

Visual Basic 6

Created by mercury

UDS EBOOK
www.updatesofts.com

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 tắt 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 găt 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 này 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 trong 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.

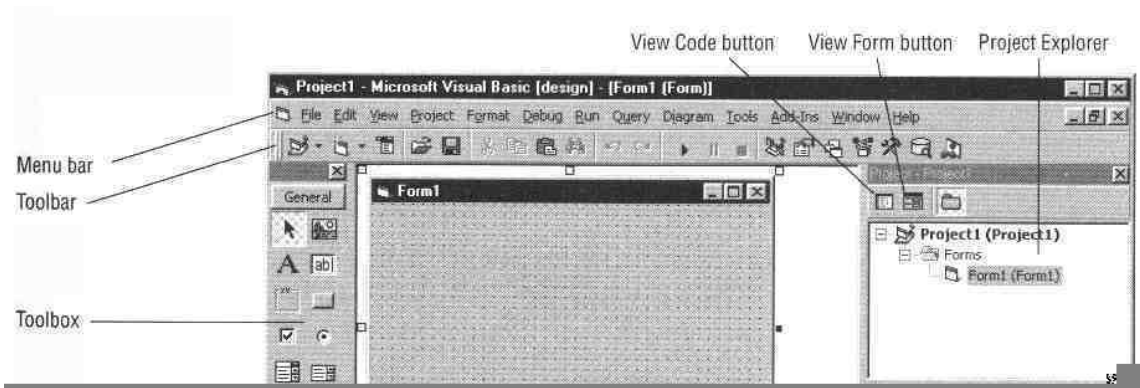
Ghi chú: Trong lúc cài VB6, nhớ chọn Graphics nếu không bạn sẽ thiếu một số hình ảnh như icons, bitmaps v.v... Đáng lẽ Microsoft cho tự động cài đặt Graphics, tức là Default (không có nói gì) thì cài đặt Graphics.

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.



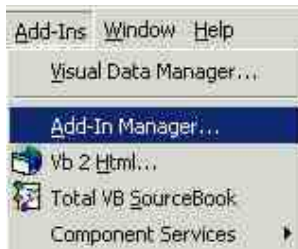
Chọn Standard EXE. Một lát sau trên màn ảnh sẽ hiện ra giao diện của môi trường phát triển tích hợp (Integrated Development Environment - IDE) giống như dưới đây:



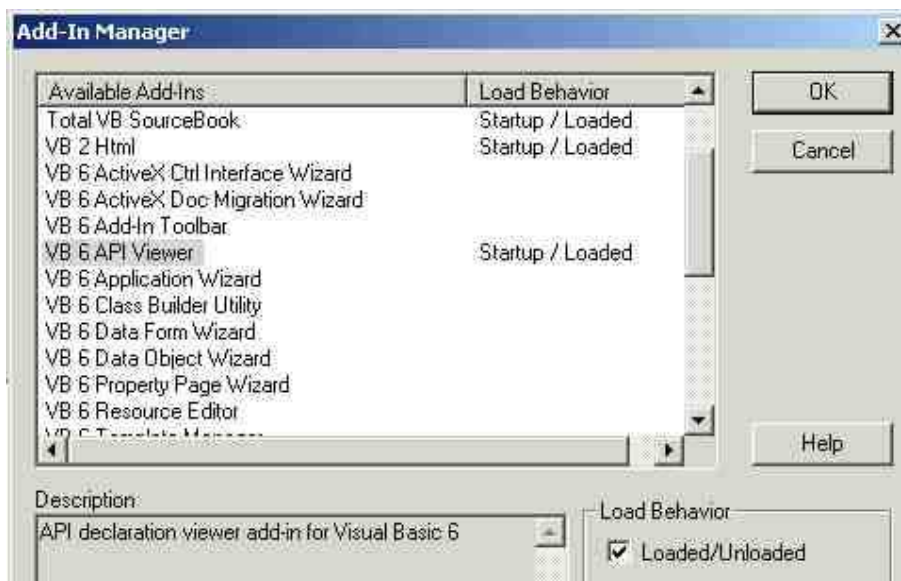
IDE của VB6 bao gồm các yếu tố sau:

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

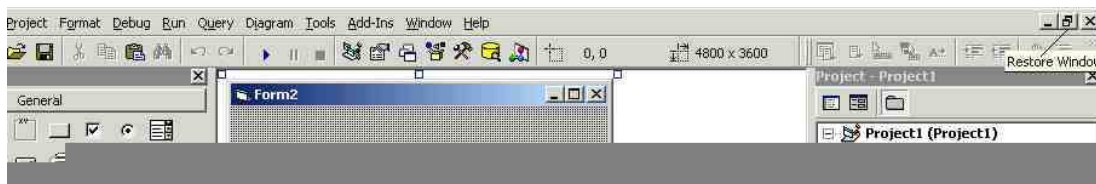
Liệt kê các đặc tính của các forms hoặc controls được chọn. Một property là một đặc tính của một object chẳng hạn như size, caption, hoặc color. Khi bạn sửa đổi một property bạn sẽ thấy hiệu quả ngay lập tức, thí dụ thay đổi property Font của một Label sẽ thấy Label ấy được display bằng Font chữ mới. 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 chia 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.

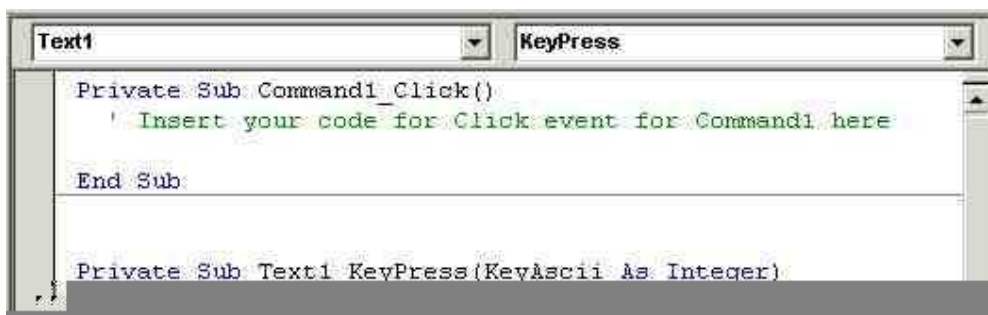


Immediate Window

Dùng để gỡ 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 này để xem code của một form mà bạn đã chọn. Window của code giống như dưới đây:



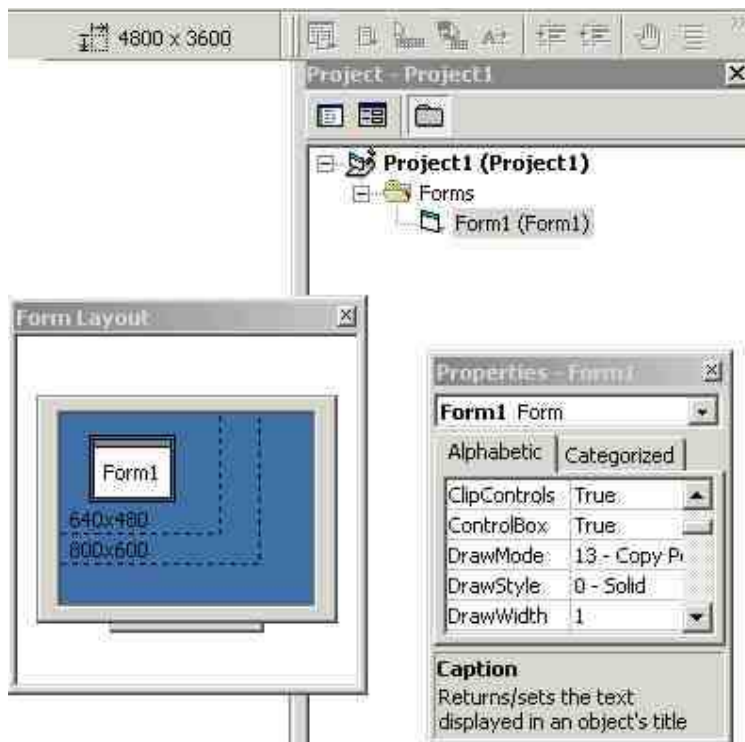
Trong Code window bạn có thể chọn display tất cả Sub của code cùng một lúc như trong hình hay display mỗi lần chỉ một Sub bằng cách click button có hình ba dòng nằm ở góc bên trái phía dưới.

View form button

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

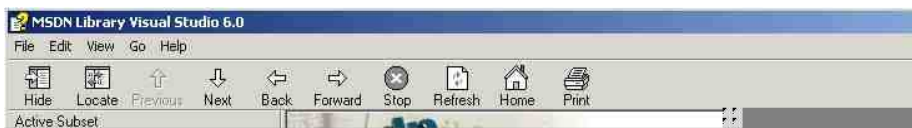
Ghi chú: Nhiều windows trong IDE như Toolbars, Toolbox, Project Explorer .v.v..có thể trôi lênh bênh (floating) hay đậu ở bên (docked). Bạn có thể thay đổi vị trí chúng bằng cách nắm vào Title Bar của window rồi dời đi. Dĩ nhiên bạn cũng có thể mở rộng hay làm nhỏ một window bằng cách dời một cạnh vertical 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 gỡ 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 này 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)

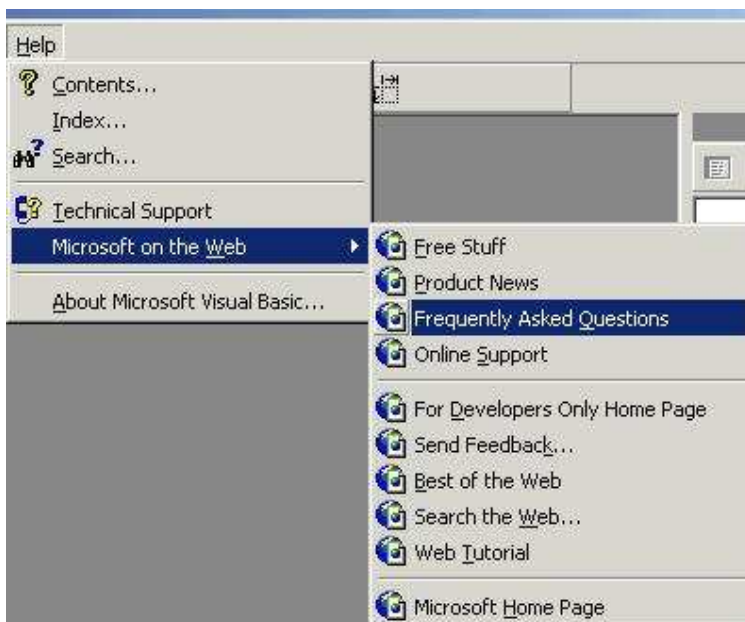
Ngoài ra, trong Help thường có **Example**. Bạn click lên chữ Example để display một thí dụ minh họa cách dùng một function hay property.

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 này 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.



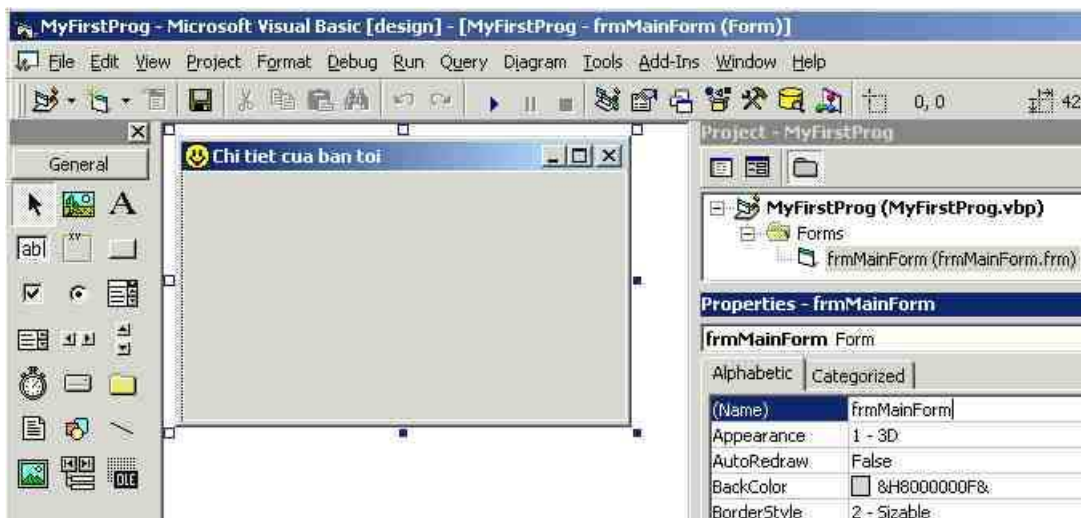
Ghi chú: Một số nội dung trên Web site của Microsoft được tối ưu hóa dành cho Microsoft Internet Explorer và không thể display đầy đủ trong một bộ trình duyệt (browser) khác. Do đó bạn nên chỉ dùng Internet Explorer làm browser trên máy bạn mà thôi.

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 Properties 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 Properties (trong hình dưới đây ta edit Caption property của form thành ra "Chi tiet cua ban toi").

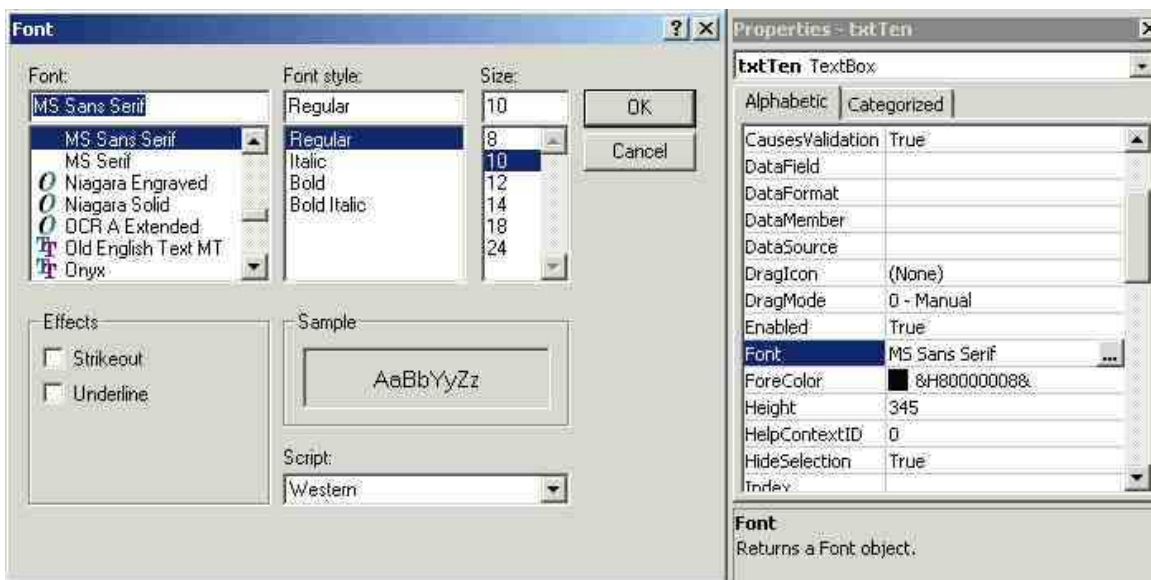


Sắp đặt các vật dụng lên Form

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 về thành cỡ 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 bức mình tại sao cái cỡ chữ tự có (default size) của các Control hơi nhỏ, bạn có thể giải quyết bằng cách sửa cỡ chữ của chính Form cho nó lớn hơn. Vì khi một Control được đặt lên một Form, nó thừa kế cỡ chữ 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 này 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 này 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
End
Begin VB.Label lblTen
    Caption       = "&Ten:"
    Height        = 375
    Left          = 210
    TabIndex      = 0
    Top           = 240
    Width         = 495
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

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

```

Sau này, 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 kèm (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 này 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 này 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 này 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 này 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 tồn 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 hệ bạn làm điều gì bí hiểm hay các cơ 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 vào đĩa" 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 này ta muốn gỡ 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 này 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 này thì nếu hồ sơ chưa có nó sẽ được dựng nên (created). Nếu hồ sơ đã có rồi thì nó sẽ bị xóa 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 đĩa (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 này 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.

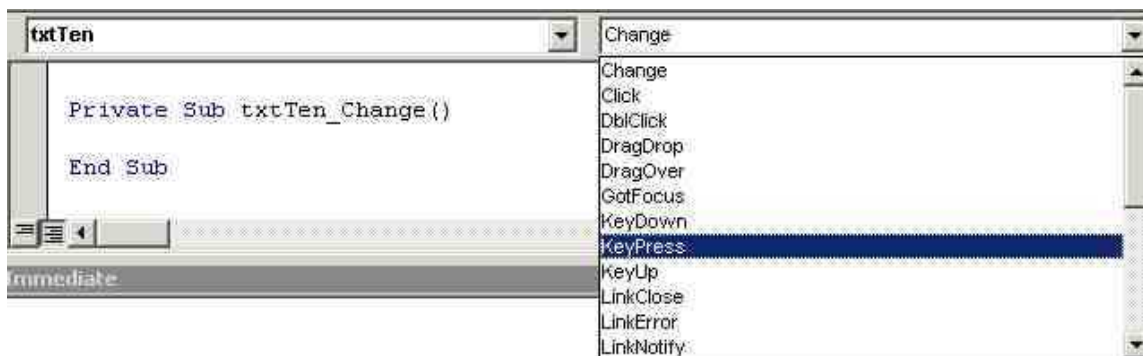
Sau khi người xử dụng đã cho tên vào rồi, cô sẽ đánh nút Tab để di chuyển cursor qua Control tiếp theo, mà ta muốn là txtDiachi. Để sắp thứ tự các Control cho sự di chuyển của cursor khi người xử dụng đánh nút Tab ta edit Property TabIndex của các Control. TabIndex bắt đầu bằng số 0. Nhiều khi người 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 gọi nút Tab
        KeyAscii = 0 ' Nuốt trọn nút Enter để Windows không còn lo cho nó
    End If
End Sub

```

Cho các Textbox khác như txtDiachi, txtTuoi bạn cũng làm tương tự 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 sử 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.

Kể 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 sử 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 này người sử dụng có thể bấm Alt-X coi như tương đương với click nút "Xuat".

Nếu nhờ 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 này để dẫn Form giựt cái nút người sử dụng đánh (keystroke) trước khi TextBox thấy. Form sẽ tráo nút Enter thành Tab rồi lẳng 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 gọi nút Tab
        KeyAscii = 0 ' Nuốt trọn nút Enter để Windows không còn lo cho nó
    End If
End Sub
```

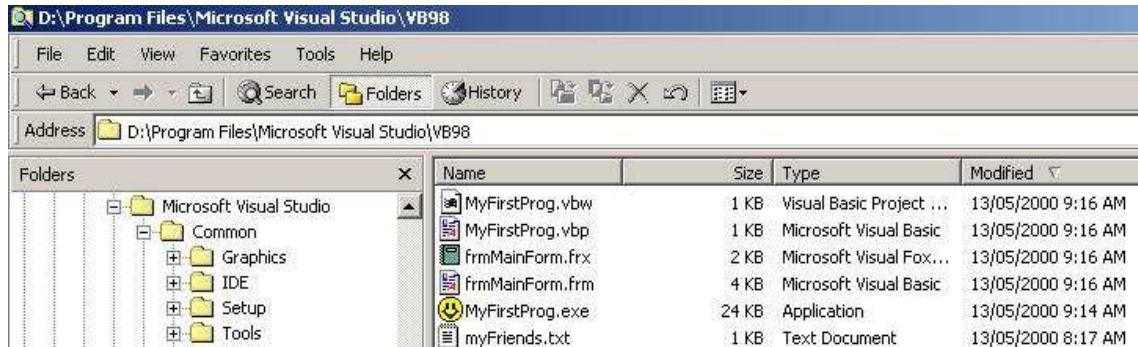
Khi bạn doubleClick lên bất cứ chỗ nào trên Form không có Control nằm, lần đầu để viết mã, VB cho bạn Private Sub Form_Load(). Bạn phải click cái Combobox bên phải, phía trên của Code Window, cho nó mở ra và chọn Event KeyPress.

Đem ra trình làng

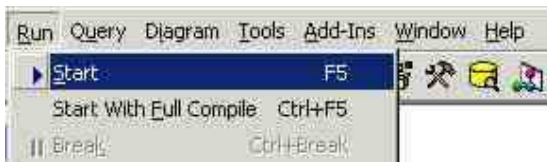
Để làm thành một hồ sơ áp dụng EXE, bạn dùng mệnh lệnh Menu **File|Make MyFirstProg.exe**. Cho thêm chút hương vị của cuộc đời tôi click Form rồi edit Property Icon, chọn cho nó từ folder:

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**.



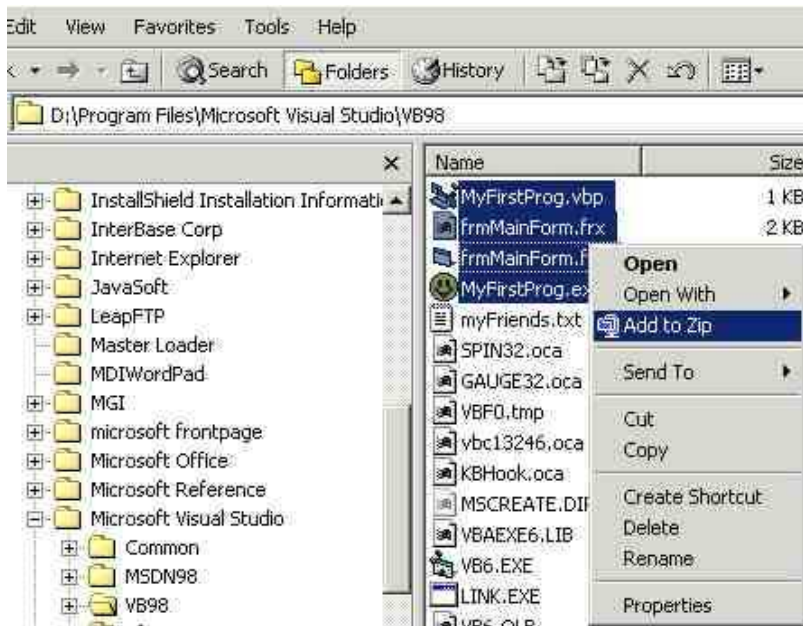
Bạn cũng có thể Click lên dấu tam giác chỉ về bên phải (nút Play của cassette) nằm trong toolbar ngay phía dưới VB menu.



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 dãy 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.



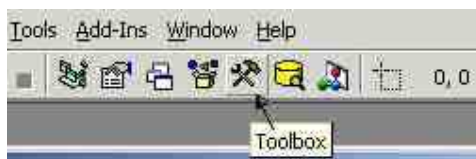
Bây giờ bạn thử khởi động VB6, đi dạo quanh nó để thử biết các phần của VB6 IDE, và thử làm theo như bài này. Nếu có thắc mắc gì thì hỏi Tutor của bạn.

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 này sẽ hiện ra trước hết để ta ra lệnh nó làm chuyện gì. Cái Form trông 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 cỡ 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 Tủ 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 bên) . Nếu một window là

dockable, sau khi bạn dời nó đi khỏi vị trí docked bình thường của nó, bạn có thể dock nó lại như cũ 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.

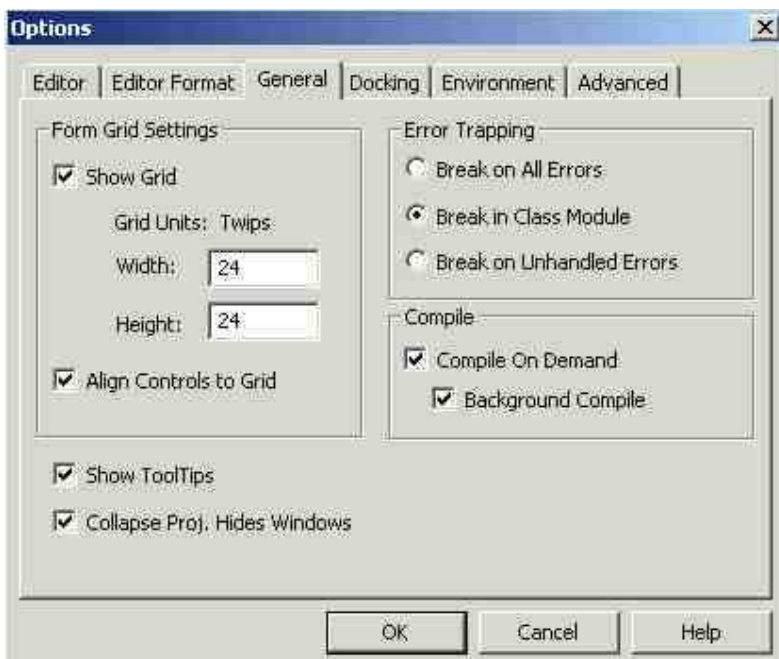
Lưu ý: Một số control có kích thước tối thiểu, bạn không thể làm cho nó nhỏ hơn được. Thí dụ như Combobox, nó phải cao đủ để display một hàng text.

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

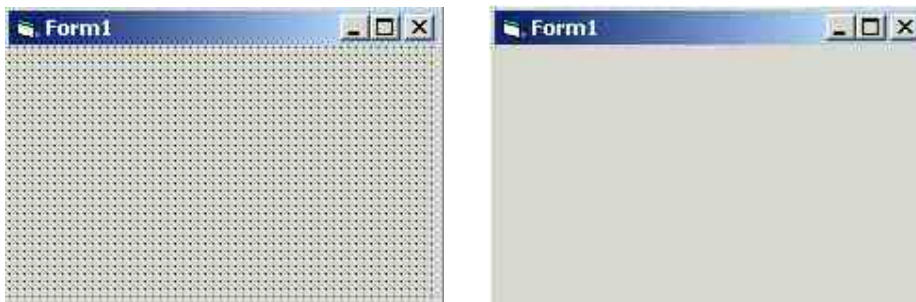
Ngoài ra, nên nhớ rằng trong lúc chương trình chạy (at run-time), trong code ta có thể thay đổi kích thước và vị trí các controls dễ dàng, thậm chí có thể làm cho chúng hiện ra hay biến mất bằng cách sửa đổi value các property left, top, width, height và visible của các controls.

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, kẻ đó chọn Tag General và clear checkbox "Show Grid":



Bạn cũng có thể nhân dịp này 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 ổ khóa trên menu bị chìm xuống.



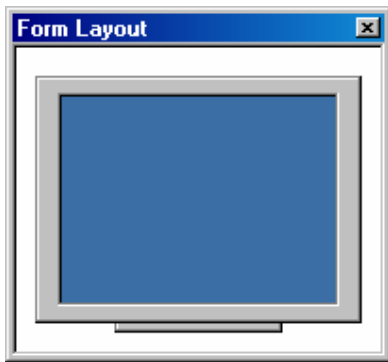
Nếu sau này bạn muốn thay đổi kích thước hoặc vị trí của chúng thì nhớ dùng Menu command **Format | Lock Controls** lại. Sau khi mở khóa, cái hình ổ khóa trên menu hiện ra bình thường.

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 hiển 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 minimize, 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 tự 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 này xảy 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 này. F
- **Form_Load:** Event này xảy 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 này. F
Bên trong Form_Load bạn không thể dùng Setfocus cho một control nào trên form vì form chưa

hắn thành hình (ra đời). Muốn làm việc ấy bạn phải delay (trì hoãn) 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 này để refresh display trên form. F
- **Form_QueryUnload:** Khi User click dấu **x** phía trên bên phải để close form thì nó generate QueryUnload event. Syntax của Sub này như dưới đây: F

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

Event này 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 này. 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 này 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 mẹo 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 nhỏ nhỏ, giống như một form. Khi ta lập trình VB6 ta lắp 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. P
 - **methods:** những gì control thực hiện được, tức là những khả năng của nó. M
 - **events:** những sự cố mà control sẽ thông báo cho chúng ta biết khi nó xảy ra với control. Khi một event xảy 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 thêm biết đến các event loại đó. Có một số Events mà chúng ta thường xử lý là: E
 - click : xảy ra khi user click lên control. Ta thường dùng nó cho CommandButton và Listbox. C
 - 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. M
- 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 : xảy 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. K

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
```

- eyDown, 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
```

- LotFocus : 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 hệ một Control GotFocus thì trước đó có một Control LostFocus. Ta có thể dùng Event này 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
```

- DagDrop : xảy 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.

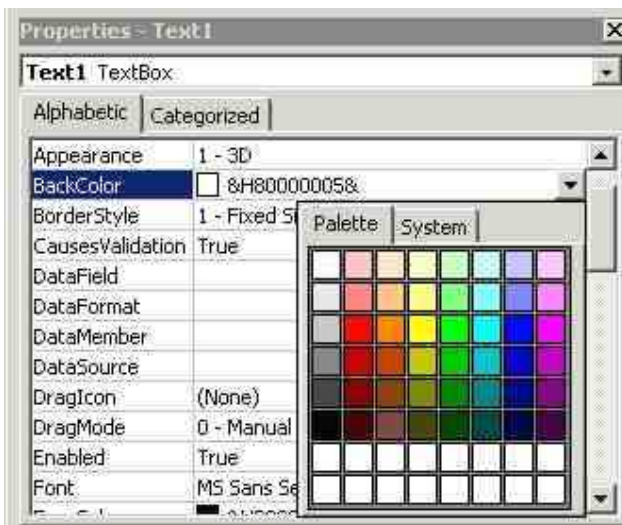
```
Private Sub Text2_DragDrop(Source As Control, X As Single, Y As Single)
    If TypeOf Source Is ListBox Then
        Text2.Text = Source.Text
    End If
End Sub
```

TextBox

TextBox là control được dùng nhiều nhất để display text và nhận keystroke của User để 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.

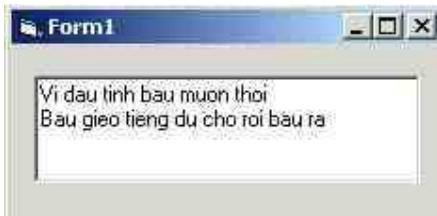


Bạn có thể chọn BackColor và ForeColor cho background và text của TextBox. Dùng Tag Palette khi chọn màu để có đúng một màu bạn muốn.



Dĩ nhiên bạn có thể lựa chọn Font và cỡ chữ cho Text với Font Property.

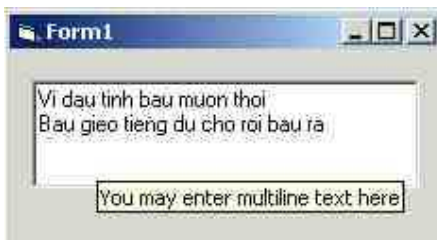
Bạn giới hạn số characters mà User có thể enter cho TextBox bằng cách set MaxLength Property. Nếu Property Multiline là True thì User có thể enter nhiều hàng. At Design time, nếu bạn muốn enter multiline thì phải nhớ bấm nút Ctrl khi press Enter mỗi khi xuống hàng. Nếu không VB6 IDE tưởng rằng bạn đã 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 dùng 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 Password.



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 CommandButton 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:

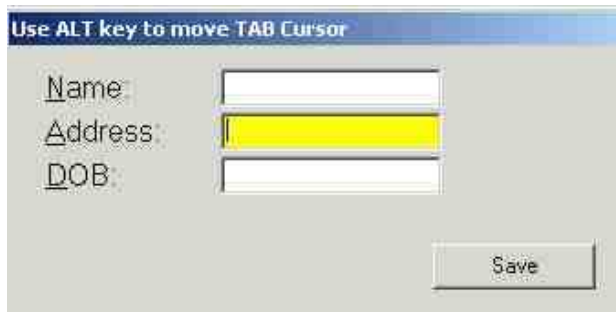
```
Private Sub CmdOperation_Click()
    If CmdOperation.Caption = "&Stop" Then
        CmdOperation.Caption = "St&art"
    Else
        CmdOperation.Caption = "&Stop"
    End If
End Sub
```

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 lõm 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

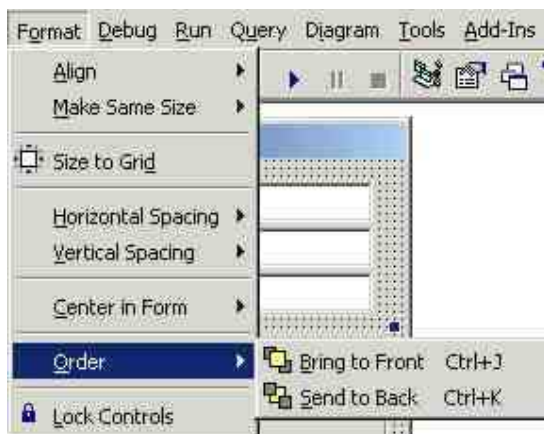
OptionButton (còn gọi là RadioButton) có hình tròn với một chấm ở giữa, thay vì hình vuông với một gạch ở giữa như CheckBox. OptionButton luôn luôn được qui tụ thành một nhóm, chứa trong một container. Container là một Control có khả năng chứa các controls khác. Frame, PictureBox, hay chính Form đều 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 này sang container khác, bạn Cut OptionButton rồi Paste nó vào 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 chính 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 xảy ra. Thí dụ như khi User click một CommandButton (Event Click) hay type nút Tab để di chuyển Cursor từ Textbox này (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 này 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 này) 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
    End If
End Sub

```



```

    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 chứ không đợi phải execute xuống tới hàng chốt.

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 hễ 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 lẫn 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

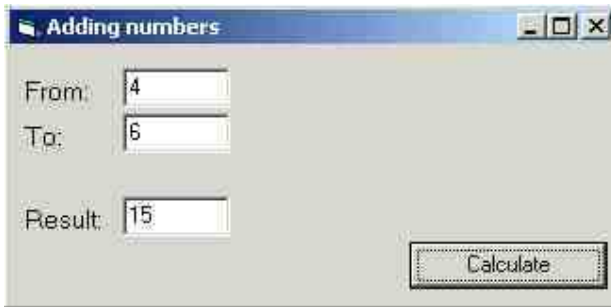
```

Cách viết này 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. Hễ khi điều kiện của một **Case** được thỏa 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**, chứ 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 này 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 này 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 chẵn 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 này 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. Kể đó i=2 thì nhỏ hơn số cuối cùng (=4) nên execution nhảy ra khỏi FOR loop.

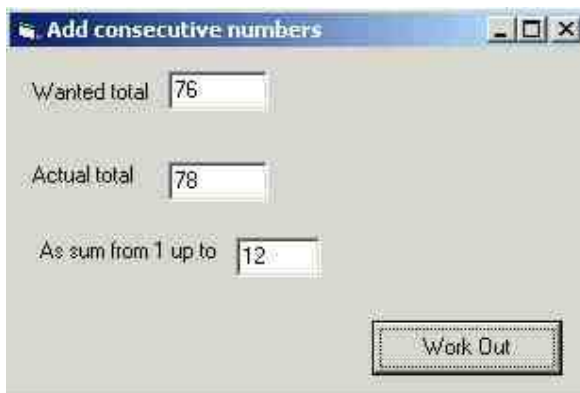
Giả dụ ta muốn lấy ra tất cả những blank space trong một text string. Ta biết con số characters trong một text string, còn gọi là chiều dài của text string có thể tính bằng cách dùng Function Len(TString). Và để nói đến character thứ i trong một Text string ta dùng Mid Function.



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 ' Display 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, chứ 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 bằng Function CInt
    Total = 0 ' Initialise Total value to zero
    i = 1 ' Initialise 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 value 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:

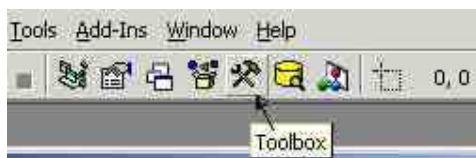
```
Form2.DisplayData
```

Chương 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 này sẽ hiện ra trước hết để ta ra lệnh nó làm chuyện gì. Cái Form trông 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 cỡ 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 Tủ 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 bên) . Nếu một window là dockable, sau khi bạn dời nó đi khỏi vị trí docked bình thường của nó, bạn có thể dock nó lại như cũ 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.

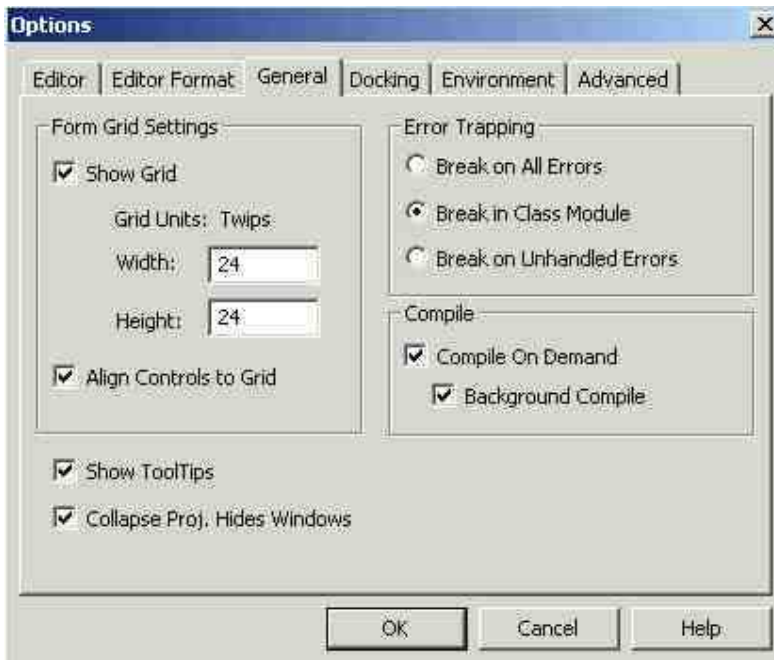
Lưu ý: Một số control có kích thước tối thiểu, bạn không thể làm cho nó nhỏ hơn được. Thí dụ như Combobox, nó phải cao đủ để display một hàng text.

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

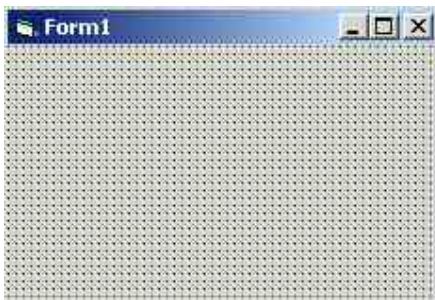
Ngoài ra, nên nhớ rằng trong lúc chương trình chạy (at run-time), trong code ta có thể thay đổi kích thước và vị trí các controls dễ dàng, thậm chí có thể làm cho chúng hiện ra hay biến mất bằng cách sửa đổi value các property left, top, width, height và visible của các controls.

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 này 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 ổ khóa trên menu bị chìm xuống.



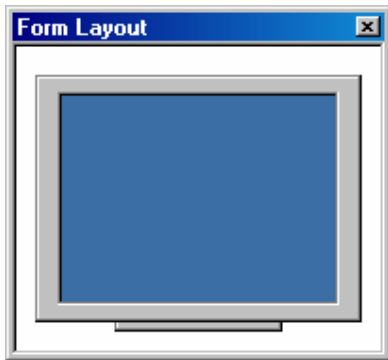
Nếu sau này bạn muốn thay đổi kích thước hoặc vị trí của chúng thì nhớ dùng Menu command **Format | Lock Controls** lại. Sau khi mở khóa, cái hình ổng khóa trên menu hiện ra bình thường.

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 hiển 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 tự 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 này xảy 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 này. F
- **Form_Load:** Event này xảy 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 này. F
 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 hoãn) 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 này để refresh display trên form. F
- **Form_QueryUnload:** Khi User click dấu **x** phía trên bên phải để close form thì nó generate QueryUnload event. Syntax của Sub này như dưới đây: F

```

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

```

Event này 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 này. 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.

Kể 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 này 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 mẹo 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 lắp 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:

- **roperties:** 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. P
 - **ethods:** những gì control thực hiện được, tức là những khả năng của nó. M
 - **vents:** những sự cố mà control sẽ thông báo cho chúng ta biết khi nó xảy ra với control. Khi một event xảy 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: E
 - lick : xảy ra khi user click lên control. Ta thường dùng nó cho CommandButton và Listbox. C
 - ouseDown, 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. M
- 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

```

Đề 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.

