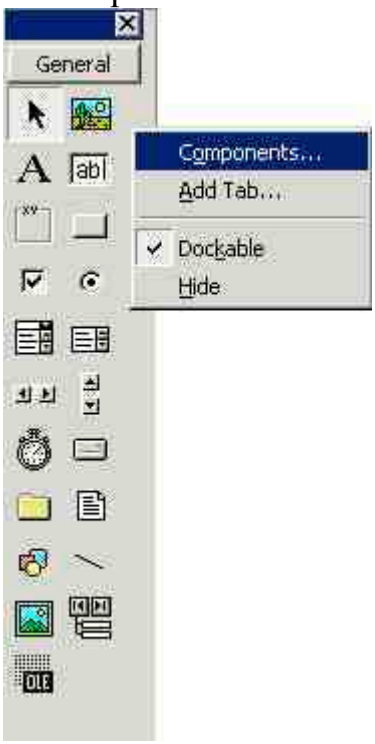
Ngoài ra bạn cũng có thể sửa đổi các toolbars theo ý thích bằng cách dùng Menu command **View | Toolbars | Customize...**



Toolbox

Đây là hộp đồ nghề với các công cụ, gọi là controls, mà bạn có thể đặt lên các form trong lúc thiết kế (design). Nếu Toolbox biến mất, bạn có thể display nó trở lại bằng cách dùng menu command **View | Toolbox**. Bạn

có thể khiến toolbox display nhiều controls hơn bằng cách chọn **Components...** từ context menu (chọn Toolbox rồi bấm nút phải của mouse để display context menu) hay dùng menu command **Project | Components**. Ngoài việc trình bày Toolbox mặc định, bạn có thể tạo cách trình bày khác bằng cách chọn **Add Tab...** từ context menu và bổ sung các control cho tab từ kết quả.



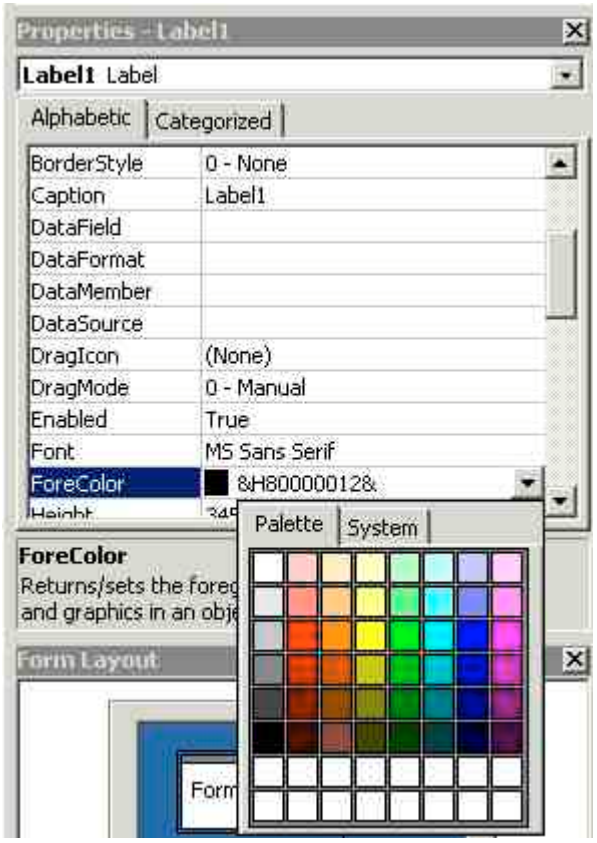
Project Explorer

Sẽ liệt kê các forms và các modules trong project hiện hành của bạn. Một ***project*** là sự tập hợp các files mà bạn sử dụng để tạo một trình ứng dụng. Tức là, trong VB6, khi nói viết một program có nghĩa là triển khai một project.

Properties window

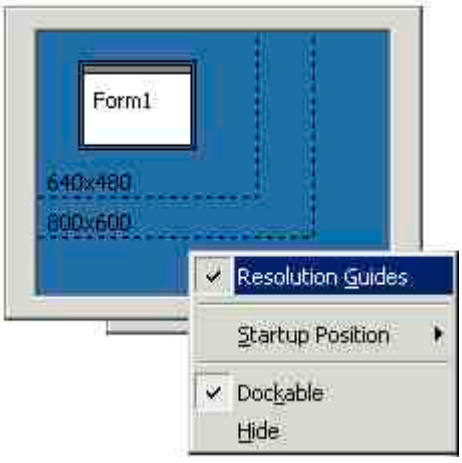
Liệt kê các đặc tính của các forms hoặc controls được chọn. Một property là một đặc tính của một object chẳng hạn như size, caption, hoặc color. Khi bạn sửa đổi một property bạn sẽ thấy hiệu quả ngay lập tức, thí dụ thay đổi property Font của một Label sẽ thấy Label ấy được display bằng Font chữ mới. Khi bạn chọn một Property của control hay form trong Properties window, phía bên phải ở chỗ value của property có thể display ba chấm (**. . .**) hay một tam giác chỉa xuống. Bấm vào đó để display một

dialog cho bạn chọn value. Thí dụ dưới đây là dialog để chọn màu cho property ForeColor của control Label1.



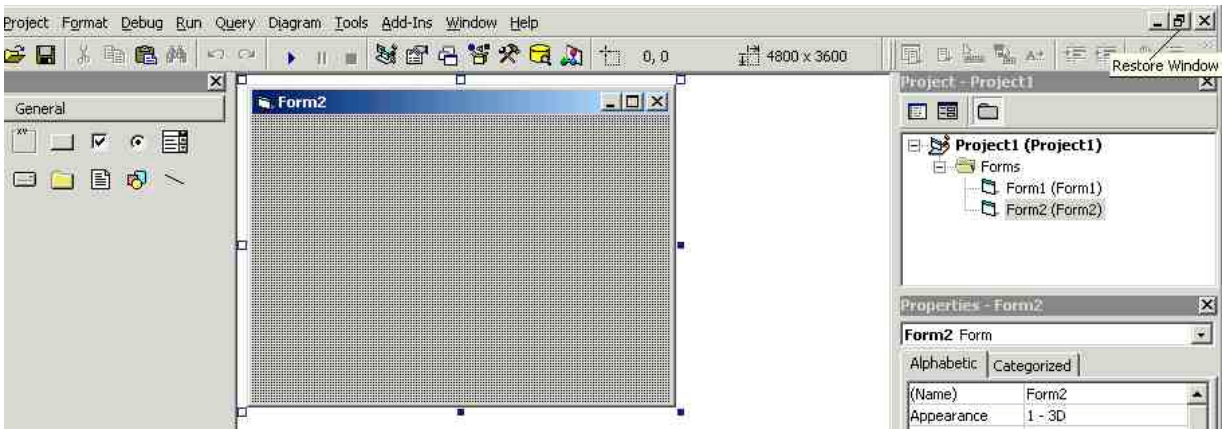
Form Layout

Bạn dùng form Layout để chỉnh vị trí của các forms khi form hiện ra lần đầu lúc chương trình chạy. Dùng context command **Resolution Guides** để thấy nếu dùng một màn ảnh với độ mịn (resolution) tệ hơn, thí dụ như 640 X 480, thì nó sẽ nhỏ như thế nào.



Form Designer

Dùng để thiết kế giao diện lập trình. Bạn bổ sung các controls, các đồ họa (graphics), các hình ảnh và một form để tạo sự ma sát mà bạn muốn. Mỗi form trong trình ứng dụng của bạn có designer form riêng của nó. Khi bạn maximise một form designer, nó chiếm cả khu làm việc. Muốn làm cho nó trở lại cỡ bình thường và đồng thời để thấy các form designers khác, click nút Restore Window ở góc bên phải, phía trên.

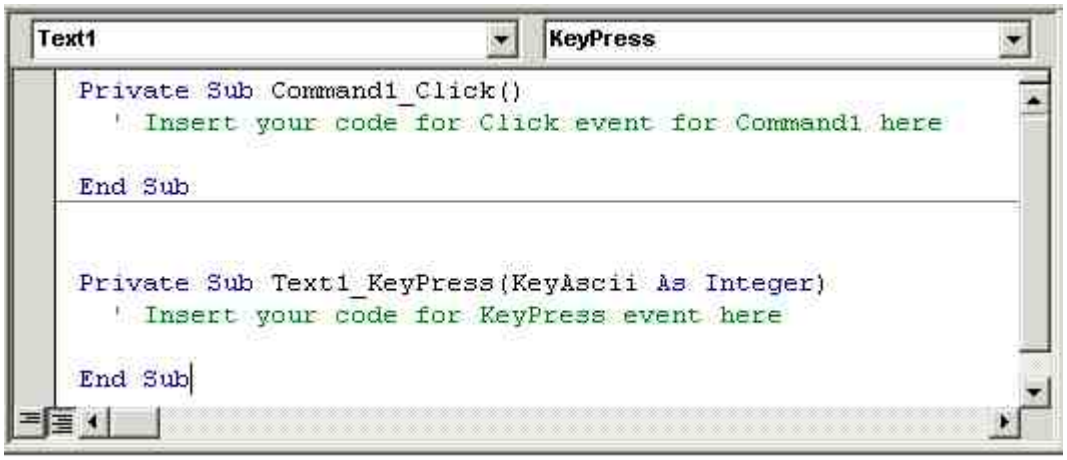


Immediate Window

Dùng để gở rối (debug) trình ứng dụng của bạn. Bạn có thể display dữ kiện trong khi chạy chương trình ứng dụng. Khi chương trình đang tạm ngừng ở một break point, bạn có thể thay đổi giá trị các variables hay chạy một dòng chương trình.

View Code button

Click lên nút nầy để xem code của một form mà bạn đã chọn. Window của code giống như dưới đây:



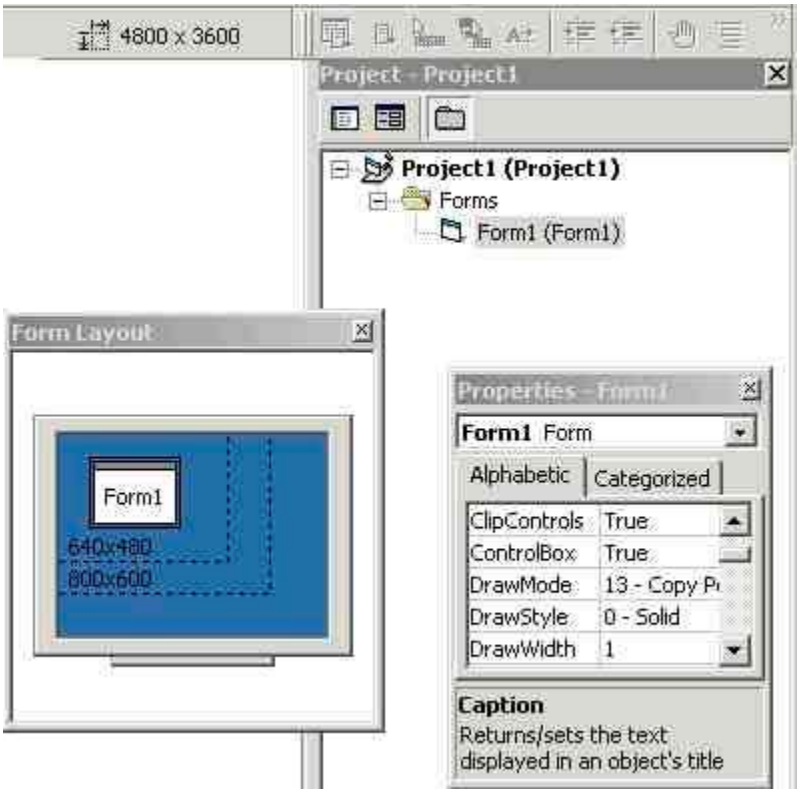
Trong Code window bạn có thể chọn display tất cả Sub của code cùng một lúc như trong hình hay display mỗi lần chỉ một Sub bằng cách click button có hình ba dòng nằm ở góc bên trái phía dưới.

View form button

Click lên nút nầy để xem form của một form mà bạn đã chọn.

Ghi chú: Nhiều windows trong IDE như Toolbars, Toolbox, Project Explorer .v.v..có thể trôi lình bình (floating) hay đậu ở bến (docked). Bạn có thể thay đổi vị trí chúng bằng cách nắm vào Title Bar của window rồi dời đi. Dĩ nhiên bạn cũng có thể mở rộng hay làm nhỏ một window bằng cách dời một cạnh vertical hay horizontal của nó. Khi để một window lên trên một window khác chúng có thể tìm cách dính nhau.

Trong hình dưới đây, Properties Window và Form Layout đã được kéo ra ngoài cho floating.



Nhận trợ giúp trong khi đang làm việc

Trong khi lập trình bạn có thể cần tìm hiểu các thông tin liên quan đến các commands, functions .v.v.. của VB6. Bạn có thể khởi động **Microsoft Developer Network | MSDN Library Visual Studio 6.0** từ nút Start, hay click **Help | Contents** từ Menu Bar của VB6, hay chọn một keyword (highlight keyword) rồi ấn F1 để đọc Help.

Nội dung Help bao gồm nhiều đặc điểm được thiết kế để thực hiện việc tìm kiếm thông tin dễ dàng hơn. Bạn có thể dựa trên **Contents** để đọc tài liệu như một quyển sách, **Index** để đọc những đoạn có nhắc đến một keyword hay **Search** để tìm một tài liệu nhanh hơn. Ví dụ, việc gở rối thông tin bắt nguồn từ nhiều đặc tính khác nhau phụ thuộc vào loại đề án mà bạn đang làm việc. Các liên kết được mô tả trong phần nầy thực hiện việc tìm kiếm dễ dàng hơn.

Ngoài ra, bạn cũng có thể click **See Also** dưới tiêu đề của chủ điểm để xem các tiêu đề của các chủ điểm mà bạn có thể đi đến hoặc liên hệ đến nhiều thông tin.

Context Sensitive Help (trợ giúp trong đúng tình huống)

Nhiều phần của VB6 là **context sensitive**, có nghĩa là lúc bối rối chỉ cần ấn nút F1 hoặc highlight keyword rồi nhấn F1 là được thông tin những gì liên hệ trực tiếp với tình huống hiện giờ của bạn.

Bạn có thể nhấn F1 từ bất kỳ phần context sensitive nào của giao diện VB6 để display thông tin Help về phần đó. Các phần context sensitive là:

- Các Windows của VB6 như Properties, Code .v.v..
- Các control trong Toolbox.
- Các Object trên một form hoặc Object tài liệu.
- Các đặc tính trong Window Properties.
- Các keywords của VB6
- Các thông báo lỗi (error messages)

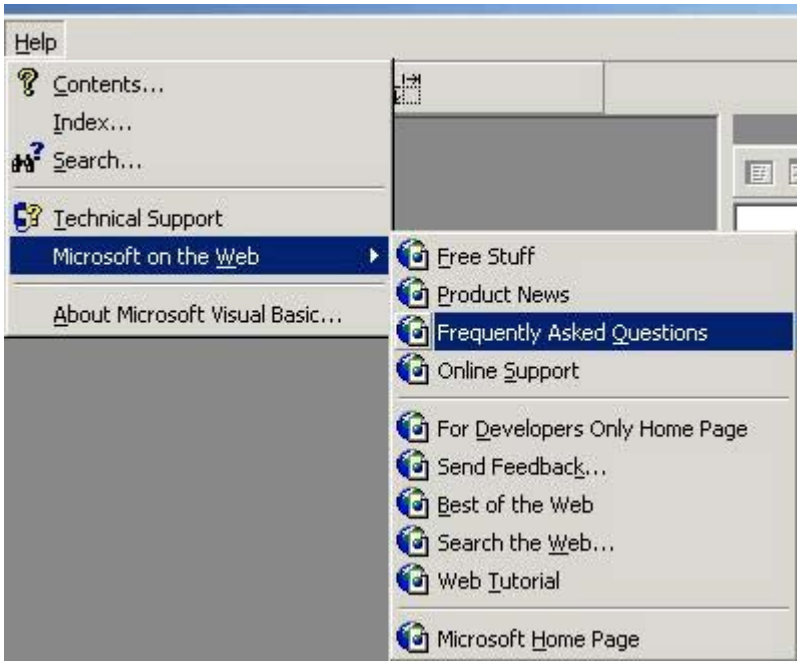
Ngoài ra, trong Help thường có **Example**. Bạn click lên chữ Example để display một thí dụ minh họa cách dùng một function hay property.

Microsoft on the Web

Web site của Microsoft chứa nhiều thông tin cập nhật cho những người lập trình VB6. Trang chủ Visual Basic đặt tại URL http://www.microsoft.com/vbasic/. Thông tin có sẵn tại địa chỉ nầy bao gồm:

- Cập nhật các đặc tính mới, các phiên bản sản phẩm, các sản phẩm liên hệ, các thuyết trình (seminar) và các hoạt động (event) đặc biệt.
- Thông tin bổ sung trên các đặc tính VB6 chứa trong các bài viết gọi là White Papers, các mách nước (tips) và các trình trợ giáo, nguồn đào tạo.
- Sản phẩm mới tải xuống (download) bao gồm sự cập nhật đến các file chương trình, các cập nhật trợ giúp, các trình điều khiển, và các file liên hệ khác của VB6.

Để truy cập Web site của Microsoft, từ menu Help chọn **Microsoft on the Web** rồi chọn menu con tùy thích như dưới đây.



Ghi chú: Một số nội dung trên Web site của Microsoft được tối ưu hóa dành cho Microsoft Internet Explorer và không thể display đầy đủ trong một bộ trình duyệt (browser) khác. Do đó bạn nên chỉ dùng Internet Explorer làm browser trên máy bạn mà thôi.

Chương Hai- Viết chương trình đầu tiên

Bạn đang làm quen với môi trường triển khai lập trình (Integrated Development Environment - IDE) của MS VB6 và rất nóng ruột muốn viết những dòng mã đầu tiên để chào mừng thế giới.

Ta thử ôn lại một số vấn đề mà có lẽ bạn đã biết rồi. Một chương trình Visual Basic gồm có phần mã lập trình và các hình ảnh (visual components). Bạn có thể thiết kế phần hình ảnh bằng cách dùng những đồ nghề (Controls hay Objects) từ Túi đồ nghề (Toolbox) nằm bên trái. Nếu bạn không thấy cái Túi đồ nghề thì dùng mệnh lệnh **Menu View|Toolbox** để bắt nó hiện ra.

Khi bạn bắt đầu thiết kế một chương trình bằng cách chọn Standard EXE, môi trường triển khai lập trình (IDE) cho bạn sẵn một Form tên là Form1. Bạn có thể đổi tên (Name) nó trong cái cửa sổ Propeties nằm phía dưới bên phải (trong hình dưới đây ta edit Name property của Form1 thành ra frmMainForm). Bạn cũng có thể sửa đề tựa (Title) của form ra cái gì có ý nghĩa hơn bằng cách đổi Caption của form cũng trong cửa sổ Propeties (trong hình dưới đây ta edit Caption property của form thành ra "Chi tiet cua ban toi").

Sắp đặt các vật dụng lên Form

Muốn đặt một Control lên Form, click hình cái Control trong Toolbox rồi Drag (bấm nút trái của con chuột rồi kéo cho thành hình chữ nhật trước khi buông nút trái ra) con chuột trên Form vẽ thành cở của Control. Những Controls bạn sẽ dùng thường nhất từ Toolbox là Label (nhãn), Textbox (hộp để đánh chữ vào) và CommandButton (nút bấm mệnh lệnh).

Trong hình trên ta có ba Label và ba Textbox. Muốn sửa chữ Label1 ra "Ten" thì edit Property Caption. Còn Textbox không dùng Property Caption mà dùng Property Text. Ta cũng có thể thay đổi các Property Caption và Text trong khi chạy chương trình (at run-time). Trong lúc thiết kế (design time) bạn có thể sửa đổi kiểu chữ của những Controls bằng cách edit Property Font của chúng trong cửa sổ Properties (click bên phải của Property Font trong Properties Window, IDE sẽ pop-up cái Font dialog để bạn lựa chọn những đặc tính của Font như trong hình dưới đây).

Nếu bạn thấy bực mình tại sao cái cở chữ tự có (default size) của các Control hơi nhỏ, bạn có thể giải quyết bằng cách sửa cở chữ của chính Form cho nó lớn hơn. Vì khi một Control được đặt lên một Form, nó thừa kế cở chử của Form.

Để sắp xếp cho một số Control thẳng hàng với nhau bạn chọn cả nhóm rồi dùng mệnh lệnh Menu **Format|Align|Lefts** .v.v..Nếu bạn chưa biết cách chọn một nhóm Control thì có hai cách. Cách thứ nhất bạn đè nút Shift trong khi click các Control bạn muốn chọn. Cái Control mà bạn chọn sau cùng sẽ là cái chuẩn để các Control khác sẽ làm giống theo. Cách thứ hai là Drag cho sợi dây thun (rubber band) bọc chung quanh các Control. Trong trường hợp các Control nầy nằm trong một container, thí dụ như một khung (Frame) hay PictureBox, thì bạn phải click Form trước, rồi đè nút Ctrl trong khi Drag rubber band bao các Control.

Chứa mọi thứ của một dự án VB

Tới đây bạn để ý thấy trong cửa sổ bên phải, phía trên, gọi là Project Explorer, có hình giống như một cái cây (tree) cho thấy ta có một Form

trong một Project (dự án). Project là một cách tiện dụng để ta sắp xếp những gì cần thiết cho một dự án. Thường thì một dự án có nhiều Form và có thể cần những thứ khác.