- K
 KeyPress : xảy 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

```

- K
 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

```

- G
 LostFocus : 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

```


○

L

LostFocus : Thường thường hệ một Control GotFocus thì trước đó có một Control LostFocus. Ta có thể dùng Event này 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
```

○

r

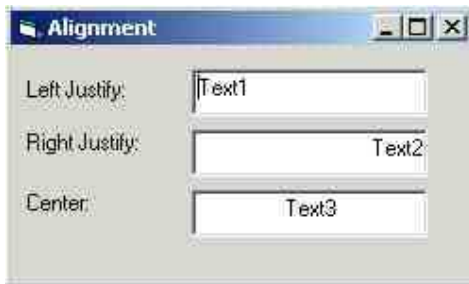
DagDrop : xảy ra khi ta drop một cái gì lên control . Parameter Source cho ta biết Control nào đã được Drag và Drop. Nhiều khi một control có thể nhận drop từ nhiều control khác nhau. Trong trường hợp đó ta phải test xem hoặc Control Type, hoặc Name hoặc Tag value của Source control là gì để tùy nghi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

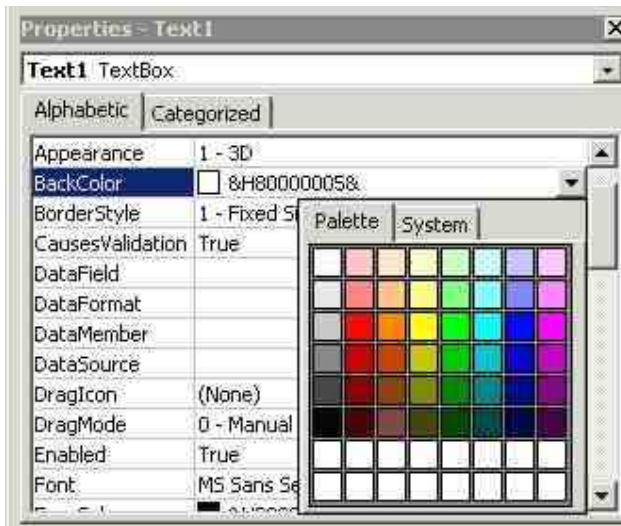
```
Private Sub Text2_DragDrop(Source As Control, X As Single, Y As Single)
    If TypeOf Source Is ListBox Then
        Text2.Text = Source.Text
    End If
End Sub
```

TextBox

TextBox là control được dùng nhiều nhất để display text và nhận keystroke của User để 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.



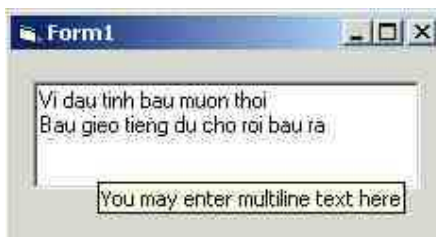
Dĩ nhiên bạn có thể lựa chọn Font và cỡ chữ cho Text với Font Property. Bạn giới hạn số characters mà User có thể enter cho TextBox bằng cách set MaxLength Property. Nếu Property Multiline là True thì User có thể enter nhiều hàng. At Design time, nếu bạn muốn enter multiline thì phải nhớ bấm nút Ctrl 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:

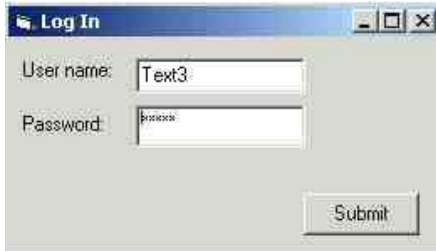
```
Private Sub Command1_Click()
    Dim TextStr
    TextStr = "Bau ra bau lay ong cau" & vbCrLf 'Note: vbCrLf = chr(13) & chr(10)
    TextStr = TextStr & "Bau cau ca bong ngat dau kho tieu"
    Text1.Text = TextStr
End Sub
```

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 dùng 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 Password.



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 CommandButton 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 (để 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 lõm 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 mờ 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.

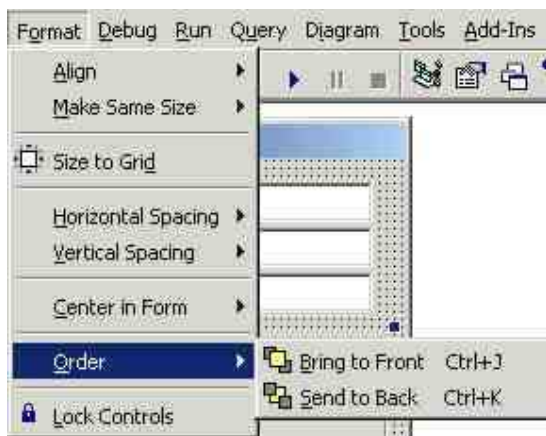
OptionButton

OptionButton (còn gọi là RadioButton) có hình tròn với một chấm ở giữa, thay vì 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 kéo để 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 này sang container khác, bạn Cut OptionButton rồi Paste nó vào 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 chính 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 xảy ra. Thí dụ như khi User click một CommandButton (Event Click) hay type nút Tab để di chuyển Cursor từ Textbox này (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 tự xử lý các dòng code

Giả dụ ta viết một chương trình Vb6 đơn giản như trong hình này 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 này) 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 chứ không đợi phải execute xuống tới hàng chốt.

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 lẫn 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
```

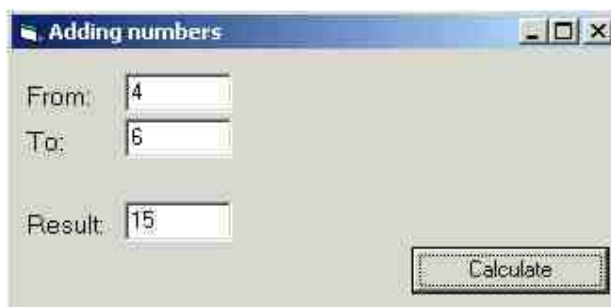
Cách viết này tương đối dễ đọc và ít nhầm lẫn khi viết code hơn là dùng nhiều ELSEIF. Phần **ELSE** trong Select Case statement thì optional (nhiệm ý), tức là có cũng được, không có cũng không sao. Hễ khi điều kiện của một **Case** được thỏa mãn thì những dòng code từ đó cho đến dòng **Case** kế dưới 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**, chứ 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 này 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 này 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 chẵn 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 này 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. Kể 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
    Next i
    TMess = TMess & Mid(txtOriginalString.Text, i, 1)
Next i

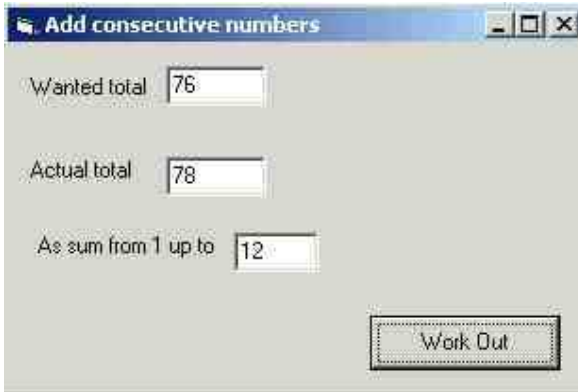
```

```

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 ' Display 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, chứ 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 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 này sẽ hiện ra trước hết để ta ra lệnh nó làm chuyện gì. Cái Form trông 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 cỡ 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 Tủ 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 bên) . Nếu một window là dockable, sau khi bạn dời nó đi khỏi vị trí docked bình thường của nó, bạn có thể dock nó lại như cũ 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.

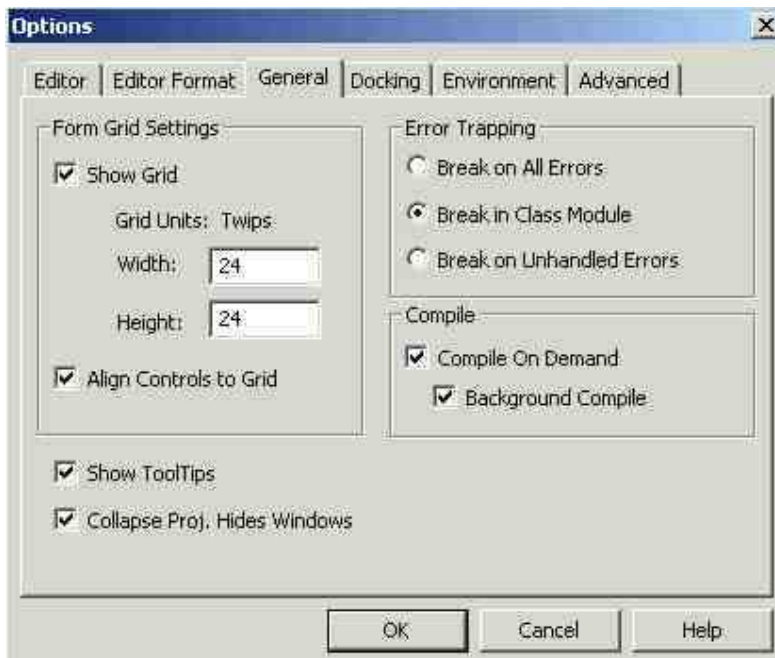
Lưu ý: Một số control có kích thước tối thiểu, bạn không thể làm cho nó nhỏ hơn được. Thí dụ như Combobox, nó phải cao đủ để display một hàng text.

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

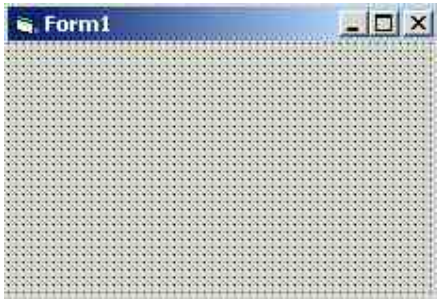
Ngoài ra, nên nhớ rằng trong lúc chương trình chạy (at run-time), trong code ta có thể thay đổi kích thước và vị trí các controls dễ dàng, thậm chí có thể làm cho chúng hiện ra hay biến mất bằng cách sửa đổi value các property left, top, width, height và visible của các controls.

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 này 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 ổ khóa trên menu bị chìm xuống.



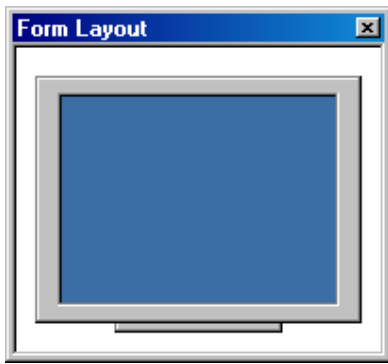
Nếu sau này bạn muốn thay đổi kích thước hoặc vị trí của chúng thì nhớ dùng Menu command **Format | Lock Controls** lại. Sau khi mở khóa, cái hình ổ khóa trên menu hiện ra bình thường.

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.

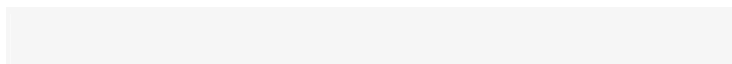
Vài Event thông dụng của Form

Nhìn từ một phương diện, Form cũng giống như Control. Ta có thể instantiate một form nhiều lần để có nhiều form tương tự 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 này xảy 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 này. F
- **Form_Load:** Event này xảy 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 này. F
 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 hoãn) 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

```

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

```

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

```

Event này 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 này. 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 này 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 mẹo 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 nhỏ nhỏ, giống như một form. Khi ta lập trình VB6 ta lắp 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. P
 - **methods:** những gì control thực hiện được, tức là những khả năng của nó. M
 - **vents:** những sự cố mà control sẽ thông báo cho chúng ta biết khi nó xảy ra với control. Khi một event xảy 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 thêm biết đến các event loại đó. Có một số Events mà chúng ta thường xử lý là: E
 - lick : xảy ra khi user click lên control. Ta thường dùng nó cho CommandButton và Listbox. C
 - ouseDown, 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. M
- 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
```

Đề 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.

- eyPress : xảy 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. K
- 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

```

- eyDown, 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

```

- LotFocus : 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 này 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

```

- DagDrop : xảy 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

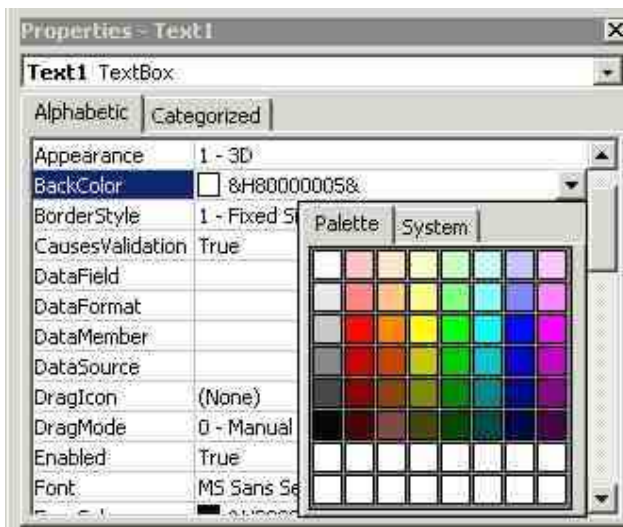
```

TextBox

TextBox là control được dùng nhiều nhất để display text và nhận keystroke của User để 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 cỡ chữ cho Text với Font Property.

Bạn giới hạn số characters mà User có thể enter cho TextBox bằng cách set MaxLength Property.

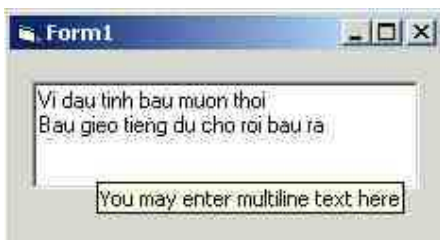
Nếu Property Multiline là True thì User có thể enter nhiều hàng. At Design time, nếu bạn muốn enter multiline thì phải nhớ bấm nút Ctrl khi press Enter mỗi khi xuống hàng. Nếu không VB6 IDE tưởng rằng bạn đã kết thúc editing.



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 dùng 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 Password.



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 CommandButton 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 (đề 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.



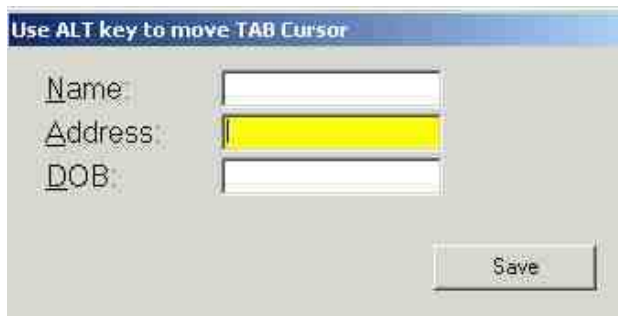
Lúc Run-time bạn có thể thay đổi hình hay Caption của CommandButton. Trong thí dụ dưới đây, Caption của CommandButton CmdOperation flip-flop giữa hai values Stop và Start:

```
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 lõm 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.

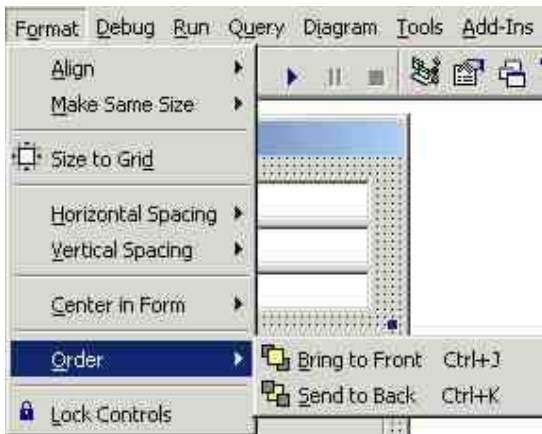
OptionButton

OptionButton (còn gọi là RadioButton) có hình tròn với một chấm ở giữa, thay vì 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 này sang container khác, bạn Cut OptionButton rồi Paste nó vào 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 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 sở 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

Rất 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. Nếu bạn chưa biết nhiều thì hãy đọc bài [Hệ thống số nhị phân](#).

Thí dụ, khi bạn ấn nút **A** trên keyboard, keyboard sẽ gửi 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 này 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. Điều này 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 này 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 này 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 này 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 này 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 chứa một con số nguyên từ -2147483648 đến 2147483647. Nếu bạn dùng Integer mà bị Overflow error khi làm toán nhân thì assign các con số vào một Long variable (sẽ cắt nghĩa variable sau này) **TRƯỚC KHI** làm toán nhân chứ 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 Integer 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 nửa 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, thúng ..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 CmdIncrX_Click()
    iCount = iCount + 1
    X = X + 1
    If X = 80 Then
        X = 0
        Y = Y + 1
    End If
End Sub

Private Sub CmdIncrY_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 CmdIncrY_Click() có declare một variable Y. Scope của variable này là chỉ nội bộ, tức là bên trong Sub CmdIncrY_Click() mà thôi. Chẳng những thế, cái **local (địa phương)** variable Y này còn che cái **global** variable Y nữa, tức là bên trong Sub CmdIncrY_Click() ta chỉ thấy local variable Y mà không thấy global variable Y. Một khi execution trong Sub CmdIncrY_Click() đã kết thúc thì local variable Y cũng biến mất luôn. Nếu bạn muốn khi trở lại execute Sub CmdIncrY_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 trong 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, nhớ 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 này rất là tiện lợi. Tuy nhiên, nếu ta sơ ý đánh vắn 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 này 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 chứa 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 = CDate(txtDate.Text) ' convert a Text String to internal Date using  
Function CDate  
        ' 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 này 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, CDate(txtDate.Text))
' txtNextMonth.text will show 8/07/2001
End Sub
```

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

```
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, CDate(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 trễ hơn. Thí dụ khoảng cách giữa hai ngày theo đơn vị tháng:

```
DateDiff("m", Now, CDate(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 này 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 này 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 INumber As Integer
Dim INumber As Integer
Dim IAge As Integer
Dim sPersonalWorth As Single
If (IAge >= 18) Then
    INumber = 1
Else
    INumber = 0
End If
If (sPersonalWorth > 1000000) Then
    INumber = 1
Else
    INumber = 0
End If
If (INumber = 1) And (INumber = 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

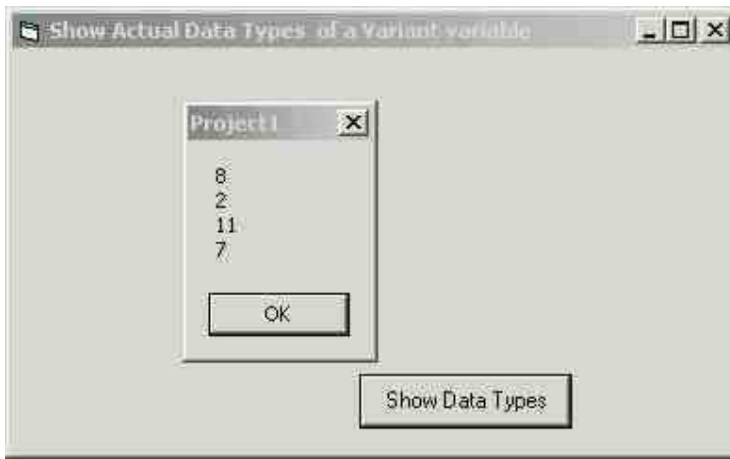
Variant variable có thể chứa Text String, Number, Date, thậm chí cả một Array (một loạt nhiều variables cùng data type). Nhìn thoáng qua nó rất tiện dụng, nhưng khi một Variant variable được dùng nhiều chỗ, trong nhiều tình huống khác nhau, bạn phải thận trọng. Lý do là vì variant variable có thể chứa 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 hớ 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 tính, nhưng sẽ tránh bị những surprise (ngạc nhiên) tốn kém sau này. Các ngôn ngữ lập trình gắt gao ấy được gọi là **strongly typed languages**, chẳng hạn như Pascal, C++, Java .v.v.. Sau này 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 Variant 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 này. Thí dụ như thay gì dùng trực tiếp một 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 này muốn thay đổi con số các xe thành ra 10, ta phải tìm và thay đổi tất cả các con số 5 này 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 này ta có thể dùng Constant như sau:

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

Sau này muốn thay đổi con số các xe, ta chỉ cần edit value của Constant. Trong khắp chương trình, ni 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 này qua loại data khác

Nhiều lúc ta cần phải convert data type của một variable từ loại này qua loại khác, VB6 cho ta một số các Functions dưới đây. Xin lưu rằng khi call các Functions này, 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 bức 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 này 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 tự 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
```

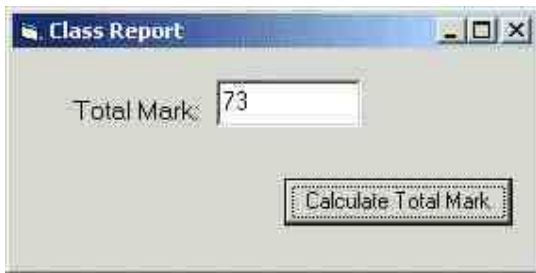
Kể đó bạn nói đến điểm của mỗi học sinh bằng cách viết **myStudentMarks(i)**, mà i là ArrayIndex. Giả dụ ta muốn tính tổng số điểm:

```
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** chứ 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

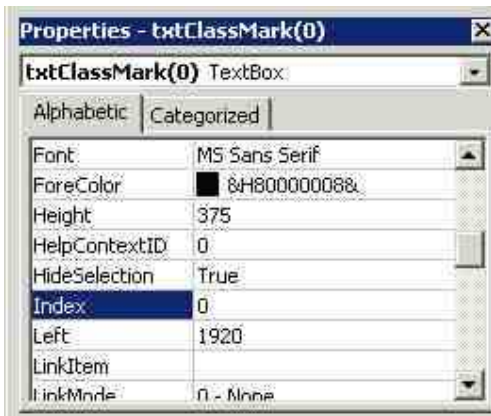
```

```

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 này 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 này 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 này 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 này rất tiện lợi, nhưng sau này 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.



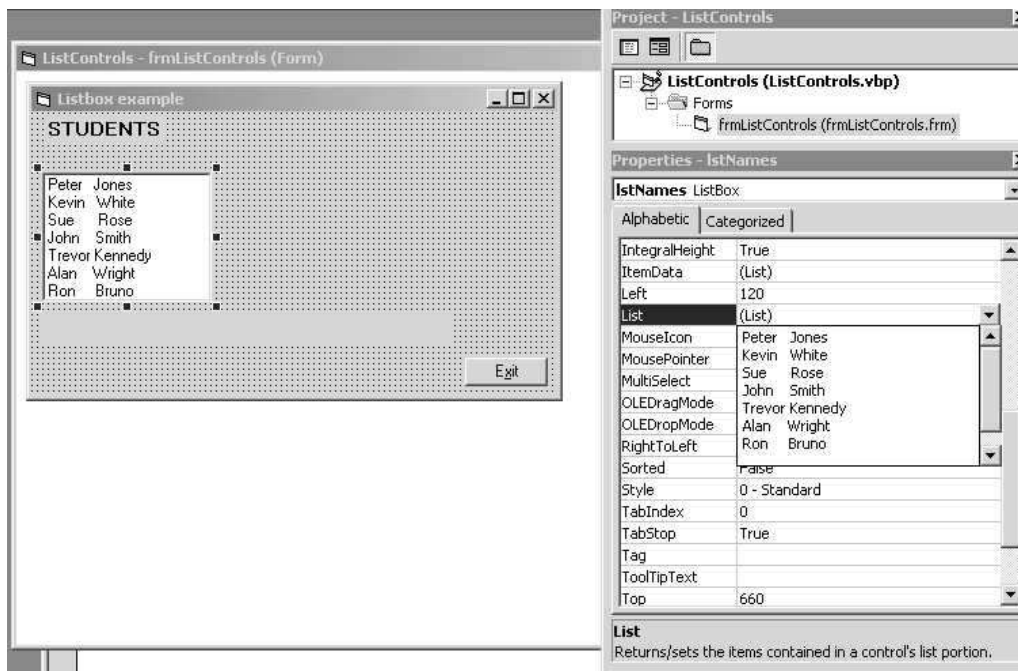
Listbox có rất nhiều công dụng vì nó rất uyển chuyển. Trong chương này ta sẽ học qua các áp dụng sau của Listbox:

- 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

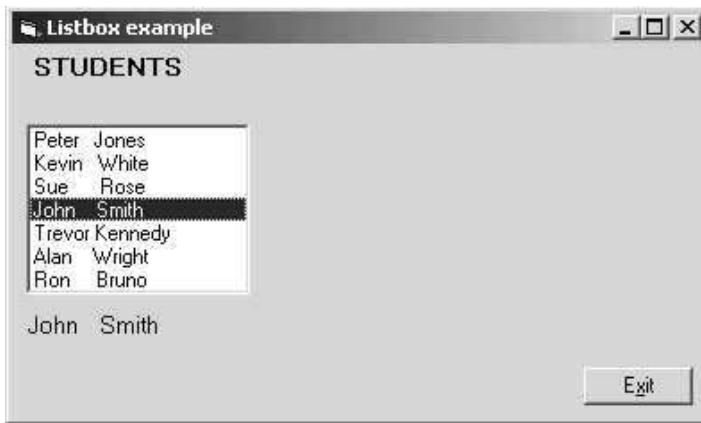
Ta hãy bắt đầu viết một chương trình gồm có một Listbox tên **lstNames** nằm trong một Form. Trong lstNames ta đánh vào tên của bảy người, mỗi lần xuống hàng nhớ đánh **Ctrl-Enter**, thay vì chỉ **Enter**, nếu không VB6 tưởng ta đã đánh xong nên close property List. Các tên này là những hàng sẽ hiện 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 này, Listbox lstNames có 7 hàng (**Items**). Con số Items này là Property **ListCount** của Listbox. Các Items của Listbox được đếm từ **0 đến ListCount-1**. Trong trường hợp này 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 này được gọi là **List**. Do đó, ta nói đến Item thứ nhất của Listbox lstNames bằng cách viết **lstNames.List(0)** , và tương tự 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 trữ 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 filename 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 lẫn lẫn 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 này 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 này còn được gọi là **FileHandle** (Handle là tay cầm). Sau khi ta Close FileNumber con số này 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 này, 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 filename 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.

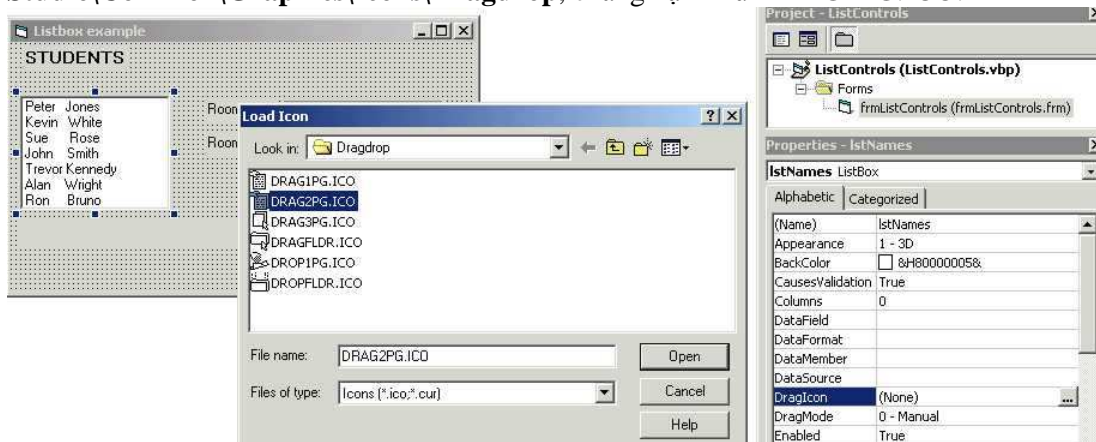
Ta đọc từng hàng của một Text file bằng cách dùng **Line Input #FileNumber**. Khi đọc đến cuối File, system sẽ cho ta value EOF(FileNumber) = True. Ta dùng value ấy để cho program nhảy ra khỏi While.. Loop.

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:



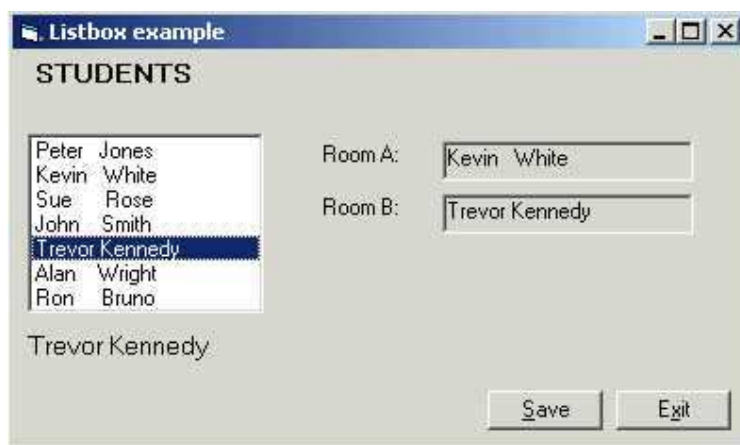
Ta sẽ dùng **Event MouseDown** của lstNames để pop-up DragIcon hình 2 trang giấy cho User Drag nó qua bên phải rồi bỏ xuống lên một trong hai lblRoom. Khi DragIcon rơi lên lblRoom, lblRoom sẽ generate **Event DragDrop**. Ta sẽ dùng Event DragDrop này để assign property Text của **Source** (tức là lstNames, 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 này. Có một giới hạn là bạn phải cho Property Sorted một value (True hay False) trong lúc design, chứ 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 này 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 cái 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 chứ 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 filename 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 này 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 này 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 vào lstNames ta lại nhớ hoán chuyển FirstName và SurName để chúng nằm đúng lại vị trí. Tức là, cái mảnh của ta là muốn biết Item nào phải nằm ở đâu trong lstNames, chứ dĩ nhiên khi display mỗi Item đều có FirstName 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 FirtName and SurName
        sFirstName = Left(anItem, 10)

```

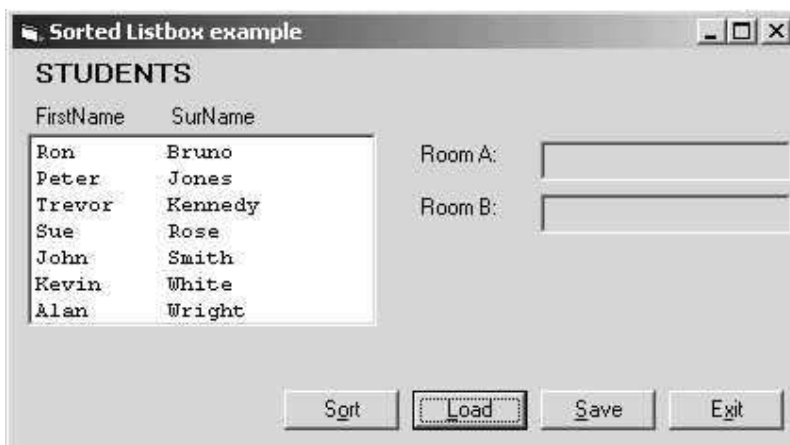


```

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 FstName 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 filename 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

```

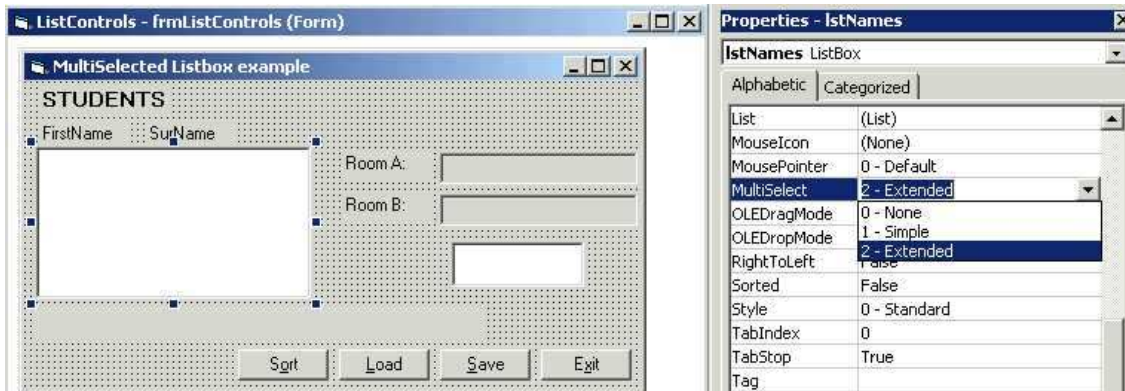
Trong bài tới ta sẽ học thêm các áp dụng khác của ListBox.

Listbox

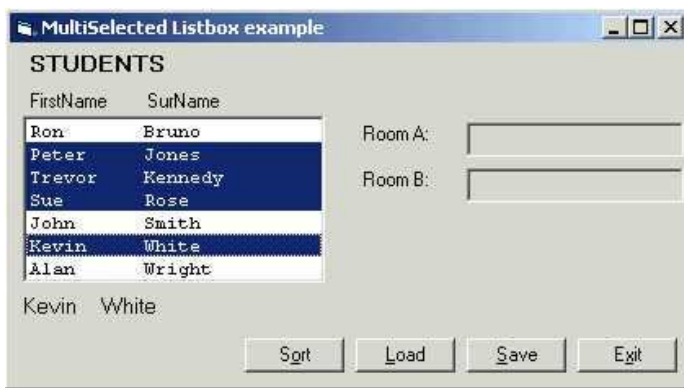
Cách dùng MultiSelect

Cho đến giờ User click vào ListBox để chọn chỉ một Item. Khi một Item được chọn thì hàng ấy trở nên highlighted với background màu xanh đậm. Nếu kế đó ta click một hàng khác thì hàng cũ được display trở lại bình thường và hàng mới được selected sẽ trở nên highlighted.

Listbox cho ta có thể select nhiều Items cùng một lúc bằng cách set **Property MultiSelect = Extended**



Đối với MultiSelected Listbox, ta chọn một nhóm Items liên tục bằng cách click Item đầu rồi nhấn nút **Shift** trong khi click Item cuối. Ta cũng có thể tiếp tục Select/Deselect thêm bằng cách ấn nút **Ctrl** trong khi click các Items. Nếu ta click một Item chưa được selected thì nó sẽ trở nên selected (highlighted màu xanh), nếu ta click một Item đã được selected rồi thì nó sẽ trở nên deselected (không còn màu xanh nữa). Thí dụ trong program bạn click "Peter Jones", kế đó ấn nút Shift trong khi click "Sue Rose", kế đó buông nút Shift ra để ấn nút Ctrl trong khi click "Kevin White", bạn sẽ có những selected Items như trong hình dưới đây:



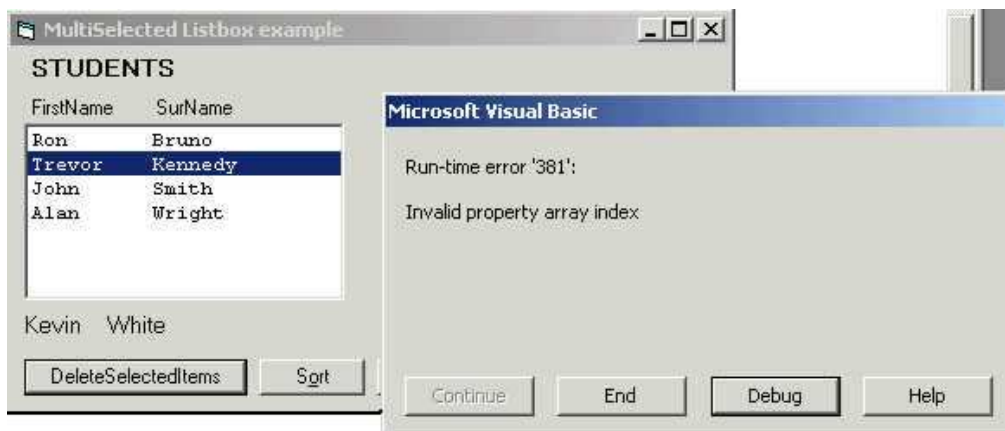
Ngoài ra bạn cũng có thể MultiSelect nhiều Items trong một Listbox bằng cách dùng mouse để drag, tức là bạn click lên Item đầu rồi tiếp tục đè mousebutton trong khi kéo mousepointer đến Item cuối cùng mới buông mousebutton ra.