Mỗi Form sẽ được chứa vào đĩa dưới dạng "frmMainForm.**frm**". Bạn save một form bằng menu command **File | Save formfilename.frm**. Nếu trong Form1 có chứa hình ảnh (thí dụ bạn dùng Properties Window để chọn một icon hình gương mặt cười làm icon cho frmMainForm) thì các hình ảnh của frmMainForm sẽ được tự động chứa trong hồ sơ "frmMainForm.**frx**". Lưu ý là không nhất thiết tên của hồ sơ (file) mà bạn phải cho biết khi chứa (save) phải giống như tên của Form mà bạn dùng trong chương trình. Tuy nhiên bạn nên dùng cùng một tên cho cả hai để sau nầy dễ tìm hồ sơ nếu có thất lạc. Theo qui ước thông thường, các Form được đặt tên bắt đầu bằng "**frm**", thí dụ như "frmMainForm".

Khi bạn **save** một Project thì có nghĩa là save tất cả hồ sơ dùng cho dự án, kể cả các Form và một hồ sơ cho chính Project, thí dụ như "MyFirstProg.vbp" ("vbp" là viết tắt chữ Visual Basic Project). Bạn save Vb6 project bằng menu command **File | Save Project**. À, muốn đổi tên Project, bạn click lên hàng trên cùng bên phải trong cửa sổ Project Explorer (Project1 (Project1.vbp)), rồi edit tên của Project trong cửa sổ Properties phía dưới. Bạn nên chứa tất cả những hồ sơ dùng cho cùng một Project trong cùng một tập (Folder/Directory).

Bạn có thể dùng Notepad để mở ra xem chơi, coi trong "frmMainForm.frm" có gì. Bạn sẽ thấy trong ấy gồm có hai phần: phần đầu là diễn tả các Control nằm trong Form, phần còn lại là mã lập trình mà bạn viết. Bạn cũng sẽ chú ý là các properties mà bạn đã sửa cho các Control đều được ghi lại trong phần đầu nói trên. VB dựa vào phần diễn tả các Control để thiết lập lại (reconstruct) hình ảnh của Form.

```
VERSION 5.00
Begin VB.Form frmMainForm
   Caption        =  "Chi tiet cua ban toi"
   ClientHeight   =  2325
   ClientLeft     =  60
   ClientTop      =  345
   ClientWidth    =  4155
   BeginProperty Font
      Name           =  "MS Sans Serif"
      Size           =  9.75
      Charset        =  0
      Weight         =  400
      Underline      =  0   'False
      Italic         =  0   'False
      Strikethrough  =  0   'False
   EndProperty
   Icon           =  "frmMainForm.frx":0000
   KeyPreview     =  -1  'True
   LinkTopic      =  "Form1"
   ScaleHeight    =  2325
   ScaleWidth     =  4155
   StartUpPosition =  3  'Windows Default
   Begin VB.CommandButton cmdXuat
      Caption        =  "&Xuat"
      Height         =  435
      Left           =  150
      TabIndex       =  7
      Top            =  1770
      Width          =  615
   End
   Begin VB.CommandButton cmdViet
      Caption        =  "&Viet vao dia"
      Height         =  465
      Left           =  2100
      TabIndex       =  6
      Top            =  1770
      Width          =  1935
   End
   Begin VB.TextBox txtTuoi
      Height         =  375
      Left           =  990
      TabIndex       =  5
      Text           =  "29"
   End
   Begin VB.Label lblTen
      Caption        =  "&Ten:"
      Height         =  375
      Left           =  210
      TabIndex       =  0
      Top            =  240
      Width          =  495
   End
End
Attribute VB_Name = "frmMainForm"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub cmdViet_Click()
Dim fileNo, myLocalFolder, myFileName
  myLocalFolder = App.Path
  If Right(myLocalFolder, 1) <> "\" Then
    myLocalFolder = myLocalFolder & "\"
  End If
  fileNo = FreeFile
  myFileName = myLocalFolder & "myFriends.txt"
  If Dir(myFileName) <> "" Then
     Open myFileName For Append As #fileNo
  Else
     Open myFileName For Output As #fileNo
  End If
  Print #fileNo, txtTen & ";" & txtDiachi & ";" & txtTuoi
  Close #fileNo
End Sub

Private Sub cmdXuat_Click()
   End
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)
  If KeyAscii = 13 Then
     SendKeys "{TAB}"
     KeyAscii = 0
  End If
End Sub
```

Sau nầy, khi đã lão luyện VB, bạn có thể dùng một chương trình tự động chế (generate) ra những hàng diễn tả các Control cho một Form.

Đó là kỹ thuật dùng trong các Wizards của VB để chế một số chương trình khởi đầu cho chúng ta từ các bảng kẻm (Template).

Thêm mã lập trình để xử lý một sự cố

Hầu hết lập trình trong Visual Basic là viết mã để xử lý các sự cố (Event). Thí dụ muốn chấm dứt chương trình, người sử dụng sẽ click nút "Xuat". Để thực hiện điều nầy trong khi triển khai chương trình bạn doubleClick (click liên tiếp 2 lần) nút "Xuat". VB IDE sẽ viết sẵn cho bạn cái vỏ của một Subroutine:

```
Private Sub cmdXuat_Click()
  End     ' Bạn chỉ viết thêm dòng nầy để kết thúc chương trình
End Sub
```

Để ý là tên (Name) của nút Xuat là "cmdXuat" ("cmd" là viết tắt chữ CommandButton), VB gắn thêm dấu gạch dưới và Event Click để làm thành tên **cmdXuat_Click** của Sub, chương trình nhỏ sẽ được xử lý khi người sử dụng click nút Xuat. Chương trình nhỏ hay Subroutine nầy còn được gọi là Event Handler cho Event Click. Hàng chữ xanh lá cây là dùng để giải thích cho lập trình viên (gọi là Comment), VB sẽ hoàn toàn không chú ý đến nó khi xử lý Sub cmdXuat_Click.

Comment có nghĩa là chú thích. Trong VB chú thích bắt đầu bằng dấu single quote **'**. Khi VB thấy dấu nầy là nó bỏ qua những gì còn lại trên dòng mã.

Là Lập trình viên chuyên nghiệp bạn nên tập thói quen dùng Comment mọi nơi để giúp người khác và chính bạn hiểu chương trình của mình. Nên nhớ là tiền phí tổn để bảo trì một chương trình thì ít nhất là tương đương với số tiền bỏ ra lần đầu để triển khai. Bảo trì có nghĩa là thăm viếng lại chương trình để sửa lỗi (fix bug) và thêm các đặc điểm cho hay hơn (enhancement).

Nói chung hể bạn làm điều gì bí hiểm hay cắc cớ thì làm ơn giải thích rõ ràng.
Nếu muốn cắt một dòng mã VB ra làm hai dòng thì chấm dứt dòng thứ nhất bằng dấu gạch dưới _.

Tiếp theo, bạn doubleClick nút "Viet vao dia" và viết những hàng mã sau:

```
Private Sub cmdViet_Click()
  Open "myFriends.txt" For Output As #2 ' Mở một hồ sơ để viết ra và gọi là cổng số 2
  ' Viết vào cổng số 2: Tên, Địa chỉ và Tuổi, ngăn   cách nhau bằng dấu chấm phẩy
  Print #2, txtTen & ";" & txtDiachi & ";" & txtTuoi
  Close #2 ' Đóng cổng số 2
End Sub
```

Trong Sub cmdViet_Click, trước hết ta mở một hồ sơ tên là "**myFriends.txt**" và gọi nó là cổng số 2. Sau khi mở hồ sơ để viết ra ta ráp Tên, Địa chỉ và Tuổi lại, ngăn cách bằng dấu chấm phẩy (**;**) để đánh dấu nhỡ sau nầy ta muốn gở riêng ba thứ ra trở lại. Dấu "**&**" là operator để ráp (concatenate) hai dòng chữ (text string) lại với nhau.

Print #2 có nghĩa là viết ra cổng số 2, tức là hồ sơ "myFriends.txt". Thứ chúng ta viết ra cổng 2 là Tên, Địa chỉ và Tuổi (txtTen & ";" & txtDiachi & ";" & txtTuoi).

Những rắc rối của việc mở một hồ sơ

Cái cổng số 2 ở trên là ta tự chọn (arbitrary). Thật ra muốn gọi cổng số mấy cũng được, miễn là chưa có phần nào khác trong cùng chương trình nầy đang dùng cổng số ấy. Đây là một cách VB làm việc cho tiện thay vì gọi nguyên một cái tên hồ sơ dài.

Nếu muốn chắc chắn không trùng số cổng với chỗ nào khác, ta có thể làm như sau:

```
fileNo = freefile
```

Rồi thay thế số 2 bằng chữ fileNo trong Sub cmdViet_Click. **freeFile** là một Function (chương trình nhỏ dùng để tính ra một thứ gì) nhờ VB cấp phát cho một con số đại diện hồ sơ chưa ai dùng.

Chữ **Output** trong câu (*Open "myFriends.txt" For Output As #2*) dùng ở đây để nói từ CPU (Central Processing Unit) ta muốn "viết ra" một hồ sơ. Khi mở một hồ sơ để viết ra kiểu nầy thì nếu hồ sơ chưa có nó sẽ được dựng nên (created). Nếu hồ sơ đã có rồi thì nó sẽ bị xoá bỏ (delete) và đồng thời một hồ sơ trống và mới sẽ được dựng nên. Động từ chuyên môn là "viết chồng lên" (**overwrite**).

Nếu ta mở một hồ sơ để "đọc vào" thì dùng chữ "**Input**" thay vì "**Output**". Còn nếu "viết thêm" vào một hồ sơ có sẵn (chớ không overwrite hồ sơ ấy) thì dùng chữ "**Append**" thay vì "Output". Trong trường hợp ấy bạn phải kiểm xem hồ sơ "myFriends.txt" đã có sẵn chưa. Bạn có thể viết như sau:

```
If Dir("myFriends.txt") <> "" then ' Nếu hồ sơ "myFriends.txt" hiện hữu
   Open "myFriends.txt" For Append As #2 ' Mở một hồ sơ để viết thêm và gọi là cổng số 2
```

```
  Else
    Open "myFriends.txt" For Output As #2 ' Mở một hồ sơ để viết ra và gọi là cổng số 2
  End If
```

Function Dir("myFriends.txt") dùng ở trên sẽ cho ta tên của hồ sơ nếu hồ sơ hiện hữu, ngược lại nó sẽ cho một dòng chữ trống (empty string), biểu hiệu là "". Tại đây, nếu lanh ý bạn sẽ hỏi hồ sơ "myFriends.txt" nằm ở folder nào. Câu trả lời là không biết chắc. Nếu bạn chưa chứa (save) chương trình vào dĩa (vì mới viết) thì nó nằm ở folder của VB6.EXE. Còn như đã chứa chương trình rồi thì có lẽ nó nằm ở folder của chương trình bạn. Muốn hồ sơ "myFriends.txt" luôn luôn đi cùng với chương trình, bạn có thể làm như sau:

```
MyLocalFolder = App.path ' Lấy folder của chương trình xử lý của bạn
If Right(MyLocalFolder,1) <> "\" then ' Nếu chữ cuối cùng không phải là backslash
   MyLocalFolder = MyLocalFolder & "\" ' thì gắn thêm một backslash ở cuối
End If
' Mở một hồ sơ với tên có folder (full pathname) để viết ra và gọi là cổng số 2
Open MyLocalFolder & "myFriends.txt" For Output As #2
```

Cuối cùng ta đóng hồ sơ lại bằng câu Close #2.. Từ rày VB có thể cấp số 2 để làm cổng cho chỗ khác trong chương trình.

Default Property của một Control

"txtTen" được dùng ở đây là viết tắt cho "txtTen.text", vì Default Property của một TextBox là text của nó. Default Property của một Control là Property được VB dùng khi bạn chỉ cho tên của Control mà thôi.
Trong khi đó Default Property của Label là Caption.

Vì txtTen được dùng như txtTen.txt để nói đến một dòng chữ, nên trong chương trình ta nhắc đến nó y như một variable (mã số) dùng cho một string. Do đó với qui ước dùng ba chữ đầu "txt" cho tên của một Textbox giúp ta nhận diện ngay nó không phải là một string variable bình thường. Hãy lưu ý sự khác biệt khi gọi một Sub trong hai trường hợp sau:

```
Call CheckmyTextbox  (txtDiachi) ' txtDiachi được xem là Textbox trọn vẹn
CheckmyTextbox  txtDiachi ' txtDiachi được xem là txtDiachi.text, một  dòng chữ
```

Thứ tự các Control trên một Form

Trong chương trình nầy ta muốn người xử dụng cho vào dữ kiện theo thứ tự "Tên, Địa chỉ, Tuổi". Khi mới vào, ta muốn cái dấu chớp tắt (cursor) nằm trong txtTen ngay để người xử dụng khỏi mất công click vào Textbox ấy khi muốn mang cursor trở lại đó. Ta nói là txtTen có cái Focus.

Sau khi người xử dụng đã cho tên vào rồi, cô sẽ đánh nút Tab để di chuyển cursor qua Control tiếp theo, mà ta muốn là txtDiachi. Để sắp thứ tự các Control cho sự di chuyển của cursor khi người xử dụng đánh nút Tab ta edit Property TabIndex của các Control. TabIndex bắt đầu bằng số 0. Nhiều khi người xử dụng thích dùng nút Enter thay vì Tab để di chuyển Cursor qua Control tiếp theo, bạn có thể làm như sau cho Textbox txtTen:

```
Private Sub txtTen_KeyPress(KeyAscii As Integer)
  If KeyAscii = 13 Then ' Nếu nút bấm là Enter
    SendKeys "{TAB}" ' giả mạo gởi nút Tab
    KeyAscii = 0 ' Nuốt trọng nút Enter để Windows không còn lo cho nó
  End If
End Sub
```

Cho các Textbox khác như txtDiachi, txtTuoi bạn cũng làm tương tợ như vậy. Khi bạn doubleClick txtTen lần đầu để viết mã, VB cho bạn **Private Sub txtTen_Change()**. Bạn phải click cái Combobox bên phải, phía trên của Code Window, cho nó mở ra và chọn Event KeyPress.

Nếu bạn muốn chương trình mình còn chuyên nghiệp hơn, bạn cho phép người xử dụng bấm nút **Alt+o** (bấm nút **Alt** xuống trong khi bấm nút **o**) để mang Cursor về txtTuoi hay **Alt+d** để mang Cursor về txtDiachi. Muốn thế bạn phải thêm vào dấu "**&**" ở phía trước các chữ **T**, **D** và **o** trong Caption của các label lblTen, lblDiachi và lblTuoi.

Kế đó bạn phải cho giá trị **TabIndex** của lblTen, txtTen, lblDiachi, txtDiachi, lblTuoi, txtTuoi liên tiếp là 0,1,2,3,4,5. Khi người xử dụng đánh Alt+o, VB sẽ mang Cursor về nhãn lblTuoi, nhưng vì không có chỗ cho nó đáp trong label nên nó phải đáp vào Control kế đó, tức là txtTuoi.

Khi ta đã cho TabIndex của các Control những giá trị kể trên thì khi Form hiện ra Cursor sẽ nằm trong TextBox txtTen vì mặc dầu lblTen có

TabIndex nhỏ nhất(0), nó không phải là chỗ Cursor đáp lên được, nên Cursor phải đáp lên textbox có TabIndex value kế đó, tức là 1.

Nếu bạn không muốn Cursor ngừng lại ở một TextBox nào thì edit Property **TabStop** của TextBox đó cho bằng False. Trong trường hợp ấy người sử dụng vẫn có thể click vào TextBox và sửa dòng chữ ở đó được như thường. Nếu bạn thật sự không muốn cho phép người sử dụng sửa gì ở TextBox thì edit Property **Enabled** bằng False hay Property **Locked** bằng True. Khi Enabled của một TextBox bằng False thì TextBox trở nên mờ đi.

Nhân tiện ta edit thêm dấu "&" ở phía trước các chữ X và V trong Caption các CommandButton "Xuat" và "Viet vao dia". Sau nầy người sử dụng có thể bấm Alt-X coi như tương đương với click nút "Xuat".

Nếu nhỡ trong Form bạn có nhiều Textbox quá, đổi nút Enter ra nút Tab cho từng Textbox một thì mất công quá. Bạn có thể làm một cái chung cho cả Form. Tức là nói rằng bạn không cần biết nút Enter vừa mới được đánh ở TextBox nào, bạn cứ nhắm mắt đổi nó ra nút Tab.

Trước hết bạn phải chọn (select) Form rồi edit Property **KeyPreview** của nó thành True. Bạn làm việc nầy để dặn Form giựt cái nút người sử dụng đánh (keystroke) trước khi TextBox thấy. Form sẽ tráo nút Enter thành Tab rồi lẵng lặng trao cho TextBox. Bạn có thể thay thế tất cả các KeyPress event handler của các TextBox bằng đoạn mã như sau:

```
Private Sub Form_KeyPress(KeyAscii As Integer)
  If KeyAscii = 13 Then ' Nếu nút bấm là Enter
    SendKeys "{TAB}" ' giả mạo gởi nút Tab
    KeyAscii = 0 ' Nuốt trọng nút Enter để Windows không còn lo cho nó
  End If
End Sub
```

Khi bạn doubleClick lên bất cứ chỗ nào trên Form không có Control nằm, lần đầu để viết mã, VB cho bạn Private Sub Form_Load(). Bạn phải click cái Combobox bên phải, phía trên của Code Window, cho nó mở ra và chọn Event KeyPress.

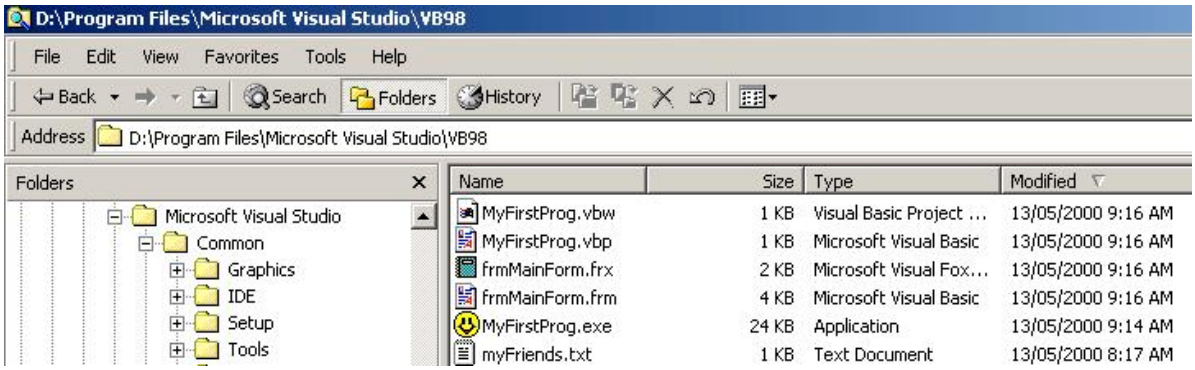
Đem ra trình làng

Để làm thành một hồ sơ áp dụng EXE, bạn dùng mệnh lệnh Menu **File|Make** MyFirstProg.exe. Cho thêm chút hương vị của cuộc đời tôi click Form rồi edit Property Icon, chọn cho nó từ folder:

D:\Program Files\Microsoft Visual Studio\Common\Graphics\Icons\Misc

một icon hình gương mặt cười. Rồi bấm mệnh lệnh Menu **File|Save Project**.

Khi dùng Explorer để xem các hồ sơ của MyFirstProg.vbp bạn sẽ thấy như dưới đây:

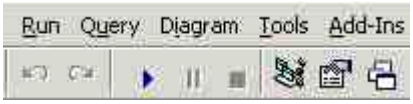


Đáng lẽ tôi dùng một folder khác thay vì VB98 để chứa dự án MyFirstProg.vbp. Hồ sơ MyFirstProg.vbw là Workspace (chỗ làm việc) dành cho VB, ta không nên động tới.

Bạn có thể làm một Shortcut cho MyFirstProg.exe với cái icon hình gương mặt cười đặt lên Desktop để chạy bên ngoài IDE của VB. Có lẽ bạn muốn Download hồ sơ:MyFirstProg.zip, nén chung tất cả các hồ sơ nói trên trong dự án MyFirstProg.vbp.

Bây giờ ngay trong VB IDE bạn có thể chạy chương trình bằng cách dùng mệnh lệnh Menu **Run|Start** hay bấm **F5**.

Bạn cũng có thể Click lên dấu tam giác chỉ về bên phải (nút Play của cassette) nằm trong toolbar ngay phía dưới VB menu.

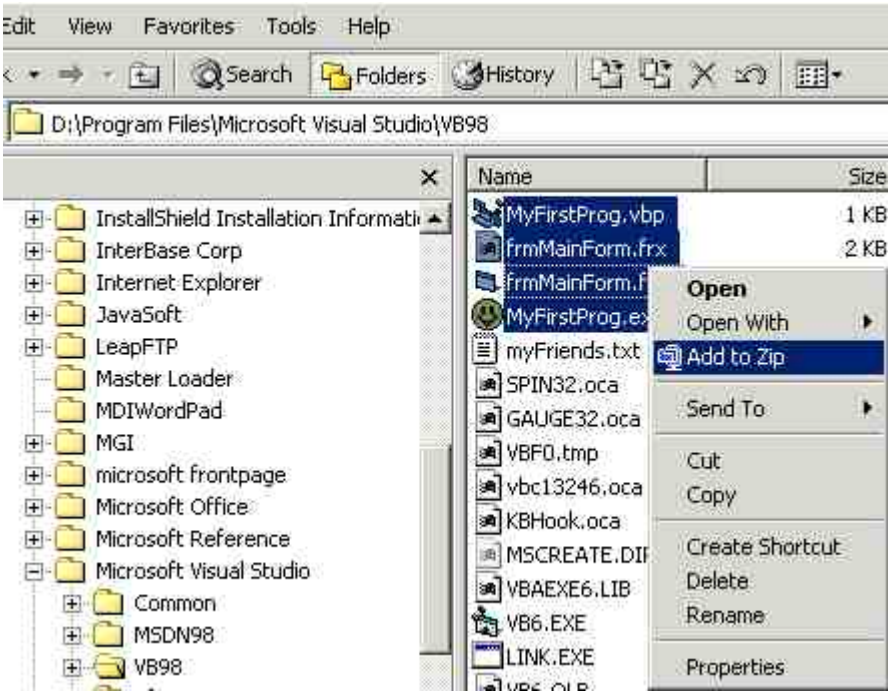


Cách nén các files trong một folder thành một zip file duy nhất

Để gởi nhiều files bằng cách đính kèm (attach) một Email trên Internet ta cần phải nén các files ấy thành một file duy nhất, gọi là Zip file. Trước hết, trong Window Explorer bạn chọn những files bạn muốn Zip chung lại. Bạn chọn nhiều files bằng cách đè nút **Ctrl** trong khi click lên tên từng file một. Nếu bạn đè lên nút **Shift**, thay vì nút Ctrl, thì cứ mỗi lúc bạn click, Window Explorer sẽ select cả một dọc tên các files nằm giữa tên hai files bạn click mới nhất. Ngoài ra bạn cũng có thể dùng Menu Command **Edit | Select All**, hay **Ctrl+A** để select tất cả các files trong một folder. Đây là trường hợp bạn sẽ dùng khi Zip tất cả các files trong một VB6 project để gởi qua Thầy/Cô.