Cái Bug ác ôn

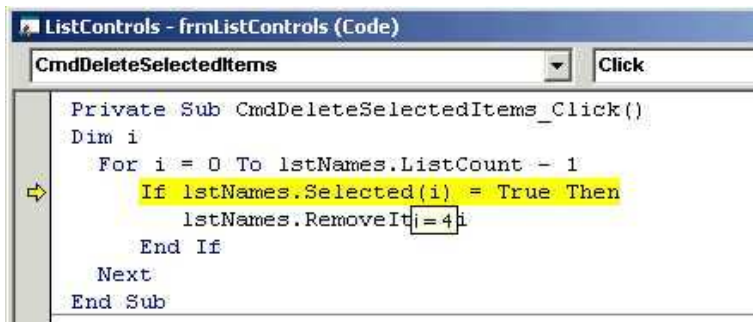
Bây giờ giả sử ta muốn delete tất cả những Items vừa được selected (highlighted). Bạn hãy đặt một CommandButton mới tên CmdDeleteSelectedItems vào Form. Ta sẽ dùng Event Click của Button này để delete những selected Items. Một selected Item của lstNames sẽ có property Selected của nó bằng True. Tức là nếu Item thứ ba (ListIndex=2) được selected thì ta có **lstNames.Selected(2) = True**. Ta có ý định sẽ iterate through mọi Items của lstNames, để xem Item nào được selected thì mình sẽ delete nó bằng cách dùng method RemoveItem. Ta sẽ viết code cho Sub CmdDeleteSelectedItems_Click() như sau:

```
Private Sub CmdDeleteSelectedItems_Click()
    Dim i
    For i = 0 To lstNames.ListCount - 1
        If lstNames.Selected(i) = True Then
            lstNames.RemoveItem i
        End If
    Next
End Sub
```

Bạn hãy chạy chương trình, click Load để populate lstNames với các tên đọc từ text file, rồi MultiSelect các tên như trong hình phía trên. Kế đó click button DeleteSelectedItems. Program sẽ té (crash) và có hình như sau:



Nếu bạn click nút Debug, program sẽ ngừng tại dòng code gặp error và highlight nó với background màu vàng. Để mousepointer lên trên chữ i của lstNames.Selected(i), VB6 sẽ popup message nhỏ nhỏ **i = 4**.



Bạn để ý thấy trong hình lúc này lstNames chỉ còn có 4 Items (Ron, Trevor, John và Alan), vì các Items kia đã bị removed.

Bạn có biết tại sao program crashed không? Đó là vì program đang refer đến property Selected của Item thứ năm (ArrayIndex i = 4) của lstNames trong khi lstNames bây giờ chỉ còn có 4 Items. Vì vậy program crashed với message **"Runtime error '381': Invalid property array index"**.

Thủ phạm của cái Bug ác ôn này là statement **For i = 0 To lstNames.ListCount - 1**. VB6 chỉ tính value của **lstNames.ListCount - 1** một lần lúc khởi sự For..Loop mà thôi (tức là lstNames.ListCount - 1 = 6), nó không lưu ý là ListCount giảm value mỗi lần một Item bị Removed. Ngoài ra ta thấy tên "Trevor Kennedy" cũng không bị removed, tức là nó bị lọt số nếu ta dùng For..Loop theo cách này. Lý do là sau khi ta Remove "Peter Jones" (Item thứ hai), "Trevor Kennedy" bị đẩy lên và trở thành Item thứ hai mới. Kể đó ta increment value của i thành 2 rồi process Item thứ ba, tức là "Sue Rose", nên "Trevor Kennedy" không hề được processed.

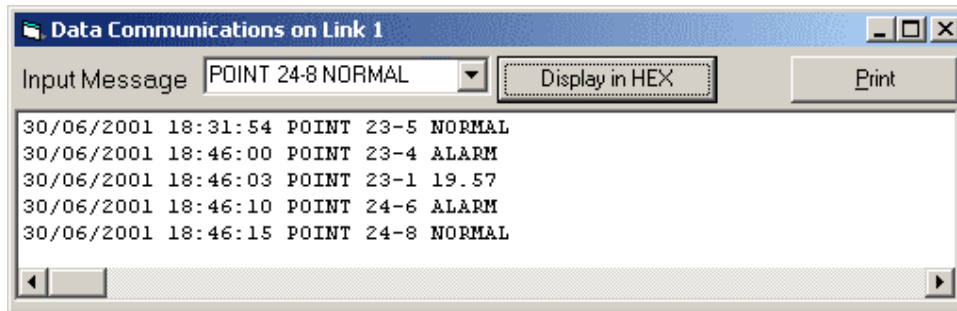
Sub CmdDeleteSelectedItems_Click cần phải được viết lại để dùng While ... Loop, thay vì For...Loop. Trong While...Loop, **lstNames.ListCount - 1** được evaluated (tính) để test ở mỗi iteration. Khi nào ta Remove một Item thì ta không increment i, vì Item ngay dưới removed Item được đẩy lên. Listing mới như sau:

```
Private Sub CmdDeleteSelectedItems_Click()  
    Dim i  
    i = 0 ' Initialise value of i to start from first Item  
    ' Note that lstNames.ListCount is evaluated freshly at each iteration  
    Do While i <= (lstNames.ListCount - 1)  
        If lstNames.Selected(i) = True Then  
            lstNames.RemoveItem i  
            ' No need to increment i here because the item below is pushed up  
        Else  
            i = i + 1 ' increment i to process the next item  
        End If  
    Loop  
End Sub
```

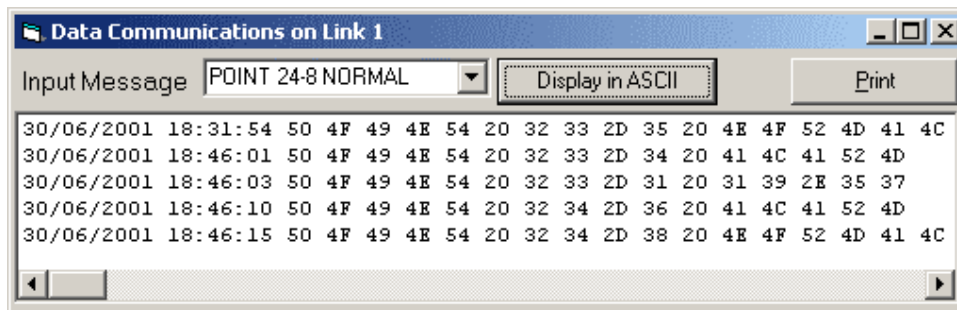
Dùng Listbox để display Event Log

Trong thí dụ sau đây ta muốn display input từ một serial COM port (Để đọc data thật qua serial COM port ta phải dùng control MsCOMM có hình telephone màu vàng trong ToolBox). Nhưng để tiện việc biểu diễn, thay vì đọc một message từ một serial COM port, ta sẽ emulate nó bằng cách dùng một ComboBox. Khi user select một hàng từ ComboBox cboInput thì ta xem cboInput.text như data đọc từ serial COM port. Lập tức lúc ấy ta sẽ display input message trong hai dạng: ASCII và HEX. Mỗi hàng input message được prefix với ngày và giờ trước khi được cho vào hai Listboxes lstASCII và lstHexadecimal.

Hình dưới đây cho thấy Form đang display trong ASCII mode.



Nếu bạn click button **Display in HEX** thì caption của button đổi thành **Display in ASCII**, lstASCII trở nên vô hình và lstHexadecimal sẽ hiện ra như sau:



Dưới đây là listing của Function HexDisplay để convert từ ASCII string ra Hexadecimal string.

```
Function HexDisplay(InASCII) As String
    ' Convert an ASCII string to HEX string
    Dim InLen, i, msg, HexStr
    InLen = Len(InASCII) ' Get length of input string
    ' Convert each ASCII character to Hex
    For i = 1 To InLen
        HexStr = Hex(Asc(Mid(InASCII, i, 1)))
        ' If HEX has only one digit then prefix it with 0
        If Len(HexStr) = 1 Then HexStr = "0" & HexStr
        msg = msg + HexStr & " "
    Next i
    HexDisplay = msg ' Return result string for Function
End Function
```

Trong program này, khi Listbox đạt đến 1000 items thì mỗi lần một hàng mới được thêm vào, hàng cũ nhất sẽ bị removed. Để cho hàng mới nhất không bị dấu ta phải nhớ cho ListIndex của Listbox bằng

Listcount-1 để Listbox tự động scrollup và highlight hàng cuối.

Mỗi khi ta thêm một hàng vào Listbox lstHexadecimal, ta cũng đồng thời viết nó vào một LogFile.

Tên của LogFile này dựa vào ngày lấy từ Computer System và có dạng như **Hex30Jun01.log**. Tức là ta sẽ dùng một LogFile khác cho mỗi ngày. Mỗi khi qua ngày mới, program tự động dùng một LogFile mới. Nhớ là khi muốn viết vào một text file theo tên gì đó, nếu file chưa hiện hữu thì ta phải create nó và viết vào, nếu file đã hiện hữu rồi ta chỉ cần append hàng mới vào cuối file (phải cẩn thận chỗ này, vì nếu không, ta vô ý overwrite cái file và mất hết những gì nó chứa trước đây).

```
Sub DisplayInHEX(inString)
    Dim Mess, LogFileName
    ' Convert ASCII to Hex
    Mess = HexDisplay(inString)
    ' Prefix with date and time and add it to the bottom of Listbox
    lstHexadecimal.AddItem Format(Now, "dd/mm/yyyy hh:nn:ss") & " " & Mess
    ' Keep only the latest 1000 events
    If lstHexadecimal.ListCount >= 1000 Then
        ' Remove the first Item, i.e. the oldest item
        lstHexadecimal.RemoveItem 0
    End If
    ' Highlight the latest item in the Listbox
    lstHexadecimal.ListIndex = lstHexadecimal.ListCount - 1
    ' Use different log file each day. Filename has format like Hex15Jun01.log
    LogFileName = "Hex" & Format(Now, "ddmmmyy") & ".log"
    ' Log to file including Date and Time
    LogEvent LogFileName, Mess, False, 2
End Sub
```

In ra content của Listbox

Dưới đây là một áp dụng của Listbox MultiSelect để in ra cả Listbox hay chỉ những hàng được selected. Sub PrintList nhận:

- Listbox mà ta muốn in
- một Boolean value mà nếu True thì in cả Listbox
- Title của Printout

```
Sub PrintList(theList As ListBox, PrintAll as Boolean, Title As String)
    ' Print the whole lot or only selected lines in a listbox
    ' PrintAll = True means printing the whole content of the listbox
    Const MaxLinesPerPage = 50
    Dim msg, i, j, PageNo, NumLines, HasSome, Margin
    HasSome = False ' Flag indicating existence of data
    Margin = Space(10) ' Make a margin of 5 characters
    Title = vbLf & vbLf & Title + vbCrLf & vbLf
```

```

NumLines = 0 ' Init number of lines on this page
PageNo = 1 ' init Page number
msg = Title ' Msg will contain everything starting with Title
Printer.FontName = "Courier New" ' Initialise Printer Fontname
Printer.FontSize = 10 ' Initialise Printer FontSize
Screen.MousePointer = vbHourglass ' Change mousepointer shape to Hourglass.
If theList.ListCount > 0 Then
    ' get here if the listbox is not empty
    For i = 0 To theList.ListCount - 1 ' Go thru each line of text in the listbox
        If theList.Selected(i) Or PrintAll Then
            ' print a line of text if it's selected or PrinAll is true
            DoEvents ' Let other processes have a chance to run
            HasSome = True
            NumLines = NumLines + 1 ' Increment count of lines
            If Left(theList.List(i), 1) = "" Then
                ' if first character is "" then use this as an indication to force a new page
                If NumLines > 0 Then
                    ' Add extra blank lines to make up a page before inserting page number
                    For j = NumLines - 1 To MaxLinesPerPage
                        msg = msg & vbCrLf
                    Next j
                    ' Insert Page number at end of page
                    msg = msg & Space$(35) & "Page-" & CStr(PageNo)
                    Printer.Print msg
                    Printer.NewPage ' Send new page.
                    NumLines = 1 ' reset Number of lines, counting this current line
                    PageNo = PageNo + 1 ' Increment Page number
                    msg = Title ' Reset Msg to contain Title for new page
                    ' Append this current line, ignoring character ""
                    msg = msg & Margin & Mid(theList.List(i), 2) & vbCrLf
                Else
                    ' Blank page so far - so just appending this line, ignoring character ""
                    msg = msg & Margin & Mid(theList.List(i), 2) & vbCrLf
                End If
            Else
                ' Normal line - just keep appending it to Msg
                msg = msg + Margin & theList.List(i) & vbCrLf
            End If
            theList.Selected(i) = False ' Clear highlight of selected line, ie. deselect it
        If NumLines > MaxLinesPerPage Then ' Start new page if page already full
            If PageNo > 1 Then ' Insert page number at the bottom, except for first page

```



```

        msg = msg + vbCrLf & Space$(35) & "Page-" & CStr(PageNo)
    End If
    Printer.Print msg ' Output all data of this page
    Printer.NewPage ' Send new page.
    NumLines = 0
    PageNo = PageNo + 1
    msg = Title
    End If
End If
Next i
End If
' Get here after going thru all lines in the listbox
If NumLines > 0 Then ' complete the last page by inserting page number
    For i = NumLines To MaxLinesPerPage
        msg = msg & vbCrLf
    Next i
    If PageNo > 1 Then
        msg = msg + vbCrLf & Space$(35) & "Page-" & Str$(PageNo)
    End If
    Printer.Print msg ' Output all data of this page
End If
If HasSome Then
    Printer.EndDoc ' Initiate the actual Print.
Else
    Beep
    MsgBox "Nothing to print, try selecting a range of lines first"
End If
Screen.MousePointer = vbDefault ' Change mousepointer shape back to normal
End Sub

```

Ta gọi PrintList để in những Items đã được selected trong Listbox lstNames như sau:

```

Private Sub CmdPrint_Click()
    PrintList lstHexadecimal, True, "*** EVENT LOG IN HEX ***"
End Sub

```

Thêm Horizontal Scrollbar vào Listbox

Có lẽ bạn để ý thấy cả hai Listboxes lstASCII và lstHexadecimal đều có Horizontal Scrollbar phía dưới. By default, Listbox không có Horizontal Scrollbar. Muốn tạo ra nó bạn phải thêm hai câu dưới đây vào một Basic module:

```

Public Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd
As Long, ByVal wParam As Long, ByVal lParam As Any) As Long
Global Const LB_SETHORIZONTALEXTENT = &H194

```


Kế đó trong Sub Form_Load gọi Function SendMessage qua Application Programming Interface (API) để yêu cầu Listbox cho hiện ra Horizontal Scrollbar.

```
Dim VLong As Long
```

```
' make a horizontal scrollbar for both Listboxes
```

```
VLong = SendMessage(lstAscii.hwnd, LB_SETHORIZONTALEXTENT, lstAscii.Width, ByVal 0)
```

```
VLong = SendMessage(lstHexadecimal.hwnd, LB_SETHORIZONTALEXTENT, lstHexadecimal.Width, ByVal 0)
```

Bạn có thể [download source code của program Eventlog.zip này](#) để có đầy đủ.

Trong bài tới ta sẽ học thêm các áp dụng còn lại của ListBox.

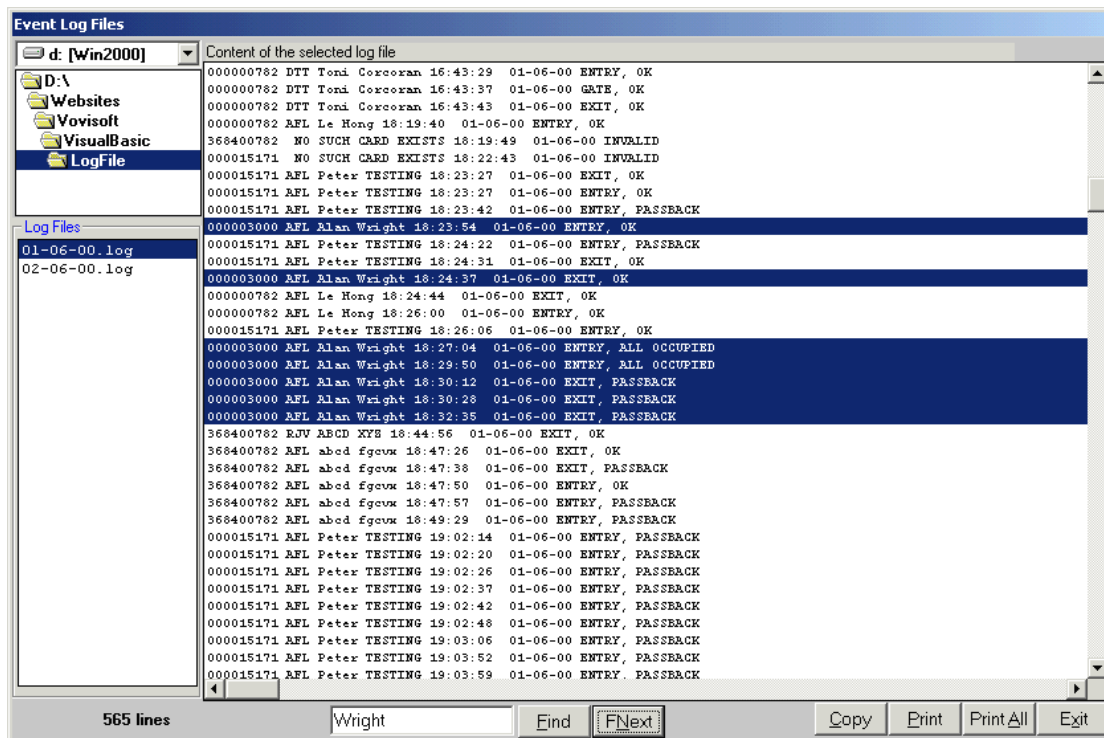
Listbox

Search trong Text File

Ta biết rằng ListBox có thể chứa rất nhiều hàng text (con số hàng tối đa là 65535). Ta đã quen với việc hiển thị content của một text file trong một ListBox. Ta đã dùng ListBox để display các Events (sự cố) xảy ra trong real-time. Giả dụ, ta ghi lại tất cả mọi Events xảy ra trong real-time của một hệ thống an ninh, tức là ta biết ai ra, vào cửa nào, lúc mấy giờ. Các Events này vừa được log xuống một Text file, vừa được cho vào một ListBox để luôn luôn hiển thị Event mới nhất ở cuối ListBox.

Khi đã có mọi Events nằm trong ListBox, ta có thể Search (tìm kiếm) xem một người nào đã đi qua những cửa nào của building bằng cách iterate qua từng hàng trong ListBox và nhận diện một Text Pattern hàng với **Function InStr**.

Trong bài mẫu dưới đây, ta đánh tên của một người vào trong TextBox rồi click nút **Find** và sau đó **Find Next** để highlight những Events trong ListBox cho thấy những lúc tên người đó xuất hiện. Trong khi tìm kiếm một Text Pattern ta có thể cho phép cả chữ Hoa, lẫn chữ Thường bằng cách covert mọi text ra Uppercase trước khi làm việc với chúng.



Listing của Sub Find_Click như sau:

```
Private Sub CmdFind_Click()
    Dim i, ALine, FText
    ' Get out if the Listbox is empty
    If EventList.ListCount = 0 Then
        MsgBox "There 's no text available"
        Exit Sub
    End If
    ' Check if user has entered the Text Pattern
    If Trim(txtFind) = "" Then
        MsgBox "Please enter the Text Pattern to search for"
        Exit Sub
    End If
    ' Clear all selected lines
    For i = 0 To EventList.ListCount - 1
        EventList.Selected(i) = False
    Next
    ' Convert the Text Pattern to Uppercase
    FText = UCase(txtFind.Text)
    ' Iterate through every line in the ListBox
    For i = 0 To EventList.ListCount - 1
        ' Convert this line to Uppercase
        ALine = UCase(EventList.List(i))
        ' If pattern exists in this line then highlight it
        If InStr(ALine, FText) > 0 Then
            EventList.Selected(i) = True ' Highlight the line
            ' Mark Current line as the Starting line for FindNext operation
            If i < EventList.ListCount - 1 Then CurrentLine = i + 1
            ' get out
            Exit Sub
        End If
    Next
    ' Only get here if Not found
    MsgBox "Not found!"
End Sub
```

Trong bài này ta có dùng một **DriveListBox** để cho User chọn một Disk drive, một **DirListBox** để user chọn một Folder/Directory và một **FileListBox** để hiển thị tên của những Files trong một Folder. Cả ba loại ListBoxes này liên kết nhau để cho ta thấy sự thay đổi ăn nhịp mỗi khi User đổi từ Disk Drive này qua Disk Drive khác, hay từ Folder này qua Folder khác. Các hàng codes thực hiện việc này rất đơn giản như sau:

```
Private Sub Drive1_Change()
```

```

' Make Path of Folder same as new Drive
Dir1.Path = Drive1.Drive
End Sub
Private Sub Dir1_Change()
' Make Path of FileList same as new Path of Folder
' The filenames in the Folder will be displayed automatically in FileListBox
FileList.Path = Dir1.Path
End Sub

```

Ta có thể chọn lùa chọn những Filenames có một Extension nào đó (thứ d như **log**) bằng cách cho **Property Pattern** của FileListBox value **"*.log"**.

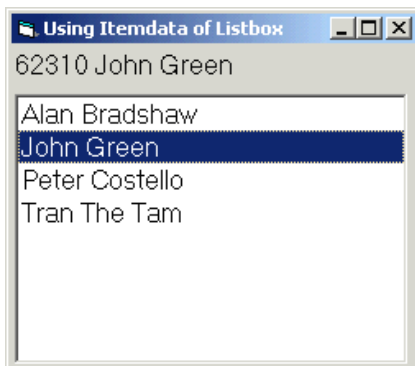
Mỗi khi User click lên tên của một File, program sẽ load content của File ấy vào ListBox **EventList** bên phải.

Sau khi selected một số hàng rồi, User có thể hoặc Print chèn ra bằng cách Click nút **Print**, hoặc Copy chèn vào Clipboard bằng cách Click nút **Copy**.

Dùng ItemData

Nếu **Property List** của ListBox được xem như một Text Array thì **ItemData** là một Number Array, và **List1.ItemData(i)** đi cặp với **List1.List(i)**. Tức là trong khi List1.List(i) hiển thị như mặt trước của một tấm bản thì List1.ItemData(i) được coi như nằm ở mặt sau của tấm bản ấy. Khi một List item thay đổi vị trí trong Listbox vì có sự biến đổi trong ListBox (thí dụ Items bị removed hay được cho thêm vào) thì ItemData của List item đó cũng đi theo với nó.

Ta thử xem thí dụ sau. Cho vào một **Sorted** Listbox tên của các nhân viên trong sở. Ngay sau khi tên một nhân viên được cho vào Listbox thì Property **NewIndex** chứa vị trí của item mới được cho vào ấy trong ListBox. Ta dùng giá trị NewIndex để assign ItemData với Số nhân viên ID. Khi User clicks lên một tên trong Listbox, program sẽ hiển thị cả Số nhân viên ID lẫn tên nhân viên. Để thử thí dụ này, bạn có thể Paste phần code dưới đây vào phần Declaration của một Form có chứa một Listbox và một Label. Nhớ set property Sorted của List1 ra True.



```

Private Sub Form_Load()
' Fill List1 and ItemData array with
' corresponding items in sorted order.
List1.AddItem "John Green" ' Add an employee name
' Use NewIndex to synchronise with Employee ID
' Assign Employee ID to ItemData of the List Item

```

```

List1.ItemData(List1.NewIndex) = 62310
List1.AddItem "Tran The Tam"
List1.ItemData(List1.NewIndex) = 42859
List1.AddItem "Alan Bradshaw"
List1.ItemData(List1.NewIndex) = 63732
List1.AddItem "Peter Costello"
List1.ItemData(List1.NewIndex) = 34127
End Sub

Private Sub List1_Click()
    ' Fetch the employee number
    Msg = List1.ItemData(List1.ListIndex) & " "
    ' Concatenate it with the employee name.
    Msg = Msg & List1.List(List1.ListIndex)
    ' Assign string to Label to display
    Label1.Caption = Msg
End Sub

```

Dùng ListBox làm Queue

Khi đi coi hát, ta thường phải đứng sắp hàng để mua vé. Cái hàng đó gọi là **Queue**. Mục đích của việc dùng Queue là để cho số người đồng cần một dịch vụ sẽ được phục vụ lần lượt theo thứ tự ai đến trước sẽ được giải quyết trước. Nguyên tắc của Queue như thế được gọi là **First-In-First-Out** (vào trước nhất, ra trước nhất). Ngược lại, nếu ai cũng muốn được phục vụ trước nhất ta sẽ có sự náo loạn, và rất cuộc có thể chẳng có ai được giải quyết.

Thí dụ ta có một Form tên frmServer, mà trong đó có một Listbox tên List1. Nếu có nhiều Forms khác trong cùng một chương trình muốn nhờ frmServer phục vụ một chuyện gì, chúng sẽ Queue bằng cách Add một Item vào cuối List1. Trong Item có chứa những chi tiết mà frmServer sẽ cần biết để phục vụ.

```

Private Sub CmdAddToQueue_Click()
    Dim myRequest As String
    Dim PersonId As String * 5
    Dim PersonName As String * 20
    ' Assign PersonId to fixed length text
    PersonId = txtPersonId.Text
    ' Assign PersonName to fixed length text
    PersonName = txtPersonName.Text
    ' Concatenate Id and Name
    myRequest = PersonId & PersonName
    ' Queue the request
    frmServer.List1.AddItem myRequest
End Sub

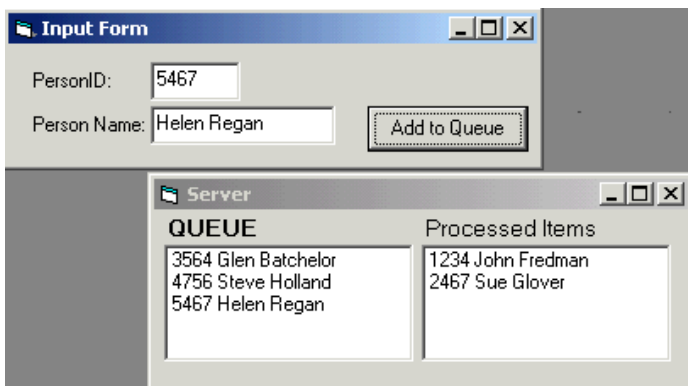
```

Bên frmServer, cứ mỗi 3 giây nó sẽ Remove Item trên hết (tức là Index=0) trong List1 và xử lý Item ấy. Trong bài này ta chỉ Remove Item 0 rồi Add nó vào List2.

```

Private Sub Timer1_Timer()
    Dim Item
    If List1.ListCount > 0 Then
        ' Look at the item at the head of the queue
        Item = List1.List(0)
        ' Process Item - just add it to List2 here
        List2.AddItem Item
        ' Remove item from queue
        List1.RemoveItem 0
    End If
End Sub

```



CheckBox Listbox

Nếu bạn chọn value của Property **Style** của Listbox là **CheckBox** thay vì **Standard** thì mọi items trong Listbox sẽ có một hộp vuông phía trước để User có thể chọn lúc chạy program. Hộp vuông của item nào được checked (đánh dấu) thì Item ấy được Selected.

Giả sử ta có một Listbox List1 với Style Checkbox và có nhiều Items để mua trong siêu thị. Khi chạy program user chọn một số items rồi click nút **Process**, program sẽ hiển thị các item đã được chọn.

Listing của **Sub CmdProcess_Click** như sau:

```

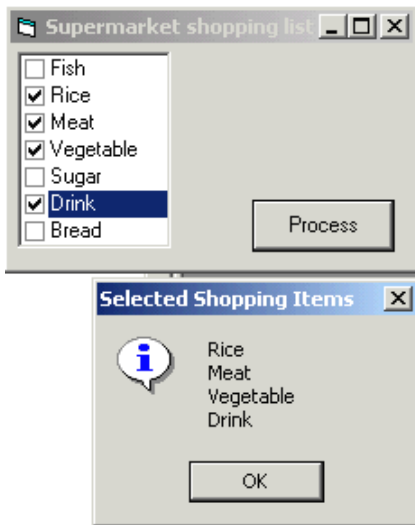
Private Sub CmdProcess_Click()
    Dim Mess As String
    ' get out if there's nothing in the list
    If List1.ListCount = 0 Then Exit Sub
    ' Iterate through every item of the checkBox Listbox
    For i = 0 To List1.ListCount - 1
        ' If item is selected then include it in the shopping list
        If List1.Selected(i) Then
            ' Append Item and a Carriage Return-LineFeed
            Mess = Mess & List1.List(i) & vbCrLf
        End If
    Next

```

```
' Display shopping list
```

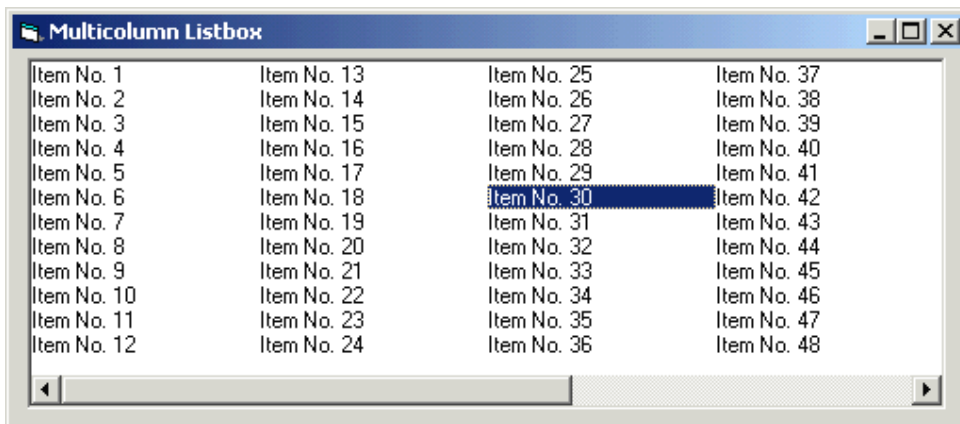
```
MsgBox Mess, vbInformation, "Selected Shopping Items"
```

```
End Sub
```



Listbox với nhiều cột

Listbox có một Property gọi là Columns. Bình thường, Property Columns có giá trị 0. Nhưng nếu bạn cho nó bằng 3 chẳng hạn, thì Listbox sẽ cố gắng hiển thị 4 cột trong phạm vi chiều ngang của Listbox. Nếu như thế vẫn không đủ hiển thị hết mọi Items trong Listbox thì sẽ có một Horizontal Scrollbar hiện ra, và khi bạn click nó qua bên phải Listbox sẽ cho hiển thị thêm các columns còn lại.



Combobox

Combobox rất giống như Listbox. Nó là tập hợp của một Textbox nằm phía trên để User cho vào data và một Listbox chứa các items mà User có thể lựa chọn khi mở nó ra. Combo box cũng có những methods như **Clear**, **AddItem** và **RemoveItem**. Tuy nhiên, Combobox không có **Property Selected**, vì khi User chọn Item nào thì Item ấy được hiển thị trong Textbox phía trên.

Combobox có 3 styles. **Drop-down Combo** Style là thông dụng nhất. Nó cho User nhiệm ý hoặc chọn một Item từ List hoặc đánh data vào Textbox. Trong hình dưới đây User đánh vào chữ Elephant thay vì chọn từ các Items có sẵn.



Drop-down List bắt buộc User phải chọn một trong những Item nằm trong List, chứ không được đánh data mới vào Textbox. Ngay cả trong lúc chạy program (at run-time) bạn cũng không thể Assign một value vào property Text của Combobox loại này. Nhưng bạn có thể làm cho Combobox hiển thị Item thứ 3 chẳng hạn bằng cách set property ListIndex của Combo bằng 2

Chương Tám - Tự tạo Object

Từ trước đến giờ, ta lập trình VB6 bằng cách thiết kế các Forms rồi viết codes để xử lý các Events của những controls trên Form khi Users click một Button hay Listbox, .v.v..

Nói chung, cách ấy cũng hữu hiệu để triển khai, nhưng nếu ta có thể hưởng được các lợi ích sau đây thì càng tốt hơn nữa:

1. Dùng lại được code đã viết trước đây trong một dự án khác
2. Dễ nhận diện được một lỗi (error) phát xuất từ đâu
3. Dễ triển khai một dự án lớn bằng cách phân phối ra thành nhiều dự án nhỏ
4. Dễ bảo trì

Mỗi lần dùng lại code, nếu để ý nguyên xi con là lý tưởng. Việc ấy được gọi là **Reusability**. Nói cho đúng ra, **dùng lại được** thật sự là khi ta chỉ cần dùng **object code**, đó là code đã được **compiled** rồi, tức là hoàn toàn không đụng đến **source code**. Vì hệ cho phép User sửa source code là tạo cơ hội cho **bugs** xuất hiện, rồi lại phải debug một lần nữa.

Sự thách đố chính của việc triển khai một dự án phần mềm lớn là thực hiện đúng thời hạn (on time), không lố tài khóa (within budget) và dễ bảo trì (ease of maintenance). Muốn đạt được các mục tiêu ấy, ta phải triển khai nhanh và làm sao cho chương trình ít có bugs, dễ bảo trì.

Giả dụ bạn đứng ra tổ chức một đám cưới. Thử tưởng tượng biết bao nhiêu chuyện phải làm: từ danh sách quan khách, thiệp mời, ẩm thực, xe cộ, chụp hình, quay phim, văn nghệ cho đến thủ tục nghi lễ, tiếp tân, hoạt náo viên ..v.v.. Nếu chỉ một mình bạn lo thật không biết làm sao nhớ cho hết. Cũng may là nhà hàng sẽ đảm trách luôn cả việc in ấn thiệp mời, ban nhạc văn nghệ và cả hoạt náo viên. Thủ tục nghi lễ thì không ai qua được bác Sáu Đạt, và bác đã nhận lời mua quà cáp, lo về tiếp tân, xe cộ và thủ tục, nghi lễ. Bác cũng sẽ liên lạc với Mục sư chủ lễ để dặn chỗ nhà thờ và sắp đặt người giựt chuông và người đàn. Anh Tư Thông có người bạn làm chủ tiệm hình, nên anh nhận trách nhiệm mượn người lo chụp hình, quay phim. Như thế việc bạn tổ chức cái đám cưới nay rút lại chỉ còn soạn danh sách quan khách, các bài diễn văn, sắp chỗ ngồi và dặn chỗ cho cặp vợ chồng mới đi hưởng tuần trăng mật.

Sở dĩ bạn cảm thấy trách nhiệm tổ chức không nặng nề vì nhà hàng, bác Sáu Đạt và anh Tư Thông tự lo gánh vác các khâu rắc rối. Cái hay ở đây là những người này tự lo quyết định mọi chi tiết của những gì cần phải làm trong khâu của họ. Chỉ khi nào cần lắm, họ mới liên lạc để lấy ý kiến của bạn. Họ giống như những người **thần** của bạn. Chắc bạn đã lưu ý rằng cái thí dụ tổ chức đám cưới này cho thấy nói chung muốn triển khai dự án lớn nào ta cần phải nhờ những người thần giúp đỡ. Quả thật, đó là cách các quản trị viên những công trình đã làm từ xưa đến nay.

Bây giờ trở lại chuyện lập trình, phải chi ta có thể tổ chức cách triển khai dự án phần mềm giống như tổ chức cái đám cưới nói trên thì tốt quá. Thật ra, không phải các lý thuyết gia phần mềm không nghĩ

đến chuyện ấy trước đây, nhưng để thực hiện được việc ấy người ta cần triển khai các phương tiện, dụng cụ thích hợp. Chỉ trong vòng 15 năm trở lại đây, việc ấy mới trở nên cụ thể qua các Operating Systems tinh vi, nhất là dùng Windows, và các ngôn ngữ lập trình như Eiffel, SmallTalk, C++ .v.v..

Lập trình theo hướng đối tượng (Object Oriented Programming)

Nói một cách nôm na, lập trình theo hướng đối tượng là thiết kế các bộ phận phần mềm của chương trình, gọi là **Objects** sao cho mỗi bộ phận có thể tự lo liệu công tác của nó giống như một người **thần** ngoài đời vậy. Chắc có lẽ bạn sẽ hỏi thế thì các **Sub** hay **Function** mà bạn đã từng viết để xử lý từng giai đoạn trong chương trình có thể đảm trách vai trò của một thần không?

Người thần chẳng những có thể làm được công tác (Subs và Functions) gì mà còn chịu trách nhiệm luôn cả mọi thứ vật dụng cần thiết (data) cho việc ấy nữa.

Có một cách định nghĩa khác cho Object là một Object gồm có data structure và các Subs/Functions làm việc trên các data ấy. Thông thường, khi ta dùng Objects ít khi giám thị chúng, ngược lại nếu khi có sự cố gì thì ta muốn chúng báo cáo cho ta biết.

Trong VB6, các Forms, Controls hay ActiveX là những Objects mà ta vẫn dùng lâu nay. Lấy thí dụ như Listbox. Một Listbox tự quản lý các items hiển thị bên trong nó. Ta biết listbox List1 đang có bao nhiêu items bằng cách hỏi List1.ListCount. Ta biết item nào vừa mới được selected bằng cách hỏi List1.ListIndex. Ta thêm một item vào listbox bằng cách gọi method AddItem của List1, ..v.v.. Nói cho đúng ra, Object là một thực thể của một Class. Nếu Listbox là một Class thì List1, List2 là các thực thể của Listbox. Cũng giống như Bà Tư Cháo Lòng và Dì Sáu Bánh Tằm là các thực thể của Class Đầu Bếp.

Ngay cả một form tên frmMyForm mà ta viết trong VB6 chẳng hạn, nó cũng là một Class. Thường thường ta dùng thẳng frmMyForm như sau:

```
frmMyForm.Show
```

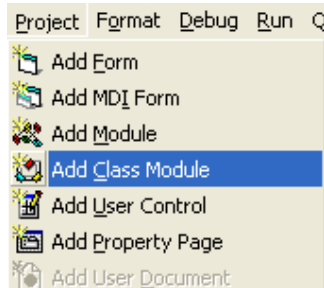
Trong trường hợp này thật ra frmMyForm tuy là một Class nhưng được dùng y như một Object. Chớ nếu muốn, ta có thể tạo ra hai, ba Objects của Class frmMyForm cùng một lúc như trong thí dụ sau:

```
Dim firstForm As frmMyForm
Dim secondForm As frmMyForm
Set firstForm = New frmMyForm
Set secondForm = New frmMyForm
firstForm.Show
secondForm.Show
```

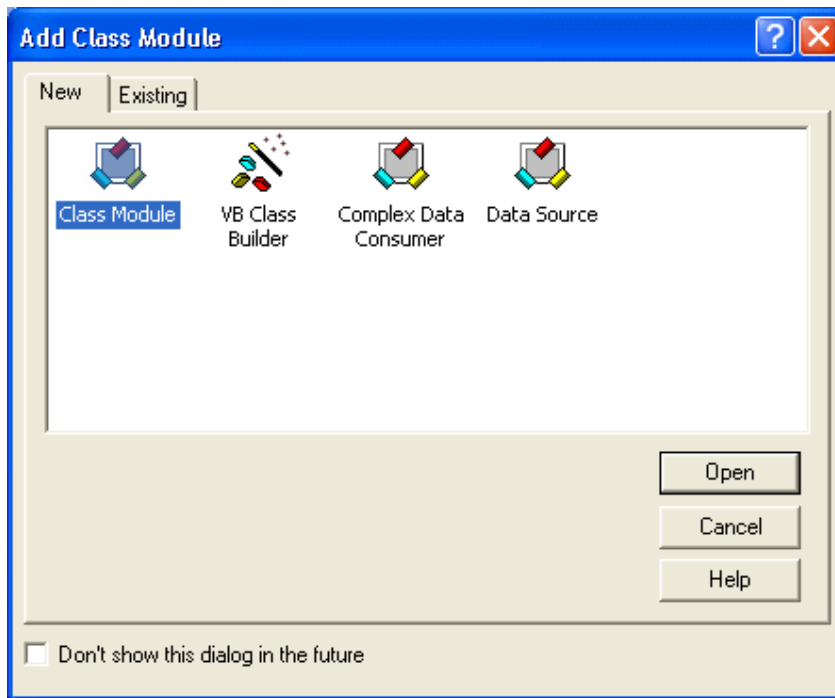
Trong thí dụ trên ta declare firstForm và secondForm là những Objects của Class frmMyForm. Sau đó ta làm nên (**instantiate**) các Objects firstForm và secondForm bằng statements **Set... = New...** firstForm và secondForm còn được gọi là các **instances** của Class frmMyForm. Class giống như cái khuôn, còn Objects giống như những cái bánh làm từ khuôn ấy. Chắc bạn đã để ý thấy trong VB6 từ dùng hai từ Class và Object lẫn lộn nhau. Điều này cũng không quan trọng, miễn là bạn nắm vững ý nghĩa của chúng.

VB6 có yểm trợ **Class** mà ta có thể triển khai và instantiate các Objects của nó khi dùng. Một Class trong VB6 có chứa **data** riêng của nó, có những **Subs** và **Functions** mà ta có thể gọi. Ngoài ra Class còn có thể Raise **Events**, tức là báo cho ta biết khi chuyện gì xảy ra bên trong nó. Cũng giống như Event Click của CommandButton, khi User clicks lên button thì nó Raise Event Click để cho ta xử lý trong Sub myCommandButton_Click(), chẳng hạn. Class trong VB6 không có hỗ trợ Visual components, tức là không có chứa những controls như TextBox, Label .v.v.. Tuy nhiên, ta có thể lấy những control có sẵn từ bên ngoài rồi đưa cho Object của Class dùng.

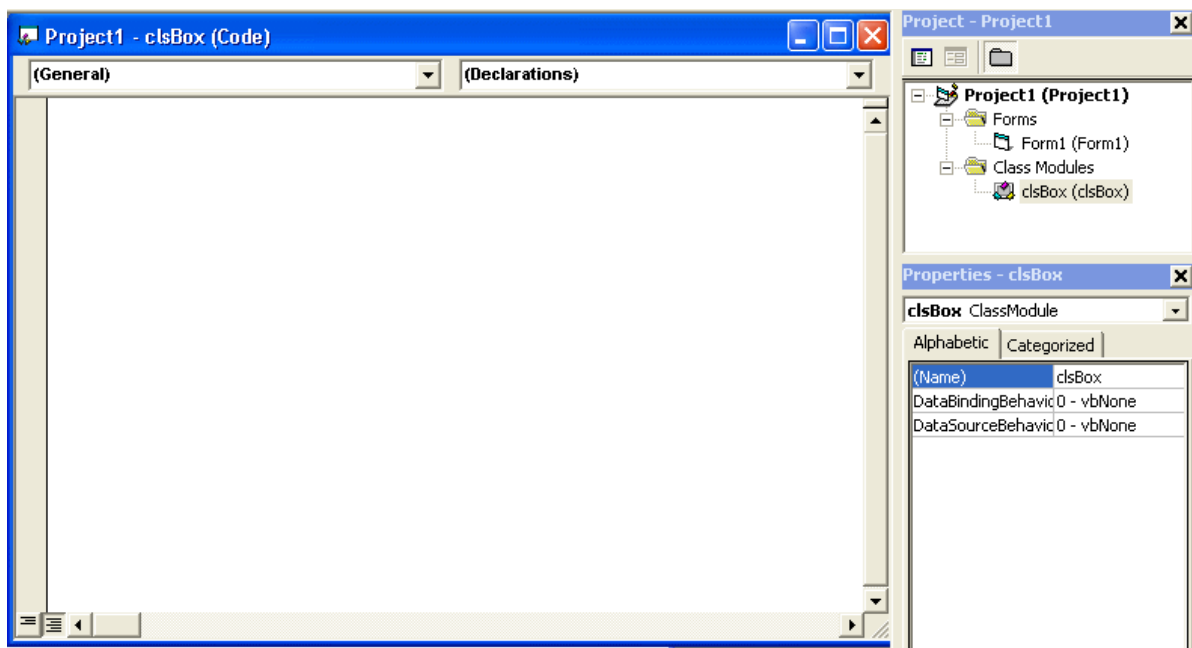
Bây giờ chúng ta hãy bắt đầu viết một Class. Bạn hãy mở một Project mới loại Standard EXE Visual Basic. Sau đó dùng Menu Command chọn **Add Class Module**:



Khi Add Class Module dialog hiện ra chọn **Class Module** và click Open.



Bạn sẽ thấy mở ra một khung trắng và Project Explorer với Properties Window. Trong Properties Window, hãy sửa Name property của Class thành clsBox như dưới đây:



Kế đó đánh vào những dòng code dưới đây, trong đó có biểu diễn cách dùng Class clsBox.

```
Option Explicit
Private mX As Integer
Private mY As Integer
Private mWidth As Integer
Private mHeight As Integer

Public Property Let X(ByVal vValue As Integer)
    mX = vValue
End Property

Public Property Get X() As Integer
    X = mX
End Property

Public Property Let Y(ByVal vValue As Integer)
    mY = vValue
End Property

Public Property Get Y() As Integer
    Y = mY
End Property

Public Property Let Width(ByVal vValue As Integer)
    mWidth = vValue
```

```

End Property

Public Property Get Width() As Integer
    Width = mWidth
End Property

Public Property Let Height(ByVal vValue As Integer)
    mHeight = vValue
End Property

Public Property Get Height() As Integer
    Height = mHeight
End Property

Public Sub DrawBox(Canvas As Object)
    Canvas.Line (mX, mY)-(mX + mWidth, mY + mHeight), , B
End Sub

Public Sub ClearBox(Canvas As Object)
    Canvas.Line (mX, mY)-(mX + mWidth, mY + mHeight), Canvas.BackColor, B
End Sub

```

Class clsBox có 4 Properties: X, Y, Width và Height. Ta sẽ instantiate một Box từ clsBox. Mỗi Box có tọa độ (X,Y) và kích thước chiều rộng và chiều cao (width, height) của nó. Thật ra ta có thể dùng Public statement để declare các biến X, Y, Width và Height. Nhưng ở đây ta cố ý declare chúng là Private, dưới dạng mX, mY, mWidth và mHeight. Khi ta muốn thay đổi các trị số của chúng, ta sẽ dùng cùng một cách viết code như bình thường (thí dụ: myBox.X = 80). Nhưng khi chương trình xử lý assignment statement ấy, nó sẽ execute một loại method (giống như Sub) gọi là **Property Let X (vValue)**. Ta thấy ở đây vValue được assigned cho mX (i.e. mX = vValue), cái Private variable của X. Như thế công việc này cũng chẳng khác gì sửa đổi một Public variable X. Tuy nhiên, ở đây ta có thể viết thêm code trong Property Let X để nó làm gì cũng được.

Bạn có nhớ trong khi thiết kế một Label, mỗi lần bạn dùng Property Window để edit Font size, forcolor hay backcolor thì chẳng những các properties ấy của Label thay đổi, mà kết quả của sự thay đổi được có hiệu lực ngay lập tức, nghĩa là Label được hiển thị trở lại với trị số mới của property. Đó là vì trong method Property có cả code bảo Label redisplay.

Ngược lại, khi ta dùng property X của Object myBox, không phải ta chỉ đọc trị số thôi mà còn execute cả cái method **Property Get X**. Nói tóm lại, Property cho ta cơ hội để execute một method mỗi khi User đọc hay viết trị số variable ấy.

Thí dụ như nếu ta muốn kiểm soát để chỉ chấp nhận trị số tọa độ X mới khi nó không phải là số âm. Ta sẽ sửa Property Let X lại như sau:

```
Public Property Let X(ByVal vValue As Integer)
    If (vValue >= 0) Then
        mX = vValue
    End If
End Property
```

Property có thể là **Read Only** hay **Write Only**. Nếu muốn một Property là Read Only thì ta không cung cấp Property Let. Nếu muốn một Property là Write Only thì ta không cung cấp Property Get. Ngoài ra nếu làm việc với **Object**, thay vì Data type thông thường, thì ta phải dùng **Property Set**, thay vì Property Let.

Thí dụ ta cho clsBox một Property mới, gọi là Font dùng object của class stdFont của VB6. Trong clsBox ta declare một Private variable mFont và viết một **Property Set Font** như sau:

```
Private mFont As StdFont
Public Property Set Font(ByVal newFont As StdFont)
    Set mFont = newFont
End Property
```

Ta sẽ dùng property Font của myBox (thuộc Class clsBox) như sau:

```
' Declare an object of Class StdFont of VB6
Dim myFont As StdFont
Set myFont = New StdFont
myFont.Name = "Arial"
myFont.Bold = True
Dim myBox As clsBox
Set myBox = New clsBox
Set myBox.Font = myFont ' Call the Property Set method
```

Class clsBox có hai Public Subs, **DrawBox** và **ClearBox**. ClearBox cũng vẽ một box như DrawBox, nhưng nó dùng BackColor của màn ảnh (canvas), nên coi như xóa cái box có sẵn. Do đó, nếu muốn, bạn có thể sửa Sub DrawBox lại một chút để nhận một Optional draw color như sau:

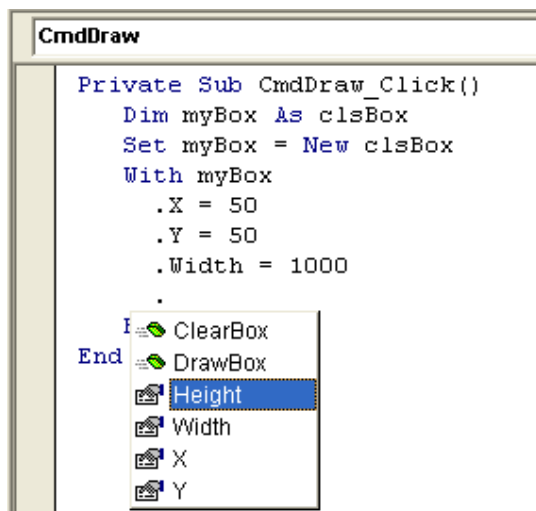
```

Public Sub DrawBox(Canvas As Object, Optional fColor As Long)
    If IsMissing(fColor) Then
        Canvas.Line (mX, mY)-(mX + mWidth, mY + mHeight), , B
    Else
        Canvas.Line (mX, mY)-(mX + mWidth, mY + mHeight), fColor, B
    End If
End Sub

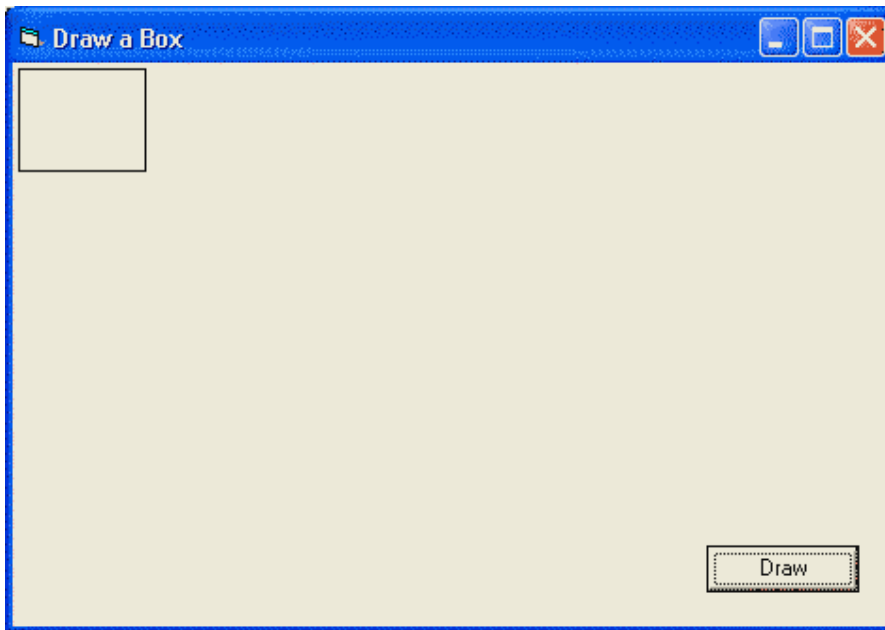
```

Trong thí dụ trên, Optional parameter fColor được tested bằng function **IsMissing**. Nếu fColor là BackColor của canvas thì ta sẽ có hiệu quả của ClearBox.

Trong form chính của chương trình dùng để test clsBox, mỗi khi ta refer đến một object thuộc class clsBox, IDE Intellisense sẽ hiển thị các Properties và Subs/Functions của clsBox như trong hình dưới đây:

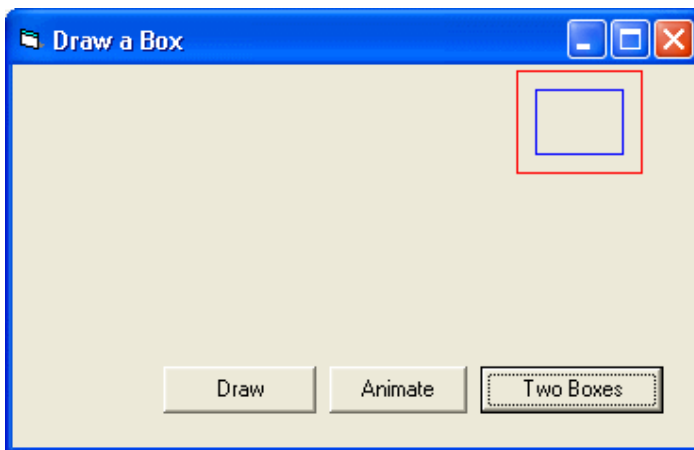


Trong chương trình này, mỗi khi ta click nút **Draw** thì một Box được instantiate, cho tọa độ X,Y và kích thước Width, Height, rồi được vẽ ra ngay trên form. Chữ **Me** trong code nói đến chính cái form **frmClass**.



Để cho chương trình thú vị hơn, khi user clicks nút **Animate**, ta sẽ cho một box màu đỏ chạy từ trái qua phải.

Khi user clicks nút **Two Boxes** ta sẽ vẽ hai boxes, hộp trong màu xanh, hộp ngoài màu đỏ, và cho chúng chạy từ trái sang phải. Ở đây ta biểu diễn cho thấy mình muốn instantiate bao nhiêu boxes từ clsBox cũng được, và dĩ nhiên mỗi box có một bộ properties với giá trị riêng của nó.



Ta có thể lập trình để cho Object báo cáo program chủ của nó khi có một biến cố (Event) xảy ra bên trong Class.

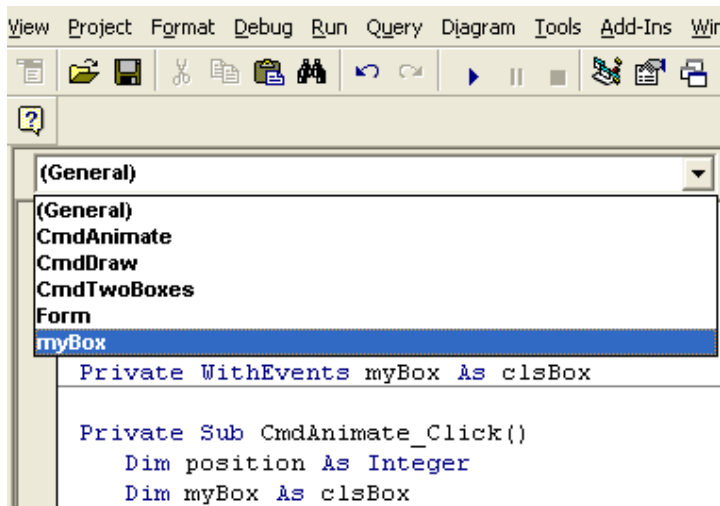
Ta thử declare một Event tên Draw trong clsBox, và viết code để mỗi khi Sub DrawBox executes thì Class sẽ Raise một event Draw.

```
Public Event Draw(X As Integer, Y As Integer)
Public Sub DrawBox(Canvas As Object, Optional fColor As Long)
    If IsMissing(fColor) Then
        Canvas.Line (mX, mY)-(mX + mWidth, mY + mHeight), , B
    Else
```

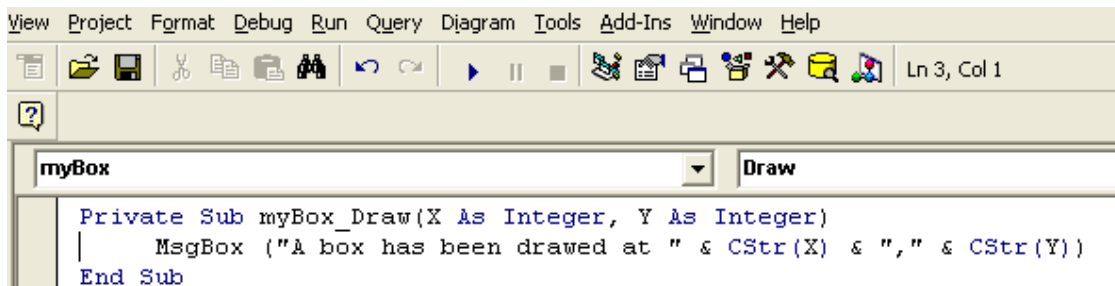
```

Canvas.Line (mX, mY)-(mX + mWidth, mY + mHeight), fColor, B
End If
RaiseEvent Draw(mX, mY)
End Sub

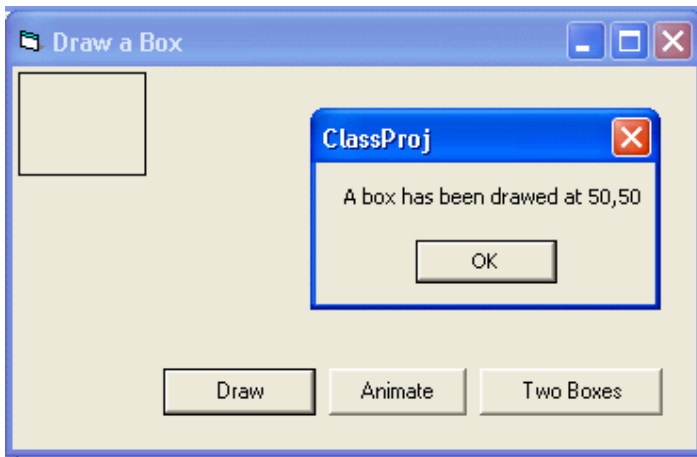
```



Bây giờ, trong frmClass thay vì chỉ declare **Dim myBox as clsBox**, ta sẽ declare **Private WithEvents myBox as clsBox**. Ngay sau đó, chữ **myBox** sẽ hiện ra trong danh sách các Object có hỗ trợ Event của frmClass. Kế đó ta sẽ viết code để handle Event Draw của myBox, tức là ta cung cấp code cho **Private Sub myBox_Draw (X as Integer, Y as Integer)**. Ở đây ta chỉ hiển thị một sử điệp báo cáo một hộp vừa được vẽ ở đâu.



Khi chạy program, mỗi lần một clsBox object executes Sub DrawBox ta sẽ thấy frmClass display một message giống như dưới đây.



Nhớ rằng, ta declare một Object với WithEvents khi ta muốn handle các Events của nó. Trong thí dụ trên frmClass là chủ của myBox và nó handles Event Draw của myBox. Tương tự như vậy, ngay cả ở bên trong một Class, nếu Class ấy được giao cho một Object có thể Raise Events (thí dụ như TextBox, ListBox, Timer .v.v..), bạn cũng có thể declare Object ấy WithEvents để nó có thể handle Events của Object.

Trong thí dụ dưới đây ta viết codes này trong một Class đã được giao cho một Textbox khi form chính gọi Sub InitObject để đưa cho Object một TextBox:

```
Private WithEvents mTextBox As TextBox

Public Sub InitObject(givenTextBox As TextBox)
    Set mTextBox = givenTextBox
End Sub

Private Sub mTextBox_KeyPress(KeyAscii As Integer)
    ' Place your code here to handle this event within the Class Object
End Sub
```

Chương Chín - Debug

Bugs là những lỗi lầm của program mà ta phát hiện khi chạy nó. Debug là công việc loại tất cả những lỗi lầm trong chương trình để nó chạy êm xuôi trong mọi hoàn cảnh.

Thông thường muốn fix một cái bug nào trước hết ta phải tìm hiểu lý do khiến nó xuất hiện. Một khi đã biết được duyên cớ rồi ta sẽ nghĩ ra cách giải quyết. Nói chung, có hai loại bugs:

1. Hoặc là program không làm đúng chuyện cần phải làm vì programmer hiểu lầm **Specifications** hay được cho tin tức sai lạc, hoặc là program bỏ sót chi tiết cần phải có. Trường hợp này ta giải quyết bằng cách giảm thiểu sự hiểu lầm qua sự nâng cấp khả năng truyền thông.
2. Program không thực hiện đúng như ý programmer muốn. Tức là programmer muốn một đàng mà bảo chương trình làm một ngã vì vô tình không viết lập trình đúng cách. Trường hợp này ta giải quyết bằng cách dùng những Software Tools (kể cả ngôn ngữ lập trình) thích hợp, và có những quá trình làm việc có hệ thống.

Trong hãng xe hơi người ta dùng từ **Quality Control** để nói đến việc chế ra chiếc xe không có lỗi làm gì cả. Để đạt mục tiêu ấy, chẳng những cần có người kiểm phẩm mà chính các nhân viên lắp ráp thậm trọng để công việc chính của người kiểm phẩm là xác nhận kết quả tốt chứ không phải tìm lỗi làm.

Có nhiều yếu tố ảnh hưởng đến chất lượng của một program như chức năng của program, cấu trúc của các bộ phận, kỹ thuật lập trình và phương pháp debug. Debug không hẳn nằm ở giai đoạn cuối của dự án mà tùy thuộc rất nhiều vào các yếu tố kể trước trong mọi giai đoạn triển khai.

Chức năng của chương trình (Program Specifications)

Dầu program lớn hay nhỏ, trước hết ta phải xác nhận rõ ràng và tỉ mỉ nó cần phải làm gì, bao nhiêu người dùng, mạng như thế nào, database lớn bao nhiêu, phải chạy nhanh đến mức nào .v.v.. Có nhiều chương trình phải bị thay đổi nữa chừng vì programmers hiểu lầm điều khách hàng muốn. Khó nhất là lúc gần giao hàng mới khám phá ra có nhiều điểm trong chương trình khách muốn một đằng mà ta làm một ngã. Do đó trong sự liên hệ với khách hàng ta cần phải hỏi đi, hỏi lại, phản hồi với khách hàng nhiều lần điều ta hiểu bằng thư từ, tài liệu, để khách xác nhận là ta biết đúng ý họ trước khi xúc tiến việc thiết kế chương trình. Nếu sau này khách đổi ý, đó là quyền của họ, nhưng họ phải trả tiền thay đổi (**variation**).

Cấu trúc các bộ phận

Program nào cũng có một kiến trúc tương tự như một căn nhà. Mỗi bộ phận càng đơn giản càng tốt và cách ráp các bộ phận phải như thế nào để ta dễ thử. Trong khi thiết kế ta phải biết trước những yếu điểm của mỗi bộ phận nằm ở đâu để ta chuẩn bị cách thử chúng. Ta sẽ không hề tin bộ phận nào hoàn hảo cho đến khi đã thử nó, dù nó đơn sơ đến đâu.

Nếu ta muốn dùng một kỹ thuật gì trong một hoàn cảnh nào mà ta không biết chắc nó chạy không thì nên thử riêng rẽ nó trước. Phương pháp ấy được gọi là **Prototype**.

Ngoài ra, ta cũng nên kế hoạch cho những trường hợp bất ngờ, điển hình là bad data - khi user bấm lung tung hay database chứa rác rến.

Nếu chương trình chạy trong **real-time** (tức là data thu nhập qua Serial Comm Port, Data Acquisition Card hay mạng), bạn cần phải lưu ý những trường hợp khác nhau tùy theo việc gì xảy ra trước, việc gì xảy ra sau. Lúc bấy giờ Logic của chương trình sẽ tùy thuộc vào trạng thái (**State**) của data. Tốt nhất là nghĩ đến những **Scenarios** (diễn tiến của những hoàn cảnh) để có thể thử từng giai đoạn và tình huống.

Ngày nay với kỹ thuật Đối Tượng, ở giai đoạn thiết kế này là lúc quyết định các Data Structures (tables, records ..v.v.) và con số Forms với Classes. Nhớ rằng mỗi Class gồm có một Data Structure và những Subs/Functions/Properties làm việc (operate) trên data ấy. Data structure phải chứa đầy đủ những chi tiết (data fields, variables) ta cần. Kế đó là những cách chương trình process data. Subs/Functions nào có thể cho bên ngoài gọi thì ta cho nó **Public**, còn những Subs/Functions khác hiện hữu để phục vụ bên trong class thì ta cho nó **Private**.

Kỹ thuật lập trình

Căn bản của programmers và các thói quen của họ rất quan trọng. Nói chung, những người hấp tấp, nhảy vào viết chương trình trước khi suy nghĩ hay cân nhắc chính chắn thì sau này bugs lòi ra khắp nơi là chuyện tự nhiên.

Dùng Subs và Functions

Nếu ở giai đoạn thiết kế kiến trúc của chương trình ta chia ra từng Class, thì khi lập trình ta lại thiết kế chi tiết về Subs, Functions .v.v., mỗi thứ sẽ cần phải thử như thế nào. Nếu ta có thể chia công việc ra từng giai đoạn thì mỗi giai đoạn có thể mà một call đến một **Sub**. Thứ gì cần phải tính ra hay lấy từ nơi khác thì có thể được thực hiện bằng một **Function**.

Thí dụ như công việc trong một tiệm giặt ủi có thể gồm có các bước sau:

1. Nhận hàng
2. Phân chia từng loại
3. Tẩy
4. Giặt
5. Ủi
6. Vô bao
7. Tính tiền
8. Giao hàng

Trong đó các bước 1,2,6 và 8 có thể là những Subs. Còn các bước 3,4,5 và 7 những Functions, thí dụ như khi ta giao cho **Function Giặt** một cái áo dơ ta sẽ lấy lại một cái áo sạch.

Nhớ rằng điểm khác biệt chính giữa một Sub và một Function là Function cho ta một kết quả mà không làm thay đổi những **parameters** ta đưa cho nó. Trong khi đó, dầu rằng Sub không cho ta gì một cách rõ ràng nhưng nó có thể thay đổi trị số (value) của bất cứ parameters nào ta pass cho nó **ByRef**. Nhắc lại là khi ta pass một parameter **ByVal** cho một Sub thì giống như ta đưa một **copy** (bản sao) của variable đó cho Sub, Sub có thể sửa đổi nó nhưng nó sẽ bị bỏ qua, không ảnh hưởng gì đến **original** (bản chính) variable.

Ngược lại khi ta pass một parameter **ByRef** cho một Sub thì giống như ta đưa bản chính của variable cho Sub để nó có thể sửa đổi vậy.

Do đó để tránh trường hợp vô tình làm cho trị số một variable bị thay đổi vì ta dùng nó trong một Sub/Function bạn nên dùng ByVal khi pass nó như một parameter vào một Sub/Function.

Thật ra, bạn có thể dùng ByRef cho một parameter pass vào một Function. Trong trường hợp đó dĩ nhiên variable ấy có thể bị sửa đổi. Điều này gọi là phản ứng phụ (**side effect**), vì bình thường ít ai làm vậy. Do đó, nếu bạn thật sự muốn vượt ngoài qui ước thông thường thì nên Comment rõ ràng để cảnh cáo người sẽ đọc chương trình bạn sau này.

Ngoài ra, mỗi programmer thường có một **Source Code Library** của những Subs/Functions ưng ý. Bạn nên dùng các Subs/Functions trong Library của bạn càng nhiều càng tốt, vì chúng đã được thử nghiệm rồi.

Đừng sợ Error

Mỗi khi chương trình có một Error, hoặc là **Compilation Error** (vì ta viết code không đúng văn phạm, ngữ vựng), hoặc là Error trong khi chạy chương trình, thì bạn không nên sợ nó. Hãy bình tĩnh đọc cái **Error Message** để xem nó muốn nói gì. Nếu không hiểu ngay thì đọc đi đọc lại vài lần và suy nghiệm xem có tìm được mạch nước nào không. Nghề programming của chúng ta sẽ gặp Errors như ăn cơm bữa, nên bạn phải tập bình tĩnh đối diện với chúng.

Dùng Comment (Chú thích)

Lúc viết code nhớ thêm **Comment** đầy đủ để bất cứ khi nào trở lại đọc đoạn code ấy trong tương lai bạn không cần phải dựa vào tài liệu nào khác mà có thể hiểu ngay lập tức mục đích của một Sub/Function hay đoạn code.

Như thế không nhất thiết bạn phải viết rất nhiều Comment nhưng hãy có điểm nào khác thường, bí hiểm thì bạn cần thông báo và giải thích tại sao bạn làm cách ấy. Có thể sau này ta khám phá ra đoạn code có bugs; lúc đọc lại có thể ta sẽ thấy dấu rằng ý định và thiết kế đúng nhưng cách lập trình có phần thiếu soát chẳng hạn.

Tính ra trung bình một programmer chỉ làm việc 18 tháng ở mỗi chỗ. Tức là, gần như chắc chắn code bạn viết sẽ được người khác đọc và bảo trì (debug và thêm bớt). Do đó, code phải càng đơn giản, dễ hiểu càng tốt. Đừng lo ngại là chương trình sẽ chạy chậm hay chiếm nhiều bộ nhớ, vì ngày nay computer chạy rất nhanh và bộ nhớ rất rẻ. Khi nào ta thật sự cần phải quan tâm về vận tốc và bộ nhớ thì điều đó cần được thiết kế cẩn thận chứ không phải dựa vào những tiêu xảo về lập trình.

Đặt tên các variables có ý nghĩa

Khổ nhất là làm việc với các variables có tên vắn tắt như K, L, AA, XY. Ta không có một chút ý niệm chúng là ai, hiện hữu để làm gì. Thay vào đó, nếu ta đặt các tên variables như NumberOfItems, PricePerUnit, Discount .v.v.. thì sẽ dễ hiểu hơn.

Một trong những bugs khó thấy nhất là ta dùng cùng một tên cho **local variable** (variable declared trong Sub/Function) và **global variable** (variable declared trong Form hay Basic Module). Local variable sẽ che lấp global variable cùng tên, nên nếu bạn muốn nói đến global variable trong hoàn cảnh ấy bạn sẽ dùng lầm local variable.

Dùng Option Explicit

Bạn nên trung tín dùng **Option Explicit** ở đầu mỗi Form, Class hay Module. Nếu có variable nào đánh vần trật VB6 IDE sẽ cho bạn biết ngay. Nếu bạn không dùng Option Explicit, một variable đánh vần trật được xem như một variable mới với giá trị 0 hay "" (empty string).

Nói chung bạn nên thận trọng khi assign một data type cho một variable với data type khác. Bạn phải biết rõ bạn đang làm gì để khỏi bị phản ứng phụ (side effect).

Desk Check

Kiểm lại code trước khi compile. Khi ta compile code, nếu không có error chỉ có nghĩa là Syntax của code đúng, không có nghĩa là logic đúng. Do đó ta cần phải biết chắc là code ta viết sẽ làm đúng điều ta muốn bằng cách đọc lại code trước khi compile nó lần đầu tiên. Công việc này gọi là **Desk Check** (Kiểm trên bàn). Một chương trình được Desk Checked kỹ sẽ cần ít debug và chứa ít bugs không ngờ trước. Lý do là mọi scenarios đã được tiên liệu chu đáo.

Soạn một Test Plan

Test Plan liệt kê tất cả những gì ta muốn thử và cách thử chúng. Khi thử theo Test Plan ta sẽ khám phá ra những bug và tìm cách loại chúng ra. Hồ sơ ghi lại lịch sử của Test Plan (trục trặc gì xảy ra, bạn đã dùng biện pháp nào để giải quyết) sẽ bổ ích trên nhiều phương diện. Ta sẽ học được từ kinh nghiệm Debug và biết rõ những thứ gì trong dự án đã được thử theo cách nào.

Xử lý Error lúc Run time

Khi EXE của một chương trình viết bằng VB6 đang chạy, nếu gặp Error, nó sẽ hiển thị một **Error Dialog** cho biết lý do vắn tắt. Sau khi bạn click OK, chương trình sẽ ngưng. Nếu bạn chạy chương trình trong VB6 IDE, bạn có dịp bảo program ngừng ở trong source code chỗ có Error bằng cách bấm button **Debug** trong Error Dialog. Tiếp theo đó bạn có thể tìm hiểu trị số các variables để đoán nguyên do của Error. Do đó, nếu bạn bắt đầu cho dùng một program bạn viết trong sổ, nếu tiện thì trong vài

tuần đầu, thay vì chạy EXE của chương trình, bạn chạy source code trong VB6 IDE. Nếu có bug nào xảy ra, bạn có thể cho program ngừng trong source code để debug.

Khi bạn dùng statement:

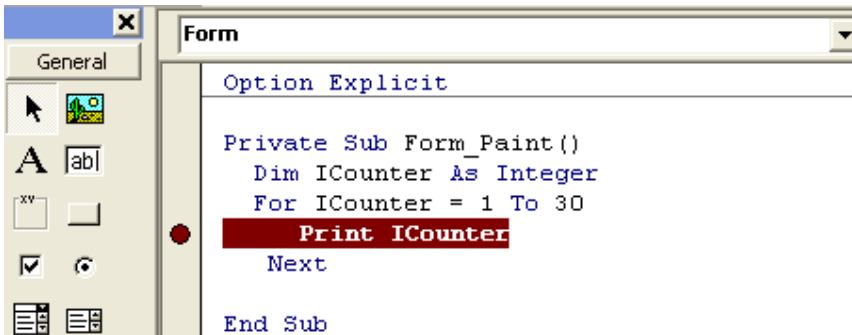
On Error Resume Next

thì từ chỗ đó trở đi, nếu chương trình gặp Error, nó sẽ bỏ qua (ignore) hoàn toàn. Điểm này tiện ở chỗ giúp chương trình EXE của ta tránh bị tét cái ạch rồi biến mất, rất là "quê" với khách hàng. Nhưng nó cũng bất lợi là khi khách hàng cho hay họ gặp những trường hợp lạ, không giải thích được (vì Error bị ignored mà không ai để ý), thì ta cũng bí luôn, có thể không biết bắt đầu từ đâu để debug. Do đó, dĩ nhiên trong lúc debug ta không nên dùng nó, nhưng trước khi giao cho khách hàng bạn nên cân nhắc kỹ trước khi dùng.

Dùng Breakpoints

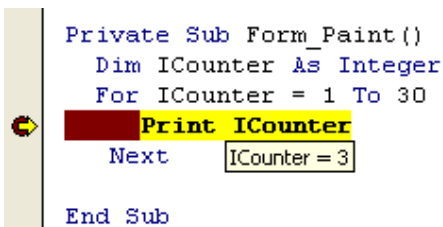
Cách hay nhất để theo dõi execution của program là dùng **Breakpoint** để làm cho program ngừng lại ở một chỗ ta muốn ở trong code, rồi sau đó ta cho program bước từng bước. Trong dịp này ta sẽ xem xét trị số của những variables để coi chúng có đúng như dự định không.

Bạn đoán trước execution sẽ đi qua chỗ nào trong code, chọn một chỗ thích hợp rồi click bên trái của hàng code, chỗ dấu chấm tròn đỏ như trong hình dưới đây:



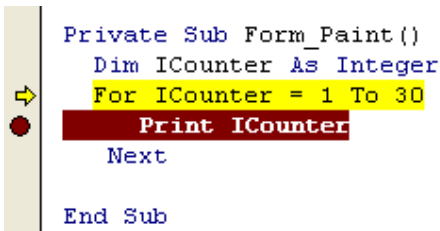
Nếu bạn click lên dấu chấm tròn đỏ một lần nữa thì là hủy bỏ nó. Một cách khác để đặt một breakpoint là để editor cursor lên hàng code rồi bấm **F9**. Nếu bạn bấm F9 lần nữa khi cursor nằm trên hàng đó thì là hủy bỏ breakpoint.

Lúc program đang dừng lại, bạn có thể xem trị số của một variable bằng cách để cursor lên trên variable ấy, tooltip sẽ hiện ra như trong hình dưới đây:



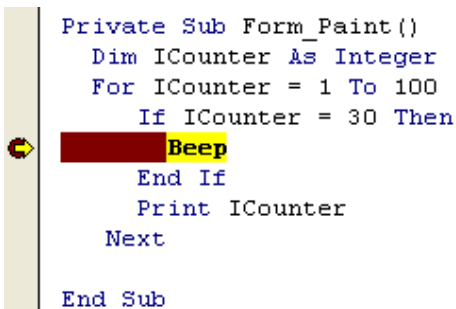
Có một số chuyện khác bạn có thể làm trong lúc này. Bạn có thể nắm dấu chấm tròn đỏ kéo (drag) nó ngược lên một hay nhiều hàng code để nó sẽ execute trở lại vài hàng code. Bạn cho program execute từng hàng code bằng cách bấm **F8**. Menu command tương đương với nó là **Debug | Step Into**. Sẽ có

lúc bạn không muốn program bước vào bên trong một Sub/Function mà muốn việc execute một Sub/Function như một bước đơn giản. Trong trường hợp đó, bạn dùng Menu command **Debug | Step Over** hay **Shift-F8**.