Sau khi đã select các file rồi, bạn right click lên các file ấy để context menu pop-up. Chọn **Add to Zip**.

Nếu bạn không thấy pop-up command Add to Zip thì là bạn chưa install chương trình Winzip. Trong trường hợp ấy, download Winzip từ Internet và install.

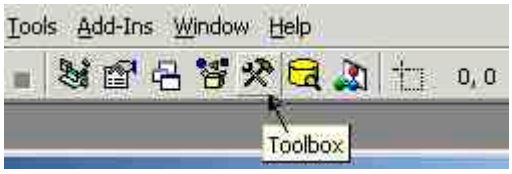


Chương Ba – Form và các Controls thông thường

Hầu hết các chương trình VB6 đều có ít nhất một Form. Khi ta chạy chương trình, Form nầy sẽ hiện ra trước hết để ta ra lệnh nó làm chuyện gì. Cái Form trống không chả làm được gì nhiều, nên ta đặt lên Form những controls như Textbox(hộp để đánh chữ vào), Label(nhãn), Commandbutton(nút bấm mệnh lệnh), .v.v.. Các controls cho ta enter các dữ kiện để chương trình dùng xử lý, và các controls cũng hiển thị (display) kết quả cho chúng ta xem.

Sắp đặt controls lên Form

Ta hãy bắt đầu thiết kế một chương trình mới (New Project) bằng cách chọn Standard EXE, môi trường triển khai lập trình (IDE) cho bạn sẵn một Form tên là Form1. Muốn đặt một Control lên Form, click hình cái Control trong Toolbox rồi Drag (bấm nút trái của con chuột rồi kéo cho thành hình chữ nhật trước khi buông nút trái ra) con chuột trên Form vẽ thành cở của Control. Một cách khác để đặt một control lên Form là doubleclick cái Control trong Toolbox, một hình control sẽ hiện ra trên Form. Kế đó bạn dời control đi đến chỗ mình muốn và resize nó. Nếu bất cứ lúc nào bạn không thấy Túi đồ nghề (Toolbox) nằm bên trái, bạn có thể dùng mệnh lệnh **Menu View|Toolbox** để bắt nó hiện ra. Có một cách khác là click lên toolbox icon trên toolbar chính của VB6.



Nên nhớ rằng Toolbox cũng là một window như các window khác. Khi nó hiện lên rồi bạn có thể nắm (bấm nút trái của con chuột và giữ như vậy chớ không buông ra) title nó để dời đi nơi khác. Bạn có thể đóng nó bằng cách click lên dấu **x** ở góc phải phía trên. Nếu right click trên Toolbox, nó sẽ display context sensitive menu, trong đó có property dockable (có thể đậu ở bến). Nếu một window là dockable, sau khi bạn dời nó đi khỏi vi trí docked bình thường của nó, bạn có thể dock nó lại như củ bằng cách double click lên title của nó.

Resize và di chuyển control

Khi bạn select một control (click lên nó), chung quanh control sẽ hiện ra resize handle, 8 nút đen dọc theo chu vi của control.

Click lên các nút đen của resize handle, bạn có thể resize control. Có một cách khác để resize control là dùng Shift + ArrowKey. Bấm nút Shift trong khi bấm một arrow key, control sẽ lớn ra hay thu hẹp theo chiều của ArrowKey.

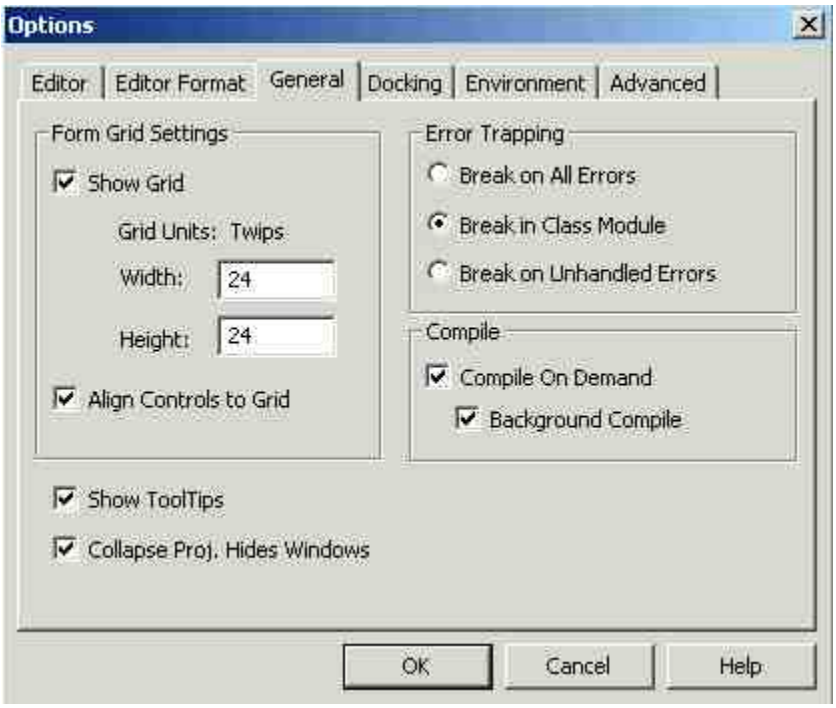
Lưu ý: Một số control có kích thước tối thiểu, bạn không thể làm cho nó nhỏ hơn được. Thí dụ như Combobox, nó phải cao đủ để display một hàng text.

Tương tự như thế, bấm nút Ctrl trong khi bấm một arrow key, control sẽ di chuyển theo chiều của ArrowKey.

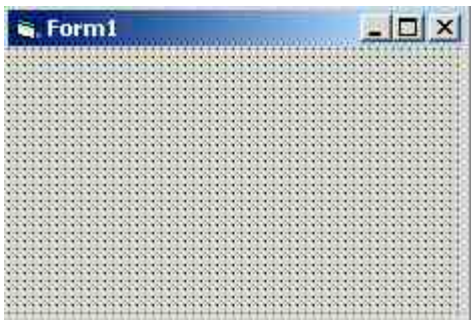
Ngoài ra, nên nhớ rằng trong lúc chương trình chạy (at run-time), trong code ta có thể thay đổi kích thước và vị trí các controls dễ dàng, thậm chí có thể làm cho chúng hiện ra hay biến mất bằng cách sửa đổi value các property left, top, width, height và visible của các controls.

Alignment Grid

Để giúp bạn sắp đặt ngay ngắn các controls trên một form, VB6 cho bạn Alignment Grid. Nó là những dấu đen của các hàng dọc và xuôi trên form. Bạn có thể làm cho các dấu đen của grid trên form biến mất bằng cách dùng menu command **Tools | Options** để display Option Dialog, kế đó chọn Tag General và clear checkbox "Show Grid":



Bạn cũng có thể nhân dịp nầy thay đổi khoảng cách chiều rộng (Width) và chiều cao (Height) của các chấm đen của grid. Kích thước nhỏ nhất của Width hay Height là 24. Hãy so sánh hai trường hợp form có và không có Show Grid như dưới đây:



Control Locking

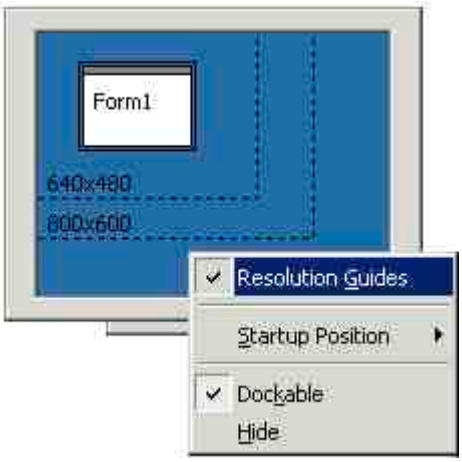
Một khi bạn đã sắp đặt kích thước và vị trí của các control trên form rồi, rất dễ ta tình cờ thay đổi các đặc tính ấy vì vô ý click lên một control. Do đó VB6 cho ta Menu command **Format | Lock Controls** để khóa chúng lại. Sau khi khóa, cái hình ống khóa trên menu bị chìm xuống.

Nếu sau nầy bạn muốn thay đổi kích thước hoặc vị trí của chúng thì nhớ dùng Menu command **Format | Lock Controls** lại. Sau khi mở khóa, cái hình ổng khóa trên menu hiện ra bình thường.

Cài đặt các Properties của Form

Nhiều property của một form ảnh hưởng đến diện mạo vật lý (physical appearance) của nó. Property Caption sẽ quyết định text được hiểu thị trong title. Nếu Property BorderStyle của form không phải là Sizable thì User không thể resize form at run-time. Property Icon quyết định hình icon được dùng trong title của form, nhất là khi form thu nhỏ (minimized). Nếu bạn không muốn cho phép User minimize hay maximize form thì set value của property MinButton, MaxButton ra False. Nếu property ControlBox là False thì form sẽ không có nút minize, maximize hay close (**x**) trên góc phải của nó, đồng thời form cũng không display cả icon bên góc trái title như trong hình dưới đây:

Vị trí đầu tiên (top,left) của form có thể được thay đổi trong design time bằng cách di chuyển hình nhỏ của nó trong window Form Layout:



Property WindowState xác định Form sẽ có kích thước bình thường (normal=0), hay minimized (=1), maximized =(2).

Lưu ý là property Font của Form sẽ được các control nằm trên nó thừa kế. Tức là khi bạn đặt một control lên form, property Font của control ấy sẽ tự động trở nên giống y như của form.

Vài Event thông dụng của Form

Nhìn từ một phương diện, Form cũng giống như Control. Ta có thể instantiate một form nhiều lần để có nhiều form tương tợ nhau. Trong thí dụ dưới đây, ta instantiate Form2 hai lần để có MyForm và YourForm:

```
Private Sub CmdCreateForms_Click()
   Dim MyForm, YourForm
   Set MyForm = New Form2
   MyForm.Caption = "This is My Form"
   MyForm.Show
   MyForm.Move 1000, 1000
   Set YourForm = New Form2
   YourForm.Caption = "YOUR FORM IS HERE"
   YourForm.Show
   YourForm.Move 2000, 2000
End Sub
```

Một Form cũng có nhiều Events rất hữu dụng.

- **Form_Initialize**: Event nầy xãy ra trước nhất và chỉ một lần thôi khi ta instantiate form đầu tiên. Ta dùng Form_Initialize event để thực hiện những gì cần phải làm chung cho tất cả các instances của form nầy.
- **Form_Load**: Event nầy xãy ra mỗi lần ta instantiate một form. Nếu ta chỉ dùng một instance duy nhất của một form trong chương trình thì Form_Load coi như

tương đương với Form_Initialize. Ta dùng Form_Load event để initialise variables, controls v.v. cho instance nầy.

Bên trong Form_Load bạn không thể dùng Setfocus cho một control nào trên form vì form chưa hẳn thành hình (ra đời). Muốn làm việc ấy bạn phải delay (trì hoản) một chút xíu bằng cách dùng Control Timer để đợi cho Form_Load được hoàn tất. Thí dụ:

```
Private Sub Form_Load()
   Timer1.Interval = 500
   Timer1.Enabled = True
End Sub
Private Sub Timer1_Timer()
   Timer1.Enabled = False  ' Timer1_Timer only execute once
   txtName.Setfocus  ' Make Tab Cursor start at TextBox txtName
End Sub
```

- **Form_Activate**: Mỗi lần một form trở nên active (current) thì nó generate một Activate event. Ta có thể dùng event nầy để refresh display trên form.
- **Form_QueryUnload**: Khi User click dấu **x** phía trên bên phải để close form thì nó generate QueryUnload event. Syntax của Sub nầy như dưới đây:

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

Event nầy cho ta một dịp để cancel Close action của form (tức là không cho User close form) bằng cách set Cancel bằng 1. UnloadMode cho ta biết ai, task hay form nào muốn close form nầy.

Ngoài ra, bạn cũng nên biết rằng một form tự động Load hay trở nên active nếu bạn nhắc đến nó, thí dụ như dùng **Form2.List1**. Khi một form đã được loaded rồi bạn có thể hide (làm cho biến mất) nó. Kế đó, khi bạn show form ra trở lại thì form không có gì thay đổi. Nhưng nếu bạn Unload một form (thí dụ bằng cách dùng **Unload Form2**), rồi sau đó load trở lại bằng cách dùng Form2.Show chẳng hạn, thì Form phải trải qua quá trình Form_Load, và dĩ nhiên form mất tất cả những gì có trước

đây. Ngoài ra, Hide/Show một form đã được loaded rồi thì rất nhanh, còn Unload/Load thì mất thì giờ hơn.

Khi bạn Show một Form chưa hiện hữu thì form sẽ được loaded và show. Đôi khi bạn muốn Load một form, rồi làm việc với nó trước khi Show, trong trường hợp đó bạn dùng **Load Form2** rồi một chập sau dùng **Form2.Show**.

MDI Form

Đôi khi bạn muốn có một MDI form, tức là một form có thể chứa nhiều form con bên trong. Dạng MDIform nầy thường được dùng trong các application như wordprocessor để có thể mở nhiều document cùng một lúc, mỗi document được hiển thị trong một form con. Để có một MDIForm bạn cần phải dùng menu command **Project | Add MDI Form**. Mỗi VB6 project chỉ có thể có tối đa một MDIform. Muốn một form trở thành một form con bạn set property MDI Child của nó thành True. At run-time bạn không thể hide (biến nó thành invisible) một MDIChild form, nhưng có thể minimize nó. Nếu bạn thật sự muốn hide nó thì phải dùng mánh lới là cho nó vị trí (top,left) số âm lớn hơn kích thước nó để nó nằm ngoài tầm hiển thị của form. Trong một chương trình dùng MDI Form, khi bạn click MDI Form nó không nhảy ra phía trước và che các form con, nhưng vẫn luôn luôn nằm ở dưới.

Controls là gì?

Controls vừa có hình, vừa có code chạy bên trong một window nho nhỏ, giống như một form. Khi ta lập trình VB6 ta lấp ráp các controls (là những vật dụng tiền chế) trên một hay nhiều form để có một chương trình nhanh chóng. Ta giao dịch với một control qua ba đặc tính của control:

- **Properties:** tập hợp các đặc tính của control mà ta có thể ấn định lúc design time hay run-time. Có nhiều properties về diện mạo, nếu ta thay đổi at design time sẽ thấy kết quả hiện ra lập tức, thí dụ Font hay màu sắc.
- **Methods:** những gì control thực hiện được, tức là những khả năng của nó.
- **Events:** những sự cố mà control sẽ thông báo cho chúng ta biết khi nó xãy ra với control. Khi một event xãy ra VB6 sẽ xử lý một

Event Handler (thí dụ như Sub Command1_Click()), miễn là chúng ta viết code sẵn trong đó. Nếu không có code thì coi như chúng ta không thèm biết đến các event loại đó. Có một số Events mà chúng ta thường xử lý là:

- Click : xãy ra khi user click lên control. Ta thường dùng nó cho CommandButton và Listbox.
- MouseDown, MouseUp : mỗi khi User bấm một mouse button là có một MouseDown Event, khi User buông nó ra thì có một MouseUp Event. Ta thường dùng MouseDown Event để Popup context sensitive menu hay bắt đầu một diễn biến Drag.

Thí dụ:

```
Private Sub Foods_MouseDown(Button As Integer, Shift As
Integer, X As Single, Y As Single)
   If Button = vbRightButton Then  ' if Right button was
pressed
      PopupMenu mnuActions  ' popup a menu
   End If
End Sub
Private Sub DrinkList_MouseDown(Button As Integer, Shift
As Integer, X As Single, Y As Single)
   DrinkList.drag  ' Displaying a drag icon to start the
drag process
End Sub
```

Để ý là Click không cho chúng ta thêm chi tiết gì về sự cố, trong khi MouseDown/MouseUp cho ta biết vị trí của cursor, button nào của Mouse được bấm và lúc ấy User có bấm nút Shift, Ctrl hay Alt không. Mỗi Click là đi đôi với một cặp MouseDown/MouseUp. Nếu bạn muốn xử lý vừa Click lẫn MouseDown thì phải cẩn thận. Thí dụ bạn muốn vừa handle Click event vừa handle Mouse Drag thì phải làm sao phân biệt hai trường hợp. Nếu không User chỉ muốn thấy kết quả của Click mà lại thấy control bắt đầu display một Drag icon thì sẽ bực mình.

KeyPress : xãy ra khi user Press một key. Ta thường dùng nó cho TextBox để loại ra (filter out) các keystrokes ta không chấp nhận. KeyPress cho ta ASCII value, một con số có giá trị từ 1 đến 255, của key.

Trong thí dụ dưới đây, một Enter key sẽ được coi như một TAB key:

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then
      KeyAscii = 0  ' Swallow the character to avoid side
effect
      SendKeys "{TAB}"  ' Emulate entering a TAB
   End If
End Sub
```

KeyDown, KeyUp : mỗi KeyPress event là cho ta một cặp KeyDown/KeyUp event. KeyDown/KeyUp cho ta KeyCode và Shift value. Để detect Function key ta cần dùng KeyDown event.

Trong thí dụ dưới đây, ta display Function key User bấm:

```
Private Sub Text3_KeyDown(KeyCode As Integer, Shift As
Integer)
   If (KeyCode >= 112) And (KeyCode <= 123) Then
      MsgBox "You pressed the Function key: F" &
Trim(Str(KeyCode - 111))
   End If
End Sub
```

- GotFocus : Control trở nên active khi nó nhận được Focus. Nó sẽ generate một GotFocus Event. Ta có thể dùng nó để đổi màu background của một text box như trong thí dụ dưới đây:

```
Private Sub Text2_GotFocus()
   Text2.BackColor = vbYellow
End Sub
```

LostFocus : Thường thường hể một Control GotFocus thì trước đó có một Control LostFocus. Ta có thể dùng Event nầy để Validate entry data hay thu xếp công chuyện cho một control vừa mất Focus.

Trong thí dụ dưới đây, nếu User không đánh vào một con số ở trong Textbox Text1 thì sẽ được thông báo và Tab Cursor sẽ trở lại Textbox Text1.

```
Private Sub Text1_LostFocus()
   If Not IsNumeric(Text1.Text) Then
      MsgBox "Please enter a number!"
      Text1.SetFocus
   End If
End Sub
```

- DragDrop : xãy ra khi ta drop một cái gì lên control . Parameter Source cho ta biết Control nào đã được Drag và Drop. Nhiều khi một control có thể nhận drop từ nhiều control khác nhau. Trong trường hợp đó ta phải test xem hoặc Control Type, hoặc Name hoặc Tag value của Source control là gì để tùy nghi xử lý.

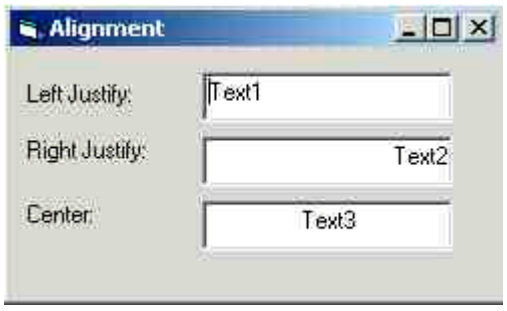
  Trong thí dụ dưới đây, khi User drop mouse xuống Textbox Text2, nếu Source là một Listbox, không cần biết Listbox nào, thì ta copy dòng được chọn trong Listbox ấy qua Textbox Text2.