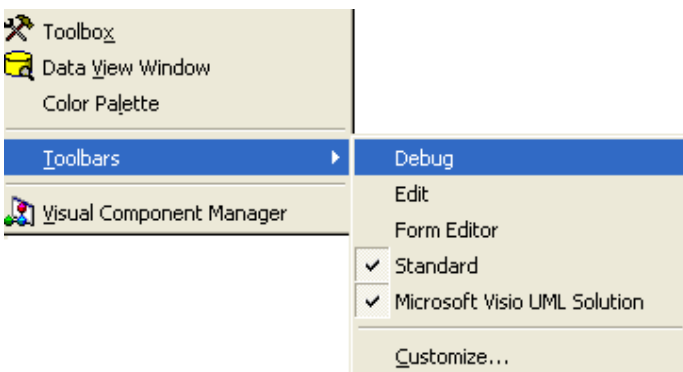
```
Private Sub Form_Paint()  
    Dim ICounter As Integer  
    For ICounter = 1 To 30  
        Print ICounter  
    Next  
End Sub
```

Nhớ là để cho program chạy lại bạn bấm **F5**, tương đương với Menu command **Run | Continue**. Có khi bạn muốn program ngừng ở giữa một For Loop khi Iterator value có một trị số khá lớn. Nếu ta để sẵn một breakpoint ở đó rồi cứ bấm F5 nhiều lần thì hơi bất tiện. Có một mảnh lối là dùng một IF statement để thử khi Iterator value có trị số ấy thì ta ngừng ở breakpoint tại statement **Beep** (thay gì statement **Print ICounter**) như trong hình dưới đây:

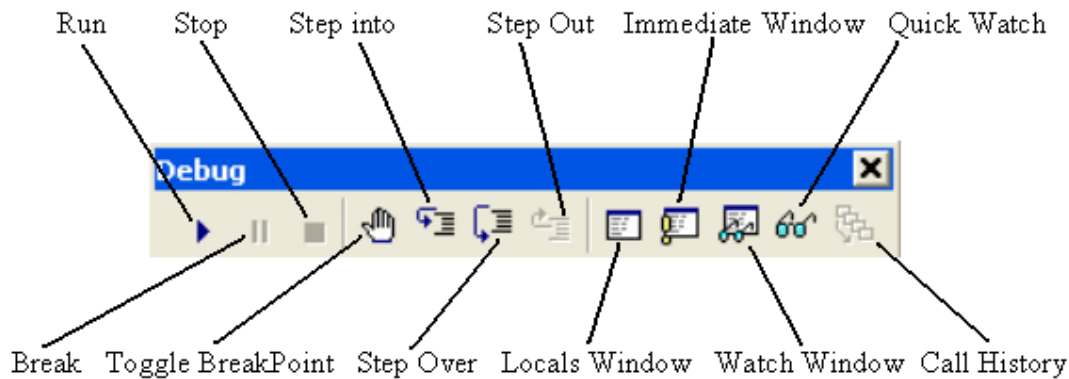


```
Private Sub Form_Paint()  
    Dim ICounter As Integer  
    For ICounter = 1 To 100  
        If ICounter = 30 Then  
            Beep  
        End If  
        Print ICounter  
    Next  
End Sub
```

Muốn hủy bỏ mọi breakpoints bạn dùng Menu command **Debug | Clear All Breakpoints**. Để tiện việc debug, bạn có thể dùng **Debug Toolbar** bằng cách hiển thị nó với Menu command **View | Toolbars | Debug**



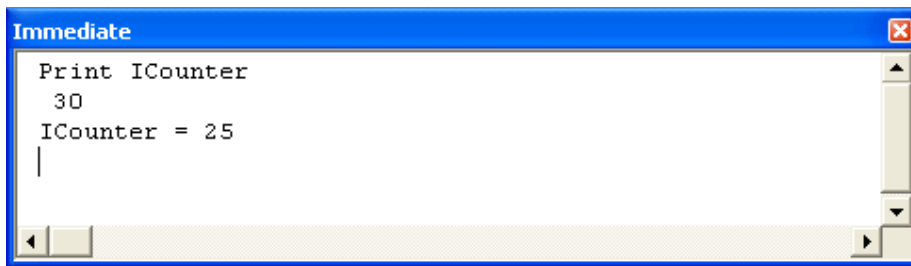
VB6 IDE sẽ hiển thị Debug Toolbar như sau:



Dùng Immediate Window

Immediate Window cho phép ta execute những VB statement trong khi program đang dừng lại. Ta có thể dùng một Print statement để hiển thị trị số của một variable hay kết quả của một Function, gọi một Sub hay thay đổi trị số một variable trước khi tiếp tục cho chương trình chạy lại.

Để hiển thị Immediate Window, dùng Menu command **View | Immediate Window**.



Thay vì đánh "**Print ICounter**" bạn cũng có thể đánh "**? ICounter**". Nhớ là mỗi VB Statement bạn đánh trong Immediate Window sẽ được executed ngay khi bạn bấm **Enter**. Bạn có thể dùng lại bất cứ VB statement nào trong Immediate Window, chỉ cần bấm Enter ở cuối hàng ấy.

Theo dấu chân chương trình (Tracing)

Đôi khi không tiện để ngừng program nhưng bạn vẫn muốn biết program đang làm gì trong một Sub. Bạn có thể để giữa code của một Sub/Function một statement giống như dưới đây:

Debug.Print Format (Now,"hh:mm:ss ") & "(Sub ProcessInput) Current Status:" & Status để program hiển thị trong Immediate Window value của Status khi nó execute bên trong Sub ProcessInput lúc mấy giờ.

Có một cách khác là thay vì cho hiển thị trong Immediate Window bạn cho viết xuống (**Log**) vào trong một text file. Dưới đây là một Sub điển hình bạn có thể dùng để Log một Event message:

```
Sub LogEvent(ByVal GivenFileName, ByVal Msg As String, HasFolder As Boolean,
IncludeTimeDate As Integer)
```

```
' Append event message Msg to a text Logfile GivenFileName
```

```
' If GivenFileName is fullPathName then HasFolder is true
```

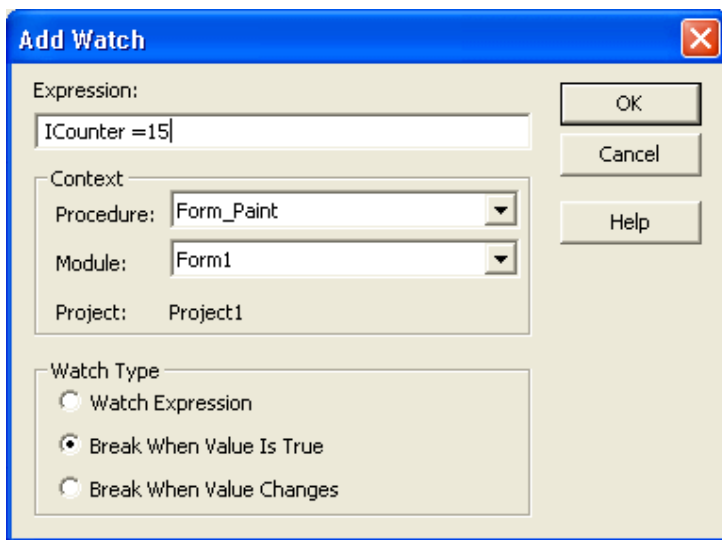
```

' IncludeTimeDate = 0 : No Time or Date
' = 1 : Prefix with Time
' = 2 : Prefix with Time and Date
Dim FileNo, LogFileName, theFolder
If HasFolder Then
    LogFileName = GivenFileName
Else
    If Right(App.Path, 1) <> "\" Then
        theFolder = App.Path & "\"
    Else
        theFolder = App.Path
    End If
    LogFileName = theFolder & GivenFileName
End If
FileNo = FreeFile
If Dir(LogFileName) <> "" Then
    Open LogFileName For Append As FileNo
Else
    Open LogFileName For Output As FileNo
End If
Select Case IncludeTimeDate
Case 0 ' No Time or Date
    Print #FileNo, Msg
Case 1 ' Time only
    Print #FileNo, Format(Now, "hh:nn:ss ") & Msg
Case 2 ' Date & Time
    Print #FileNo, Format(Now, "dd/mm/yyyy hh:nn:ss ") & Msg
End Select
Close FileNo
End Sub

```

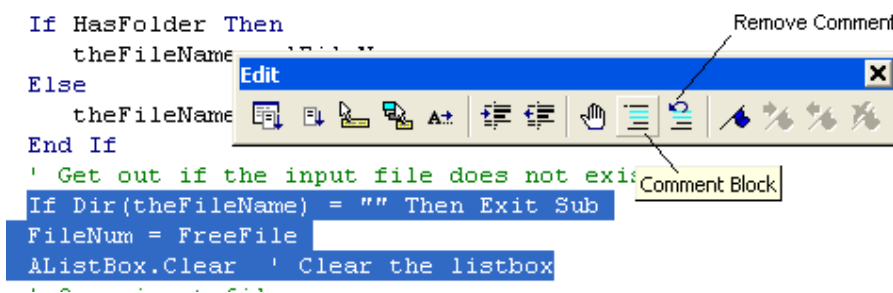
Dùng Watch Window

Đôi khi bạn muốn program ngừng không phải ở một chỗ nào nhất định, nhưng khi trị số của một variable hay của một expression là bao nhiêu, có thể là bạn không biết tại sao một variable tự nhiên có một trị số như vậy. Câu hỏi: **Ai là thủ phạm?** . Thí dụ bạn muốn program ngừng lại khi **ICounter = 15**. Bạn có thể dùng Menu command **Debug | Add Watch**. VB6 IDE sẽ hiển thị dialog dưới đây. Bạn đánh **ICounter = 15** vào textbox **Expression** và click option box **Break When Value Is True** trong hộp **Watch Type**. Làm như vậy có nghĩa là ta muốn program ngừng khi ICounter bằng 15.



Dùng Phương Pháp Triệt Khai (Elimination Method)

Có một phương pháp rất thông dụng khi debug là Comment Out những hàng code nghi ngờ để xem bug có biến mất không. Nó được gọi là **Elimination Method**. Nếu bug biến mất thì những hàng code đã được comment out là thủ phạm. Bạn có thể Comment Out một số hàng cùng một lúc bằng cách highlight các hàng ấy rồi click **Comment Block** trên Edit ToolBar.



Khi dùng Elimination Method bạn phải cân nhắc Logic của code bạn trong khi quyết định Comment Out những hàng nào, nếu không, đó là một phương pháp khá nguy hiểm.

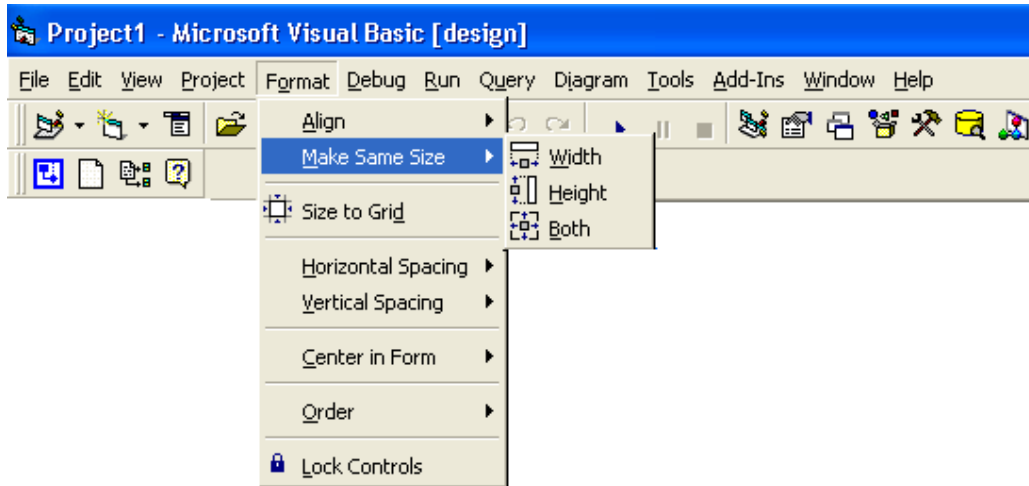
Ngoài ra, Menu Command **View | Locals Window** liệt kê cho bạn trị số của tất cả variables trong một Sub/Function và **View | Call Stack** liệt kê thứ bậc các Sub gọi lần lượt từ ngoài vào trong cho đến vị trí code đang ngừng hiện thời.

Chương Mười - Dùng Menu

Menu trong Windows là nơi tất cả các commands của một program được sắp xếp thứ tự theo từng loại để giúp ta dùng dễ dàng.

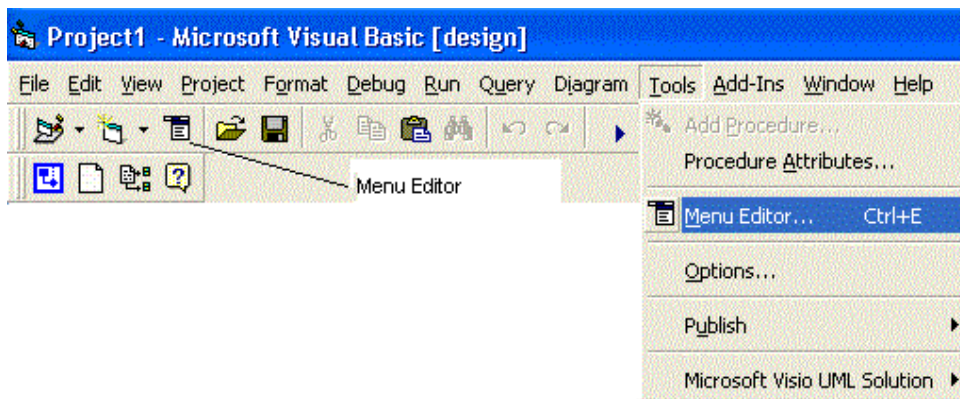
Có hai loại menu ta thường gặp: **drop-down (thả xuống)** menu và **pop-up (hiện lên)** menu. Ta dùng drop-down menu làm Menu chánh cho chương trình. Thông thường nó nằm ở phía trên chóp màn ảnh. Nằm dọc theo chiều ngang là Menu Bar, nếu ta click lên một command trong Menu Bar thì program sẽ thả xuống một menu với những MenuItems nằm dọc theo chiều thẳng đứng. Nếu ta click lên

MenuItem nào có dấu hình tam giác nhỏ bên phải thì program sẽ popup một Menu như trong hình dưới đây (khi ta click **Format | Make Same Size**):

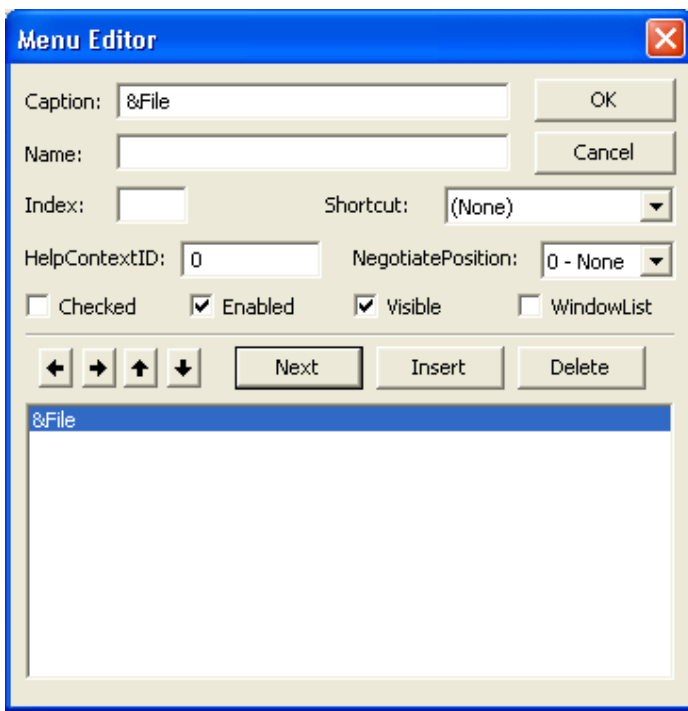


Main Menu

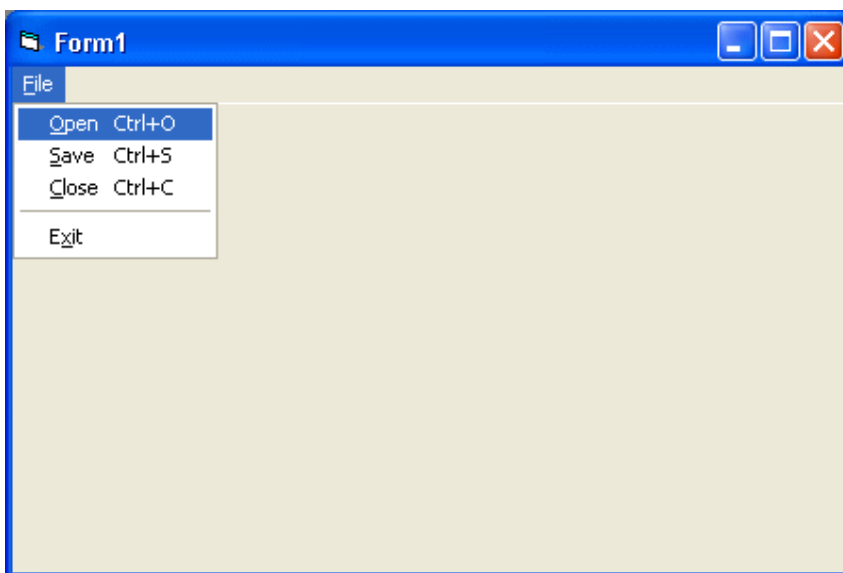
Ta dùng **Menu Editor** để tạo hoặc sửa một Menu cho program. Menu thuộc về một Form. Do đó, trước hết ta select một Form để làm việc với Designer của nó (chớ không phải code của Form). Kế đó ta dùng Menu Command **Tools | Menu Editor** hay click lên icon của Menu Editor trên Toolbar để làm cho Menu Editor hiện ra.



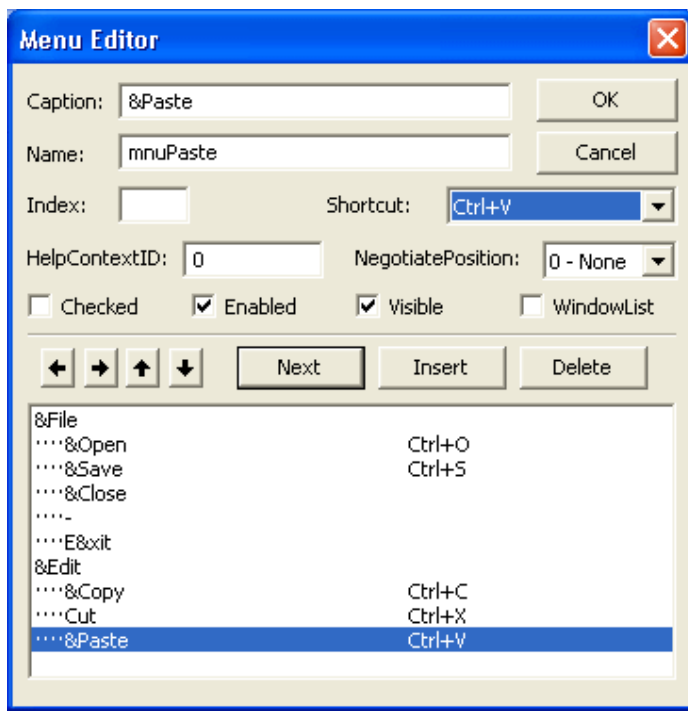
Đầu tiên có một vệt màu xanh nằm trong khung trắng của Menu Editor, nơi sẽ hiển thị Caption của Menu Command đầu tiên của Form. Khi ta đánh chữ **&File** vào Textbox **Caption**, nó cũng hiện ra trên vệt xanh nói trên. Kế đó, bạn có thể đánh tên của Menu Command vào Textbox **Name**. Dù ta cho Menu Command một tên nhưng ta ít khi dùng nó, trừ trường hợp muốn nó visible/invisible (hiện ra/biến mất). Bình thường ta dùng tên của MenuItem nhiều hơn.



Để có một Menu như trong hình dưới đây ta còn phải edit thêm vào các MenuItem's Open, Save, Close và Exit.



Hình dưới đây cho thấy tất cả các MenuItem's của Menu Command File đều nằm thụt qua bên phải với bốn dấu chấm (....) ở phía trước. Khi ta click dấu tên chỉ qua phải thì MenuItem ta đang Edit sẽ có thêm bốn dấu chấm, tức là thụt một bậc trong Menu (Nested).



Tương tự như vậy, khi ta click dấu tên chỉ qua trái thì MenuItem ta đang Edit sẽ mất bốn dấu chấm, tức là trở lại một bậc trong Menu.

Nếu muốn cho User dùng Alt key để xử dụng Menu, bạn đánh thêm dấu **&** trước character bạn muốn trong menu Caption. Thí dụ **Alt-F** sẽ thả xuống Menu của Menu Command File.

Nếu bạn đặt cho MenuItem **&Open** tên **mnuOpen**, thì khi bạn Click lên Caption nó trên Form trong lúc thiết kế, VB6 IDE sẽ hiển thị cái vỏ của **Sub mnuOpen_Click()**, giống như Sub cmdButton_Click() của một CommandButton:

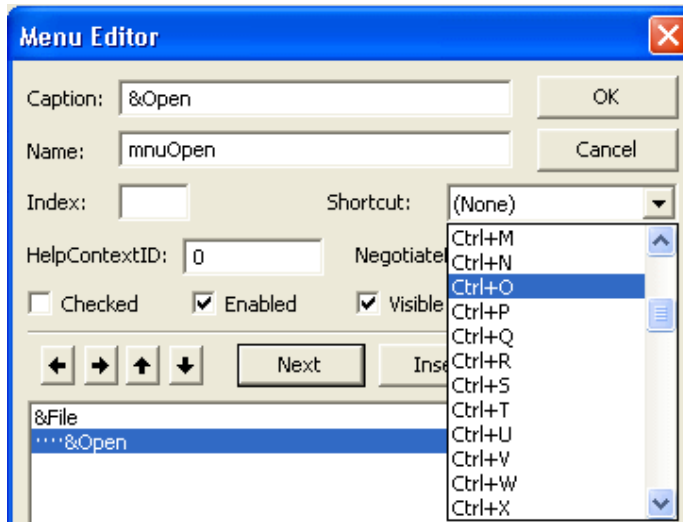
```
Private Sub mnuOpen_Click()
    MsgBox "You clicked mnuOpen"
End Sub
```

Trong thí dụ trên ta đánh thêm một Statement để hiển thị một message đơn giản **"You clicked mnuOpen"**. Bạn có thể đặt cho một MenuItem tên gì cũng được, nhưng người ta thường dùng prefix **mnu** để dễ phân biệt một menuItem Event với một CommandButton Event. Do đó, ta có những tên mnuFile, mnuOpen, mnuSave, mnuClose, mnuExit.

Cái gạch ngang giữa MenuItems Close và Exit được gọi là **Menu Separator**. Bạn có thể nhét một Menu Separator bằng cách cho Caption nó bằng **dấu trừ (-)**.

Ngoài Alt key ta còn có thể cho User dùng Shortcut của menuItem. Để cho MenuItem một Shortcut,

bạn chọn cho nó một Shortcut từ ComboBox **Shortcut** trong Menu Editor.
Trong hình dưới đây ta chọn **Ctrl+O** cho mnuOpen.



By default, menuItem được Enabled và Visible. Lúc thiết kế bạn có thể cho MenuItem giá trị khởi đầu của Enabled và Visible bằng cách dùng Checkboxes Enabled và Visible.

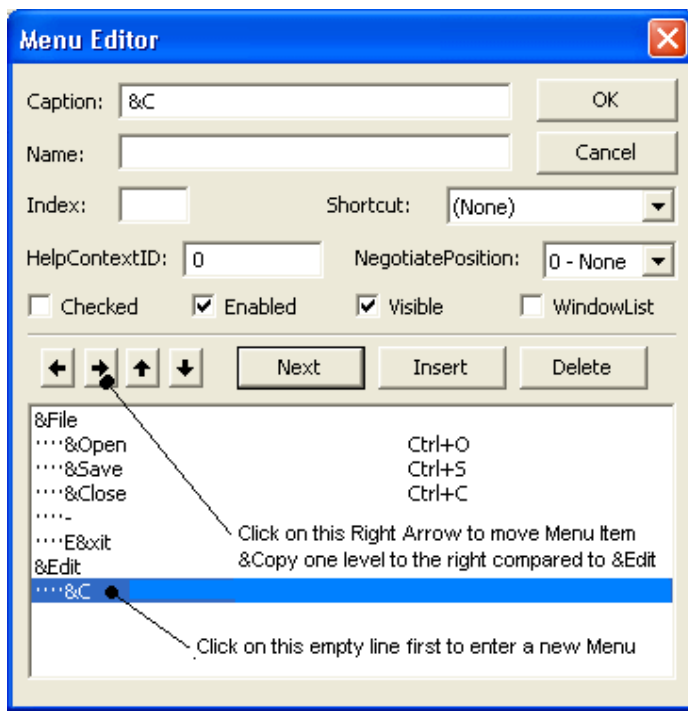
Trong khi chạy program (at runtime), bạn cũng có thể thay đổi các values Enabled và Visible như sau:

```
mnuSave.Enabled = False
```

```
mnuOpen.Visible = False
```

Khi một MenuItem có Enabled=False thì nó bị mờ và user không dùng được.

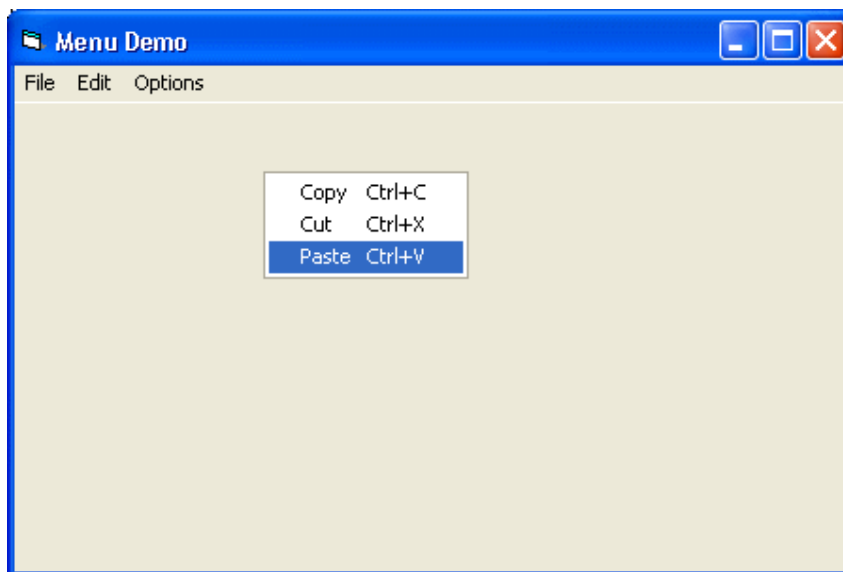
Bạn dùng các dấu mũi tên chỉ lên và xuống để di chuyển MenuItem đã được selected lên và xuống trong danh sách các MenuItems. Bạn dùng button **Delete** để hủy bỏ MenuItem đã được selected, **Insert** để nhét một MenuItem mới ngay trên MenuItem đã được selected và **Next** để chọn MenuItem ngay dưới MenuItem đã được selected.



Pop-up Menu

Đối với User, đang khi làm việc với một Object trong Windows tiện nhất là ta có thể làm hiển thị **Context Menu** (Menu áp dụng cho đúng tình huống) bằng một Mouse click. Thông thường đó là Right Click và cái Context Menu còn được gọi là **Pop-up Menu**. Chính cái Pop-Up menu thật ra là Drop-down menu của một Menu Bar Command. Bình thường Menu Bar Command ấy có thể visible hay invisible (tàn hình).

Trong hình dưới đây, khi User Right click trên Form, mnuEdit sẽ hiện lên. Nếu bình thường bạn không muốn cho User dùng nó trong Main Menu thì bạn cho nó invisible:



Code làm cho Popup menu hiện lên được viết trong Event Mousedown của một Object mà tình cờ ở đây là của chính cái Form:

```

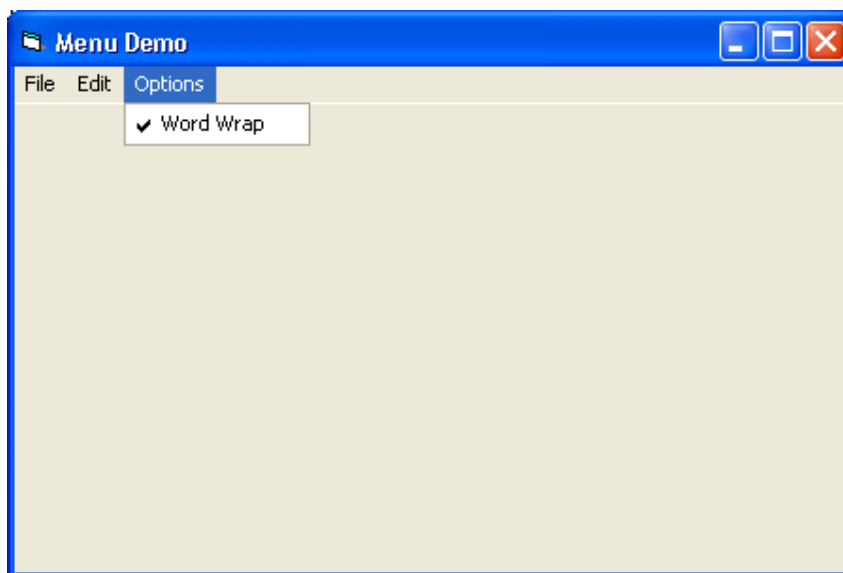
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' Popup the Edit Menu if User clicked the Right Button of the Mouse
    If Button = vbRightButton Then
        PopupMenu mnuEdit
    End If
End Sub

```

Ngay cả khi bạn muốn cho mnuEdit bình thường là invisible, bạn cũng nên để cho nó visible trong lúc đầu để tiện bỏ code vào dùng để xử lý Click Events của những MenuItems thuộc về mnuEdit như mnuCopy, mnuCut và mnuPaste.

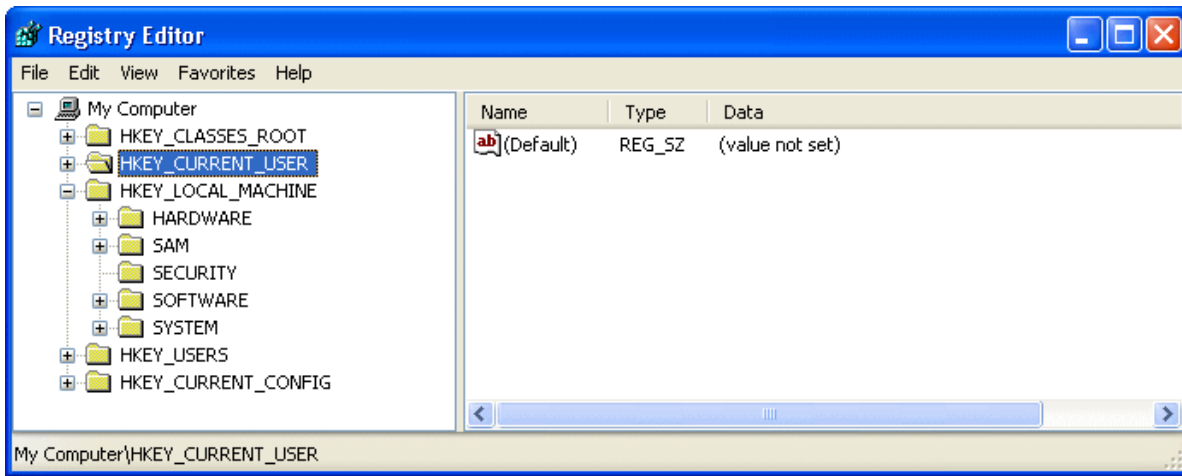
Chứa menu Settings trong Registry

Giả sử program bạn cho User một Option WordWrap như dưới đây:



Bạn muốn Program nhớ Option mà User đã chọn, để lần tới khi User khởi động program thì Option WordWrap còn giữ nguyên giá trị như cũ.

Cách tiện nhất là chứa value của Option WordWrap như một **Key** trong **Registry**. Registry là một loại cơ sở dữ liệu đặc biệt của Windows Operating System dùng để chứa những dữ kiện liên hệ đến Users, Hardware, Configurations, ActiveX Components ..v.v. dùng trong computer. Trong Registry, data được sắp đặt theo từng loại theo đẳng cấp. Bạn có thể Edit trực tiếp trị số các Keys trong Registry bằng cách dùng **Registry Editor**.



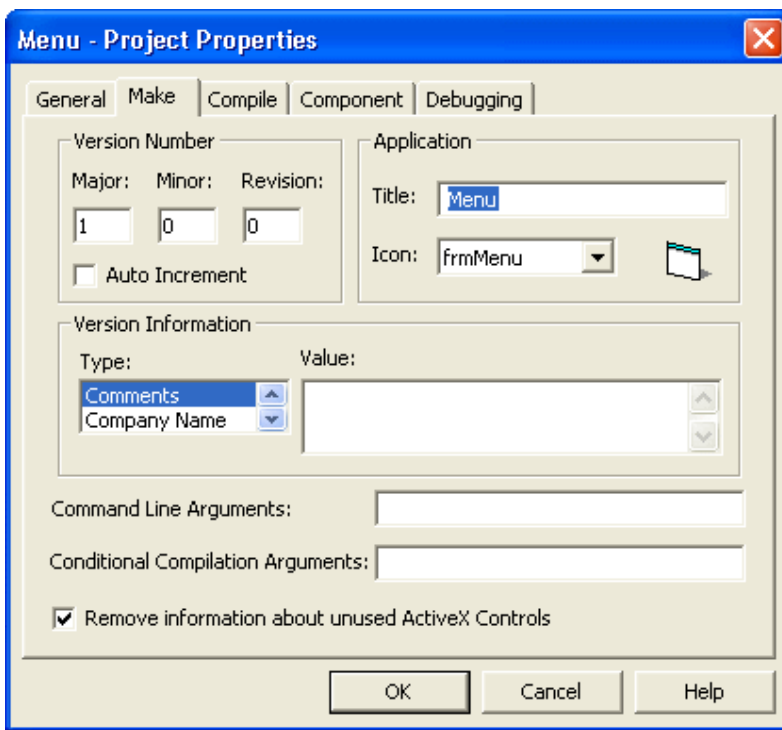
Trong program này ta cũng nên tiện bắt program nhớ luôn vị trí của Form khi program ngừng lại, để lần tới khi User khởi động program thì program sẽ có vị trí lúc đầu giống y như trước.

Ta sẽ dùng **Sub SaveSetting** để chứa **Checked** value của mnuWordWrap và **Left, Top** của Form. Code ấy ta sẽ để trong **Sub Form_QueryUnload** vì nó sẽ được executed trước khi Form Unload.

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

Private Sub SaveSettings()
    ' Save Location of the form
    SaveSetting App.Title, "Location", "Left", Me.Left
    SaveSetting App.Title, "Location", "Top", Me.Top
    ' Save the setting of WordWrap in menu
    SaveSetting App.Title, "Settings", "WordWrap", mnuWordWrap.Checked
End Sub
```

App.Title là Tựa đề của program. Thông thường nó là tên của VB Project, nhưng bạn có thể sửa nó trong **Project Property Dialog** (Tab **Make**) :



Khi chứa value của một thứ gì (ta gọi là **Key**) vào Registry bạn có thể sắp đặt cho nó nằm trong **Section** nào tùy ý. Ở đây ta đặt ra hai Sections tên **Location** để chứa Top,Left của Form và tên **Settings** để chứa Key mnuWordWrap.Checked.

Muốn cho program có các giá trị của Keys chứa trong Registry khi nó khởi động ta chỉ cần dùng **Function GetSetting** trong **Sub Form_Load** để đọc vào từ Registry như dưới đây:

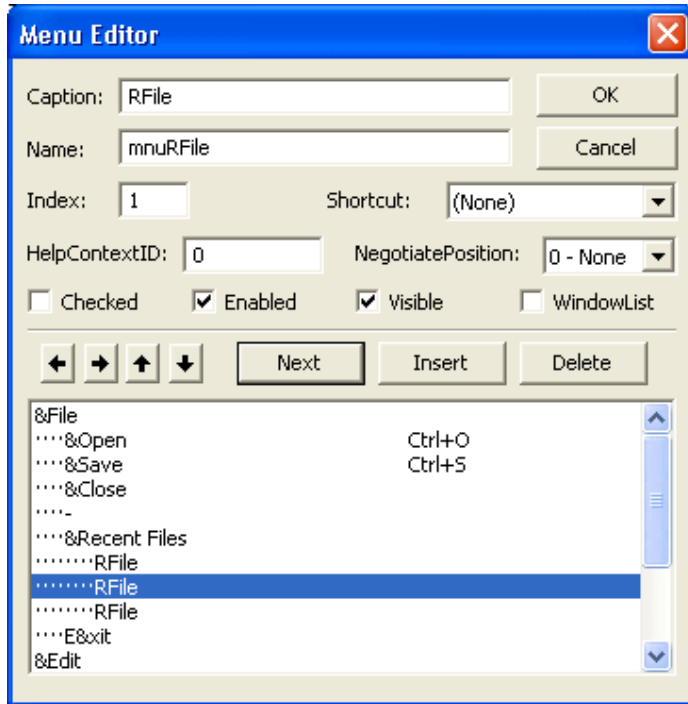
```
Private Sub Form_Load()
    ' Initialise Location of the form by reading the Settings from the Registry
    Me.Left = Val(GetSetting(App.Title, "Location", "Left", "0"))
    Me.Top = Val(GetSetting(App.Title, "Location", "Top", "0"))
    ' Initialise setting of WordWrap in the menu
    mnuWordWrap.Checked = ( GetSetting(App.Title, "Settings", "WordWrap", "False") = "True"
)
End Sub
```

Lúc đầu khi chưa có gì trong Registry thì **"0"** (string "0" được converted bởi Val ra 0) là default value cho Left và Top, còn **"False"** là default value của mnuWordWrap.Checked.

Ngoài ra ta cũng muốn program nhớ tên của ba Files User dùng gần đây nhất. Tức là trong Drop-down của Menu Command File sẽ có MenuItem **Recent Files** để hiển thị từ một đến ba tên Files, cái mới nhất nằm trên hết. Trước hết, ta cần tạo ra 3 SubmenuItem có cùng tên mnuRFile nhưng mang **Index** bằng 0,1 và 2 (bạn đánh vào Textbox **Index**). Ta sẽ dùng Captions của chúng để hiển thị tên các Files.

Lúc chưa có Filename nào cả thì MenuItem **Recent Files** sẽ bị làm mờ đi (tức là `mnuRecentFiles.Enabled = False`).

Ta sẽ chứa tên các Files như một String trong Section Settings của Registry. Ta phân cách tên các Files bằng delimiter character |. Thí dụ: "LattestFileName.txt|OldFileName.txt|OldestFilename.txt"
Mỗi lần User Open một File ta sẽ thêm tên File ấy vào trong Registry và bất cứ lúc nào chỉ giữ lại tên của 3 Files mới dùng nhất.

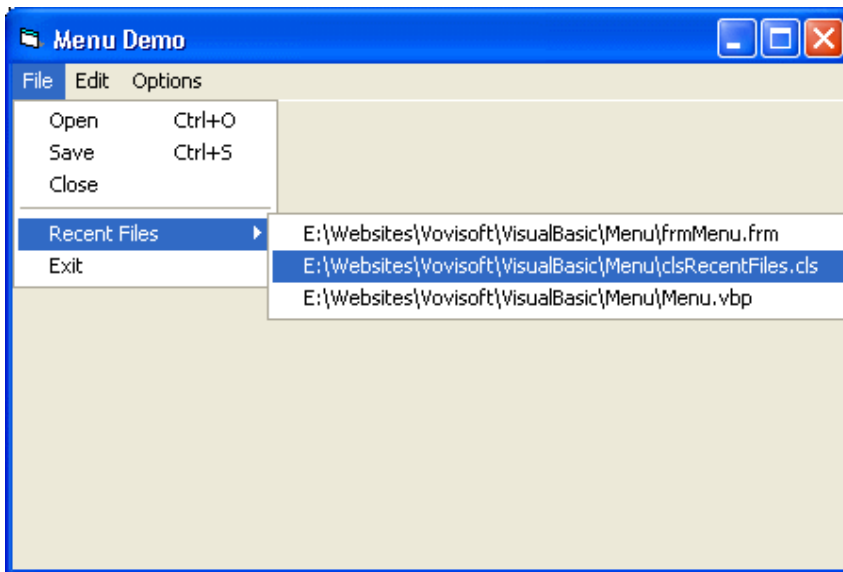


Dưới đây là code dùng để thêm tên File mới dùng nhất vào Registry:

```
Private Sub mnuOpen_Click()  
    ' Initialise Folder in Common Dialog  
    CommonDialog1.InitDir = App.Path  
    ' Launch the dialog  
    CommonDialog1.ShowOpen  
    ' Save the Filename in the Registry, using Object myRecentFiles  
    myRecentFiles.AddFile CommonDialog1.FileName  
End Sub
```

Code dùng trong Sub Form_Load để đọc tên RecentFiles và hiển thị trong Menu:

```
'  
Set myRecentFiles = New clsRecentFiles  
' Pass the form handle to it  
' This effectively loads the most recently used FileNames to menu
```



Ta sẽ dùng một Class tên **clsRecentFiles** để đặc biệt lo việc chứa tên Files vào Registry và hiển thị tên các Files ấy trong Menu. Bên trong clsRecentFiles ta cũng dùng clsString, là một Class giúp ta ngắt khúc String trong Registry ra tên của các Files dựa vào chỗ các delimiter character |.

```
' Author: Le Duc Hong http://www.vovisoft.com
' Class Name: clsRecentFiles
' This Class saves the most Recent FileNames used in the Registry in form of
' a String delimited by |.
' Up to MaxFiles Filenames maybe stored.
' You need to pass the Form that contains the menu to it.
' The assumption is that you have created an array of MenuItems named mnuRFile
' to display the FileNames
'
Const MaxFiles = 3 ' Maximum number of FileNames to remember
Private myForm As Form
Private RecentFiles As clsString
Public Sub Init(TForm As frmMenu)
    Set myForm = TForm
    Set RecentFiles = New clsString
    ' Read the Most Recent Filename String from the Registry
    RecentFiles.Text = GetSetting(App.Title, "Settings", "RecentFiles", "")
    ' Assign the Delimiter character and tokenise the String (i.e. split it) into FileNames
    RecentFiles.Delimiter = "|"
    UpdateMenu
End Sub
```

```

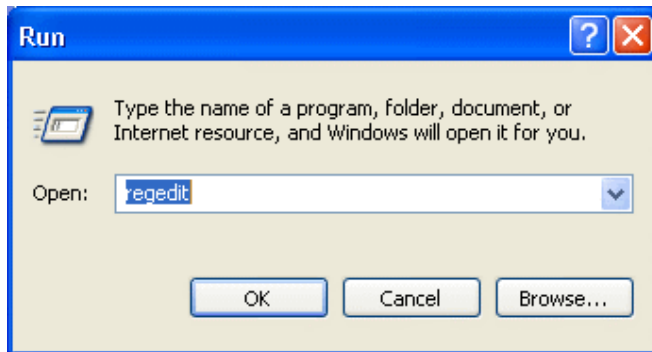
Public Sub AddFile(FileName As String)
    ' Add the latest FileName to the list and update the Registry
    ' Prefix the FileName to the existing MostRecentFileName String
    RecentFiles.Text = FileName & "|" & RecentFiles.Text
    ' Discard the oldest FileNames if the total number is greater than MaxFiles
    If RecentFiles.TokenCount > MaxFiles Then
        Dim TStr As String
        Dim i As Integer
        ' Reconstitute the String that contains only the most recent MaxFiles FileNames
        For i = 1 To MaxFiles
            TStr = TStr & RecentFiles.TokenAt(i) & "|"
        Next
        ' Remove the last delimiter character on the right
        RecentFiles.Text = Left(TStr, Len(TStr) - 1)
    End If
    ' Update the String in the Registry
    SaveSetting App.Title, "Settings", "RecentFiles", RecentFiles.Text
    UpdateMenu
End Sub

Private Sub UpdateMenu()
    ' Display the most recent Filenames in the menu
    Dim i As Integer
    ' If there is no FileNames to display then disable the MenuItem entry
    If RecentFiles.TokenCount = 0 Then
        myForm.mnuRecentFiles.Enabled = False
        Exit Sub
    Else
        ' Otherwise enable the MenuItem entry
        myForm.mnuRecentFiles.Enabled = True
    End If
    ' Assign FileName to Caption of mnuRFile array and make the MenuItem elements visible
    For i = 1 To RecentFiles.TokenCount
        myForm.mnuRFile(i - 1).Caption = RecentFiles.TokenAt(i) ' Assign to Caption
        myForm.mnuRFile(i - 1).Visible = True ' Make the MenuItem visible
        If i = MaxFiles Then Exit For ' This line maybe unnecessary
    Next
    ' Make the rest of the MenuItem array mnuRFile invisible if there are less than MaxFiles
    If RecentFiles.TokenCount < MaxFiles Then
        For i = RecentFiles.TokenCount To MaxFiles - 1
            myForm.mnuRFile(i).Visible = False
        Next
    End If
End Sub

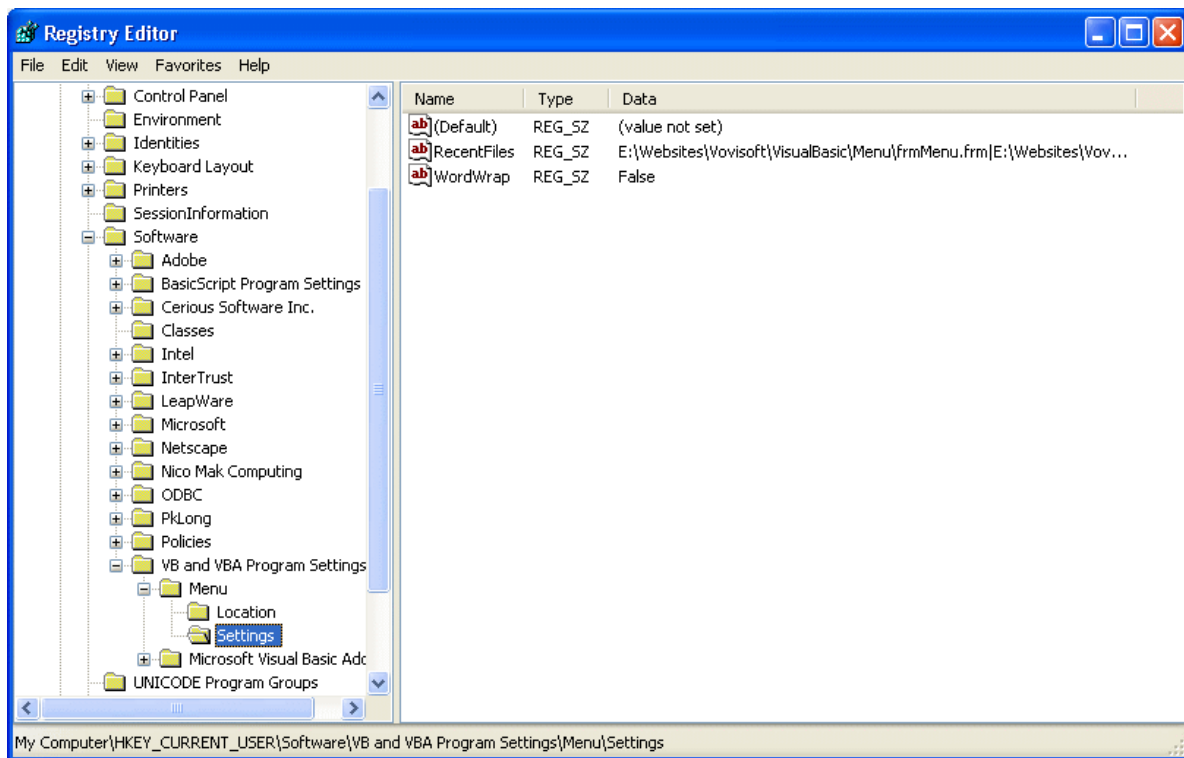
```

Next
End If
End Sub

Bạn có thể chạy Line Command **RegEdit** sau khi click **Start | Run**



để xem chi tiết của các Keys mà program đã chứa trong Sections **Location** và **Settings** của Folder **HKEY_CURRENT_USER\Software\VB and VBA Program Settings\Menu**



Chương Mười Một - Dùng Dialogs

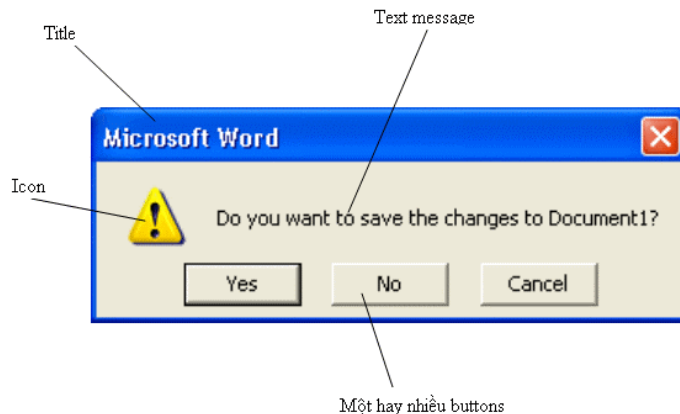
Dialogs (giao thoại) được dùng để hiển thị tin tức và nhận mouse hay keyboard input từ users tùy theo tình huống. Chúng được dùng để tập trung sự chú ý của users vào công tác đương thời của program nên rất hữu dụng trong các chương trình của Windows.

Có nhiều dạng Dialogs, mỗi thứ áp dụng cho một hoàn cảnh riêng biệt. Trong chương này ta sẽ bàn qua 4 loại Dialogs chính và nghiên cứu về khi nào và cách nào ta dùng chúng:

1. Message Boxes
2. Input Boxes
3. Common Dialogs
4. Custom Dialogs

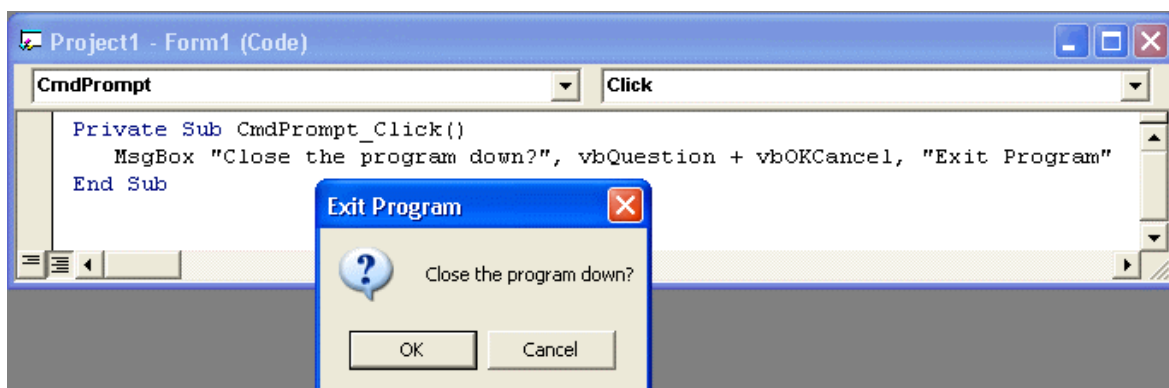
Message Boxes

Message Boxes được dùng để nhắc nhở user một chuyện gì, và đòi hỏi một phản ứng nào đó từ user. Thí dụ như khi ta chấm dứt program MSWord mà chưa lưu trữ hồ sơ thì MSWord sẽ nhắc ta lưu trữ nó bằng Dialog dưới đây:



Trong trường hợp này user có thể click một trong 3 buttons. Nếu click **Yes** thì sẽ xúc tiến việc lưu trữ hồ sơ trước khi kết thúc program MSWord. Nếu click **No** thì MSWord sẽ lặng lẽ kết thúc. Nếu click **Cancel** thì có nghĩa user đổi ý việc chấm dứt program và trở lại tiếp tục dùng MSWord.

Ta dùng routine **MsgBox** để hiển thị Message Box như coding trong hình dưới đây:



Parameter (thông số) thứ nhất của MsgBox là text message **Close the program down?**, parameter thứ nhì là tập hợp của icon (vbQuestion) và số buttons (vbOKCancel) bằng cách cộng hai constants: **vbQuestion + vbOKCancel** (hai buttons OK và Cancel), parameter thứ ba là title (tiêu đề) của Dialog.

Trong thí dụ MSWord bên trên Constant của icon và buttons là **vbExclamation + vbYesNoCancel**

(ba buttons Yes, No và Cancel).

Ta chọn số và loại buttons theo bảng dưới đây:

Constant	Các buttons
vbOKOnly	OK
vbOKCancel	OK Cancel
vbYesNo	Yes No
vbRetryCancel	Retry Cancel
vbYesNoCancel	Yes No Cancel
vbAbortRetryIgnore	Abort Retry Ignore

Constant của các icons ta có thể dùng là **vbCritical**, **vbQuestion**, **vbExclamation** và **vbInformation**. Khi một Message Box được mở ra, cả program ngừng lại và đợi user phản ứng. Ta nói Message Box được hiển thị trong **Modal Mode**, nó dành mọi sự chú ý và tạm ngưng các execution khác trong cùng program. Sau khi user click một button, Message Box sẽ biến mất và program sẽ tiếp tục chạy từ hàng code ngay dưới hàng MsgBox.

Trong thí dụ trên ta dùng MsgBox như một Sub, nhưng ta cũng có thể dùng MsgBox như một Function để biết user vừa mới click button nào. Function MsgBox returns một value (trả về một giá trị) mà ta có thể thử biết để theo đó thì hành. Thí dụ như:

```
Private Sub CmdPrompt_Click()  
    Dim ReturnValue As Integer  
    ReturnValue = MsgBox("Close the program down", vbQuestion + vbOKCancel, "Exit  
Program")  
    Select Case ReturnValue  
        Case vbOK  
            MsgBox "You clicked OK"  
        Case vbCancel  
            MsgBox "You clicked Cancel"  
    End Select  
End Sub
```

Các trị số Visual Basic intrinsic constants mà Function MsgBox returns là:

Trị số	Tên	Const
1	OK	VbOK
2	Cancel	vbCancel
3	Abort	vbAbort
4	Retry	vbRetry
5	Ignore	vbIgnore
6	Yes	vbYes
7	No	VbNo

Bạn có thể hiển thị Text message trong Message Box thành nhiều hàng bằng cách dùng Constant **vbCrLf** (CarriageReturn và LineFeed) để đánh dấu những chỗ ngắt khúc như sau:

```
MsgBox "This is the first line" & vbCrLf & " followed by the second line"
```

Nếu bạn thấy mình thường dùng MsgBox với cùng một icon và những buttons, nhưng có Text message khác nhau, bạn có thể viết một Global Subroutine trong .BAS module để dùng lại nhiều lần. Thí dụ bạn có một Global Sub như sau:

```
Public Sub DisplayError(ByVal ErrMess As String )
    MsgBox ErrMess, vbCritical + vbOKOnly, "Error"
End Sub
```

Mỗi lần muốn hiển thị một Error message bạn chỉ cần gọi Sub DisplayError với Text message mà không sợ dùng lầm lẫn icon. Sau này muốn đổi cách hiển thị Error message chỉ cần edit ở một chỗ. Nếu user muốn bạn lưu trữ tất cả mọi errors xảy ra lúc run-time, bạn chỉ cần thêm vài hàng code trong Sub DisplayError để viết Error message vào một text file.

Input Boxes

Với Message Boxes, user chỉ có thể click lên một button. Đôi khi ta muốn user đánh vào thêm một ít dữ kiện, trong trường hợp ấy ta có thể dùng **Input Boxes**.

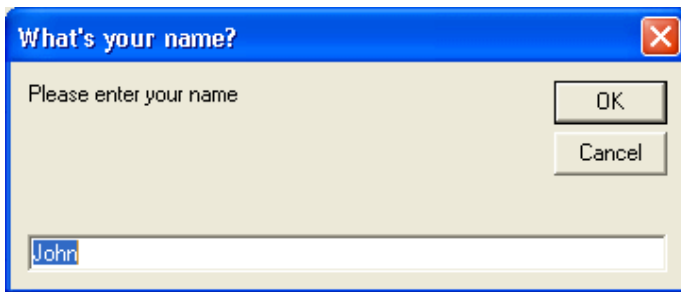
Input Boxes giống giống Message Box, nhưng nó chuyên nhận input data từ user và không hiển thị một icon. Thí dụ:

```

Private Sub CmdGreeting_Click()
    Dim strReply As String
    strReply = InputBox$("Please enter your name", "What 's your name?", "John", 2000, 1000)
    MsgBox "Hi " & strReply & ", it 's great to meet you!", vbOKOnly, "Hello"
End Sub

```

Để ý các parameters của **Function InputBox\$**. Parameter thứ nhất là Text message, parameter thứ hai là Title của Dialog, parameter thứ ba là Default Input Value. Đây là value được hiển thị sẵn trong Input Box khi nó xuất hiện, nếu đó là input user thường đánh vào thì user chỉ cần click nút **OK** là đủ. Hai parameters cuối cùng là Optional (nhiệm ý, có cũng được, không có cũng không sao). Nó là X,Y coordinates của Input Box trong đơn vị **twips**. Hệ thống tọa độ lấy góc trên bên trái làm chuẩn với X=0, Y=0.



Input Box có hai dạng Functions:

- **InputBox\$** - returns một String đăng hoàng
- **InputBox** - returns một String nằm trong Variant variable

Nếu bạn click nút Cancel thì returned Value là empty string, bạn có thể test empty string để nhận diện trường hợp này.

Dưới đây là một thí dụ dùng Function InputBox:

```

Private Sub CmdFortuneTeller_Click()
    Dim varValue As Variant
    Dim intAge As Integer
    varValue = InputBox("Please enter your age", "How old are you?", "18")
    If IsNumeric(varValue) Then
        intAge = Val(varValue)
        If intAge < 20 Then
            MsgBox "You are a young and ambitious person", vbOKOnly, "Observation"
        Else
            MsgBox "You are a matured and wise person", vbOKOnly, "Observation"
        End If
    End If
End Sub

```



```

End If
Else
    MsgBox "Oh oh! - please type your age!", vbCritical + vbOKOnly, "Input Error"
End If
End Sub

```

Khi nào nên dùng Input Boxes

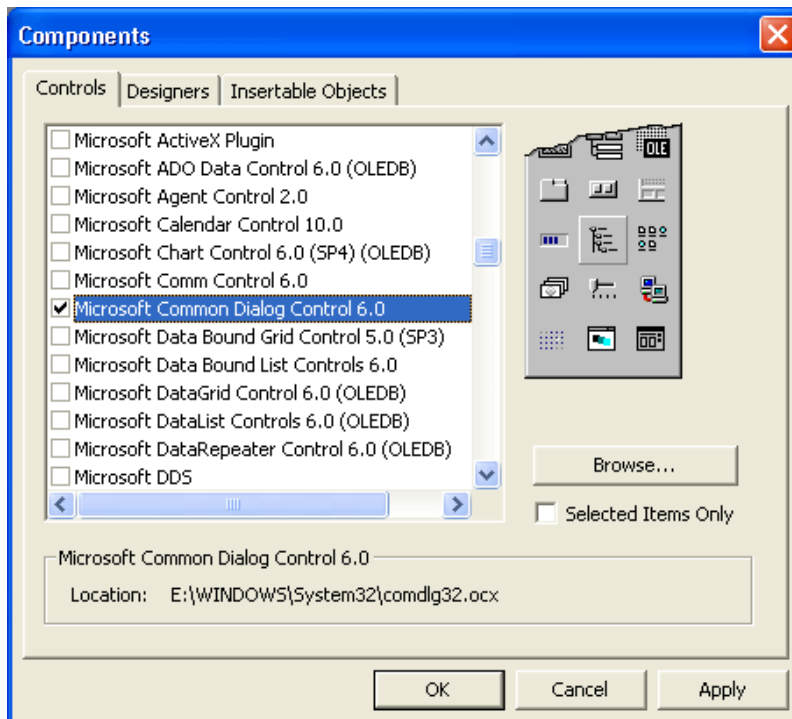
Mặc dầu Input Boxes rất dễ dùng, trên thực tế rất ít khi ta dùng nó vì những lý do sau đây:

- Ta không thể làm gì được trong lúc user input data, phải đợi sau khi user click OK thì mới bắt đầu xử lý input textstring. Ngược lại nếu ta dùng một Textbox trong một Form thông thường, ta có thể code trong các Event handlers của Events **KeyPress** hay **Change** để kiểm soát các keystrokes của user.
- Input Boxes chỉ cho ta đánh vào một text string duy nhất. Nhiều khi ta muốn user đánh vào nhiều thứ nên cần phải có một form riêng.
- Sau cùng, Input Boxes xem không đẹp mắt. Program dùng Input Boxes có vẻ như không chuyên nghiệp, do đó ta cần phải dùng **Custom Dialogs**.

Common Dialogs

Bạn có để ý thấy hầu như mọi programs trong Windows đều có cùng những dialogs để Open và Save files ? Và hầu như tất cả programs đều có cùng dialogs để chọn màu, font chữ hay để in ? Đó là vì các Dialogs thông dụng ấy thuộc về Common Dialog Library của MSWindows và cho phép các program gọi.

Muốn dùng các Dialogs ấy trong VB6 ta phải reference **Comdlg32.ocx** bằng IDE Menu command **Project | Components...** rồi chọn và Apply **Microsoft Common Dialog Control 6.0**.



Microsoft Common Dialog Control 6.0 cho ta sáu dạng Dialogs tùy theo gọi Method nào:

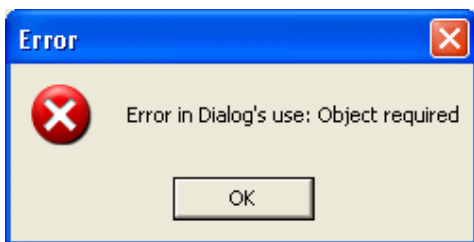
Tên	Method
Open File	ShowOpen
Save File	ShowSave
Color	ShowColor
Font	ShowFont
Print	ShowPrinter
Help	ShowHelp

Open và Save File Dialogs

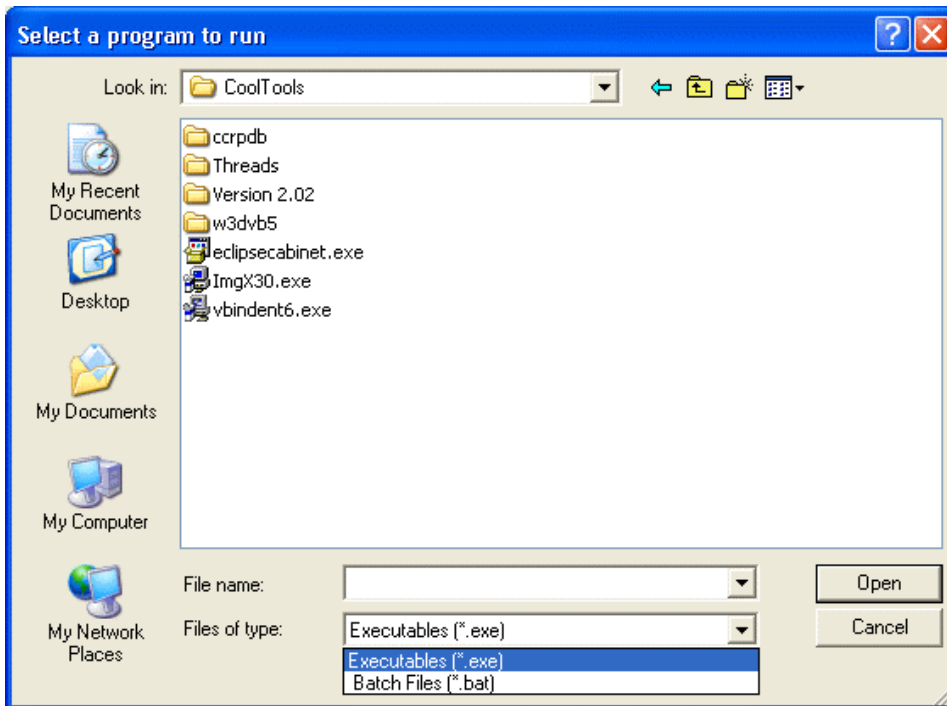
Bạn hãy mở một Project mới với một button tên CmdOpen trong Form1 và đánh vào code sau đây cho Sub CmdOpen_Click:

```
Private Sub CmdOpen_Click()
    On Error GoTo DialogError
    With CommonDialog1
        .CancelError = True ' Generate Error number cdlCancel if user click Cancel
        .InitDir = "E:\VB6" ' Initial (i.e. default ) Folder
        .Filter = "Executables (*.exe) | *.exe| Batch Files (*.bat)| *.bat"
        .FilterIndex = 1 ' Select ""Executables (*.exe) | *.exe" as default
        .DialogTitle = "Select a program to run"
        .ShowOpen ' Launch the Open Dialog
        MsgBox "You selected " & .FileName, vbOKOnly + vbInformation, "Open Dialog"
    End With
    Exit Sub
DialogError:
    If Err.Number = cdlCancel Then
        MsgBox "You clicked Cancel!", vbOKOnly + vbInformation, "Open Dialog"
        Exit Sub
    Else
        MsgBox "Error in Dialog's use: " & Err.Description, vbOKOnly + vbCritical, "Error"
        Exit Sub
    End If
End Sub
```

Hãy chạy program ấy và click button **Open**, program sẽ hiển thị error message dưới đây:



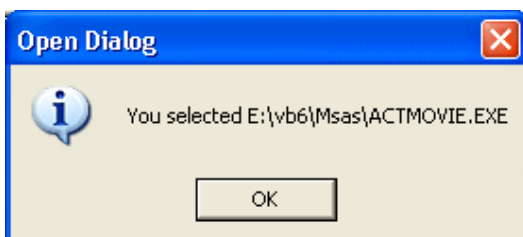
Đó là vì ta quên bỏ một Microsoft Common Dialog Control 6.0 vào Form1. Vậy bạn hãy doubleclick icon của nó trong ToolBox. Bây giờ hãy chạy program lại và click button Open để hiển thị **Open Dialog**.



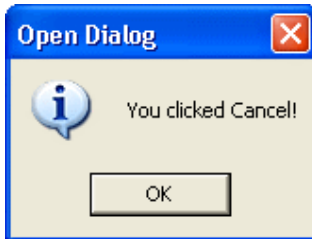
Bạn có thể chọn folder nào tùy ý bằng cách di chuyển từ folder này qua folder khác hay thay đổi disk drive. Nếu bạn click vào bên phải của combobox **File of type**, nó sẽ dropdown để cho thấy bạn có thể chọn một trong hai loại Files như liệt kê trong statement:

```
.Filter = "Executables (*.exe) | *.exe| Batch Files (*.bat)| *.bat"
```

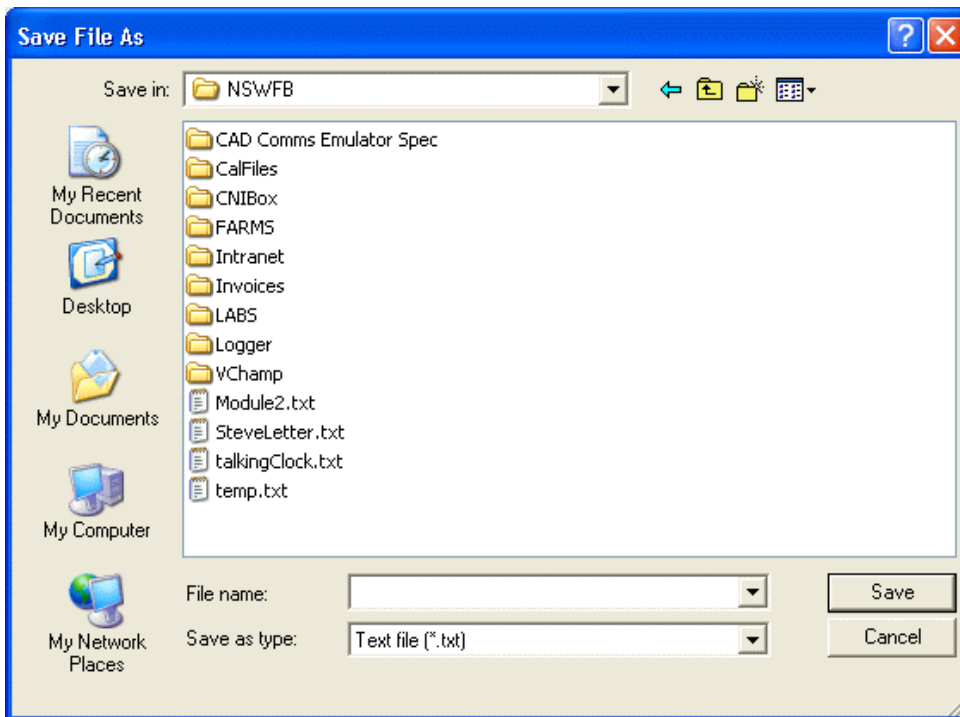
Sau khi chọn một Filename có sẵn hay đánh một tên vào **File name** textbox, bạn click **Open**. Sau đó, `CommonDialog1.FileName` sẽ chứa tên file bạn đã chọn hay đánh vào.



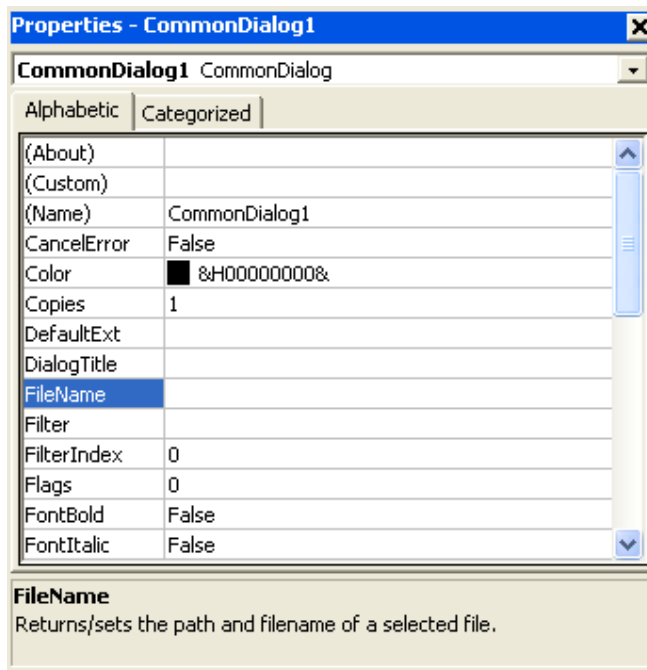
Vì ta cho **.CancelError = True** nên nếu user click Cancel program sẽ generate một Error số **32755 (cdlCancel)**. Ở đây ta bắt Error ấy bằng cách dùng **On Error GoTo DialogError** và thử Err.Number= cdlCancel để hiển thị Error message dưới đây:



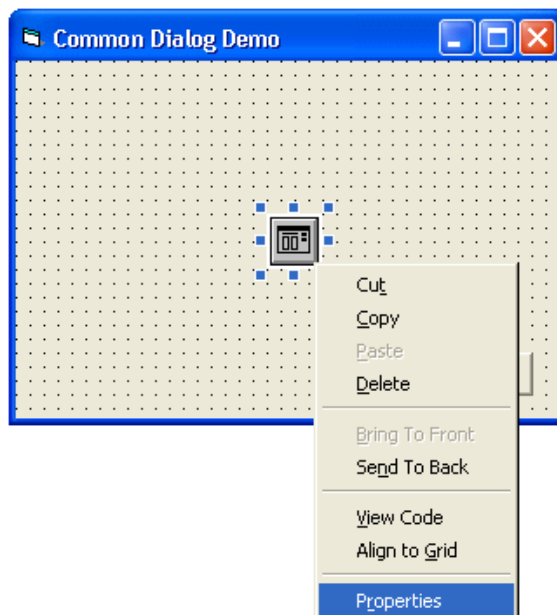
Save Dialog cũng tương tự như Open Dialog, ta dùng method **ShowSave** để hiển thị nó.



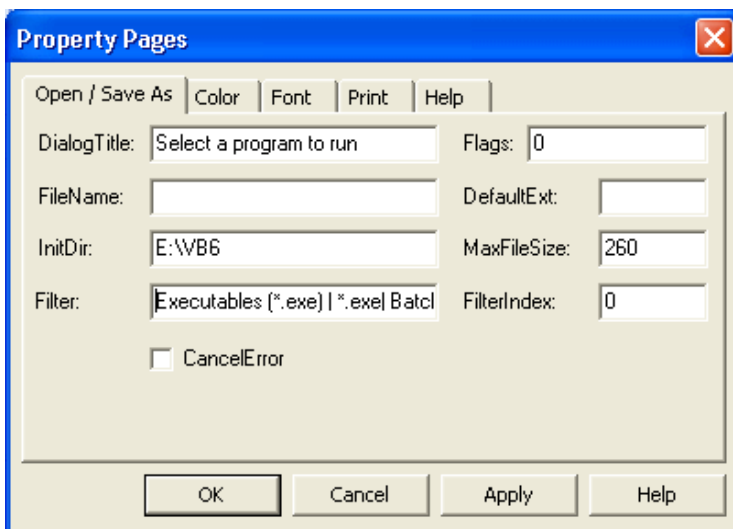
Trong thí dụ trên ta định nghĩa các properties của CommonDialog1 bằng code. Bạn cũng có thể dùng Properties Windows để định nghĩa chúng như dưới đây:



Ngoài ra, bạn cũng có thể dùng các trang Properties của CommonDialog1 để định nghĩa Properties lúc thiết kế bằng cách right click CommonDialog1 trên Form1 rồi chọn **Properties**:

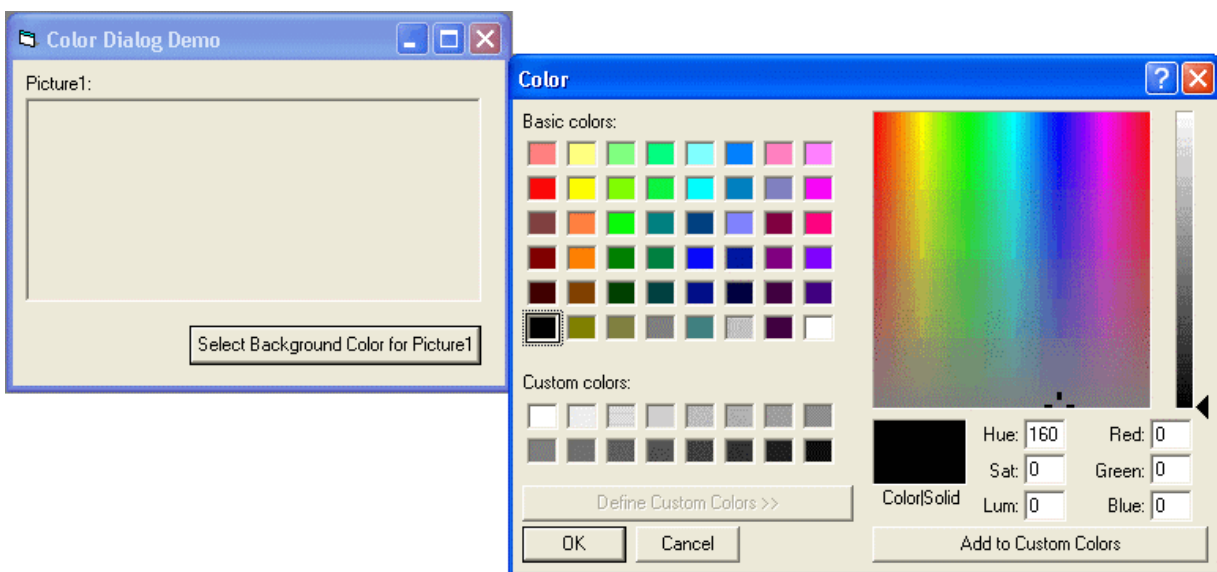


Properties Pages Dialog sẽ hiển thị với Tab **Open/Save As** có sẵn lúc đầu, bạn có thể đánh các tin tức như sau:

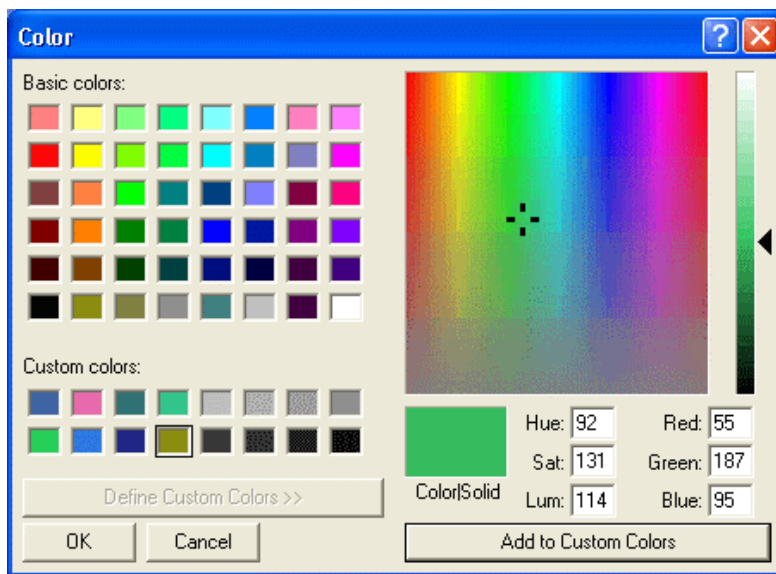


Color Dialog

Color Dialog cho user một cách chọn màu rất dễ dùng. Ngoài những màu có sẵn, user có thể tự tạo ra một màu rồi cho nó thêm vào trong bảng màu được cung cấp, gọi là **Windows Palette** bằng cách click button **Add to Custom Colors**.



Bạn tạo ra một màu bằng cách click chỗ có màu theo ý trong bảng màu lớn hình vuông rồi nắm hình tam giác bên phải kéo lên, kéo xuống để thay đổi độ đậm của màu như hiển thị trong hộp vuông **ColorSolid**. Khi vừa ý với màu hiển thị, bạn click button **Add to Custom Colors**, màu ấy sẽ được cho thêm vào nhóm **Custom Colors** nằm phía dưới, bên trái.



Ta dùng method **ShowColor** để hiển thị Color Dialog. Sau khi user đã chọn một màu rồi, ta có thể trực tiếp assign nó cho property ForeColor hay BackColor của một control. Trong thí dụ dưới đây cái màu mà user vừa chọn được assigned cho background của picturebox Picture1:

```
Private Sub CmdSelectColor_Click()
    On Error GoTo NoColorChosen
    With CommonDialog1
        .CancelError = True
        ' Entire dialog box is displayed, including the Define Custom Colors section
        .Flags = cdlCCFullOpen
        .ShowColor ' Launch the Color Dialog
        Picture1.BackColor = .Color ' Assign selected color to background of Picture1
    End With
    Exit Sub
NoColorChosen:
    ' Get here if user clicks the Cancel button
    MsgBox "You did not select a color!", vbInformation, "Cancelled"
    Exit Sub
End Sub
```

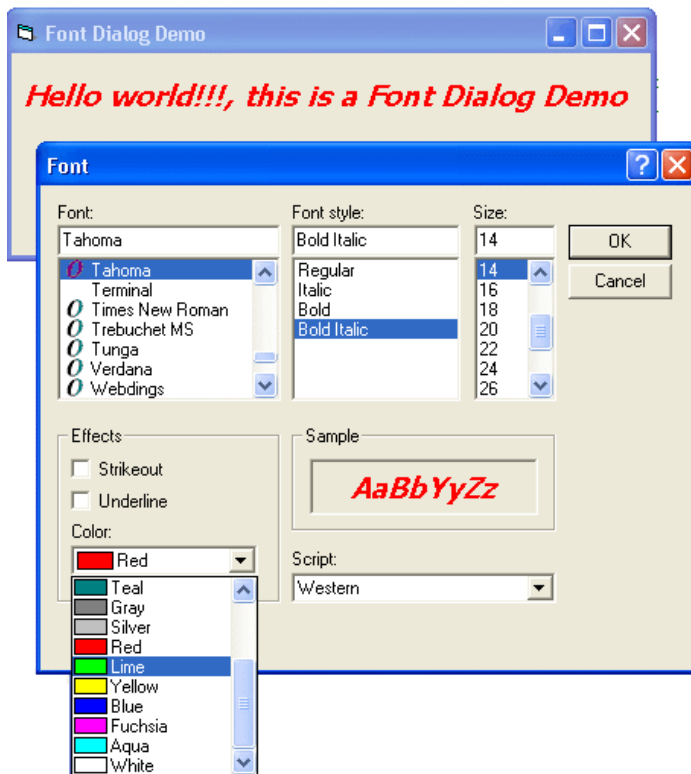
Font Dialog

Font Dialog cho ta chọn Font cho màn ảnh hay printer và chọn màu để dùng cho chữ của Font. Ta dùng method **ShowFont** để hiển thị FontDialog. Các chi tiết trình bày trong Font Dialog tùy thuộc vào trị số của Flags như sau:

Constant	Trị số	Hiệu quả
cdlCFScreenFonts	1	Chỉ hiển thị các Fonts printer hỗ trợ

cdlCFPrinterFonts	2	Chỉ hiển thị các Fonts của màn ảnh, chưa chắc tất cả đều được printer hỗ trợ
cdlCFBoth	3	Hiển thị các Fonts màn ảnh và printer
cdlCFScalableOnly	&H20000	Chỉ hiển thị các scalable Fonts như TrueType fonts mà bạn đã cài vào máy

Nếu bạn muốn cho user nhiệm ý để chọn màu thì thêm 256 vào trị số của Flags.



Dưới đây là code để cho user chọn Font và màu của Label1.