```
Private Sub Text2_DragDrop(Source As Control, X As
Single, Y As Single)
   If TypeOf Source Is ListBox Then
      Text2.Text = Source.Text
   End If
End Sub
```

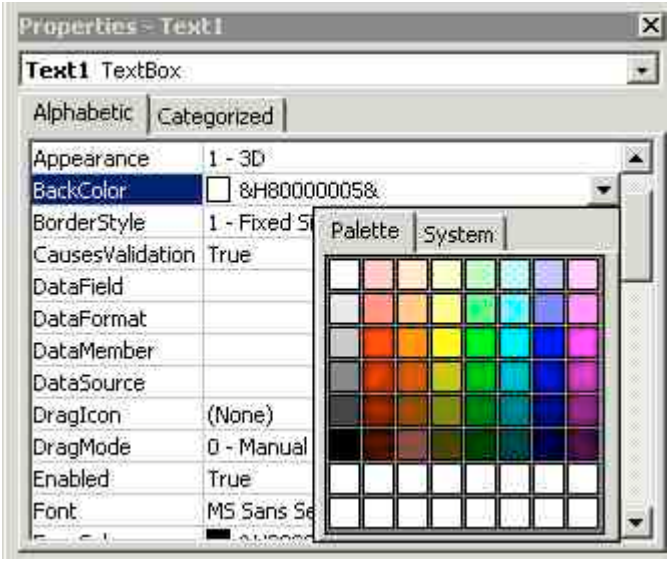
TextBox

TextBox là control được dùng nhiều nhất để display text và nhận keystroke của Userđể sửa đổi text có sẵn hay cho vào text mới. Property chính và default của Textbox là **text**, tức là thường thường Text2.text có thể được viết tắt là Text2. Ta có thể disable (khiến nó bất lực, không phản ứng gì hết và không cho sửa đổi) một text box bằng cách set Property Enable ra False (chữ sẽ bị mờ đi), hay Lock (không cho sửa đổi) một text box bằng cách set Property Locked ra True (chữ không bị mờ).

Text có thể được Align (Alignment Property) để display bên trái, chính giữa hay bên phải của hộp nó.

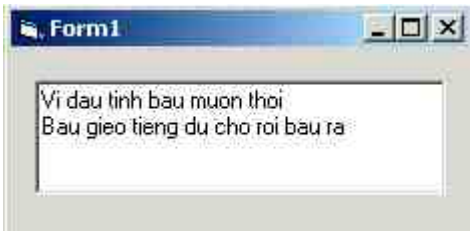


Bạn có thể chọn BackColor và ForeColor cho background và text của TextBox. Dùng Tag Palette khi chọn màu để có đúng một màu bạn muốn.



Dĩ nhiên bạn có thể lựa chọn Font và cở chữ cho Text với Font Property. Bạn giới hạn số characters mà User có thể enter cho TextBox bằng cách set MaxLength Property.

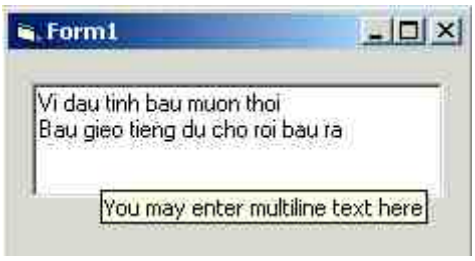
Nếu Property Multiline là True thì User có thể enter nhiều hàng. At Design time, nếu bạn muốn enter multiline thì phải nhớ bấm nút Ctrl khi press Enter mỗi khi xuống hàng. Nếu không VB6 IDE tưởng rằng bạn đã kết thúc editing.



Muốn assign cho text box multiline text thì phải nhét vào mỗi cuối hàng CarriageReturn và LineFeed characters. Thí dụ như:

```
Private Sub Command1_Click()
   Dim TextStr
   TextStr = "Bau ra bau lay ong cau" & vbCrLf  ' Note: vbCrLf =
chr(13) & chr(10)
   TextStr = TextStr & "Bau cau ca bong ngat dau kho tieu"
   Text1.Text = TextStr
End Sub
```

Nếu bạn muốn mách nước cho User về cách dùng một textbox nào đó thì có thể dùng Property ToolTipText để nó display mách nước mỗi khi mouse cursor nằm lên textbox.



Dùng Property TabIndex để ấn định thứ tự cho Tab Cursor dừng mỗi khi User bấm nút TAB để dời TAB Cursor đến Textbox kế tiếp. Nếu bạn không muốn Tab Cursor dừng ở một TextBox nào thì set Property TabStop nó thành False. Tab Cursor không dừng ở Textbox có Property Enabled bằng False, nhưng vẫn dừng ở Textbox có property Locked bằng True.

Nếu bạn muốn dùng Textbox làm một Password field thì set Property PasswordChar bằng "*". Làm như thế sẽ ép buộc Textbox display mọi character bằng PasswordChar, tức là "*", để người khác không đọc được trong khi User enter một Paswword.

Properties SelLength, SelStart và SelText

Nếu bạn muốn biết được tình hình hiện thời của Textbox: SelText cho bạn dãy chữ đang được selected. SelStart cho bạn vị trí của insertion point (chỗ cursor flashing). SelLength cho biết con số characters đã được selected.

Nếu bạn muốn sửa đổi text trong Textbox: SelText cho bạn nhét vào một dãy chữ. SelStart cho bạn ấn định vị trí bắt đầu của dãy chữ bạn sắp select. SelLength ấn định số characters bạn muốn chọn, bắt đầu từ SelStart.

Dưới đây là một thí dụ trong đó ta highlight text tìm được:

```
Private Sub Form_Click ()
   Dim Search, Where  ' Declare variables.
   ' Get search string from user.
   Search = InputBox("Enter text to be found:")
   Where = InStr(Text1.Text, Search)  ' Find the given string in
Text1.Text.
   If Where > 0 Then  ' If found,
      Text1.SelStart = Where - 1  ' set selection start and
      Text1.SelLength = Len(Search)  ' set selection length.
   Else
      MsgBox "String not found."  ' Notify user.
   End If
End Sub
```

CommandButton

CommandButton rất tiện cho ta dùng vào việc xử lý một chuyện gì khi User click lên button. Event ta dùng thường nhất cho CommanButton là Click. Ta dùng Property Caption của CommandButton để enter cái gì ta muốn display trên button. Nếu muốn cho phép User dùng ALT+E (đè nút Atl trong lúc bấm nút E) để generate event click thì nhét dấu "**&**" trước chữ E trong Caption của button. Caption sẽ display chữ E với một gạch dưới.

Ngoài ra ta cũng có thể cho thêm một cái hình vào CommandButton bằng cách chọn một icon cho property Picture và set Property Style ra Graphical, thay vì Standard.

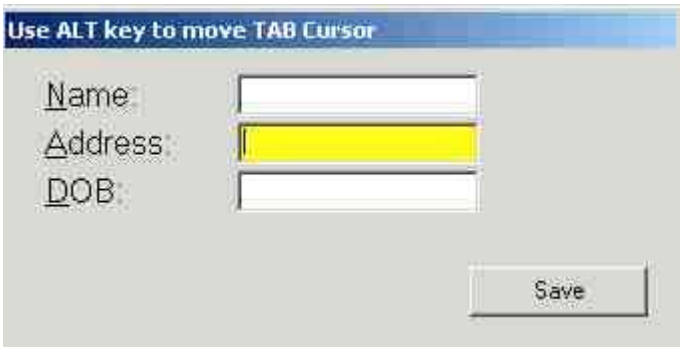
Lúc Run-time bạn có thể thay đổi hình hay Caption của CommandButton. Trong thí dụ dưới đây, Caption của CommandButton CmdOperation flip-flop giữa hai values Stop và Start:

```
Private Sub CmdOperation_Click()
   If CmdOperation.Caption = "&Stop" Then
      CmdOperation.Caption = "St&art"
   Else
      CmdOperation.Caption = "&Stop"
   End If
End Sub
```

Label

Mục đích chính của Label là để display, không cho User Edit như Textbox. Do đó ta có thể dùng Property Font, ForeColor và Backcolor để làm cho nó đẹp. Ngoài ra Property BorderStyle có thể cho Label lỏm xuống nếu bạn set nó bằng Fixed Single. Nếu set property BackStyle bằng Transparent sẽ tránh trường hợp Backcolor của Label làm cho không đẹp.

Label cũng có Property Tabindex. Nếu bạn muốn dùng ALT key để mang Tab Cursor về một Textbox, hãy để một Label với TabIndex bằng TabIndex của TextBox trừ 1. Giả sử Label có Caption là "&Address" thì ALT+A sẽ mang Tab Cursor về TextBox màu vàng như trong thí dụ dưới đây:



Ngoài ra nhớ rằng bạn có thể thay đổi Caption của Label lúc run-time.

CheckBox

CheckBox được dùng để User xác nhận có đặc tính nào một cách nhanh chóng. Property Value của CheckBox có thể là Checked (làm cho hộp vuông có dấu, bằng 1), Unchecked (làm cho hộp vuông trống không, bằng 0) hay Grayed (làm cho hộp vuông có dấu màu nhạt, bằng 2). Một khi biết rằng CheckBox có Value bằng 1, ta có thể đọc Caption của CheckBox để dùng nếu cần.

Bạn có thể dùng Property Alignment để làm cho Caption đứng bên phải (Left Justify) hay bên trái (Right Justify) của hộp vuông.

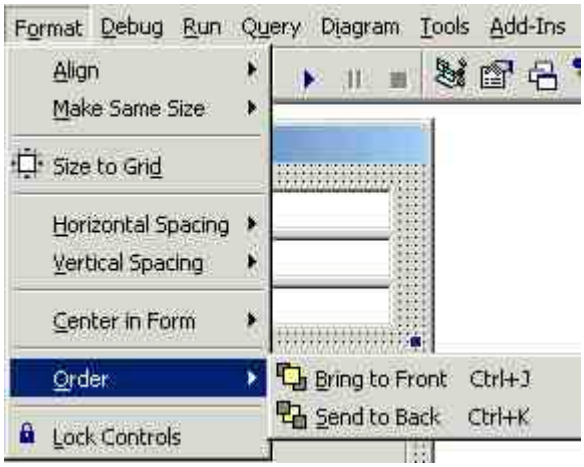
OptionButton

OptionButton (còn gọi là RadioButton) có hình tròn với một chấm ở giữa, thay gì hình vuông với một gạch ở giữa như CheckBox. OptionButton luôn luôn được qui tụ thành một nhóm, chứa trong một container. Container là một Control có khả năng chứa các controls khác. Frame, PictureBox, hay chính Form đều là Container. Sau khi đặt một Container lên Form, nếu muốn để một OptionButton lên Container, trước hết ta phải Select container, rồi kế đó chọn OptionButton. Sở dĩ, tất cả OptionButtons phải nằm trong một container là vì bất cứ lúc nào, nhiều nhất là một OptionButton trong container có value True (vòng tròn có chấm ở giữa).

Muốn biết một OptionButton có thật sự nằm trong một container, bạn thử kéo cái container đi chỗ khác. Nếu OptionButton bị dời theo container thì nó nằm trong container. Một cách khác là thử kéo OptionButton ra khỏi container. Nếu kéo ra được thì nó không nằm trong container.

Muốn di chuyển một OptionButton từ container nầy sang container khác, bạn Cut OptionButton rồi Paste nó vô container kia.

Đôi khi một container nằm che trên một control khác. Muốn mang một container ra phía sau các controls khác bạn Select container rồi dùng Menu command **Format | Order | Send to Back**.



Chương Bốn - Viết Code

Trong ba chương đầu chúng ta đã học qua ba bộ phận chánh của một chương trình Visual Basic 6.0. Đó là:

- **Forms** là cái nền hay khung để ta xây dựng User Interface.
- **Controls** là những viên gạch để ta dùng xây dựng User Interface.
- **Event procedures** là code nằm phía sau những hình ảnh, nó là chất keo dùng để dán các Controls lại với nhau để tạo thành chương trình áp dụng của ta.

Như ta đã thấy, tất cả các code được xử lý (executed) khi có một Event xãy ra. Thí dụ như khi User click một CommandButton (Event Click) hay type nút Tab để di chuyển Cursor từ Textbox nầy (Event Lostfocus) qua Textbox khác (Event GotFocus). Các nhóm code xử lý là :

```
Private Sub Command1_Click()
   ...
End Sub
Private Sub Text1_LostFocus()
   ...
End Sub
và
Private Sub Text2_GotFocus()
   ...
End Sub
```

Trong khi lập trình, mỗi lần ta double click lên một Control của một Form là VB6 IDE tự động generate cho ta cái vỏ từ hàng **Private Sub**

***Control_Event*()** cho đến **End Sub** để chúng ta điền những hàng code của mình vào chính giữa.

Điều khiển thứ tự xử lý các dòng code

Giả dụ ta viết một chương trình Vb6 đơn giản như trong hình nầy với hai Textbox tên txtName, txtAge và một nút tên CmdEnter nằm trong một form tên Form1:

Thông thường các dòng code được xử lý theo thứ tự từ trên xuống dưới. Thí dụ như để kiểm xem các dữ kiện vừa được cho vào các Textbox có tương đối hợp lý hay không, khi User click nút CmdEnter, ta xử lý Sub dưới đây:

```
Private Sub CmdEnter Click()
   ' Make sure the Name field is not blank
   If txtName.Text = "" Then
      MsgBox "Please enter Name"
      Exit Sub  ' Terminate this Sub
   End If
   ' Make sure a number is supplied for Age
   If Not IsNumeric(txtAge.Text) Then
      MsgBox "Please enter a number for Age"
      Exit Sub  ' Terminate this Sub
   End If
End Sub
```

Cái Sub nói trên có chữ **Private** nằm phía trước, ý nói chỉ nội trong cùng một form chứa Control CmdEnter (tức là Form1 trong trường hợp nầy) ta mới có thể gọi (dùng) Sub CmdEnter_Click().

Thí dụ ta muốn khi User bấm key "Enter" trên bàn phím sau khi cho vào chi tiết ở Textbox txtAge thì coi như User đã click nút CmdEnter. Ta viết như sau:

```
Private Sub txtAge_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then
      KeyAscii = 0  ' swallow Key Enter to avoid side effect
      CmdEnter_Click  ' Call Private Sub CmdEnter_Click from the same
form
   End If
End Sub
```

Khi ta dùng câu **CmdEnter_Click** làm một dòng code (còn gọi là gọi Sub CmdEnter_Click) thì coi như tương đương với nhét tất cả 10 dòng codes giữa hai hàng **Private Sub CmdEnter_Click()** và **End Sub** tại chỗ câu CmdEnter_Click, như viết lại dưới đây:

```
Private Sub txtAge_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then
      KeyAscii = 0  ' Swallow Key Enter to avoid side effect
      ' Make sure the Name field is not blank
      If txtName.Text = "" Then
         MsgBox "Please enter Name"
         Exit Sub  ' Terminate this Sub
      End If
      ' Make sure a number is supplied for Age
      If Not IsNumeric(txtAge.Text) Then
         MsgBox "Please enter a number for Age"
         Exit Sub  ' Terminate this Sub
      End If
   End If
End Sub
```

Có một cách nói khác là khi execution đi đến hàng **CmdEnter_Click** thì nó nhảy vào **Private Sub CmdEnter_Click()** để execute cho đến hết rồi nhảy trở lại hàng kế tiếp trong **Private Sub txtAge_KeyPress(KeyAscii As Integer)** Trong **Private Sub CmdEnter_Click()** nếu User không đánh gì vào Textbox txtName thì chương trình sẽ display message "Please enter Name" rồi **Exit Sub**. Đây là cách nhảy ngay ra khỏi Sub chớ không đợi phải execute xuống tới hàng chót.

Dùng IF....THEN statement

Trong **Private Sub CmdEnter_Click()** ở trên ta thấy có hai chỗ dùng **IF...THEN** để thử xem một điều kiện gì có được thỏa mãn không. Nếu điều kiện là đúng vậy, tức là **True** thì ta thực hiện những gì được viết từ hàng **IF...THEN** cho đến hàng **END IF**. Ngược lại, nếu điều kiện không đúng thì execution nhảy xuống tới dòng code nằm ngay dưới dòng **END IF**. Tức là có khi execution sẽ đi ngang qua, có khi không đi ngang qua những dòng code ở giữa câu **IF...THEN** và câu **END IF**. Điều kiện trong IF Statement là phần nằm giữa hai chữ **IF** và **THEN**. Nó được gọi là **Logical Expression**. Ta có:

```
   txtName.text = ""  ' content of Textbox txtName is nothing, i.e.
an empty string
và
   NOT IsNumeric(txtAge.text)  ' content of TextBox txtAge is not a
number
```

Trong Logical Expression thứ nhì ta dùng Function IsNumeric để được cho biết rằng txtAge.text có phải là một con số hay không. Vì ta chỉ than phiền khi txtAge không phải là một con số nên ta phải để thêm chữ NOT phía trước. Tức là khi

```
   IsNumeric(txtAge.text) = False
```

```
thì
   NOT IsNumeric(txtAge.text) = True
```

Nếu giữa **IF...THEN** và **END IF** chỉ có một dòng code bạn có thể nhập dòng code lên với **IF...THEN** và không dùng **END IF**. Tức là:

```
   If theColorYouLike = vbRed Then
      MsgBox "You 're a lucky person!"
   End If
is equivalent with
   If theColorYouLike = vbRed Then MsgBox "You 're a lucky person!"
```

Một Logical Expression có thể đơn giản (simple) như trong các thí dụ trên hay rắc rối hơn nếu ta ráp nhiều simple Logical Expression lại với nhau bằng cách dùng những từ **OR** và **AND**. Khi hai Logical Expression được ráp lại bằng chữ **OR (HAY)** thì chỉ cần ít nhất một trong hai Expression là TRUE là Logical Expression tổng hợp cũng là TRUE. Cái TRUE Table cho OR như sau:

A	B	A OR B
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

Trong thí dụ dưới đây nếu một người 25 tuổi trở lên HAY có lợi tức trên 30 ngàn đô la một năm thì cho mượn tiền được :

```
If (PersonAge >= 25) Or (PersonIncome >= 30000) Then
   LendPersonMoney
End If
```

Để ý cách dùng các dấu ngoặc đơn giống như trong toán đại số. Thông thường hể cái gì nằm trong ngoặc thì mình tính trước. Nếu có nhiều lớp dấu ngoặc thì tính theo thứ tự từ trong ra ngoài. Như trong bài trên ta tính xem **PersonAge >= 25** xem là TRUE hay FALSE, rồi tính xem **PersonIncome >= 30000** xem là TRUE hay FALSE, trước khi tính kết quả tổng hợp, dựa vào cái TRUE table cho OR.

Khi hai Logical Expression được ráp lại bằng chữ **AND (Và)** thì chỉ khi nào cả hai Expression đều là TRUE, Logical Expression tổng hợp mới là TRUE. Cái TRUE Table cho AND như sau:

A	B	A AND B

FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

Trong thí dụ dưới đây nếu học sinh 18 tuổi trở lên và cha mẹ kiếm 100 ngàn trở lên một năm thì đăng ký học sinh ở một đại học tư:

```
If (StudentAge >= 18) And (ParentIncome >= 100000) Then
   EnrollStudentAtPrivateUniversity
End If
```

Một Logical Expression có thể tập hợp cả OR lẫn AND như trong thí dụ dưới đây nếu học sinh 18 tuổi trở lên và cha mẹ kiếm 100 ngàn trở lên một năm HAY học sinh có Intelligent Quotient cao hơn 160 thì đăng ký học sinh ở một đại học tư:

```
If ((StudentAge >= 18) And (ParentIncome >= 100000)) Or (StudentIQ >
160) Then
   EnrollStudentAtPrivateUniversity
End If
```

Hai dấu ngoặc đơn nằm bên ngoài của:

```
  ((StudentAge >= 18 ) And (ParentIncome >= 100000))
```

không cần thiết vì theo qui ước, ta tính AND expression trước khi tính OR expression, nhưng nó giúp ta đọc dễ hơn.

Dùng IF....THEN..ELSE statement

Hãy xem thí dụ:

```
If (StudentPassmark > 75) Then
   ' Part A
   EnrollStudentAtPublicSchool
Else
   ' Part B
   EnrollStudentAtPrivateSchool
End If
```

Nếu học sinh đậu với số điểm trên 75 thì cho học trường công, NẾU KHÔNG thì phải học trường tư. Tức là nếu **StudentPassmark > 75** là TRUE thì xử lý phần A, nếu không thì xử lý phần B. Để ý phần A gồm những dòng code nằm giữa dòng **If (StudentPassmark > 75) then** và **else**. Còn phần B gồm những dòng code nằm giữa dòng **else** và **end if**.

Ta có thể ráp chữ **ELSE** với chữ **IF** để dùng như trong thí dụ sau đây:

<
```
If (StudentPassmark > 75) Then
   EnrollStudentAtPublicSchool
ElseIf (StudentPassmark >= 55) Then
   EnrollStudentAtSemipublicSchool
Else
   EnrollStudentAtPrivateSchool
End If
```

Nếu học sinh đậu với số điểm trên 75 thì cho học trường công, Nếu từ 55 điểm đến 75 điểm thì cho học trường bán công, nếu không (tức là điểm đậu dưới 55) thì phải học trường tư.

Nếu ở tỉnh nhỏ, không có trường tư, ta không có quyết định cho học trò đậu dưới 55 điểm học ở đâu thì bỏ phần **ELSE** trong thí dụ trên. Phần chương trình trở thành:

```
If (StudentPassmark > 75) Then
   EnrollStudentAtPublicSchool
ElseIf (StudentPassmark >= 55) Then
   EnrollStudentAtSemipublicSchool
End If
```

Ta có thể dùng ELSEIF nhiều lần như sau:

```
If (TheColorYouLike = vbRed) Then
   MsgBox "You 're a lucky person"
ElseIf (TheColorYouLike = vbGreen) Then
   MsgBox "You 're a hopeful person"
ElseIf (TheColorYouLike = vbBlue) Then
   MsgBox "You 're a brave person"
ElseIf (TheColorYouLike = vbMagenta) Then
   MsgBox "You 're a sad person"
Else
   MsgBox "You 're an average person"
End If
```

Execution đi lần lượt từ trên xuống dưới, nếu một điều kiện IF là TRUE thì xử lý phần của nó rồi nhảy xuống ngay dưới dòng **END IF**. Chỉ khi một điều kiện IF không được thỏa mãn ta mới thử một điều kiện IF bên dưới kế đó. Tức là nếu bạn thích màu đỏ lẫu màu tím (magenta) thì chương trình sẽ display "You're a lucky person", và không hề biết "You're a sad person".

Dùng SELECT CASE statement

Thí dụ có nhiều ELSEIF như trên có thể được viết lại như sau:

```
Select Case TheColorYouLike
```

```
Case vbRed
   MsgBox "You 're a lucky person"
Case vbGreen
   MsgBox "You 're a hopeful person"
Case vbBlue
   MsgBox "You 're a brave person"
Case vbMagenta
   MsgBox "You 're a sad person"
Else
   MsgBox "You 're an average person"
End Select
```

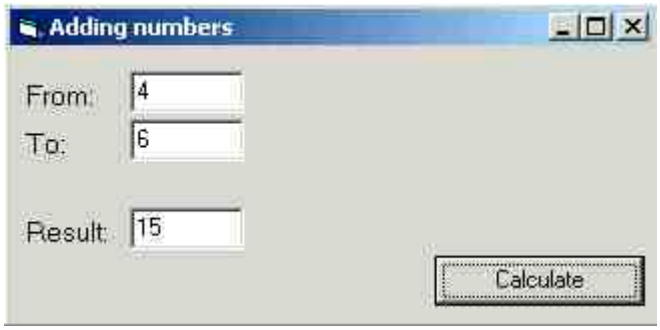
Cách viết nầy tương đối dễ đọc và ít nhầm lẫn khi viết code hơn là dùng nhiều ELSEIF. Phần **ELSE** trong Select Case statement thì optional (nhiệm ý), tức là có cũng được, không có cũng không sao. Hể khi điều kiện của một **Case** được thoả mãn thì những dòng code từ đó cho đến dòng **Case** kế dưới hay **Else** được xử lý và tiếp theo execution sẽ nhảy xuống dòng nằm ngay dưới dòng **End Select**.

Nhớ là dưới cùng ta viết **End Select**, chớ không phải **End If**. Các Expression dùng cho mỗi trường hợp **Case** không nhất thiết phải đơn giản như vậy. Để biết thêm chi tiết về cách dùng Select Case, bạn highlight chữ **Case** (doubleclick chữ Case) rồi bấm nút **F1**.

Dùng FOR statement

Trong lập trình, nói về Flow Control (điều khiển hướng đi của execution) ta dùng hai loại statement chính: **Branch statements** như IF..THEN..ELSE (kể cả Select Case) và **Iterative statements** (lập đi, lập lại) như FOR và WHILE LOOP (Vòng). Ta sẽ nói đến WHILE Loop trong phần kế tiếp. Trong khi Branch statement cho phép ta execute trong nhánh nầy hay nhánh kia tùy theo value của Logical Expression thì Iterative statement cho ta execute một phần code lập đi, lập lại nhiều lần cho đến khi một điều kiện được thỏa mãn.

Giả dụ ta viết một chương trình đơn giản để tính tổng số các con số giữa bất cứ hai con số nào (coi chừng lớn quá). Cái form của chương trình giống như dưới đây:



Sau khi cho hai con số **From (Từ)** và **To (Cho đến)** ta click nút Calculate và thấy kết quả hiện ra trong Textbox txtTotal. Cái Sub tính tổng số được liệt ra dưới đây:

```
Private Sub CmdTotal_Click()
   Dim i, FromNo, ToNo, Total
   FromNo = CInt(txtFromNumber.Text)   ' Convert Text string ra
internal number b?ng Function CInt
   ToNo = CInt(txtToNumber.Text)   ' Convert Text string ra internal
number b?ng Function CInt
   Total = 0   ' Initialise Total value to zero
   For i = FromNo To ToNo   ' Iterate from FromNo to ToNo
      Total = Total + i   ' Add the number to the Total
   Next
   txtTotal.Text = CStr(Total)   ' Convert internal number ra Text
string
End Sub
```

Trong thí dụ trên, **FOR** loop bắt đầu từ dòng **For i = FromNo To ToNo** và chấm dứt ở dòng **Next**. Khi execution bắt đầu Total bằng 0, i bằng FromNo. Execution sẽ đi qua hết những dòng trong FOR loop rồi value của i sẽ được tăng lên 1, rồi execution sẽ bắt đầu lại ở đầu loop. Trong thí dụ nầy vì FromNo=4 và ToNo=6 nên execution sẽ đi qua cái FOR loop 3 lần. Lần thứ nhất i=4, lần thứ nhì i=5 ,và lần thứ ba thì i=6. Sau đó, khi i=7 thì nó lớn hơn ToNo (=6) nên execution nhảy ra khỏi FOR loop. Kết quả là Total=15 và được display trong Textbox txtTotal, sau khi được converted từ internal number ra text string với Function CStr.

Nếu ta chỉ muốn cộng những số chẳn từ 4 đến 16 ta có thể làm cho i tăng value lên 2 (thay vì 1) mỗi khi đến cuối loop. Tức là i=4,6,8 .v.v..Ta sẽ thêm chữ **STEP** trong FOR statement như sau:

```
For i = 4 To 16 Step 2   ' Iterate from 4 to 16 with Step=2
   Total = Total + i   ' Add the number to the Total
Next
```

Total sẽ bằng 4+6+8+10+12+14+16= 70. Trong thí dụ trên ta cũng có thể dùng STEP số âm như sau:

```
For i = 16 To 4 Step -2  ' Iterate from 16 to 4 with Step=-2
   Total = Total + i  ' Add the number to the Total
Next
```

Trong trường hợp nầy FOR loop bắt đầu với i=16. Khi đến cuối loop lần thứ nhất value của i bị bớt 2 và trở thành 14. Sau đó i bị giảm giá trị dần dần đến 4. Kế đó i=2 thì nhỏ hơn số cuối cùng (=4) nên execution nhảy ra khỏi FOR loop.

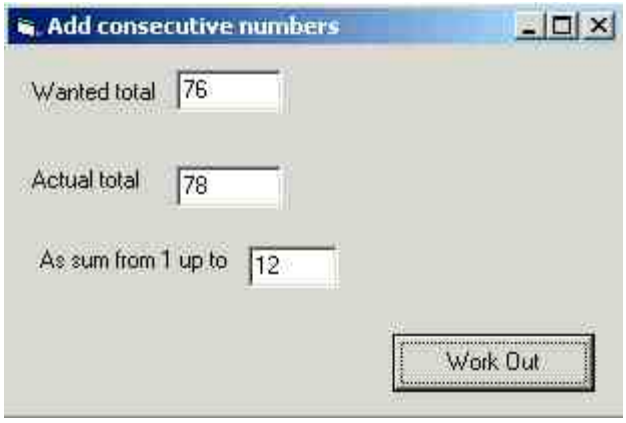
Giả dụ ta muốn lấy ra tất cả những blank space trong một text string. Ta biết con số characters trong một text string, còn gọi là chiều dài của text string có thể tính bằng cách dùng Function Len(TString). Và để nói đến character thứ i trong một Text string ta dùng Mid Function.

Khi User click button **Remove Blank Spaces** chương trình sẽ execute Sub dưới đây:

```
Private Sub CmdRemoveBlankSpaces_Click()
   Dim i, TLen, TMess
   TMess = ""  ' Initialise temporary String to null string
   For i = 1 To Len(txtOriginalString.Text)  ' Iterate from the first
chracter to the last character
      of the string
      ' Check if chracter is NOT a blank space
      If Mid(txtOriginalString.Text, i, 1) <> " " Then
         ' Character is not a blank space - so append it to TMess
         TMess = TMess & Mid(txtOriginalString.Text, i, 1)
      End If
   Next
   txtResultString.Text = TMess  ' Disaplay TMess by assigning it to
txtResultString.text
End Sub
```

Thông thường, ta dùng FOR loop khi biết trước execution sẽ đi qua loop một số lần nhất định. Nhưng thỉnh thoảng, khi một điều kiện được thỏa

mãn ta có thể ép execution nhảy ra giữa chừng khỏi FOR loop, chớ không đợi cho đến đủ số lần đi qua loop. Thí dụ như ta muốn biết phải cộng bao nhiêu số kế tiếp từ 1 trở lên để được tổng số vừa lớn hơn hay bằng 76.



Khi User click button **Work Out**, Sub dưới đây sẽ được xử lý:

```
Private Sub cmdWorkOut_Click()
   Dim i, Total, WantedTotal
   WantedTotal = CInt(txtWantedTotal.Text)   ' Convert Text string ra
internal number b?ng Function
   CInt
   Total = 0   ' Initialise Total value to zero
   For i = 1 To 30
      Total = Total + i   ' Add the number to the Total
      If Total >= WantedTotal Then Exit For   ' Jump out of FOR loop
   Next
   txtActualTotal.Text = CStr(Total)   ' Display the Actual Total
   txtUptoNumber.Text = CStr(i)   ' Display the highest number
End Sub
```

Dùng DO WHILE Loop statement

Khi ta không biết chắc là execution sẽ đi qua loop bao nhiêu lần thì tốt nhất là dùng DO WHILE Loop statement. Khác với FOR Loop, trong DO WHILE Loop ta phải tự lo initialisation (tức là mới vô đầu i bằng bao nhiêu) và tự lo tăng value của parameter i. Nếu Logical Expression là True thì execute những dòng code từ **DO WHILE** cho đến **Loop**.

Thí dụ mới vừa qua có thể viết lại bằng cách dùng DO WHILE Loop như sau:

```
Private Sub cmdWorkOut_Click()
   Dim i, Total
   WantedTotal = CInt(txtWantedTotal.Text)   ' Convert Text string ra
internal number b?ng Function
   CInt
   Total = 0   ' Initialise Total value to zero
```

```
   i = 1  ' Intialise at the first character
   Do While (Total < WantedTotal)  ' Logical Expression is (Total <
WantedTotal)
      Total = Total + i  ' Add the number to the Total
      i = i + 1  ' Increment the vakue of i
   Loop
   txtActualtotal.Text = CStr(Total)  ' Display the Actual Total
   txtUptonumber.Text = CStr(i - 1)  ' Display the highest number
End Sub
```

Trong khi Total hãy còn nhỏ hơn WantedTotal thì ta tiếp tục đi qua While Loop. Giả dụ ta có các hàng text chứa giá tiền các thứ có thể bỏ vào ổ bánh mì thịt với giá như sau:

```
Chicken Roll      45c
Roast Beef         55c
Tomato Sauce        5c
```

Bây giờ ta muốn viết code để lấy ra giá tiền từ những hàng Text string như trên. Ta sẽ đi từ bên phải lần lần qua trái cho đến khi tìm được một blank space.

```
Private Sub WorkOutPrice_Click()
   Dim i, TStr, PriceInCents, Price
   TStr = "Chicken Roll 45c"
   i = Len(TStr)  ' Starting from the rightmost character of the text
string
   ' Going from right to left, look for the first blank character
   Do While (Mid(TStr, i, 1) <> " ")
      i = i - 1  ' Keep walking to the left
   Loop
   PriceInCents = Mid(TStr, i + 1)  ' String including character "c"
   ' Discard the rightmost character which is "c" and convert the
price string to single number
   Price = CSng(Left(PriceInCents, Len(PriceInCents) - 1))
   txtPrice.Text = CStr(Price)  ' Display the highest number
End Sub
```

Dùng Function

Function là một dạng subroutine giống giống như Sub. Chỉ khác ở chỗ Function cho ta một kết quả, cho nên cách dùng Function hơi khác với Sub. Ta viết một variable bên trái dấu =, được assigned kết quả của một Function. Thí dụ như ta dùng Trim Function để loại bỏ những blank space ở hai đầu của text string TString:

```
ResultString = Trim(TString)
```

Ta đưa cho Function Trim một text string called TString. Sau khi Function Trim được executed, ta có kết quả nhưng TString không hề thay đổi. Ngược lại, khi ta gọi một Sub, tất cả những parameter ta đưa cho Sub đều có thể thay đổi trừ khi ta tuyên bố một parameter nào đó là ByVal.

Trong thí dụ sau, một copy của StringA được đưa cho Sub nên sau khi execute ProcessString, StringA không hề bị thay đổi.

```
Sub ProcessString (ByVal StringA, ConditionA, ConditionB)
```

Public Sub và Function

Khi ta dùng chữ Public (thay vì Private) phía trước một Sub hay Function, ta cho phép code nằm ở một Form hay Basic Module khác có thể gọi (hay dùng) Sub hay Function đó. Thí dụ trong Form2 ta có định nghĩa DisplayData là:

```
Public Sub DisplayData
   . . . .
End Sub
```

Trong Form1, ta gọi DisplayDta như sau:

```
Form2.DisplayData
```

Chương Năm - Các loại dữ kiện

Công việc chính của tất cả các chương trình VB6 chúng ta viết là chế biến các dữ kiện để trình bày. Thí dụ một thầy giáo dùng một chương trình để tính điểm trung bình của học sinh trong một môn thi. Thầy tuần tự cho điểm của từng học sinh vào và sau cùng bấm một nút bảo chương trình tính điểm trung bình cho cả lớp. Chương trình sẽ display điểm thi của từng học sinh bên cạnh tên của học sinh ấy, tổng số học sinh, tổng số điểm, điểm thấp nhất, điểm cao nhất và điểm trung bình:

Tên họ	Điểm
Lê Quang Vinh	15.50
Trần văn Thành	16.00
Nguyễn Thị Hương	17.50
Võ Tự Cường	14.00
Phạm Văn Khá	18.00
Cao Xuân Tiên	13.00
Tổng số học sinh:	6
Tổng số điểm:	94.00
Điểm thấp nhất:	13.00
Điểm cao nhất:	18.00
Điểm trung bình:	15.66

Ta có thể tạm chia quá trình xử lý của một chương trình ra làm ba giai đoạn:

1. **Tiếp nhận dữ kiện:** Đây là giai đoạn ta cho dữ kiện vào chương trình (**Input data**) hoặc bằng cách điền vào một form, hoặc đọc dữ kiện từ một cơ sỡ dữ kiện (**Database**) hoặc nhận dữ kiện qua đường dây viễn thông, .v.v..

2. **Chế biến dữ kiện:** Một khi đã có dữ kiện đầy đủ rồi ta sẽ sắp xếp, cộng, trừ, nhân, chia theo cách đã định trước để đi đến kết quả.

3. **Trình bày, báo cáo:** Kết quả cần phải được display trên màn ảnh cách gọn ghẽ, thứ tự hay được in ra, ta còn gọi là **Report**.

Như vậy trong mọi giai đoạn của chương trình ta đều làm việc với dữ kiện. Trong thí dụ nói trên ta làm việc với hai loại dữ kiện: **"dòng chữ" (text string)** cho tên học sinh và **"số" (number)** cho các điểm. Sở dĩ ta phải phân biệt các data types vì mỗi loại data có những chức năng riêng

của nó. Thí dụ ta không thể cộng hai text string lại với nhau như hai con số, nhưng ta có thể ghép hai text string lại với nhau, thí dụ như ghép chữ **house** với chữ **wife** thành ra chữ **housewife**. Chốc nữa ta sẽ bàn thêm về data types, nhưng bây giờ ta thử tìm hiểu data được chứa trong computer như thế nào.

Dữ kiện được chứa theo quy ước

Rốt cuộc lại, tất cả data đều được chứa dưới dạng các con số. Mỗi con số đại diện cho một thứ gì đó, tùy theo quy ước của người dùng. Chúng ta biết bộ trí nhớ (**memory**) của computer chứa những **byte** data, thí dụ như computer của bạn có 32MB, tức là khoảng hơn 32 triệu bytes. Thật ra một byte gồm có 8 **bits**, mỗi bit đại diện một trong hai trị số: **1 và 0**, hay **Yes và No** , dòng điện chạy qua **được hay không được** .v.v.. Bit là đơn vị trí nhớ nhỏ nhất của memory.

Một byte có thể chứa một con số từ 0 đến 255, tức là 2^8 -1 (2 lũy thừa 8 bớt 1) . Khi dùng bits ta đếm các số trong hệ thống nhị phân.

Thí dụ, khi bạn ấn nút **A** trên keyboard, keyboard sẽ gởi về computer con số **65 (01000001 trong nhị phân)** . Nếu bạn đang dùng một Notepad chẳng hạn, bạn sẽ thấy chữ **A** hiện ra. Bạn hỏi tại sao letter A được biểu diễn bằng số 65? Xin trả lời rằng đó là quy ước quốc tế. Quy ước được áp dụng cho tất cả các keys của bàn phím được gọi là **ASCII**. Theo quy ước nầy digit "1" được biểu diễn bằng con số 48 (00110001) và nút Enter bằng số 13 (00010011).

Chắc có lẽ bạn đã đoán ra rằng theo quy ước ASCII, mỗi pattern (dạng) của 8 bits (1 byte) sẽ biểu diễn một text character. Bây giờ ta thử tính xem các mẫu tự alphabet và digits sẽ chiếm bao nhiêu patterns trong số 256 patterns ta có thể biểu diễn bằng 1 byte. Từ A đến Z có 26 characters. Nhân đôi để tính cho lowercase (chữ thường) và uppercase (chữ hoa) thành ra 52. Cộng với 10 digits từ 0 đến 9 thành ra 62. Cộng thêm chừng ba mươi ngoài các symbols ta dùng chỉ đến chừng 100 patterns mà thôi. Tức là nói một cách khác nếu số patterns ta dùng dưới 128 thì chỉ cần 7 bits (chớ không đến 8 bits) cũng đủ rồi.

Thật ra từ nãy giờ ta chỉ nói đến các characters có thể display hay in ra

được (**printable characters**). Các con số ASCII từ 1 đến 31 không in ra được nhưng được dùng một cách đặc biệt, thí dụ như 7 là BELL (tiếng bíp), 12 là qua trang mới, 10 là xuống hàng, 13 là Enter/CarriageReturn, .v.v.. Chúng được gọi là các **Control Characters**.

Khi xem qua các Font chữ trong Windows, bạn sẽ thấy cho cùng một con số 65, không phải Font nào cũng display chữ A. Thí dụ như Font Symbol nó display đủ thứ dấu hiệu. Điểm nầy nhắc chúng ta lại rằng mối liên hệ giữa một con số bên trong (internal number) và một dấu hiệu được display chẳng qua là một quy ước mà thôi.

Giả sử chúng ta dùng những con số ASCII còn trống để biểu diễn các chữ Việt Nam có dấu và chịu khó ngồi vẽ thêm các Vietnamese characters cần thiết trong Font thì ta có thể display chữ Việt được. Đúng vậy, đó là cách các khoa học gia Việt Nam đã dùng để display tiếng Việt trong MSWindows, điển hình là **VPS, VISCII**.

Không phải memory của computer chỉ chứa data thường mà thôi. Nó còn chứa chính chương trình, gọi là **executable code** trong **machine language** (ngôn ngữ của máy). Ngày xưa, khi memory của computer còn ít, người ta có thể cho vào từng byte của code một chương trình. Họ lập trình bằng **Assembly language**. Mỗi hàng code trong Assembly language có thể được dịch thẳng ra code trong machine language. CPU của mỗi manufacturer có một assembly language khác nhau. Các công ty Computer nổi tiếng ngày xưa là IBM, Digital, CDC. Đến thời buổi Microcomputer ta có Motorola, Intel, nhưng tựu trung, nếu không biết trước code của machine language nào, ta không thể nhận ra gì cả khi nhìn vào memory dump (in ra snapshot của memory) của một computer.

Text String

Nếu ta ghép nhiều characters lại với nhau ta có một Text String. Trong VB6, Text String được viết thành một dãy chữ với dấu ngoặc kép ở hai đầu, thí dụ: **"Hello, world"**

Tưởng tượng ta ghép ba mẫu tự alphabet đầu tiên lại với nhau: **ABC**, trong memory Text String nầy được biểu diễn bằng con số

010000010100001001000011 (trong binary) hay 414243 (trong Hex, mỗi nhóm 4 bits tương đương với một Hex digit).

VB6 cho ta những Function rất tiện lợi để làm việc với Text String. Để ghép hai Text String lại với nhau ta dùng operator **&**. Thí dụ:

```
FirstWord = "Hello"
SecondWord = "World"
Greeting = FirstWord & SecondWord ' Greeting bây giờ là "HelloWorld"
```

nếu muốn có một blank space ở giữa hai chữ trên ta viết như sau:

```
Greeting = FirstWord & " " & SecondWord
```

Muốn biết một Text String đang chứa bao nhiêu characters ta dùng **Function**

Len. Thí dụ:

```
Greeting = "Hi John!"

iLen = Len(Greeting) ' iLen bây giờ bằng 8
```

Để trích ra một phần của Text String (tức là trích ra một **SubString**) ta dùng các **Functions Left, Right và Mid**.

```
Today = "24/05/2001"

' Lấy ra 2 characters từ bên trái của String Today

StrDay = Left(Today,2) ' StrDay bây giờ bằng "24"

' Lấy ra 4 characters từ bên phải của String Today

StrYear = Right(Today,4) ' StrYear bây giờ bằng "2001"

' Lấy ra 2 characters bắt đầu từ character thứ tư của String Today, character đầu tiên từ bên trái là thứ nhất

StrMonth = Mid(Today,4,2) ' StrMonth bây giờ bằng "05"

' Lấy ra phần còn lại bắt đầu từ character thứ tư của String Today

StrMonthYear = Mid(Today,4) ' StrMonthYear bây giờ bằng 05/2001"
```

Trong tất cả các trường hợp trên Text String Today không hề bị thay đổi, ta chỉ trích ra một SubString của nó mà thôi.

Nếu ta muốn thay đổi chính Text String Today ta có thể assign value mới cho nó hay dùng Function Mid ở bên trái dấu **Assign (=)**, thí dụ:

```
Today = "24/05/2001"

' Thay thế character thứ 3 của Today bằng "-"

Mid(Today,3,1) = "-"

' Thay thế 2 characters bắt đầu từ character thứ 4 của Today bằng "10"

Mid(Today,4,2) = "10"

' Thay thế character thứ 6 của Today bằng "-"

Mid(Today,6,1) = "-" ' Today bây giờ bằng "24-10-2001"
```

Ta cũng có thể đạt được kết quả như trên bằng cách lập trình như sau:

```
Today = "24/05/2001"
Today = Left(Today,2) & "-10-" & Right(Today,4)
```

Ngoài ra có hai Function rất thông dụng cho Text String là **Instr** và **Replace**. Instr cho ta vị trí (position) của một pattern trong một Text String. Thí dụ ta muốn biết có dấu * trong một Text String hay không:

```
myString = "The *rain in Spain mainly..."

Position = Instr(myString,"*") ' Position sẽ là 5


Nếu trong myString không có dấu "*" thì Position sẽ bằng 0
```

Bây giờ ta thử tách ra Key và Value trong thí dụ sau:

```
KeyValuePair = "BeatlesSong=Yesterday"

Pos = Instr(KeyValuePair, "=")

Key = Left(KeyValuePair, Pos-1)

Value = Mid(KeyValuePair, Pos+1)
```

Muốn thay đổi tất cả dấu "/" thành dấu "-" trong một Text String ta có thể dùng **Function Replace** như sau:

```
Today = "24/05/2001"
Today = Replace (Today, "/", "-")
```

Muốn biết trị số ASCII của một character ta dùng **Function Asc** và ngược lại để có một Text Character với một trị số ASCII nào đó ta dùng **Function Chr**.

```
ASCIINumberA = Asc("A") ' ASCIINumberA bây giờ bằng 65

LineFeedChar = Chr(10)
StrFive = Chr(Asc("0") + 5) ' ta có digit "5"
```

Text String trong VB6 dùng một byte cho mỗi ASCII character. Sau nầy khi ta lập trình trong VB7, một character có thể là Unicode character, trong trường hợp đó nó được biểu diễn bằng 2 bytes. VB6 không support Unicode nên không phải là môi trường thích hợp để lập trình cho Unicode tiếng Việt. Trong VB7 mỗi loại Text String có Encoding method riêng của nó để yểm trợ Unicode nếu cần.

Các loại số

Từ nãy giờ ta chỉ bàn về Text String và cách chứa của nó trong memory. Nên nhớ rằng **"123"** là một Text String và nó được biểu diễn trong memory bằng con số 001100010011001000110011 trong Binary hay 313233 trong Hex. Như vậy có cách nào biểu diễn con số 123 mà không dùng Text String không? Dĩ nhiên là được. Con số **123 là 7B trong Hex** hay 01111011 trong Binary, và ta có thể chứa con số nầy vừa tiện lợi để làm toán, vừa ít tốn memory hơn là chứa Text String "123". Nhớ là ta cần Text String để display hay in ra, còn khi làm toán cộng, trừ, nhân, chia ta lại cần cái dạng raw number hay internal number của nó.

Để convert một Text String ra Internal number ta có thể dùng các **Functions Val, CInt**(ra Integer) hay **CSng**(ra Single). Ngược lại, để convert từ internal number ra Text String ta có thể dùng **Function CStr**.

```
Dollars = "500"
ExchangeRatePerDollar = "7000"
tempValue= Val(Dollars) * Val(ExchangeRatePerDollar)
VNDong = CStr(tempValue)
MsgBox "Amount in VN Dong is " & VNDong
```

Thật ra VB6 support nhiều loại data để dùng chứa những con số. Trước hết ta có số nguyên (**Integer** và **Long**). Cùng là số nguyên nhưng Integer dùng 2 bytes trong memory để chứa một con số nguyên từ -32768 đến 32767. Để ý là 32768 = 2^15 (2 lũy thừa 15) , tức là trong memory các con số từ 32768 đến 65535 được dùng để biểu diễn các số âm. Một lần nữa, nhớ rằng một con số trong memory để biểu diễn một thứ gì chẳng qua chỉ là theo quy ước mà thôi.

Còn Long dùng 4 byte để để chứa một con số nguyên từ -2147483648 đến 2147483647. Nếu bạn dùng Integer mà bị Oveflow error khi làm toán nhân thì assign các con số vào một Long variable (sẽ cắt nghĩa variable sau nầy) **TRƯỚC KHI** làm toán nhân chớ đừng để kết quả một bài toán nhân quá lớn trước khi Assign nó vào một Long variable. Thí dụ:

```
' Thay gì viết
Dim Result as Long
Result = 30345 * 100 ' sẽ bị overflow error

' Hãy viết như sau:
Dim Result as Long
Result = 30345
Result = Result * 100 ' không bị overflow error
```

Để tính toán cho chính xác ta cần một loại data có thể chứa số sau decimal point. VB6 cho ta **Single** và **Double**. Single dùng 4 bytes, Double dùng 8 bytes. Thông thường, bạn sẽ hiếm khi cần nhắc đến Double.

Khi display một số Single hay Double bạn cần dùng **Function Format** để convert từ Single ra Text String một cách uyển chuyển. Thí dụ

```
Dollars = "500.0"
ExchangeRatePerDollar = "7000.0"
'Dùng Function CSng để convert String ra Single
tempValue= CSng(Dollars) * CSng(ExchangeRatePerDollar)

'Dùng Function Format để có các dấu phẩy ở ngàn và triệu và phải có 2 digits sau decimal point.

VNDong = Format (tempValue, "#,###,###.00")
MsgBox "Amount in VN Dong is " & VNDong
```

VB6 cho ta hai cách **chia**, đó là **/** dùng cho Single/Double và **** dùng cho Integer.
5 / 3 cho ta 1.6666666
5 \ 3 cho ta 1

Function Round được dùng để bỏ bớt các con số nằm phía sau decimal point. Thí dụ:

Round (12.3456789, 4)

chỉ giữ lại 4 con số sau decimal point và cho ta 12.3457

Numeric data type **Currency** chỉ chứa nhất định 4 số sau decimal point. Nó không có ích lợi đặc biệt gì.

Variable

Variable là những chỗ chứa data tạm thời trong memory để ta dùng trong quá trình biến chế data của chương trình. Khi ta **Declare (khai báo)** một variable loại data gì là ta dành ra một chỗ trong memory để chứa một miếng data loại ấy. Nhớ là tùy theo loại data ta sẽ cần nhiều hay ít memory, một Interger chỉ cần 2 bytes, còn một Single cần đến 4 bytes, trong khi một String thì cần nhiều memory hơn nữa. Thí dụ như:

```
Dim strFullName as String
Dim ICount as Integer
Dim sRate as Single
```

Khi bạn tìm cách cho hai data type khác nhau làm việc, thí dụ như làm toán chia một Text String bởi một con số thì có thể bị **Mixed mode error**. Tuy nhiên nếu Text String ấy gồm những digits thì có thể VB6 sẽ tự động convert Text String ra một con số trước khi dùng nó trong một bài toán. Ngược lại, dĩ nhiên ta không thể ghép một con số vào một Text String, nhưng VB6 có thể convert con số ra một Text String of digits trước khi ghép Text String ấy vào String kia.

Mặc dầu VB6 rất tế nhị trong việc đoán ra ý định của chúng ta trong khi coding nhưng ta phải thận trọng trong cách dùng Data type để tránh gặp phải những bất ngờ.

Vấn đề đặt tên cho variable rất quan trọng. Bạn nên đặt tên variable và

các Function, Sub như thế nào để khi đọc code ta thấy dễ hiểu như đọc một bài luận văn. Thường thường, để dễ nhận diện data type của một variable người ta gắn phía trước tên variable các **prefix** như **str** cho String, **I** cho Integer, **s** cho Single ..v.v.. Khi ráp nhiều chữ rời thành tên một variable, thường thường người ta làm cho letter đầu tiên của mỗi chữ thành ra **Hoa (Capital)**, thí dụ như **TotalSalesOfTheMonth**.

Có một Tip nho nhỏ là đừng ngại đặt tên variable quá dài. Khi đánh máy nữa chừng tên của một variable bạn có thể đánh **Ctrl-Space** để IDE đánh nốt phần còn lại của tên variable, nếu không có sự trùng hợp với một tên variable/Sub/Function nào khác.

Nếu công tác lập trình giống như nấu ăn, bạn có thể nghĩ đến variable như các cái rổ, thau ta cần có để việc chuẩn bị thức ăn được tiện lợi. Trước khi bắt tay vào công tác ta xin với chủ nhà cho mình bao nhiêu cái rổ, thau, thún .v.v..(đó là **Declare variables**). Ta để mỗi loại thức ăn vào một rổ hay thau khác nhau, chớ không để thịt chung với rau cải (cũng như không thể cộng Text String với con số). Khi ta bỏ thêm một trái cà vào rổ cà thì số trái cà trong rổ tăng lên 1. Một lát sau ta lấy ra vài trái cà để dùng thì số trái cà trong rổ bị giảm đi. Khi không cần dùng nữa ta bỏ hay cất mấy trái cà còn lại rồi dẹp cái rổ đi chỗ khác.

Trị giá của một variable thường hay thay đổi trong quá trình xử lý data. Đến một lúc nào đó variable không còn hiện hữu. Phạm vi hoạt động của một variable được gọi là **scope**. Nếu code nằm ngoài phạm vi của một variable thì không thể dùng đến variable ấy được. Dưới đây là listing của một chương trình VB6 ngắn:

```
Option Explicit
Dim iCount As Integer
Dim X As Integer
Dim Y As Integer

Private Sub CmdIncreX_Click()
   iCount = iCount + 1
   X = X + 1
   If X = 80 Then
      X = 0
      Y = Y + 1
   End If

End Sub
Private Sub CmdIncreY_Click()
   Dim Y
   iCount = iCount + 1
```

```
        Y = Y + 1
    End Sub
```

Trong listing trên Scope của iCount, X, Y là toàn bộ listing, tức là ở đâu cũng có thể nói đến, dùng, thấy các variables đó. Tuy nhiên trong Sub CmdIncreY-Click() có declare một variable Y. Scope của variable nầy là chỉ nội bộ, tức là bên trong Sub CmdIncreY-Click() mà thôi. Chẳng những thế, cái **local (địa phương)** variable Y nầy còn che cái **global** variable Y nữa, tức là bên trong Sub CmdIncreY-Click() ta chỉ thấy local variable Y mà không thấy global variable Y. Một khi execution trong Sub CmdIncreY-Click() đã kết thúc thì local variable Y cũng biến mất luôn. Nếu bạn muốn khi trở lại execute Sub CmdIncreY-Click() mà local variable Y vẫn còn y nguyên với giá trị gì nó có từ trước, bạn nên Declare rằng nó **Static**, như:

```
Static Y as Integer
```

Nói tóm lại, Local variable của Sub hay Function chỉ hoạt động và hiện hữu bên trong Sub/Function. Global variable của một Form hay Module thì áp dụng cho cả Form/Module trừ khi bị che lại bởi một local variable có cùng tên bên tron một Sub/Function. Ngoài ra khi ta Declare một Global variable là Public thì các Form/Module khác cũng thấy và dùng nó được luôn. Theo nguyên tắc của Software Engineering thì vì lý do an ninh ta chỉ cho phép người khác thấy cái gì cần thấy thôi. Do đó, ta không nên Declare các variable **Public** bừa bãi, nhỡ khi có một variable bị thay đổi value một cách bí mật mà ta không đoán được thủ phạm là ai. Nhớ rằng Declare Public cũng giống như để nhà không đóng cửa vậy.

Ngoài ra, câu **Option Explicit** ở đầu Listing được dùng để tuyên bố rằng tất cả mọi variables dùng trong Form/Module đều cần phải được Declare. Nhớ là VB6 không đòi hỏi ta phải Declare một variable trước khi dùng nó. Thường thường, tùy theo tình huống, VB6 có thể đoán ra được Data Type của variable khi ta dùng nó lần đầu tiên. Nếu code đòi hỏi value của một variable khi nó được dùng lần đầu thì VB6 tự động coi nó như một Empty String (String không có character nào cả, viết là **""**) nếu nó là Text String Data type hay con số 0 nếu nó là một con số. Điều nầy rất là tiện lợi. Tuy nhiên, nếu ta sơ ý đánh vần lộn một variable thì VB6 tự động initialise nó ra Empty String hay 0. Đây có thể không phải là điều ta muốn. Nếu có dùng Option Explicit thì việc nầy sẽ bị lộ tẩy ngay vì tất cả mọi variable đều phải được tuyên bố chính thức. Do đó, đã là VB6

programmer, bạn hãy xem việc dùng Option Explicit như là một điều răn của Chúa, hãy vâng giữ cách trung tín.

Ngày và Giờ

Có một loại data type được dùng để chứ cả ngày lẫn giờ, đó là Date. Ta có thể biết hiện thời là ngày nào, mấy giờ bằng cách gọi **Function Now**. Sau đó ta dùng các Function Day, Month và Year để lấy ra ngày, tháng và năm như sau:

```
Private Sub CmdCheckDate_Click()
   Dim dDate As Date
   If IsDate(txtDate.Text) Then  ' eg: txtDate = "26/12/01"
      dDate = CVDate(txtDate.Text)  ' convert a Text String to
internal Date using Function CVDate
      ' Day, Month and Year are automatically converted to String
      MsgBox "Day = " & Day(dDate) & ", Month = " & Month(dDate) & ",
Year = " & Year(dDate)
   Else
      MsgBox "Invalid date. Please try again."
   End If
End Sub
```

Trong Listing trên ta dùng Function IsDate để kiểm xem txtDate.text có hợp lệ không. Lưu ý là IsDate không phân biệt ngày theo Mỹ (dạng mm/dd/yy) hay theo Âu Châu (dạng dd/mm/yy), do đó dùng IsDate trong công việc kiểm soát nầy không an toàn lắm.
Để display ngày giờ theo đúng cách mình muốn bạn có thể dùng **Function Format** như sau:

```
MsgBox "NOW IS " & Format (Now, "ddd dd-mmm-yyyy hh:nn:ss")

' will display
    NOW IS Fri 08-Jun-2001 22:10:53
```

Bạn có thể dùng **mm** để display tháng bằng một con số. Sở dĩ ta dùng **nn** thay vì **mm** cho phút là vì **mm** đã được dùng cho tháng.

Ta có thể thêm bớt các đơn vị của ngày, tháng, v.v. bằng cách dùng **Function DateAdd**. Thí dụ:

```
Private Sub CmdNextMonth_Click()
   txtDate.Text = Format(Now, "dd/mm/yy")   ' 08/06/01
   txtNextMonth.Text = DateAdd("m", 1, CVDate(txtDate.Text))
   ' txtNextMonth.text will show 8/07/2001
End Sub
```

Dưới đây là cách tính ra ngày cuối của tháng nầy:

```
Private Sub CmdLastDayOfMonth_Click()
   Dim dNextMonthDate, dFirstDayNextMonth
   dNextMonthDate = DateAdd("m", 1, Now)   ' Add one month to get same
day next month
   ' Get first day of next month
   dFirstDayNextMonth = 1 & "/" & Month(dNextMonthDate) & "/" &
Year(dNextMonthDate)
   ' Subtract one day to get Last day of this month
   txtLastDayOfMonth.Text = DateAdd("d", -1,
CVDate(dFirstDayNextMonth))
End Sub
```

Ta có thể tính khoảng cách giữa hai ngày theo đơn vị ngày, tháng v.v.. bằng cách dùng **Function DateDiff**. Kết quả có thể là âm, dương hay 0 tùy theo ngày nào trể hơn. Thí dụ khoảng cách giữa hai ngày theo đơn vị tháng:

```
DateDiff("m", Now, CVDate(dNextMonthDate))
```

Trong chương tới ta sẽ học về Data types Boolean, Variant và Data Array.

Chương Sáu - Dùng dữ kiện

Trong chương 5 ta học qua các điểm căn bản về việc dùng variables. Vì công việc chính của một chương trình là xử lý data chứa trong variables, cho nên nếu VB6 cho ta càng nhiều phương tiện để làm việc với variables thì càng tiện lợi. Trong chương nầy ta sẽ học:

- Boolean variable, tại sao nó hữu dụng
- Variant variable, cách làm việc với nó.
- Cách biến đổi (convert) từ loại data type nầy qua loại data khác
- Arrays của variables và Arrays của controls
- Cách tạo một data type theo ý mình

Boolean Variables

Boolean là loại data chỉ có thể lấy một trong hai values: **True** hay **False**. Khi học về Statement **IF...THEN** trong chương 4, ta đã nói sơ qua về Boolean data type. Cái phần ở giữa hai chữ IF và THEN được gọi là **Logical Expression** và kết quả của một Logical Expression là một Boolean value. Nếu điều kiện được thỏa mãn thì value là True, nếu không thì là False.

Bạn hỏi nếu một variable chỉ có thể có hai values, tại sao ta không thể dùng Integer và giới hạn cách dùng trong vòng hai values 1 và 0 thôi, cần gì phải đặt ra Boolean data type. Làm như vậy cũng được, nhưng cái khác biệt chính là khi ta operate trên 2 variables ta phải biết rõ rằng *để làm việc với Integer* ta dùng **+, -, *,** \ trong khi *với Boolean* ta dùng **OR, AND, NOT, XOR**. Thử xem thí dụ dưới đây:

```
' Use Integer with values 1 or 0
Dim IAnumber As Integer
Dim IBnumber As Integer
Dim IAge As Integer
Dim sPersonalWorth As Single
If (IAge >= 18) Then
   IAnumber = 1
Else
   IAnumber = 0
End If
If (sPersonalWorth > 1000000) Then
   IBnumber = 1
Else
```

```
   IBnumber = 0
End If
If (IAnumber = 1) And (IBnumber = 1) Then
   StandForTheElection
End If
'================================
' Use Boolean
Dim bAdult As Boolean
Dim bRich As Boolean
Dim IAge As Integer
Dim sPersonalWorth As Single
bAdult = (IAge >= 18)
bRich = (sPersonalWorth > 1000000)
If bAdult And bRich Then
   StandForTheElection
End If
```

Trong thí dụ trên, ta lập trình để nếu một người thỏa mãn hai điều kiện: vừa trưởng thành (18 tuổi trở lên) , vừa giàu có (có trên 1 triệu bạc) thì có thể ra ứng cử

Nếu ta dùng Integer, thứ nhất chương trình đọc khó hiểu, thứ hai cái Logical Expression của IF statement vẫn phải làm việc với operator AND.

Trong khi đó nếu dùng Boolean thì chương trình có vẻ tự nhiên và dễ đọc như tiếng Anh thông thường.

Variant Variables

Variant variable có thể chứa Text String, Number, Date, thậm chí cả một Array (một loạt nhiều variables cùng data type). Nhìn thoáng qua nó rất tiện dụng, nhưng khi một Variant variable được dùng nhiều chỗ, trong nhiều tình huống khác nhau, bạn phải thận trọng. Lý do là vì variant variable có thể chứa những loại data types khác nhau, nên khi bạn operate hai variable có data type khác nhau, Visual Basic 6 cố gắng biến đổi một trong hai variable thành data type của variable kia để làm việc, kết quả là thỉnh thoảng bạn sẽ bị kẹt.

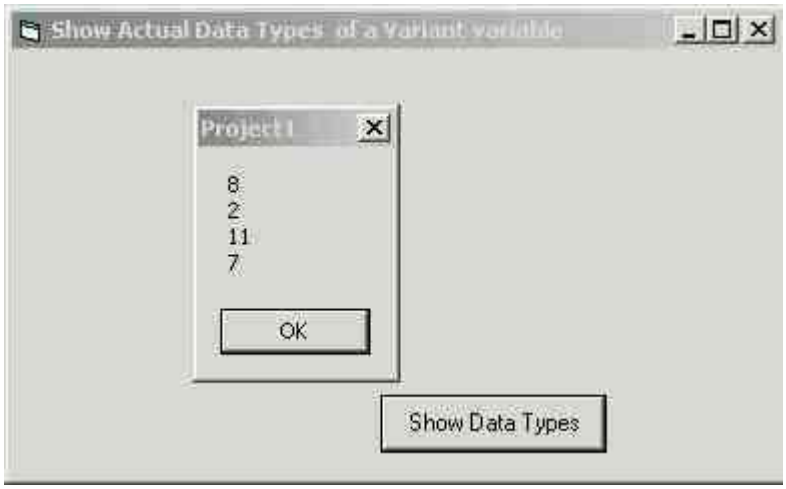
Các tay Software Engineers thuần túy rất ghét lập trình với data không được Declare rõ ràng. Họ không muốn bị hố vì vô tình. Thà rằng để Language Compiler bắt gặp trước những trường hợp bạn vô tình operate trên hai variables có data type khác nhau. Có khi ta bực mình vì Compiler khó tánh, nhưng sẽ tránh bị những surprise (ngạc nhiên) tốn kém sau nầy. Các ngôn ngữ lập trình gắt gao ấy được gọi là **strongly typed languages**,

chẳng hạn như Pascal, C++, Java .v.v.. Sau nầy nếu ta dùng .NET thì các ngôn ngữ C#, VB.NET (VB7) đều là strongly typed. Trong VB7, Microsoft cho lưu đài biệt tích Variant variables của VB6.

Công việc Declare một Variant variable cũng giống như Declare một data type khác. Chỉ có điều ta có thể biết data type thật sự đang được chứa bên trong một Varaint variable bằng cách dùng **Function VarType** như dưới đây:

```
Private Sub cmdShowDataTypes_Click()
   Dim sMess As String
   Dim vVariant As Variant
   vVariant = "Nguoi Tinh khong chan dung"  ' Assign a String to
vVariant
   sMess = VarType(vVariant) & vbCrLf  ' use vbCrLF to display the
next string on a new line
   vVariant = 25  ' Assign an Integer to vVariant
   sMess = sMess & VarType(vVariant) & vbCrLf
   vVariant = True  ' Assign an Boolean value to vVariant
   sMess = sMess & VarType(vVariant) & vbCrLf
   ' Assign an Date to vVariant
   vVariant = #1/1/2001#  ' enclose a Date string with #, instead of
" as for normal Text String
   sMess = sMess & VarType(vVariant)
   MsgBox sMess
End Sub
```

Khi ta click button ShowDataTypes chương trình sẽ display giá trị của các Data Types trong mỗi trường hợp:



Sau đây là bảng liệt kê những VarTypes thông dụng:

Giá trị VarType	Chú thích
0-vbEmpty	Không có gì trong variant
1-vbNull	Không có valid (hợp lệ) data trong variant
2-vbInteger	Variant chứa Integer

4-vbSingle	Variant chứa Single
7-vbDate	Variant chứa Date/Time
8-vbString	Variant chứa String
9-vbObject	Variant chứa một Object
11-vbBoolean	Variant chứa Boolean

Để làm việc với đủ loại VarTypes bạn có thể dùng Select Case như sau:

```
Private Sub Process_Click()
   Select Case VarType(vVariant)
   Case vbString
      ' ...
   Case vbBoolean
      ' ...
   Case vbInteger
      ' ...
   Case vbDate
      ' ...
   End Select
End Sub
```

Constants (Hằng số)

Variables rất tiện dụng để chúng ta dùng chứa các data có thể biến đổi value trong suốt quá trình xử lý của chương trình. Nhưng đôi khi chúng ta muốn có một loại variable mà value không bao giờ thay đổi, VB6 cho ta **Constant** để dùng vào việc nầy. Thí dụ như thay gì dùng trực tiếp môt con số hay một Text String ở nhiều chỗ trong chương trình, ta đặt tên Constant và cho nó một value tại một chỗ nhất định. Thí dụ ta viết chương trình cho 5 chiếc xe chạy đua. Để khởi hành các chiếc xe ta dùng một FOR...LOOP đơn giản như:

```
For ICar = 1 To 5
   Call StartCar (ICar)
Next
```

Tương tự như vậy ở nhiều nơi khác trong chương trình, mỗi lần nói đến con số các xe ta dùng số 5. Nếu sau nầy muốn thay đổi con số các xe thành ra 10, ta phải tìm và thay đổi tất cả các con số 5 nầy thành ra 10. Nếu không thận trọng ta có thể thay đổi một con số 5 dùng cho chuyện gì khác, chớ không phải cho con số các xe, thành ra 10 - như vậy ta vô tình tạo ra một bug. Để tránh vấn đề nầy ta có thể dùng Constant như sau:

```
Const NUMBER_OF_CARS = 10
For ICar = 1 To NUMBER_OF_CARS
   Call StartCar (ICar)
Next
```

Sau nầy muốn thay đổi con số các xe, ta chỉ cần edit value của Constant. Trong khắp chương trình, nơi nào nhắc đến con số các xe ta dùng chữ NUMBER_OF_CARS, vừa dễ hiểu, vừa tránh lầm lẫn.

Biến đổi (convert) từ loại data type nầy qua loại data khác

Nhiều lúc ta cần phải convert data type của một variable từ loại nầy qua loại khác, VB6 cho ta một số các Functions dưới đây. Xin lưu rằng khi call các Functions nầy, nếu bạn đưa một data value bất hợp lệ thì có thể bị error.

Conversion Function	Chú thích
CBool ()	Đổi parameter ra True hay False. Nếu Integer value khác 0 thì được đổi thành True
CByte ()	Đổi parameter ra một con số từ 0 đến 255 nếu có thể được, nếu không được thì là 0.
CDate ()	Đổi parameter ra Date
CDbl ()	Đổi parameter ra Double precision floating point number
CInt ()	Đổi parameter ra Integer
CSng ()	Đổi parameter ra Single precision floating point number
CStr ()	Đổi parameter ra String

Ngoài các Function nói trên bạn cũng có thể dùng **Function Val** để convert một String ra Number. Lưu ý là khi Function Val process một String nếu nó gặp một character nào không phải là digit hay decimal point thì nó không process tiếp nữa. Do đó nếu Input String là "$25.50" thì Val returns con số 0 vì **$** không phải là một digit. Nếu Input String là "62.4B" thì Val returns 62.4.

CDbl là Function dùng để convert một String ra số an toàn nhất. Input String có thể chứa các dấu **,** và **.** (thí dụ: 1,234,567.89) tùy theo nơi bạn ở trên thế giới (thí dụ như Âu Châu hay Mỹ). CSng cũng làm việc giống như CDbl nhưng nếu con số lớn hơn 1 triệu nó có thể bị bug.

Cái bug bực mình nhất của CSng là nếu Input String không có gì cả (tức là InputString="") thì Function CSng cho bạn Type Mismatch Error. Do đó để khắc phục cái khuyết điểm nầy bạn có thể viết cho mình một Function tạm đặt tên là CSingle để dùng thế cho CSng như sau:

```
Function CSingle(strNumber) As Single
   If Trim(strNumber) = "" Then
      CSingle = 0#
   Else
```

```
      CSingle = CSng(strNumber)
   End If
End Function
```

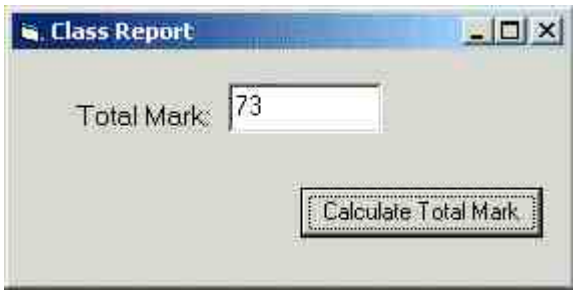
Arrays

Khi bạn có nhiều variables tương tợ nhau, thí dụ như điểm thi của 10 học sinh, nếu phải đặt tên khác nhau cho từng variable (thí dụ: HoaMark, TaiMark, SonMark, TamMark, NgaMark, HuongMark .v.v..) thì thật là cực nhọc và bất tiện. Bạn có thể dùng **Array** để có một tên chung cho cả nhóm, rồi nói đến điểm của từng người một bằng cách dùng một con số gọi là **ArrayIndex**. Bạn sẽ Declare như sau:

```
Dim myStudentMarks(10) as Integer
```

Kế đó bạn nói đến điểm của mỗi học sinh bằng cách viết **myStudentMarks(i)**, mà i là ArrayIndex. Giả dụ ta muốn tính tổng số điểm:

```
Sub CmdCalculateTotal_Click()
   Dim myStudentMarks(10) As Integer  ' Declare array, assuming
students' marks are Integers
   Dim TotalMark As Integer  ' Use this variable to accumulate the
marks
   Dim i As Integer  ' Use i as ArrayIndex
   myStudentMarks(1) = 6  ' First student's mark
   myStudentMarks(2) = 7  ' Second student's mark
   myStudentMarks(3) = 5
   myStudentMarks(4) = 9
   myStudentMarks(5) = 6
   myStudentMarks(6) = 8
   myStudentMarks(7) = 9
   myStudentMarks(8) = 10
   myStudentMarks(9) = 6
   myStudentMarks(10) = 7
   TotalMark = 0  ' This statement is not required as VB6 initialises
TotalMark to 0
   ' Go through all students and add each student's mark to the Total
   For i = 1 To 10
      TotalMark = TotalMark + myStudentMarks(i)
   Next
   txtTotal.Text = CStr(TotalMark)  ' Convert to String for display
by assigning to Textbox txtTotal
End Sub
```



Khi ta Declare **Dim myStudentMarks(10) as Integer** thật ra ra ta có một Array với 11 **Array Elements** chớ không phải 10, vì Array bắt đầu với ArrayIndex value=0. Có điều trong thí dụ trên ta cố ý không nhắc đến ArrayElement 0. Nếu thật sự muốn có chính xác 10 Elements thôi, ta có thể Declare như sau:

```
Dim myStudentMarks (1 To 10 ) As Integer
```

Loại Array ta vừa dùng qua là Single Dimention. Nếu trong thí dụ trên ta muốn tính điểm của học sinh trong 3 lớp học, ta có thể Declare Double Dimention Array như sau:

```
' Four classes, each has up to 6 students
Dim myStudentMarks(3, 5) As Integer  ' Note that each dimension
starts at 0
' or
' Three classes, each has up to 5 students
Dim myStudentMarks(1 To 3, 1 To 5) As Integer
```

Nhân tiện nói chuyện về Array cho variables, ta cũng có thể dùng Array cho controls cùng một loại trong một Form. Nếu ta có nhiều Label controls (hay Textbox controls) với những chức năng giống nhau trong chương trình, ta có thể dùng cùng một tên cho các controls, nhưng khác **Property Index** value của chúng.

Bạn có thể create một Array of Labels bằng cách Copy cái Label rồi Paste nó lên Form. VB6 sẽ hỏi nếu bạn muốn có một Array of controls. Nếu bạn trả lời Yes, VB6 sẽ tự động cho Label thứ nhất Index value=0 và Label mới vừa được Pasted Index value=1. Sau đó nếu bạn tiếp tục Paste cái Label đã có Index rồi thì VB6 không hỏi nữa và vui vẻ tăng Index value lên cho các Labels sau. Do đó nếu bạn gọi Label thứ nhất là lblClass rồi Copy và Paste nó 2 lần bạn sẽ có một Array of 3 Labels tên lblClass và các Index values 0, 1, 2.

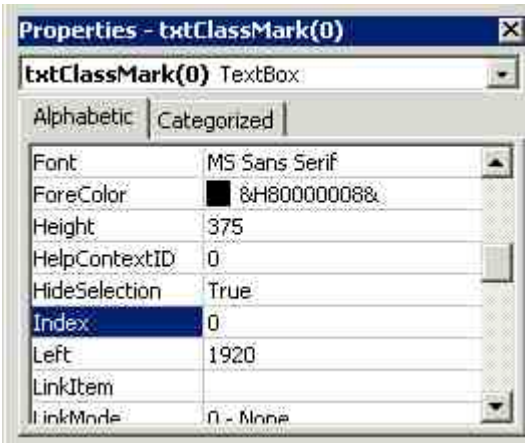
Trong thí dụ sau đây, ta create một Array of Labels tên **lblClass** và một Array of Textboxes tên **txtClassMark**. Trong Sub Form_Load ta generate Captions của các Labels.

```
Private Sub Form_Load()
   Dim i As Integer
   For i = 0 To 2
      ' Label Index starts at 0, but Class number starts at 1
      lblClass(i) = "Mark of Class " & CStr(i + 1)
   Next
End Sub

Sub CmdCalculateTotal_Click()
   ' Three classes, each has up to 5 students
   Dim myStudentMarks(1 To 3, 1 To 5) As Integer
   Dim TotalMark As Integer
   Dim ClassMark As Integer
   Dim i As Integer  ' Use as ArrayIndex for Class
   Dim j As Integer  ' Use as ArrayIndex for StudentMark
   ' Students' marks of first class
   myStudentMarks(1, 1) = 6
   myStudentMarks(1, 2) = 7
   myStudentMarks(1, 3) = 5
   myStudentMarks(1, 4) = 9
   myStudentMarks(1, 5) = 6
   ' Students' marks of second class
   myStudentMarks(2, 1) = 8
   myStudentMarks(2, 2) = 8
   myStudentMarks(2, 3) = 6
   ' Students' marks third class
   myStudentMarks(3, 1) = 5
   myStudentMarks(3, 2) = 7
   myStudentMarks(3, 3) = 8
   myStudentMarks(3, 4) = 6
   For i = 1 To 3
      ClassMark = 0  ' Intialise ClassMark of class i to 0
      ' Now go through each Student in Class i
      For j = 1 To 5
         ClassMark = ClassMark + myStudentMarks(i, j)  ' Accumulate
ClassMark of class i
         TotalMark = TotalMark + myStudentMarks(i, j)
      Next
      ' Display ClassMark of class i. Note that txtClassMark Index
starts at 0, NOT 1
      txtClassMark(i - 1).Text = CStr(ClassMark)
   Next
   txtTotal.Text = CStr(TotalMark)  ' Display TotalMark
End Sub
```

Ghi chú: Nếu bạn có một Array of Textboxes gồm chỉ có 2 Textboxes, rồi sau đó bạn Delete một Textbox và muốn dùng Textbox còn lại làm cái Textbox duy nhất, bạn vẫn phải refer (nhắc đến nó) bằng cách dùng Index (thí dụ: **txtClassMark(0)**), dù rằng bây giờ nó là Textbox duy nhất mang tên ấy. Nếu bạn muốn dẹp cái vụ Index thì bạn phải vào Properties Window để Delete cái Index value của ArrayTextbox. Nếu bạn không lưu

ý điểm nầy thì có khi bạn sẽ bứt tóc không hiểu tại sao mình dùng đúng tên mà VB6 vẫn nhất định rằng cái Textbox bạn nói đến không hiện hữu, vì VB6 cần Index value của Textbox.



Thỉnh thoảng, khi Declare Array bạn không biết rõ mình sẽ cần bao nhiêu Elements cho mỗi dimension. Trong trường hợp ấy bạn có thể dùng **Dynamic Arrays** và Declare một Array như sau:

```
Private myStudentMarks() As Integer
```

Vì bạn không để một con số ở giữa hai dấu ngoặc đơn nên VB6 biết là bạn muốn dùng Dynamic Array và dimension của nó có thể sẽ thay đổi trong tương lai. Khi nào muốn thay đổi dimension của Dynamic Array bạn dùng **ReDim** keyword:

```
ReDim myStudentMarks(10)

ReDim Preserve myStudentMarks(10)
```

Cả hai statements trên đều đổi dimension của Array myStudentMarks thành 11 (từ Element 0 đến Element 10), nhưng trong statement thứ nhì chữ **Preserve** giữ nguyên values của các Elements của Array. Khi làm việc với Array thỉnh thoảng bạn cần biết các Elements thấp nhất và cao nhất bằng cách dùng **LBound** và **UBound**.

```
Private MyArray(10 to 20) As String
LowestNum = LBound(MyArray)   ' LBound returns 10
HighestNum = UBound(MyArray)   ' UBound returns 20

Private YourArray( 2 to 5, 10 to 15) As Integer
LowestNumOfFirstDimension = LBound(YourArray,1)   ' LBound returns 2
HighestNumOfSecondDimension = UBound(YourArray,2)   ' UBound returns
15
```

Ngoài ra nếu dùng Dynamic Array, bạn có thể assign một Array nầy cho một Array khác, thay vì phải dùng FOR ...LOOP để copy từng Array Element.

```
MyArray = HisArray
```

Data Type của bạn

Bạn có thể gom các mảnh Data của cùng một vật nào đó thành một nhóm và đặt tên cho loại Data Type ấy như sau:

```
Type EmployeeRec  ' EmployeeRec as name of this new Data Type
   Firstname As String
   Surname As String
   Salary As Single
   DateOfBirth As Date
End Type

' Now declare a variable of the new data type
Dim MyEmployee As EmployeeRec
MyEmployee.Firstname = "David"
MyEmployee.Surname = "Smith"
MyEmployee.Salary = 25000
MyEmployee.DateOfBirth = #14/6/1963#
```

Trong Software Engineering, người ta gọi loại Data Type nầy là **Structured Data Type** để phân biệt nó với các loại Simple Data Type như Integer, Boolean, Single .v.v.. Bạn để ý cách nói đến một mảnh data, MyEmployee.Firstname, giống như là Property Firstname của một control tên MyEmployee.

Có một cách viết khác để tránh typing nhiều lần chữ MyEmployee bằng cách dùng keyword **With** như sau:

```
With MyEmployee
   .Firstname = "David"
   .Surname = "Smith"
   .Salary = 25000
   .DateOfBirth = #14/6/1963#
End With
```

Mặc dầu định nghĩa và dùng Structured Data Type cách nầy rất tiện lợi, nhưng sau nầy ta có thể dùng **Class** để đạt được cùng một mục tiêu mà còn hữu hiệu hơn nữa. Trong Class chẳng những ta định nghĩa những mảnh data mà còn đề ra cách xử lý chúng nữa.

Chương Bảy - Dùng List Controls

Có hai loại List controls dùng trong VB6. Đó là Listbox và Combobox. Cả hai đều display một số hàng để ta có thể lựa chọn. Listbox chiếm một khung chữ nhật, nếu chiều ngang nhỏ thì có khi không display đầy đủ một hàng, nếu chiều dài không đủ để display tất cả mọi hàng thì Listbox tự động cho ta một vertical scroll bar để cho biết còn có nhiều hàng bị che và ta có thể xem các hàng ấy bằng cách dùng vertical scroll bar.

Combobox thường thường chỉ display một hàng, nhưng ta có thể chọn display bất cứ hàng nào khác. Combobox giống như một tập hợp của một Textbox nằm phía trên và một Listbox nằm phía dưới.

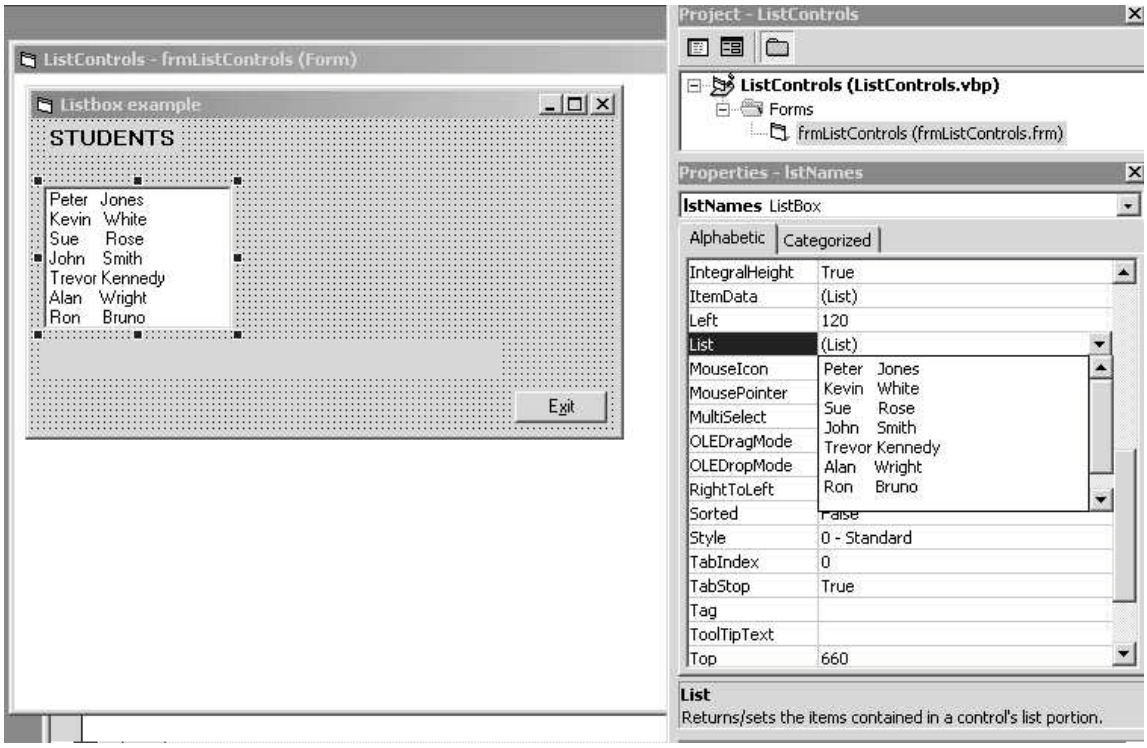
Listbox có rất nhiều công dụng vì nó rất uyển chuyển. Trong chương nầy ta sẽ học qua các áp dụng sau của Listbox:

- Display nhiều sự lựa chọn để User selects bằng cách click hay drag-drop
- Những cách dùng Property Sorted
- Cách dùng Multiselect
- Dùng để display Events
- Dùng để Search hay process text
- Cách dùng Itemdata song song với các Items của List
- Dùng làm Queue

Listbox

Display nhiều sự lựa chọn

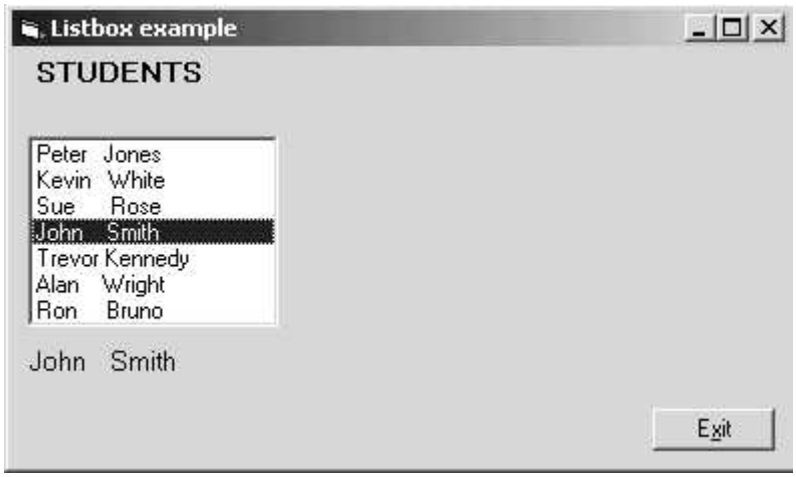
Ta hãy bắt đầu viết một chương trình gồm có một Listbox tên **lstNames** nằm trong một Form. Trong lstNames ta đánh vào tên của bảy người, mỗi lần xuống hàng nhớ đánh **Ctrl-Enter**, thay vì chỉ **Enter**, nếu không VB6 tưởng ta đã đánh xong nên close property List. Các tên nầy là những hàng sẽ hiện ra trong Listbox khi ta bắt đầu chạy program.



Ngoài lstNames ta cho thêm một Label với Caption STUDENTS để trang hoàng, và một Label khác tên **lblName**. Mỗi khi User click lên hàng tên nào ta muốn display hàng tên ấy trong lblName. Sau cùng ta cho vào một CommandButton tên **CmdExit** để cho User phương tiện Stop cái program. Ta sẽ có chương trình như sau:

```
Private Sub lstNames_Click()
   ' Assign the selected line of Listbox lstNames to Caption of Label
lblName
   lblName.Caption = lstNames.List(lstNames.ListIndex)   ' or =
lstNames.text
End Sub
Private Sub CmdExit_Click()
   End
End Sub
```

Giả sử ta click vào tên John Smith trên Listbox, ta sẽ thấy tên ấy cũng được display trong Label lblName.



Trong thí dụ nầy, Listbox lstNames có 7 hàng (**Items**). Con số Items nầy là Property **ListCount** của Listbox. Các Items của Listbox được đếm từ **0 đến ListCount-1**. Trong trường hợp nầy là từ 0 đến 6.

Khi User click lên một hàng, Listbox sẽ generate **Event lstNames_Click**. Lúc bấy giờ ta có thể biết được User vừa mới Click hàng nào bằng cách hỏi Property **ListIndex** của lstNames, nó sẽ có value từ 0 đến ListCount-1. Lúc program mới chạy, chưa ai Click lên Item nào của Listbox thì ListIndex = -1.

Những Items trong Listbox được xem như một Array của String. Array nầy được gọi là **List**. Do đó, ta nói đến Item thứ nhất của Listbox lstNames bằng cách viết **lstNames.List(0)** , và tương tợ như vậy, Item cuối cùng là **lstNames.List(lstNames.ListCount-1)**.

Ta có thể nói đến item vừa được Clicked bằng hai cách: hoặc là lstNames.List(lstNames.ListIndex), hoặc là lstNames.text.

Save content của Listbox

Bây giờ để lưu trử content của lstNames, ta thêm một CommandButton tên **CmdSave**. Ta sẽ viết code để khi User click nút CmdSave program sẽ mở một Output text file và viết mọi items của lstNames vào đó:

```
Private Sub CmdSave_Click()
   Dim i, FileName, FileNumber
   ' Obtain Folder where this program's EXE file resides
```

```
   FileName = App.Path
   ' Make sure FileName ends with a backslash
   If Right(FileName, 1) <> "\" Then FileName = FileName & "\"
   FileName = FileName & "MyList.txt"  ' name output text file
MyList.txt
   ' Obtain an available filenumber from the operating system
   FileNumber = FreeFile
   ' Open the FileName as an output file , using FileNumber as
FileHandle
   Open FileName For Output As FileNumber
   ' Now iterate through each item of lstNames
   For i = 0 To lstNames.ListCount - 1
      ' Write the List item to file. Make sure you use symbol # in
front of FileNumber
      Print #FileNumber, lstNames.List(i)
   Next
   Close FileNumber  ' Close the output file
End Sub
```

App là một Object đặc biệt đại diện cho chính cái program đang chạy. Ở đây ta dùng Property Path để biết lúc program đang chạy thì execute module EXE của nó nằm ở đâu. Lý do là thường thường ta để các files liên hệ cần thiết cho program lẩn quẩn hoặc ngay trong folder của program hay trong một subfolder, chẳng hạn như data, logs, .v.v.. App còn có một số Properties khác cũng rất hữu dụng như PrevInstance, Title, Revision ..v.v.

Nếu mới started một program mà thấy App.PrevInstance = True thì lúc bấy giờ cũng có một copy khác của program đang chạy. Nếu cần ta End program nầy để tránh chạy 2 copies của program cùng một lúc. App.Title và App.Revision cho ta tin tức về Title và Revision của program đang chạy.

Để viết ra một Text file ta cần phải Open nó trong mode Output và tuyên bố từ rày trở đi sẽ dùng một con số (FileNumber) để đại diện cái File thay vì dùng chính FileName. Để tránh dùng một FileNumber đã hiện hữu, tốt nhất ta hỏi xin Operating System cung cấp cho mình một con số chưa ai dùng bằng cách gọi **Function FreeFile**. Con số FileNumber nầy còn được gọi là **FileHandle** (Handle là tay cầm). Sau khi ta Close FileNumber con số nầy trở nên FREE và Operating System sẽ có thể dùng nó lại. Do đó bạn phải tránh gọi FreeFile liên tiếp hai lần, vì OS sẽ cho bạn cùng một con số. Tức là, sau khi gọi FreeFile phải dùng nó ngay bằng cách Open một File rồi mới gọi FreeFile lần kế để có một con số khác. Để ý cách dùng chữ **Input, Output** cho files là relative (tương đối) với vị trí của program (nó nằm trong memory của computer). Do đó từ trong

memory viết ra hard disk thì nói là Output. Ngược lại đọc từ một Text file nằm trên hard disk vào memory cho program ta thì gọi là Input.

Load một Text file vào Listbox

Trong bài nầy, thay vì đánh các Items của Listbox vào Property List của lstNames ta có thể populate (làm đầy) lstNames bằng cách đọc các Items từ một Text file. Ta thử thêm một CommandButton tên **CmdLoad**. Ta sẽ viết code để khi User click nút CmdLoad program sẽ mở một Input text file và đọc từng hàng để bỏ vào lstNames:

```
Private Sub CmdLoad_Click()
   Dim i, FileName, FileNumber, anItem
   ' Obtain Folder where this program's EXE file resides
   FileName = App.Path
   ' Make sure FileName ends with a backslash
   If Right(FileName, 1) <> "\" Then FileName = FileName & "\"
   FileName = FileName & "MyList.txt"
   ' Obtain an available filenumber from the operating system
   FileNumber = FreeFile
   ' Open the FileName as an input file , using FileNumber as
FileHandle
   Open FileName For Input As FileNumber
   lstNames.Clear  ' Clear the Listbox first
   ' Now read each line until reaching End-Of-File, i.e. no more data
   Do While NOT EOF(FileNumber)
      Line Input #FileNumber, anItem  ' Read a line from the Text
file into variable anItem
      lstNames.AddItem anItem  ' Add this item to the bottom of
lstNames
   Loop
   Close FileNumber  ' Close the input file
End Sub
```

Để **đọc từ một Text file** ta cần phải Open nó trong **mode** Input. Trước khi populate lstNames ta cần phải delete tất cả mọi items có sẵn bên trong. Để thực hiện việc đó ta dùng method Clear của Listbox. Sau đó ta dùng method AddItem để cho thêm từng hàng vào trong Listbox. By default, nếu ta không nói nhét vào ở chỗ hàng nào thì AddItem nhét Item mới vào dưới chót của Listbox.

Nếu muốn nhét hàng mới vào ngay trước item thứ 5 (ListIndex = 4), ta viết:

```
lstNames.AddItem newItemString, 4  ' newItemString contains "Ross
Close", for example
' To insert a new Item at the beginning of the Listbox, write:
lstNames.AddItem newItemString, 0
```

Nhớ là mỗi lần bạn Add một Item vào Listbox thì ListCount của Listbox increment by 1.

Muốn delete một item từ Listbox ta dùng method RemoveItem, thí dụ như muốn delete item thứ ba (ListIndex=2) của lstNames, ta viết:

```
lstNames.RemoveItem 2
```

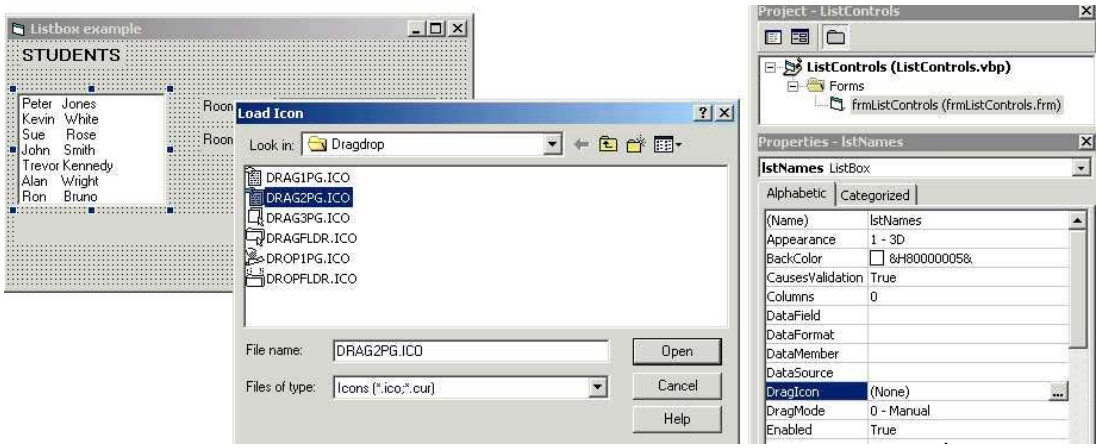
Mỗi lần bạn RemoveItem từ Listbox the ListCount của Listbox decrement by 1. Do đó nếu bạn dùng cái Test dựa vào ListCount của một ListBox để nhảy ra khỏi một Loop thì phải coi chừng tránh làm cho value ListCount thay đổi trong Loop vì AddItem hay RemoveItem. Ta đọc từng hàng của một Text file bằng cách dùng **Line Input #FileNumber**. Khi đọc đến cuối File, system dẽ cho ta value EOF(FileNumber) = True. Ta dùng value ấy để cho program nhảy ra khỏi While.. Loop.

Câu **Do While NOT EOF(FileNumber)** có nghĩa **Trong khi chưa đến End-Of-File của Text File đại diện bởi FileNumber** thì đọc từ hàng và bỏ vào Listbox. Giống như "Trong khi chưa trả hết nợ nhà vợ thì phải tiếp tục ở rể".

Drag-Drop

Ta đã học qua Click Event của Listbox. Bây giờ để dùng Drag-Drop cho Listbox bạn hãy đặt 2 Labels mới lên Form. Cái thứ nhất tên gì cũng được nhưng có Caption là **Room A**. Hãy gọi Label thứ hai là **lblRoom** và cho Property **BorderStyle** của nó bằng Fixed Single. Kế đến select cả hai Labels (Click a Label then hold down key Ctrl while clicking the second Label) rồi click copy và paste lên Form. VB6 sẽ cho bạn Array của hai lblRoom labels.

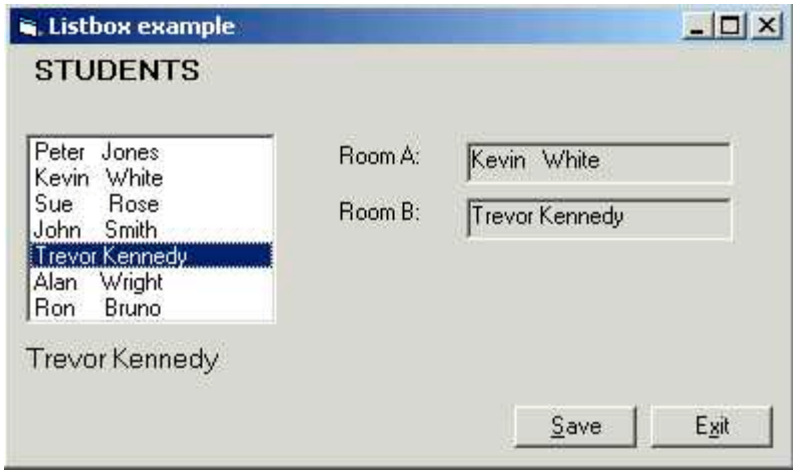
Để cho lstNames một DragIcon, bạn click lstNames, click Property DragIcon để pop-up một dialog cho bạn chọn một dragdrop icon từ folder **C:\Program Files\Microsoft Visual Studio\Common\Graphics\Icons\Dragdrop**, chẳng hạn như DRAG2PG.ICO:



Ta sẽ dùng **Event MouseDown** của lstNames để pop-up DragIcon hình 2 trang giấy cho User Drag nó qua bên phải rồi bỏ xuống lên một trong hai lblRoom. Khi DragIcon rơi lên lblRoom, lblRoom sẽ generate **Event DragDrop**. Ta sẽ dùng Event DragDrop nầy để assign property Text của **Source** (tức là lstNames, cái control từ nó phát xuất Drag action) vào Property Caption của lblRoom. Lưu ý vì ở đây ta dùng cùng một tên cho cả hai lblRoom nên chỉ cần viết code ở một chỗ để handle Event DragDrop.

```
Private Sub lstNames_MouseDown(Button As Integer, Shift As Integer, X
As Single, Y As Single)
   ' Start Pop-up DragIcon and start Drag action
   lstNames.Drag
End Sub
Private Sub lblRoom_DragDrop(Index As Integer, Source As Control, X
As Single, Y As Single)
   ' Assign Property Text of Source (i.e. lstNames) to Label's
Caption
   lblRoom(Index).Caption = Source.Text
End Sub
```

Kết quả sau khi Drag hai tên từ Listbox qua Labels là như sau:



Dùng Property Sorted

Trong thí dụ trên ta có thể quyết định vị trí của một Item mới khi ta nhét nó vào Listbox. Đôi khi ta muốn các Items của Listbox được tự động sắp theo thứ tự Alphabet. Bạn có thể set **Property Sorted = True** để thực hiện chuyện nầy. Có một giới hạn là bạn phải cho Property Sorted một value (True hay False) trong lúc design, chớ trong khi chạy program bạn không thể làm cho Property Sorted của Listbox thay đổi. Giả dụ ta muốn sort các Items của một Listbox khi cần. Vậy thì ta làm sao? Giải pháp rất đơn giản. Bạn tạo một Listbox tên lstTemp chẳng hạn. Cho nó Property Visible= False (để không ai thấy nó) và Property Sorted=True. Khi cần sort lstNames chẳng hạn, ta copy content của lstNames bỏ vào lstTemp, đoạn Clear lstNames rồi copy content (đã được sorted) của lstTemp trở lại lstNames.

Lưu ý là ta có thể AddItem vào một Listbox với Property Sorted=True, nhưng không thể xác định nhét Item vào trước hàng nào, vì vị trí của các Items do Listbox quyết định khi nó sort các Items.

Ta hãy cho thêm vào Form một CommandButton mới tên **CmdSort** và viết code cho Event Click của nó như sau:

```
Private Sub CmdSort_Click()
   Dim i
   lstTemp.Clear  ' Clear temporary Listbox
   ' Iterate though every item of lstNames
   For i = 0 To lstNames.ListCount - 1
      ' Add the lstNames item to lstTemp
      lstTemp.AddItem lstNames.List(i)
   Next
   lstNames.Clear  ' Clear lstNames
   ' Iterate though every item of lstTemp
   For i = 0 To lstTemp.ListCount - 1
      ' Add the lstTemp item to lstNames
      lstNames.AddItem lstTemp.List(i)
   Next
   lstTemp.Clear  ' Tidy up - clear temporary Listbox
End Sub
```

Nhân tiện, ta muốn có option để sort các tên theo FirstName hay Surname. Việc nầy hơi rắc rối hơn một chút, nhưng nguyên tắc vẫn là dùng cái sorted Listbox vô hình tên lstTemp.

Bạn hãy đặt lên phía trên lstName hai cál Labels mới tên lblFirstName và lblSurName và cho chúng Caption "FirstName" và "SurName". Từ đây ta Load file "MyList.txt" vào lstNames bằng cách Click button CmdLoad chớ không Edit Property List của lstNames để enter Items lúc design nữa. Ngoài ra ta dùng dấu phẩy (,) để tách FirstName khỏi

SurName trong mỗi tên chứa trong file MyList.txt. Content của file MyList.txt bây giờ trở thành như sau:

```
Peter,Jones
Kevin,White
Sue,Rose
John,Smith
Trevor,Kennedy
Alan,Wright
Ron,Bruno
```

Ta sẽ sửa code trong Sub CmdLoad_Click lại để khi nhét tên vào lstNames, FirstName và SurName mỗi thứ chiếm 10 characters. Để các chữ trong Items của lstNames sắp hàng ngay ngắn ta đổi Font của lstNames ra Courier New. Courier New là một loại Font mà chiều ngang của chữ **m** bằng chữ **i**, trong khi hầu hết các Fonts khác như Arial, Times Roman ..v.v. là **Proportional Spacing**, có nghĩa là chữ m rộng hơn chữ i. Listing mới của Sub CmdLoad_Click trở thành như sau:

```
Private Sub CmdLoad_Click()
   Dim i, Pos
   Dim FileName, FileNumber, anItem
   Dim sFirstName As String * 10  ' fixed length string of 10
characters
   Dim sSurName As String * 10  ' fixed length string of 10
characters
   ' Obtain Folder where this program's EXE file resides
   FileName = App.Path
   ' Make sure FileName ends with a backslash
   If Right(FileName, 1) <> "\" Then FileName = FileName & "\"
   FileName = FileName & "MyList.txt"
   ' Obtain an available filenumber from the operating system
   FileNumber = FreeFile
   ' Open the FileName as an input file , using FileNumber as
FileHandle
   Open FileName For Input As FileNumber
   lstNames.Clear  ' Clear the Listbox first
   ' Now read each line until reaching End-Of-File, i.e. no more data
   Do While Not EOF(FileNumber)
      Line Input #FileNumber, anItem  ' Read a line from the Text
file
      ' Now separate FirstName from SurName
      Pos = InStr(anItem, ",")  ' Locate the comma ","
      ' The part before "," is FirstName
      sFirstName = Left(anItem, Pos - 1)
      sFirstName = Trim(sFirstName)  ' Trim off any unwanted blank
spaces
      ' The part after "," is SurName
      sSurName = Mid(anItem, Pos + 1)
      sSurName = Trim(sSurName)  ' Trim off any unwanted blank spaces
      lstNames.AddItem sFirstName & sSurName  ' Add this item to the
bottom of lstNames
   Loop
   Close FileNumber  ' Close the input file
End Sub
```

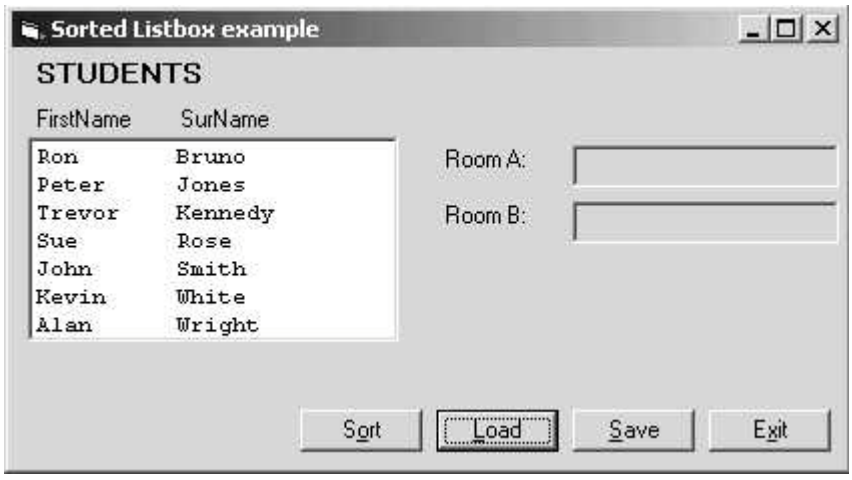
Vì FirstName nằm ở bên trái của mỗi Item nên sort theo FirstName cũng giống như sort cả Item. Việc ấy ta đã làm bằng Sub CmdSort_Click rồi, do đó khi User click Label lblFirstName ta chỉ cần gọi CmdSort_Click như sau:

```
Private Sub lblFirstName_Click()
   CmdSort_Click
End Sub
```

Để sort theo SurName ta cần phải tạm thời để SurName qua bên trái của Item trước khi bỏ vào lstTemp. Ta thực hiện chuyện nầy bằng cách hoán chuyển vị trí của FirstName và SurName trong Item trước khi bỏ vào lstTemp. Sau đó, khi copy các Items từ lstTemp để bỏ vô lại lstNames ta lại nhớ hoán chuyển FirstName và SurName để chúng nằm đúng lại vị trí. Tức là, cái mánh của ta là muốn biết Item nào phải nằm ở đâu trong lstNames, chớ dĩ nhiên khi display mỗi Item đều có FisrtName bên trái. Code để sort tên theo SurName cũng giống như CmdSort_Add nhưng thêm thắt chút ít như sau:

```
Private Sub lblSurName_Click()
   Dim i, anItem
   Dim sFirstName As String * 10  ' fixed length string of 10
characters
   Dim sSurName As String * 10  ' fixed length string of 10
characters
   lstTemp.Clear  ' Clear temporary Listbox
   ' Iterate though every item of lstNames
   For i = 0 To lstNames.ListCount - 1
      anItem = lstNames.List(i)
      ' Identify FistName and SurName
      sFirstName = Left(anItem, 10)
      sSurName = Mid(anItem, 11)
      ' Swap FirstName/SurName positions before adding to lstTemp
      lstTemp.AddItem sSurName & sFirstName
   Next
   lstNames.Clear  ' Clear lstNames
   ' Iterate though every item of lstTemp
   For i = 0 To lstTemp.ListCount - 1
      anItem = lstTemp.List(i)
      ' Identify FistName and SurName
      sSurName = Left(anItem, 10)  ' SurName now is on the left
      sFirstName = Mid(anItem, 11)
      ' Add FirstName/SurName in correct positions to lstNames
      lstNames.AddItem sFirstName & sSurName
   Next
   lstTemp.Clear  ' Tidy up - clear temporary Listbox
End Sub
```

Các Items trong lstNames sorted theo SurName hiện ra như sau:



Nhân tiện đây ta sửa cái Sub CmdSave_Click lại một chút để Save Items theo sorted order mới nếu cần:

```
Private Sub CmdSave_Click()
   Dim i, FileName, FileNumber, anItem
   ' Obtain Folder where this program's EXE file resides
   FileName = App.Path
   ' Make sure FileName ends with a backslash
   If Right(FileName, 1) <> "\" Then FileName = FileName & "\"
   ' Call Output filename "MyList.txt"
   FileName = FileName & "MyList.txt"
   ' Obtain an available filenumber from the operating system
   FileNumber = FreeFile
   ' Open the FileName as an output file , using FileNumber as
FileHandle
   Open FileName For Output As FileNumber
   ' Now iterate through each item of lstNames
   For i = 0 To lstNames.ListCount - 1
      anItem = lstNames.List(i)
      anItem = Trim(Left(anItem, 10)) & "," & Trim(Mid(anItem, 11))
      ' Write the List item to file. Make sure you use symbol # in
front of FileNumber
      Print #FileNumber, anItem
   Next
   Close FileNumber  ' Close the output file
End Sub
```