```
Private Sub CmdSelectFont_Click()
    On Error GoTo NoFontChosen
    CommonDialog1.CancelError = True
    ' Causes the dialog box to list only the screen fonts supported by the system.
    CommonDialog1.Flags = cdlCFScreenFonts + 256 ' Add 256 to include Color option
    CommonDialog1.ShowFont ' Launch the Font Dialog
    With Label1.Font
        .Bold = CommonDialog1.FontBold
        .Italic = CommonDialog1.FontItalic
        .Name = CommonDialog1.FontName
        .Size = CommonDialog1.FontSize
        .Strikethrough = CommonDialog1.FontStrikethru
    End With
End Sub
```

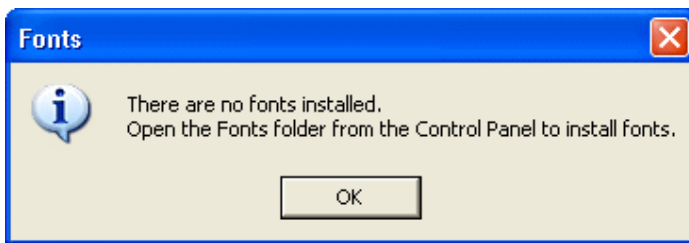


```

.Underline = CommonDialog1.FontUnderline
End With
Label1.ForeColor = CommonDialog1.Color
Label1.Caption = "Hello world!!!, this is a Font Dialog Demo"
Exit Sub
NoFontChosen:
MsgBox "No font was chosen!", vbInformation, "Cancelled"
Exit Sub
End Sub

```

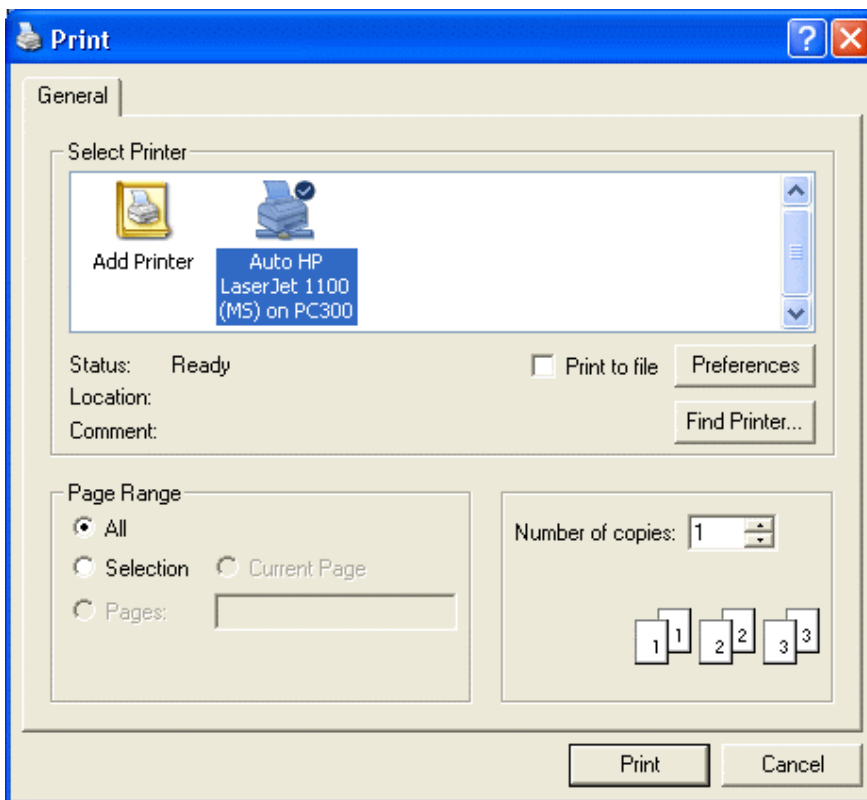
Chú ý: Nếu bạn quên cho Flags một trong những hằng số nói trên program sẽ cho một Error message như sau:



Print Dialog

Print Font cho ta một giao diện cũng giống như trong Microsoft Office để chọn những nhiệm ý về việc in. Với Print Dialog ta có thể chọn printer nào với những đặc tính nào bằng cách click button **Properties** hay button **Preferences**. Ta cũng có thể quyết định in từ trang nào đến trang nào của document và in bao nhiêu copies. Chỉ có điều phải lưu ý là nếu user dùng Print Dialog để chọn một Printer khác mà trong Print Dialog ta đã chọn Property **PrinterDefault = True** thì Printer ấy sẽ trở thành Default Printer và nó cũng sẽ có hiệu lực vĩnh viễn trong cả Windows cho đến khi user thay đổi lại.

Khác với Color và Font Dialogs, Print Dialog không đòi hỏi ta phải cho một trị số của Property Flags. Ta chỉ cần dùng Method **ShowPrinter** để hiển thị Print Dialog. Ba properties thường được dùng nhất sau khi user chọn các nhiệm ý của Print Dialog là **Copies**, **FromPage** và **ToPage**. Để cho user các default values của những properties này, bạn có thể để sẵn các trị số trước khi hiển thị Print Dialog.



Dưới đây là code mẫu dùng print Dialog:

```
Private Sub CmdSelectPrinter_Click()
    With CommonDialog1
        .FromPage = 1
        .ToPage = 1
        .Copies = 1
        .ShowPrinter
    End With
End Sub
```

Help Dialog

Ta dùng method **ShowHelp** để hiển thị các thông tin giúp đỡ, nhưng nhớ phải cho CommonDialog ít nhất trị số của các properties **HelpFile** và **HelpCommand**.

```
Private Sub CmdHelp_Click()
    CommonDialog1.HelpFile = "YourProgram.hlp"
    CommonDialog1.HelpCommand = cdlHelpContents
    CommonDialog1.ShowHelp
End Sub
```

Để biết thêm chi tiết về cách dùng ShowHelp, highlight chữ **HelpContext** trong source code VB6 rồi ấn phím **F1** và chọn **MsComDlg**.

Custom Dialogs

Nhiều khi Message Box, Input Box hay các dạng Common Dialogs vẫn không thích hợp cho hoàn cảnh lập trình. Trong trường hợp ấy bạn có thể dùng một Form bình thường để làm thành một Dialog cây nhà, lá vườn. Nó hơi mất công hơn một chút, nhưng thứ nhất nó có những màu sắc giống như các Forms khác trong chương trình, và thứ hai ta muốn làm gì tùy ý. Chỉ có cái bất lợi là chương trình sẽ dùng nhiều tài nguyên hơn, nói thẳng ra là cần thêm một ít memory.

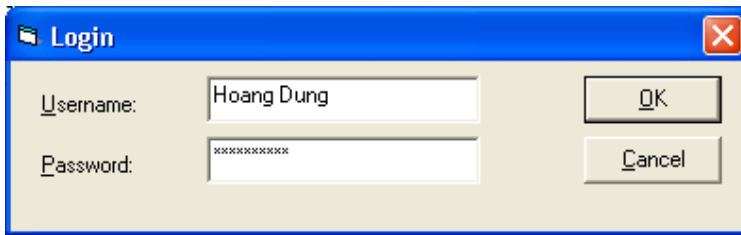
Sau đây ta thử triển khai một Login Form tổng quát, có thể dùng trong nhiều trường hợp. Khi khởi động, program này sẽ hiển thị một Login form yêu cầu user đánh vào tên và mật khẩu. Sau đó, nếu tên và mật khẩu hợp lệ thì cái Form chính của program mới hiện ra. Cách ta thực hiện là cho program khởi động với một **Sub Main** trong .BAS Module. Sub Main sẽ gọi **Sub GetUserInfo** (cũng nằm trong cùng Module) để hiển thị form frmLogin trong Modal mode để nó làm việc cùng một cách như Message Box, Input Box hay Common Dialogs.

Khi form frmLogin được dấu kín bằng statement **Me.Hide** thì execution trong Sub GetUserInfo sẽ tiếp tục để chi tiết điền vào các textboxes txtUserName và txtPassword được trả về local variables strUserName và strPassword. Mã nguồn của Sub Main và Sub GetUserInfo được liệt ra dưới đây:

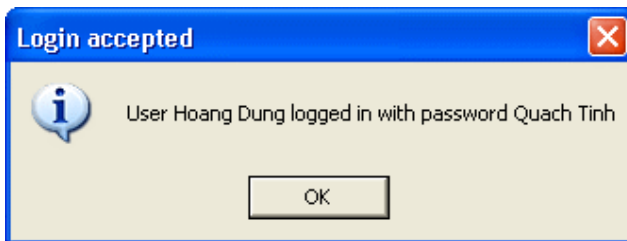
```
Sub Main()  
    Dim strUserName As String  
    Dim strPassword As String  
    ' Call local Sub getUserInfo to obtain UserName and Password  
    GetUserInfo strUserName, strPassword  
    If strUserName = "" Then  
        MsgBox "Login failed or aborted", vbInformation, "login Aborted"  
    Else  
        MsgBox "User " & strUserName & " logged in with password " & strPassword,  
vbInformation, "Login accepted"  
        ' Check UserName and Password here  
        ' If valid password then show the Main form of the program which is implemented  
separately...  
        ' frmMain.Show  
    End If  
End Sub  
  
Private Sub GetUserInfo(ByRef sUserName As String, ByRef sPassword As String)  
    ' Invoke frmLogin form in Modal mode  
    frmLogin.Show vbModal  
    ' As soon as frmLogin is hidden, the execution gets here  
    sUserName = frmLogin.txtUserName ' assign the form's txtUserName to sUserName  
    sPassword = frmLogin.txtPassword ' assign the form's txtPassword to sPassword  
    Unload frmLogin ' Unload form frmLogin
```

End Sub

Login form được hiển thị như dưới đây:



Sau khi user điền chi tiết và click **OK**, tạm thời ta chỉ hiển thị một thông điệp để xác nhận các chi tiết ấy.



Trong tương lai, bạn có thể viết thêm code để kiểm tra xem tên và mật khẩu có hiệu lực không. Có một vài chi tiết về form frmLogin để nó làm việc giống giống một Common Dialog:

- Ta cho **property BorderStyle** của frmLogin là **Fixed Dialog**.
- Ta cho **Property PasswordChar** của textbox txtPassword bằng "*" để khi user điền mật khẩu, ta chỉ thấy một hàng dấu hoa thị.
- Ta cho **Property StartupPosition** của form là **CenterScreen**.
- **Property Default** của button **cmdOK** là True để khi user ấn phím **Enter** trong form là coi như tương đương với click button cmdOK.
- Tương tự như thế, **Property Cancel** của button **cmdCancel** là True để khi user ấn phím **Esc** trong form là coi như tương đương với click button cmdCancel.

Tạm thời coding của event click của cmdOK và cmdCancel chỉ đơn giản như liệt kê dưới đây:

```
Private Sub CmdCancel_Click()  
    ' Clear txtUserName and txtPassword  
    txtUserName = ""  
    txtPassword = ""  
    ' Hide the Form to get out of Modal mode  
    Me.Hide  
End Sub  
  
Private Sub CmdOK_Click()  
    ' Hide the Form to get out of Modal mode  
    Me.Hide
```

Chương Mười Hai - Dùng Đồ Họa (Phần I)

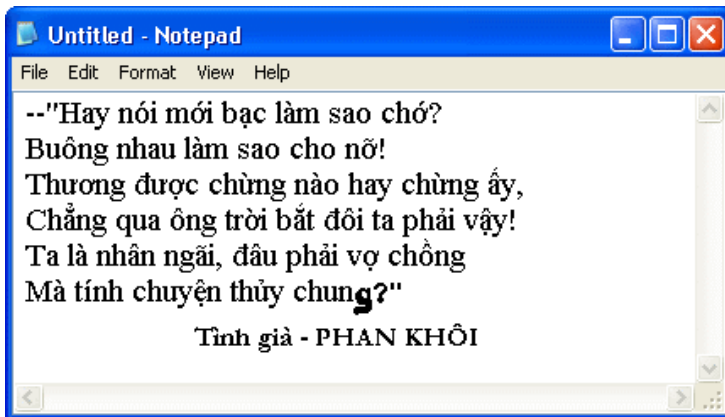
Tục ngữ Anh có câu: "Một hình ảnh đáng giá một ngàn chữ (a picture is worth a thousand words)", ý nói khi ta dùng hình ảnh để diễn tả sẽ giúp người xem hiểu nhanh hơn khi ta chỉ có nói thôi.

Visual Basic 6 có cho ta một số phương tiện về đồ họa (graphics) để trang điểm cho các cửa sổ phong phú, thân thiện, dễ làm việc với, và thú vị. Dù rằng các phương tiện về đồ thị này không nhanh đủ cho ta viết những chương trình trò chơi (games) nhưng tương đối cũng đủ khả năng để đáp ứng các nhu cầu cần thiết thông thường.

Khi nói đến đồ họa, ta muốn phân biệt nó với Text thông thường. Thí dụ ta dùng Notepad để edit một bài thơ trong một cửa sổ. Trong lúc bài thơ đang được hiển thị ta có thể sửa đổi dễ dàng bằng cách dùng bàn phím để đánh thêm các chữ mới vào, dùng các nút **Delete**, **Backspace** để xóa các chữ. Đó là ta làm việc với **Text**.

Bây giờ, trong khi bài thơ còn đang hiển thị, ta dùng một chương trình Graphic như **PhotoImpact Capture** của **ULead** để chụp cái hình cửa sổ của bài thơ (active window) thành giống như một photo, thì ta có một **Graphic**. Sau đó, muốn sửa đổi bài thơ từ graphic này ta phải dùng một graphic editor như **MSPaint**, **PaintShopPro**, v.v.. Các chữ trong hình cũng có cùng dạng graphic như ta thấy một photo, nên muốn edit phải dùng một cọ với màu sơn.

Dưới đây là graphic của một cửa sổ Notepad sau khi được thêm chữ **g** và dấu chấm hỏi ở cuối bằng cách dùng MSPaint.



Màu (color) và độ mịn (resolution)

Ta nói một tấm hình tốt vì nó có màu sắc sảo và rõ ràng. Bạn có còn nhớ trong ngày Lễ khai mạc Thế Vận Hội Moscow, người ta cho hiển thị nhiều hình rất hay bằng cách nhờ khán giả, trong một khu hình chữ nhật, mỗi người cầm đưa lên một tấm cạt-tông màu. Hàng ngàn tấm cạt-tông đưa lên ráp lại thành ra một hình tuyệt đẹp.

Một graphic trong Windows cũng gồm có nhiều đốm nhỏ, mỗi đốm, được gọi là một **pixel**, có khả năng hiển thị 16, 256, ... màu khác nhau.

Độ mịn (resolution)

Thông thường độ mịn (resolution) của màn ảnh ta dùng là 800x600, tức là chiều ngang có 800 pixels và chiều cao có 600 pixels. Sau này, để xem các hình rõ hơn ta còn dùng độ mịn 1028x768 với cạt SuperVGA và Monitor tốt. Ta nói cạt SuperVGA có đến 2MB RAM, tại sao phải cần đến 2MB để hiển thị graphic đẹp?

Nếu màu của mỗi pixel được biểu diễn bởi một byte dữ kiện thì với một byte ta có thể chứa một con số từ 0 đến 255. Người ta đồng ý với nhau theo một quy ước rằng số 0 tượng trưng cho màu đen, số 255 tượng trưng cho màu trắng chẳng hạn. Nếu độ mịn của màn ảnh là 1024x768 thì ta sẽ cần $1024 \times 768 = 786432$ bytes, tức là gần 0,8 MB.

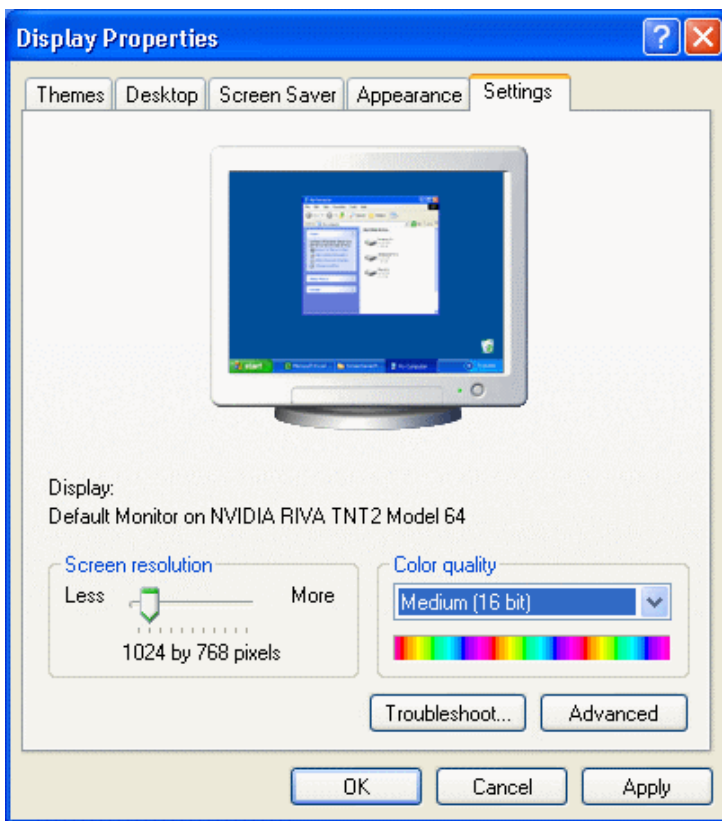
Một byte có 8 bits. Đôi khi ta nghe nói 16 bit color, ý nói thay vì một byte, người ta dùng đến 2 bytes cho mỗi pixel. Như vậy mỗi pixel này có khả năng hiển thị $2^{16} = 65536$ màu khác nhau. Muốn dùng 16 bit color cho SuperVGA, ta cần phải có $1024 \times 768 \times 2 = 1572864$ bytes, tức là gần 1,6 MB. Đó là lý do tại sao ta cần 2MB RAM. Lưu ý là RAM của VGA (Vector Graphic Adapter) card không liên hệ gì với RAM của bộ nhớ computer.

Không ngờ các cụ Ăng-Lê ngày xưa đã biết Tin Học nên nói trước: "Một hình ảnh đáng giá một ngàn chữ". Chữ **word** thời IBM gồm có 4 bytes, nên một màn ảnh đáng giá 400 ngàn chữ, như vậy các cụ không chính xác lắm, nhưng coi như đúng.

Nên nhớ rằng cùng một graphic hiển thị trên hai màn ảnh có cùng độ mịn, thí dụ như 800x600, nhưng kích thước khác nhau, thí dụ như 14 inches và 17 inches, thì dĩ nhiên hình trên màn ảnh 17 inches sẽ lớn hơn, nhưng nó vẫn có cùng một số pixels, có điều pixel của nó lớn hơn pixel của màn ảnh 14 inches.

Nói một cách khác, nếu ta dùng màn ảnh lớn hơn thì graphic sẽ lớn hơn nhưng không có nghĩa là nó rõ hơn. Muốn thấy rõ chi tiết, ta phải làm cho graphic có độ mịn cao hơn. Trở lại câu chuyện Thế Vận Hội Moscow, muốn có hình rõ hơn, thì trong cùng một diện tích, ta phải nhồi bừa con ngòi xịch lại gần nhau để khoảng đất chứa nhiều người hơn và mỗi người cầm một tấm cạt-tông nhỏ hơn.

Ta thay đổi **Display Properties** của một màn ảnh bằng cách right click lên desktop rồi select **Properties**, kế đó click Tab **Settings** rồi chọn **Screen resolution** và **Color quality** giống như hình dưới đây:



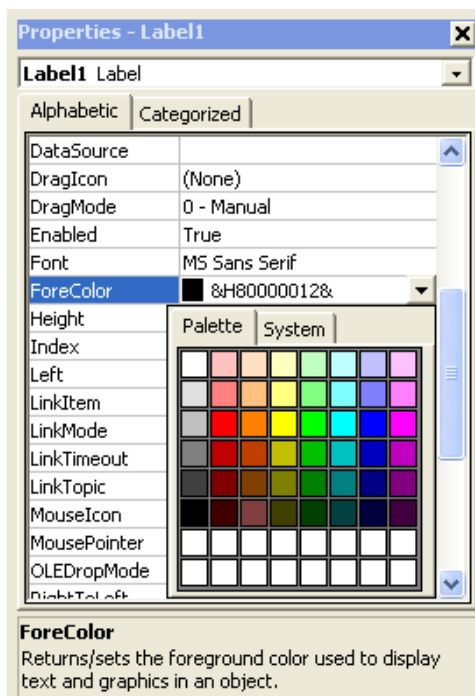
Khi ta tăng độ mịn của màn ảnh, các hình ảnh sẽ nhỏ lại vì kích thước của pixel được thu nhỏ lại. Do đó, ta có thể cho hiển thị nhiều thứ hơn trên desktop. Phẩm chất của các graphic vẫn không thay đổi, mặc dầu hình nhỏ hơn. Nhớ là muốn hình rõ hơn thì khi cấu tạo và chứa graphic, ta phải dùng một độ mịn cao. Giống như khi chụp hình, muốn hình đẹp ta cần cái máy chụp hình dùng phim lớn của thợ chuyên nghiệp và focus kỹ lưỡng, thay vì dùng máy rẻ tiền tự động, chỉ đưa lên là bấm chụp được.

Màu (color)

Khi ta dùng chỉ có một bit (chỉ có trị số 0 hay 1) cho mỗi pixel thì ta chỉ có trắng hay đen. Lúc ấy ta có thể dùng một byte (8 bits) cho 8 pixels. Dầu vậy, nếu độ mịn của graphic cao đủ, thì hình cũng đẹp. Thử xem các tuyệt tác photos trắng đen của Cao Đàm, Cao Lĩnh thì biết. Các máy Fax dùng nguyên tắc scan hình giấy cỡ A4 ra thành những pixels trắng đen rồi gởi qua đường dây điện thoại qua đầu kia để tái tạo lại hình từ những dữ kiện pixels.

Visual Basic 6 cho ta chỉ định một con số vào mỗi màu VB có thể hiển thị, hay chọn trực tiếp một màu từ Dialog. Có bốn cách:

- Bạn chỉ định trực tiếp một con số hay chọn một màu từ cái Palette.



- Bạn chọn một trong các hằng số định nghĩa sẵn trong VB, gọi là **intrinsic color constants** (intrinsic có nghĩa nôm na là cây nhà lá vườn hay in-built), chẳng hạn như **vbRed** , **vbBlue**. Danh sách của intrinsic color constants lấy từ VB6 online help được liệt kê dưới đây:

Constant	Value	Description
vbBlack	0x0	Black
vbRed	0xFF	Red
vbGreen	0xFF00	Green
vbYellow	0xFFFF	Yellow
vbBlue	0xFF0000	Blue
vbMagenta	0xFF00FF	Magenta
vbCyan	0xFFFF00	Cyan
vbWhite	0xFFFFFFFF	White

- Dùng **Function QBColor** để chọn một trong 16 màu. Function QBColor xuất phát từ thời Quick Basic (QBasic) của Microsoft. QBasic là tiền thân của Visual Basic. Trong QBasic bạn có thể dùng các con số 1,2,3 .. để chỉ định các màu Blue, Green, Cyan , .v.v..Function QBColor giản tiện hóa cách dùng màu, user không cần phải bận tâm về cách trộn ba thứ màu căn bản Red, Green, Blue. Bạn viết code một cách đơn giản như:

```
Label1.ForeColor = QBColor(2)
QBColor(Color As Integer) As Long
```

Dưới đây là trị số các màu ta có thể dùng với Function QBColor.

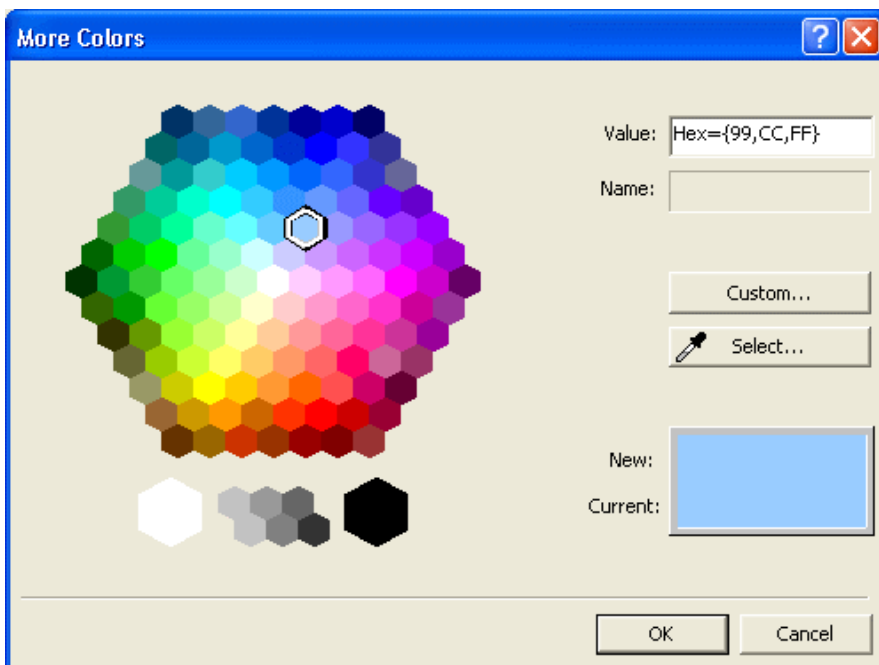
Trị số	Màu	Trị số	Màu
--------	-----	--------	-----

0	Black	8	Gray
1	Blue	9	Light Blue
2	Green	10	Light Green
3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Yellow	14	Light Yellow
7	White	15	Bright White

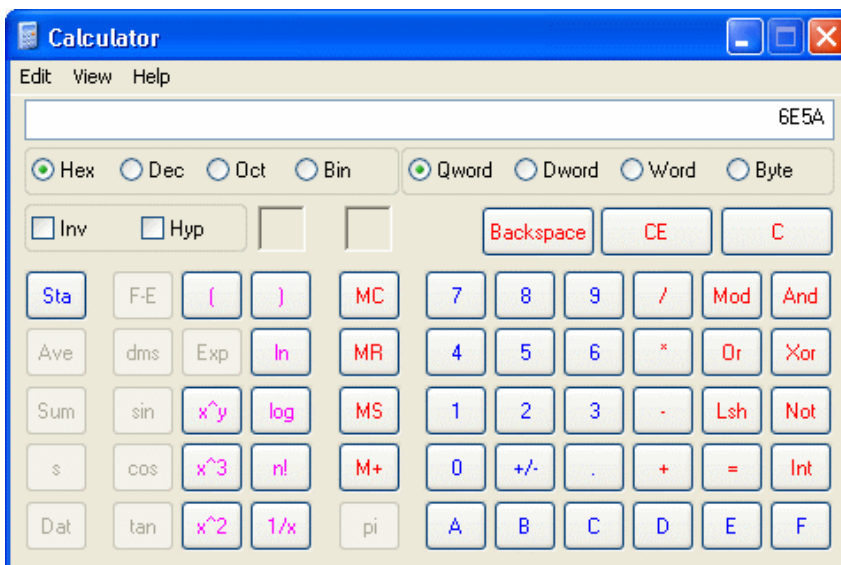
- Dùng **Function RGB** để trộn ba màu **Red**, **Green** và **Blue**. Trong cái bảng liệt kê các intrinsic color constants phía trên, nếu để ý bạn sẽ thấy vbWhite(0xFFFFFFFF) là tổng số của vbRed(0x0000FF), vbGreen(0x00FF00) và vbBlue(0xFF0000). Một màu được biểu diễn bằng sự pha trộn của ba thành phần màu căn bản, mỗi màu bằng một byte có trị số từ 0 đến 255. 0 là không dùng màu ấy, 255 là dùng tối đa màu ấy.

Hệ thống số ta dùng hằng ngày là Thập Phân. Trị số 0xFF của vbRed là con số 255 viết dưới dạng Thập lục phân (Hexadecimal hay **Hex** cho gọn và ở đây được đánh dấu bằng **0x** trước con số để phân biệt với số Thập phân). Trong hệ thống số Hex ta đếm từ 0 đến 9 rồi A,B,C,D,E,F rồi qua số hàng thập lục 10, 11,..., 19, 1A, 1B, ..1E, 1F, 20, 21..v.v. Tức là thay vì chỉ dùng 10 symbols từ 0 đến 9 trong Thập phân, ta dùng 16 symbols từ 0 đến F. Muốn biết thêm về hệ thống số Hex hãy đọc bài [Cơ số Nhị Phân](#).

Trong hình dưới đây là một thí dụ cho thấy màu xanh nhạt đã được chọn gồm ba thành phần Blue(0x990000= 153*256*256), Green(0xCC00= 204*256) và Red(0xFF= 255):



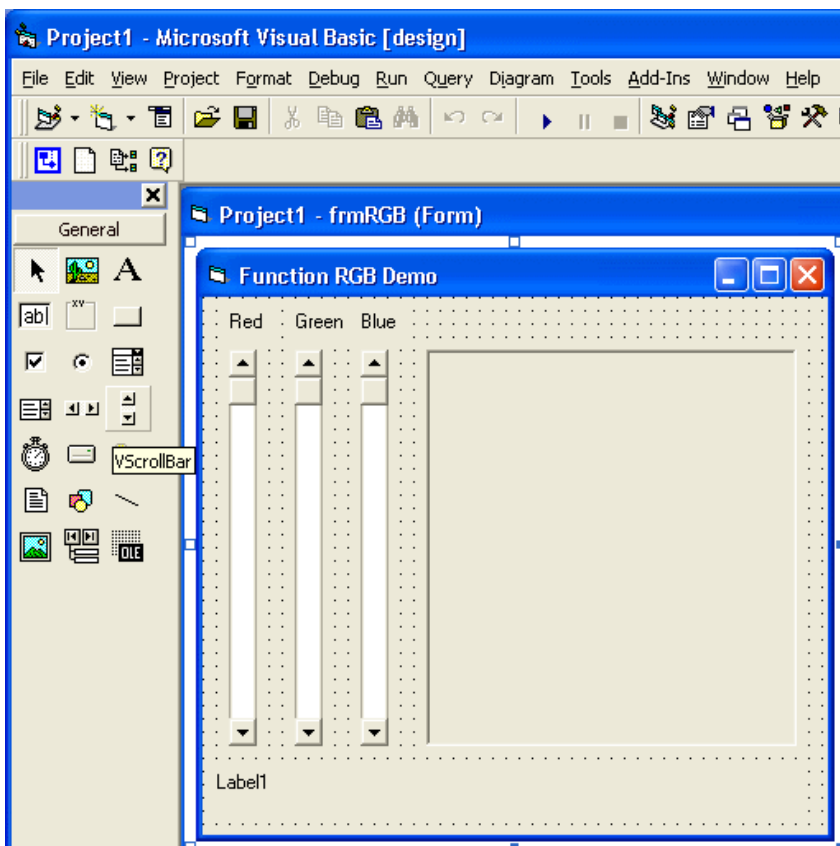
Ghi chú: Bạn có thể dùng Windows Calculator để hoán chuyển số giữa các dạng Decimal, Binary và Hexadecimal. Chọn **View|Scientific** thay vì **View|Standard**.



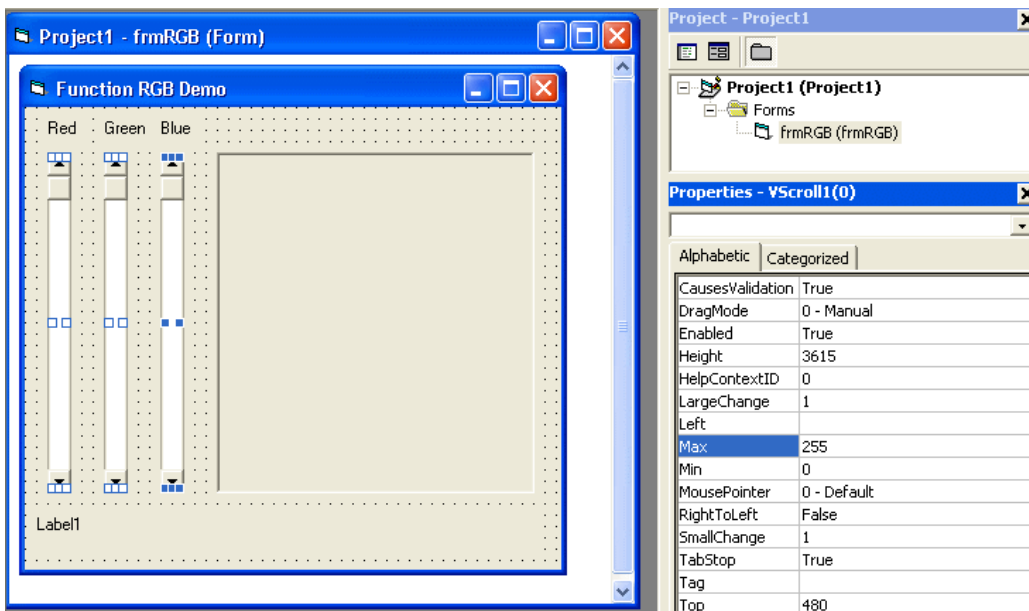
Function RGB

Để áp dụng Function RGB, ta sẽ viết một chương trình VB6. Bạn hãy khởi động một chương trình VB6 mới, bỏ vào một Label tên Label1 với Caption **Red** và một Vertical Scroll tên **VScroll1**. Kế đó select cả hai Label1 và VScroll1 rồi Copy và Paste hai lần để là thêm hai cặp. Đổi Caption của hai Label mới này ra **Green** và **Blue**. Bây giờ ta có một Array ba Vertical Scrolls cùng tên VScroll1, với index là 0,1 và 2.

Đặt một PictureBox tên **picColor** vào bên phải ba cái VScrolls. Thêm một Label phía dưới, đặt tên nó là **lblRGBValue**, nhớ clear caption của nó, đừng có để chữ Label1 như dưới đây:



Bây giờ select cả ba VScrolls và edit value của **property Max** trong cửa sổ Properties thành **255**, ý nói khi kéo cái bar của một VScroll1 lên xuống ta giới hạn trị số của nó từ Min là 0 đến Max là 255.



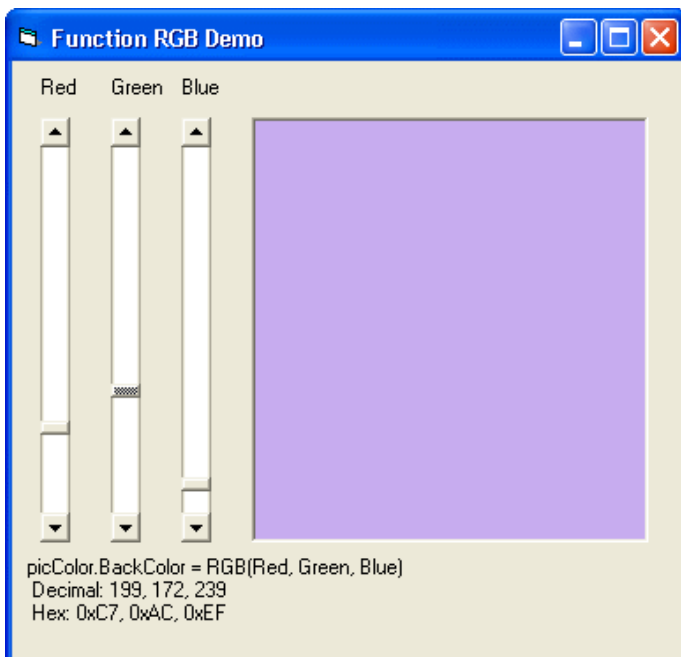
Chuyện chính ta phải làm là viết code để xử lý **Event Change** của các VScrolls. Vì chúng là một Array nên ta có thể dùng một Sub duy nhất để handle events đến từ cả ba VScrolls. Mỗi lúc một trong 3 VScrolls thay đổi trị số ta sẽ trộn ba màu Red, Green, Blue biểu diễn bởi trị số của 3 VScrolls thành màu **BackColor** của PictureBox **picColor**. Đồng thời ta cho hiển thị trị số của ba thành phần màu Red, Green và Blue trong Label **lblRGBValue**. Bạn hãy double click lên một trong 3 VScrolls rồi viết code như sau:

```

Private Sub VScroll1_Change(Index As Integer)
    ' Use Function RGB to mix 3 colors VScroll1(0) for Red,
    ' VScroll1(1) for Green and VScroll1(2) for Blue
    ' and assign the result to BackColor of PictureBox picColor
    picColor.BackColor = RGB(VScroll1(0).Value, VScroll1(1).Value, VScroll1(2).Value)
    ' Variable used to prepare display string
    Dim strRGB As String
    ' Description of what is displayed
    strRGB = "picColor.BackColor = RGB(Red, Green, Blue) " & vbCrLf
    ' Values of Red, Green, Blue in Decimal
    strRGB = strRGB & " Decimal: " & VScroll1(0).Value & ", " & VScroll1(1).Value & ", " & VScroll1(2).Value & vbCrLf
    ' Values of Red, Green, Blue in Hexadecimal
    strRGB = strRGB & " Hex: 0x" & Hex(VScroll1(0).Value) & ", 0x" & Hex(VScroll1(1).Value) & ", 0x" & Hex(VScroll1(2).Value)
    ' Assign the resultant string to caption of Label lblRGBValue
    lblRGBValue.Caption = strRGB
End Sub

```

Bạn hãy khởi động chương trình rồi nắm các bar của 3 VScrolls kéo lên, kéo xuống để xem kết quả. Cửa sổ của chương trình sẽ có dạng giống như dưới đây:



Color Mapping

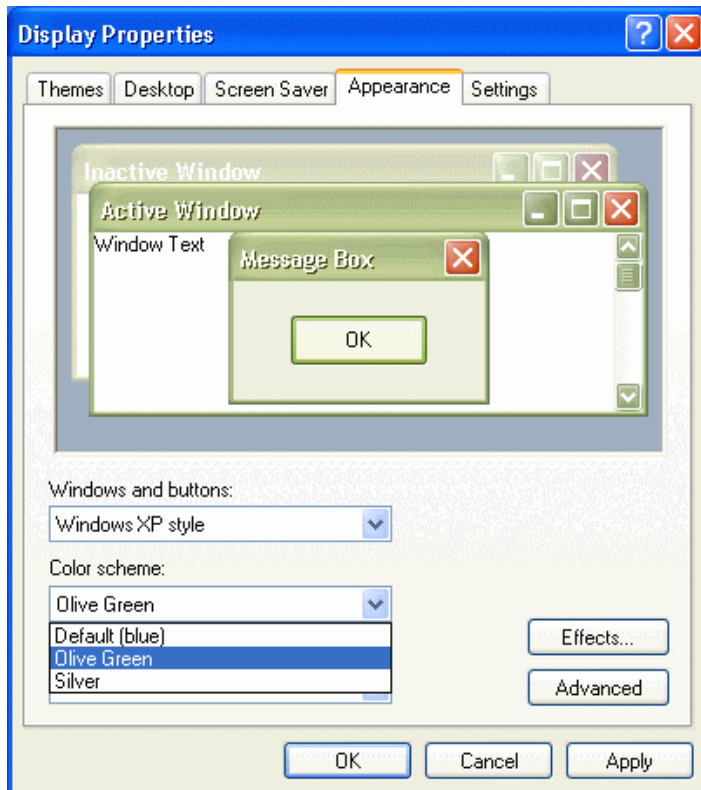
Nếu dùng Hex Calculator đổi con số 0xFFFFFFFF ra decimal ta sẽ được 16777215, nếu kể cả số 0 ta sẽ có tổng cộng 16777216 màu. Lúc này ta bàn về 8bit (1 byte) và 16bit (2 bytes) color, nhưng ở đây ta nói chuyện 3 byte color. Như thế có thể màn ảnh không đủ khả năng để cung cấp mọi màu mà

Function RGB tính ra. Vậy VGA card sẽ làm sao?

Thí dụ một cạt VGA chỉ hỗ trợ đến 8 bits. Nó sẽ cung cấp 256 màu khác nhau. Nếu Function RGB đòi hỏi một màu mà VGA card có thể cung cấp chính xác thì tốt, nếu không nó sẽ tìm cách dùng hai hay ba đóm gần nhau để trộn màu và cho ta ảo tưởng màu ta muốn. Công tác này được gọi là **Color Mapping** và cái màu được làm ra được gọi là **custom color**.

Dùng Intrinsic Color Constants

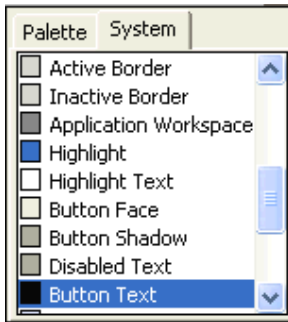
Một trong những features của MSWindows là cho ta chọn Color Scheme của Windows theo sở thích. Bình thường, Color Scheme của Windows là Blue, nhưng ta có thể chọn Olive Green hay Silver, nếu ta muốn.



Chỉ khổ nỗi nếu ta đã dùng một màu đỏ đậm để hiển thị tuyệt đẹp thứ gì trong chương trình VB6 mà bây giờ user tự nhiên thay đổi Color Scheme thành Olive Green chẳng hạn khiến cho màu đỏ đậm ấy coi chẳng giống ai trong cái Color Scheme mới.

Để tránh trường hợp ấy, thay vì nói thẳng ra là màu gì (xanh hay đỏ) ta nói dùng màu **vbActiveTitlebar** hay **vbDesktop**, .v.v.

Dùng Intrinsic Color Constant sẽ bảo đảm màu ta dùng sẽ được biến đổi theo Color Scheme mà user chọn để khỏi bị trường hợp cái màu trở nên chẳng giống ai. Lúc thiết kế, ta cũng có thể chọn Intrinsic Color Constant từ Tab **System** khi chọn màu.



Graphic files

Khi một hình Graphic được lưu trữ theo dạng số pixels với màu của chúng như đã nói trên thì ta gọi là một **Bit Map** và tên file của nó trong disk có extension **BMP** thí dụ như **House.bmp**. Lưu trữ kiểu này cần rất nhiều memory và rất bất tiện để gởi đi hay hiển thị trên một trang Web. Do đó người ta dùng những kỹ thuật để giảm thiểu lượng memory cần để chứa graphic nhưng vẫn giữ được chất lượng của hình ảnh. Có hai dạng Graphic files rất thông dụng trên Web, mang tên với extensions là **JPG** và **GIF**. Đặc biệt với GIF files ta có thể chứa cả hoạt họa (animation), tức là một GIF file có thể chứa nhiều hình (gọi là **Frames**) để chúng lần lượt thay nhau hiển thị, cho người xem có cảm tưởng một vật đang di động.

In trên màn ảnh

VB6 có **method Print** cho ta in thẳng trên Form, PictureBox hay Printer. Ba loại control này được coi như những khung vải mà họa sĩ vẽ lên.

Bạn hãy khởi động một chương trình VB6 mới. Đặt lên form một PictureBox tên Picture1 và một button tên CmdPrintTenLines với Caption **Print Ten Lines**. DoubleClick lên button này và viết code dưới đây:

```
Private Sub CmdPrintTenLines_Click()
    Dim i As Integer
    ' String variable used for display
    Dim strLine As String
    ' Write 10 lines, one under the other
    For i = 1 To 10
        strLine = "This is line " & CStr(i)
        Me.Print strLine ' Print on Form
        Picture1.Print strLine ' Print on Picture1
    Next
End Sub
```

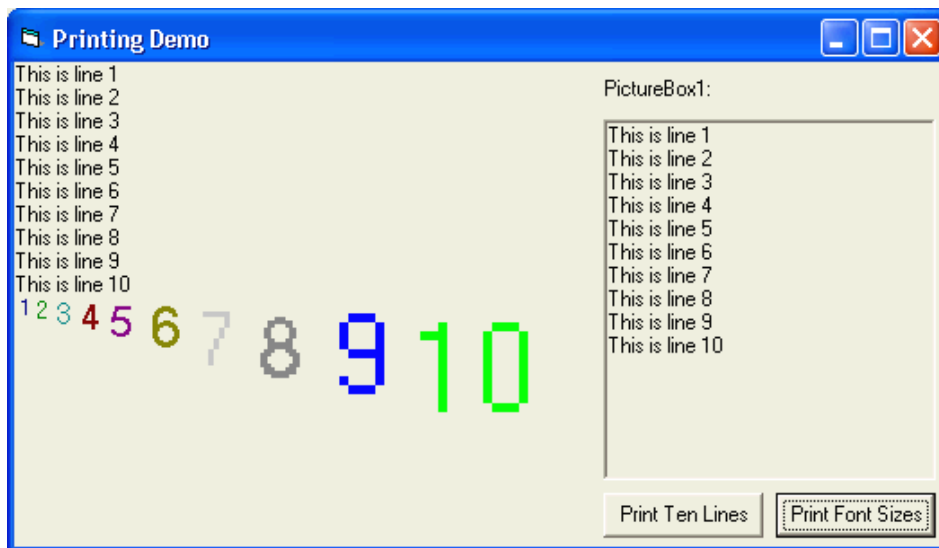
Bạn hãy chạy thử program rồi click nút **Print Ten Lines**. Trong trường hợp này ta dùng default Font và Color để in 10 hàng. Sau mỗi Print, chương trình tự động xuống hàng. Kế đó, thêm một button tên CmdPrintFontSizes với Caption **Print Font Sizes**. DoubleClick lên button này và viết code dưới đây:

```

Private Sub CmdPrintFontSizes_Click()
    Dim i As Integer
    ' Print numbers 1 to 10, one after the other on the same line
    For i = 1 To 10
        ' Define Font size
        Me.Font.Size = Me.Font.Size + i
        ' Define Color using Function QBColor
        Me.ForeColor = QBColor(i)
        ' Print without moving to next line. Note the semicolon ";"
        Me.Print Str(i);
    Next
End Sub

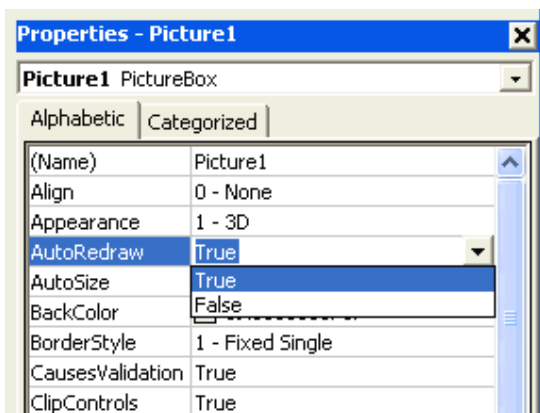
```

Trong Sub CmdPrintFontSizes_Click, ta thay đổi cỡ kiểu chữ để cho các con số được in ra lớn lên dần dần và thay đổi màu của các con số bằng cách dùng **function QBColor**. Để in các con số liên tục không xuống hàng ta dùng method Print với semicolon (;). Bạn hãy chạy chương trình lại. Click nút **Print Ten Lines** rồi click nút **Print Font Sizes**, kết quả sẽ giống như dưới đây:



Bây giờ bạn thử **minimize** cửa sổ của chương trình, kế đó restore nó lại kích thước cũ. Bạn sẽ thấy các hàng ta in lúc này không còn trong form hay PictureBox nữa.

Lý do là khi ta Print lên form hay PictureBox, các hình ấy được vẽ trong graphic địa phương chứ không được VB6 kể là một phần của cửa sổ. Muốn tránh trở ngại này ta phải dặn VB6 nhớ vẽ lại bằng cách set **property AutoRedraw** của form và Picture1 ra **True**.



Hệ thống tọa độ

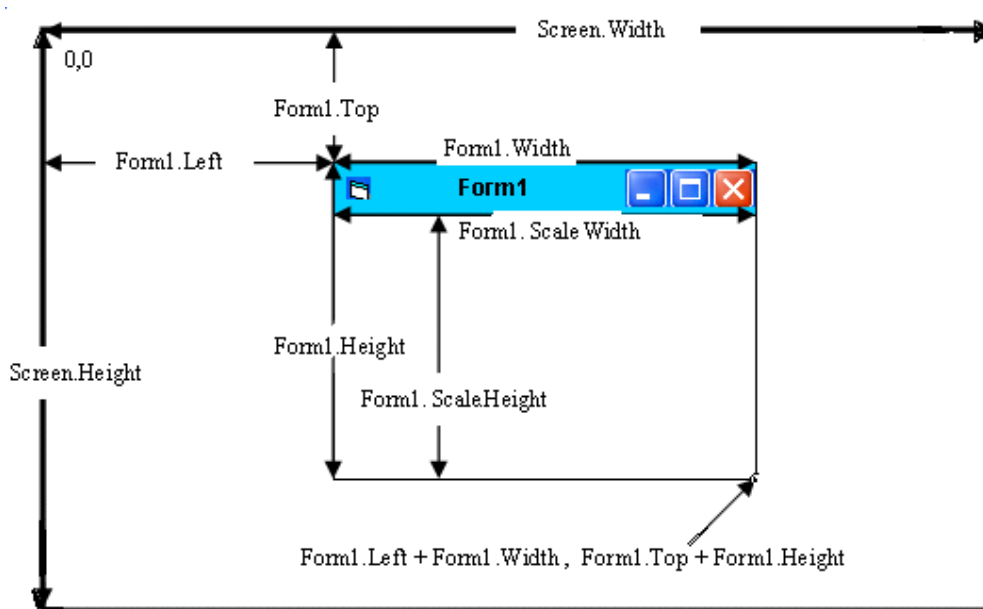
Khi đặt một Object hay vẽ một cái gì lên màn ảnh (screen) hay form .v.v.. ta cần phải chỉ định Object ấy nằm chỗ nào kể từ (with reference to) cái góc Trên Trái (Top Left) của màn ảnh hay form.

Cái góc Trên Trái là Trung tâm tọa độ của screen hay form. Ở đó tọa độ X và Y đều bằng 0, ta viết là **0,0**. Nếu ta đi lần qua phải theo chiều rộng của screen thì tọa độ X tăng lên. Nếu ta đi dọc xuống dưới theo chiều cao của screen thì tọa độ của Y tăng lên.

Kể đến là đơn vị đo lường ta dùng để biểu diễn khoảng cách. Trong bài trước ta đã nói đến độ mịn của màn ảnh (screen resolution) dựa vào **pixel**. Ta có thể dùng đơn vị pixel để nói một Object có tọa độ X và Y mỗi chiều bao nhiêu pixels tính từ trung tâm tọa độ.

Như thế, ngay cả trên cùng một màn ảnh khi ta tăng độ mịn nó lên thì một Object đã được đặt lên màn ảnh theo đơn vị pixel sẽ xích qua trái và lên trên một ít vì kích thước một pixel bây giờ nhỏ hơn lúc trước một chút.

Hình dưới đây minh họa các kích thước của màn ảnh và Form.

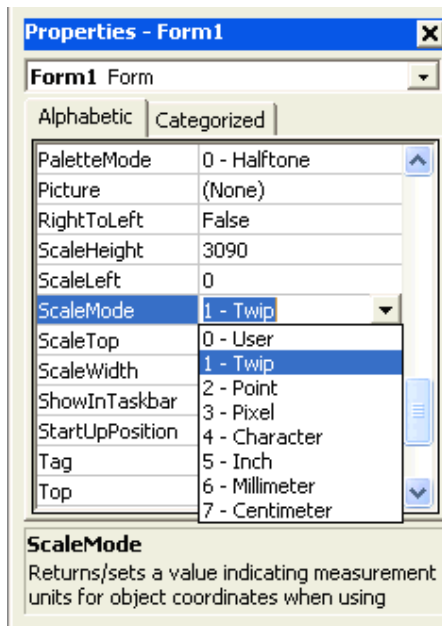


Điểm cần biết là có những phần như **title bar** và **border** của một form ta không thể vẽ lên được. Do đó diện tích còn lại của form được gọi là **Client Area**. Chiều rộng và chiều cao của Client Area được gọi là **ScaleWidth** và **ScaleHeight**.

Nếu muốn khoảng cách từ một Object đến trung tâm tọa độ, hay kích thước của chính Object, không hề thay đổi dù ta có tăng, giảm độ mịn của màn ảnh hay in hình ra printer (thí dụ ta muốn nó luôn luôn dài 5cm chẳng hạn) thì ta dùng hệ thống tọa độ theo đơn vị **twips** của form.

Twips là Default Coordinate System của VB6. Trong hệ thống này mỗi điểm là tương đương với 1/567 centimeter. Do đó, nếu bạn vẽ một đường dài 567 twips nó sẽ hiển thị dài 1cm trên màn ảnh, và khi bạn in nó ra, nó cũng dài 1cm trên giấy. Tức là độ dài thật của Object không tùy thuộc vào loại màn ảnh (độ mịn cao hay thấp) hay printer. Người ta nói nó là **Device independent** coordinate system (Hệ thống tọa độ độc lập với dụng cụ). Nói một cách khác Twips cho ta thật sự **what you see is what you get (WYSIWYG - thấy sao có vậy)**, rất thích hợp với Desktop publishing.

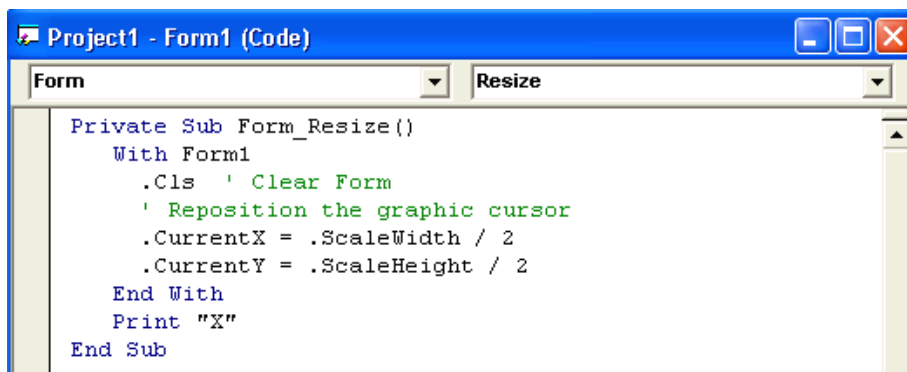
Bạn có thể thay đổi hệ thống tọa độ của một form bằng cách edit **property ScaleMode** qua cửa sổ Properties như sau:



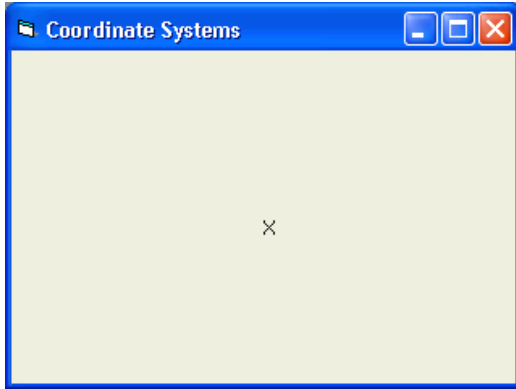
Ghi chú: Thay đổi trị số ScaleMode không có hiệu lực ngay mà chỉ ảnh hưởng những gì được thiết kế sau đó.

Giống như khi ta Edit Text trong Notepad, Text Cursor (thanh | chớp chớp) là vị trí hiện tại, nơi sẽ hiển thị cái chữ ta đánh sắp tới, trong graphic ta có một Cursor vô hình, nơi sẽ hiển thị cái gì ta sắp **Print**. Ta chỉ định vị trí của graphic cursor ấy bằng cách cho trị số của **CurrentX** và **CurrentY**.

Bạn hãy khởi động một dự án VB6 mới và viết code cho **Event Resize** của form chính như sau:



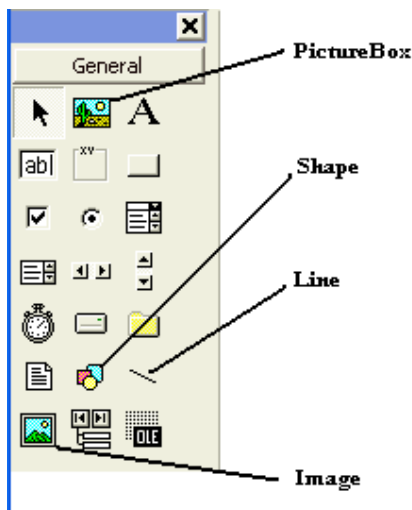
Thử chạy chương trình và Resize form. Mỗi khi bạn Resize form, chữ **X** sẽ được dời đến vị trí khoảng chính giữa của Client Area của form.



Dùng Graphics

Đã có một chút căn bản về graphics của VB6, bây giờ ta có thể đặt những graphics lên form. Có hai cách để làm chuyện ấy:

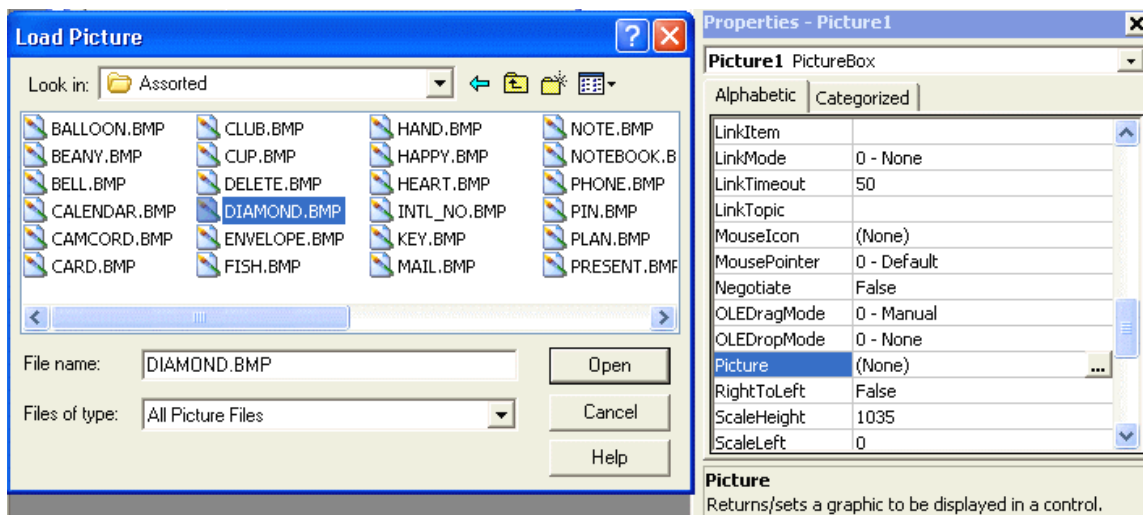
- Dùng **Graphical Controls**: Ta có **PictureBox** và **Image** có thể chứa hình ảnh. Trong khi **Line** và **Shape** có thể vẽ đường thẳng hay các hình chữ nhật, tròn .v.v.. trên form, lúc thiết kế.
- Dùng **Graphics Methods**: Đây là những mệnh lệnh cho ta vẽ trực tiếp lên form lúc run-time. Các mệnh lệnh VB6 cho ta là **Cls**, **Pset**, **Point**, **Line** và **Circle**.



Tùy theo hoàn cảnh, bạn có thể lựa chọn cách nào tiện dụng.

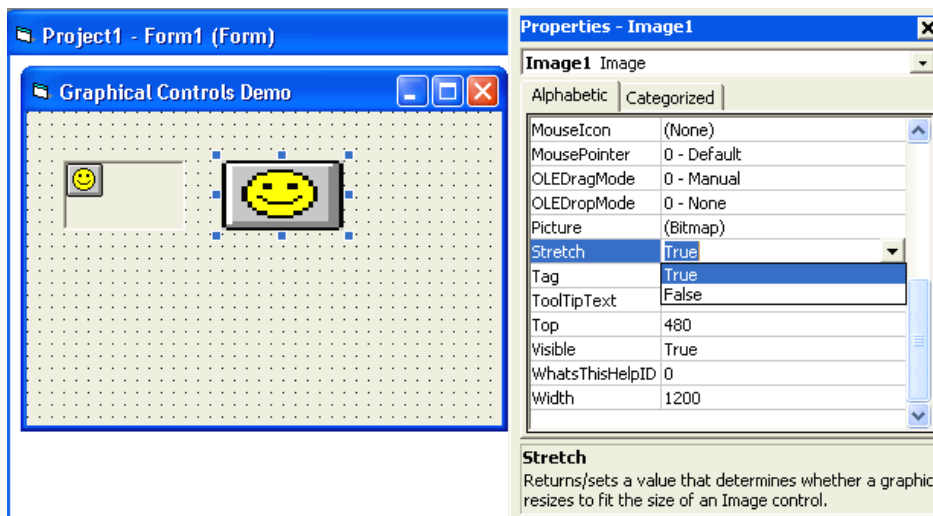
PictureBox và Image

Dùng PictureBox hay Image là cách dễ nhất để hiển thị một graphic trong form. Lúc thiết kế, bạn có thể đánh thẳng tên của graphic vào **property Picture** trong cửa sổ Properties. Form cũng nhận property Picture. Bạn cũng có thể click lên bên phải chữ property Picture để browse và chọn một graphic, thường là Bitmap hay Icon.



Sự khác biệt chính giữa Image và PictureBox là Image có **property Stretch** mà ta có thể set thành True để kéo giãn graphic ra cho chiếm trọn diện tích của Image. Image là một graphic control **lightweight** (nhẹ ký), tức là nó không đòi hỏi nhiều memory và chạy nhanh hơn PictureBox. Lý do là PictureBox là một container, tức là nó có thể chứa các controls khác. Ngoài ra, PictureBox cũng cho phép ta vẽ lên trên nó giống như trên form.

Trong hình dưới đây, trong lúc thiết kế ta đặt một PictureBox và một Image cùng một cỡ lên cùng một form. Kế đó ta assign cùng một picture hình **happy.bmp** cho cả hai. Riêng với Image, ta set property **Stretch** của nó ra **True**.



Chỉ định hình ảnh lúc run-time

Trong lúc program đang chạy, ta có thể thay đổi hình ảnh chứa trong PictureBox hay Image bằng cách dùng **Function LoadPicture**. Nhớ là ta không thể assign trực tiếp vào Property Picture của hai graphical controls này. Lý do là Property Picture chỉ là một cách thân thiện cho ta chỉ định một graphic trong lúc thiết kế. Khi một hình ảnh đã được chỉ định rồi, VB6 chứa cả hình ấy vào file có cùng tên với file của form nhưng với extension **.frx**. Tức là nếu tên của form là **Form1** thì graphic của Property Picture được chứa chung với các graphics khác của form trong file **Form1.frx**.

Do đó, vì VB6 program chứa luôn graphic chung với nó, ta không cần phải nhắc đến tên của graphic file khi dùng hay deploy, tức là không cần đính kèm tên graphic file trong Setup file cho người ta

install. Dưới đây là code mẫu để lúc run-time ta load một graphic tên **sad.bmp** nằm trong Subfolder tên **images** của App.path vào Image control tên Image1.

```
Private Sub CmdLoad_Click()  
    Dim LocalDir As String  
    ' Assign Folder where program resides to LocalDir  
    LocalDir = App.Path  
    ' Append right backslash if last character is not "\"  
    If Right(LocalDir, 1) <> "\" Then  
        LocalDir = LocalDir & "\"  
    End If  
    ' Load graphic "sad.bmp" from SubFolder "images" into Image1  
    Image1.Picture = LoadPicture(LocalDir & "images\sad.bmp")  
End Sub
```

Dĩ nhiên, nếu ta muốn load graphic lúc run-time thì phải cung cấp graphic file riêng.

Control Shape

Control Shape cho phép bạn vẽ những hình đơn giản như đường thẳng, hộp, vòng tròn trên form, lúc thiết kế. Sau khi DoubleClick lên control Shape trong Toolbox để thêm một control Shape vào form, bạn chọn loại Shape của nó từ cửa sổ Properties rồi nắm vào một góc của Shape trên form drag lớn nhỏ tùy ý.

Muốn sơn bên trong một Shape, bạn chọn màu từ **property FillColor**. Property FillColor cũng giống như BackColor của các controls khác, nhưng nó chỉ có hiệu lực khi bạn cho **property FillStyle** một trị số khác hơn là **1- Transparent** (trong suốt), thí dụ như **0- Solid** (dày đặc).

Trong khi các Graphical Controls như Shape, Line cho ta vẽ hình lúc thiết kế thì **Graphics Methods** cho ta vẽ những thứ ấy lúc run-time. Ta cũng có thể chấm từng đốm (pixel) hay copy cả một Picture từ chỗ này đến chỗ khác.

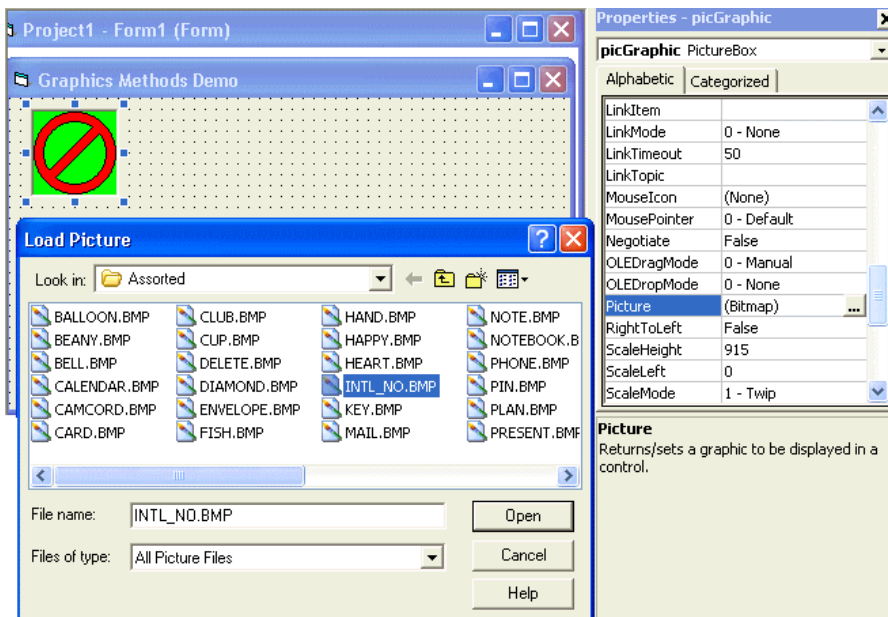
Chỉ cần một chút kinh nghiệm bạn có thể làm hoạt họa (animation) hay tạo visual effects tuyệt diệu mà không cần phải đụng đến **Windows API (Application Programming Interface)** để dùng **Function BitBlt**.

Method PaintPicture

Method PaintPicture cho phép bạn copy rất nhanh một khối dữ kiện đồ họa, nói nôm na là một khu vực trong một hình graphic trên form, PictureBox hay Printer đến một nơi khác. Thí dụ bạn copy một hình từ chỗ này đến chỗ khác trong form, hay từ form/PictureBox ra Printer Object để một chốc sau bạn in nó ra.

Bạn hãy khởi động một dự án VB6 mới và DoubleClick lên PictureBox Icon trong ToolBox để đặt một PictureBox lên form. Đặt tên PictureBox ấy là **picGraphic** và set property Visible của nó ra False để ta không thấy nó lúc run-time.

Bây giờ load một hình vào property Picture của picGraphic bằng cách Browse một Bitmap file từ cửa sổ Properties. Ở đây ta chọn **INTL_NO.BMP** từ folder **(Program Files)\Microsoft Visual Studio\Common\Graphics\Bitmaps\Assorted**



Trong chương trình này ta muốn hễ khi đè nút trái của Mouse xuống và di chuyển Mouse cursor thì khi cursor đi đến đâu, hình INTL_NO được vẽ đến đó.

Ta sẽ dùng một **Flag** để đánh dấu nút-trái-của-Mouse-Down, đặt tên là **flgMouseDown**. Khi nhận được **Event MouseDown** ta set flgMouseDown thành **True**, và khi nhận được **Event MouseUp** ta reset flgMouseDown thành **False**. Mỗi lần nhận được Event MouseMove thì nếu flgMouseDown là True ta sẽ PaintPicture INTL_NO.

Để xóa background của form, ta thêm một button tên **CmdClearForm** để chạy graphic **method Cls**. Dưới đây là liệt kê code mẫu:

```

' Flag that indicates that the Mouse's left button is depressed
Dim flgMouseDown As Boolean

Private Sub Form_Load()
    ' Initialise flgMouseDown to False
    flgMouseDown = False
End Sub

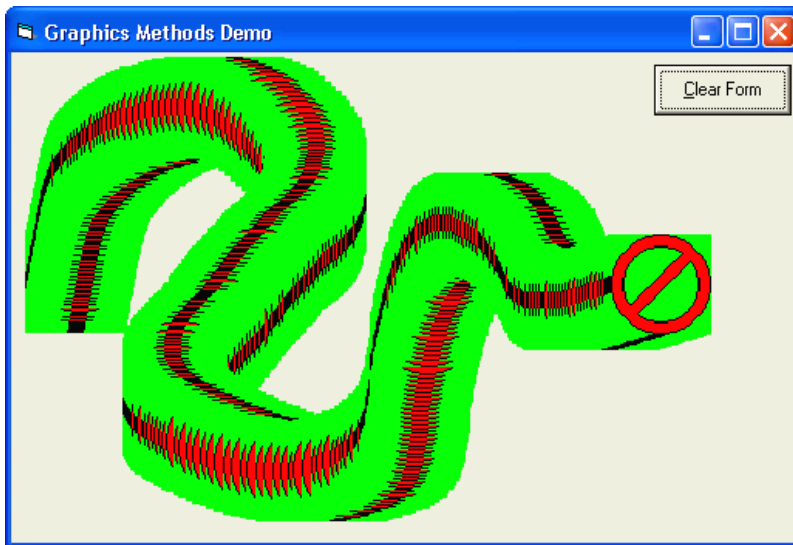
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' Set Flag flgMouseDown
    flgMouseDown = True
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' Paint picGraphic if flgMouseDown is True
    If flgMouseDown Then
        ' Paint full-size picGraphic at Mouse cursor location
        PaintPicture picGraphic.Picture, X, Y, picGraphic.Width, picGraphic.Height
    End If
End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' Reset Flag flgMouseDown
    flgMouseDown = False
End Sub

Private Sub CmdClearForm_Click()
    ' Clear the form
    Cls
End Sub

```



Lưu ý là bạn phải declare variable **flgMouseDown** bên ngoài các Subs để mọi Sub đều thấy và có thể dùng nó. Muốn biết thêm chi tiết về cách dùng **method PaintPicture**, trong VB6 IDE DoubleClick lên chữ PaintPicture trong code editor để highlight chữ ấy rồi bấm nút **F1**.

Method PSet

Ta dùng **method PSet** (đến từ chữ **Point Set**) để vẽ một pixel lên form. Ta cần cho biết PSet ở đâu và với màu gì, tức là ta cho nó tọa độ X,Y của pixel và một màu tính từ **function RGB**.

Dưới đây là code để vẽ pixels đủ màu lên form một cách bất chừng (randomly) về vị trí và màu sắc khi user clicks lên form chính:

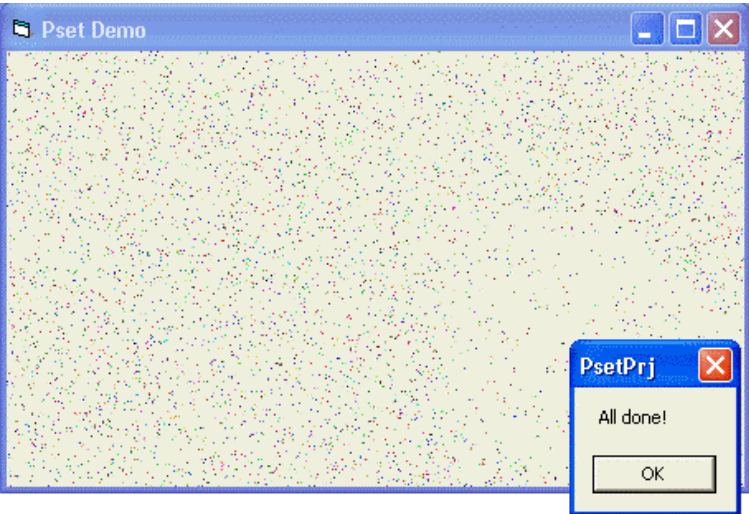
```
Private Sub Form_Click()
    Dim i As Integer
    ' Variables for pixel coordinates
    Dim iXCoord As Integer
    Dim iYCoord As Integer
    ' Variable for primary colours
    Dim iRed As Integer
    Dim iGreen As Integer
    Dim iBlue As Integer
    ' Start the Random number generation
    Randomize
    ' Plot 2000 dots randomly
    For i = 1 To 2000
        ' get a random XCoord.
        ' Note that Rnd(1) returns a real number between 0 and 1, eg: 0.384
        iXCoord = Int(Rnd(1) * ScaleWidth)
        ' get a random YCoord.
        iYCoord = Int(Rnd(1) * ScaleHeight)
```



```
' Get a random number between 0 and 254 for each primary colour
iRed = Int(Rnd(1) * 255)
iGreen = Int(Rnd(1) * 255)
iBlue = Int(Rnd(1) * 255)
' Plot the pixel at iXCoord,iYCoord
PSet (iXCoord, iYCoord), RGB(iRed, iGreen, iBlue)
Next
MsgBox ("All done!")
End Sub
```

Trong thí dụ trên ta dùng **method Randomize** để generate sẵn trong bộ nhớ các con số real bất chừng từ 0 đến 0.999. Sau đó mỗi lần ta gọi **Function Rnd(1)** là nó sẽ trả về một con số real lấy bất chừng từ bộ số do method Randomize generated. Do đó, **Rnd(1) * ScaleWidth** sẽ cho ta một con số real có trị số từ 0 đến ScaleWidth. Muốn đổi con số real đó ra Integer, ta dùng **Function Int**.

Khi khởi động chương trình và Click lên form ta sẽ có hình giống như dưới đây:



Mách nước: Để xóa một đốm bạn Pset lại tại chỗ ấy một đốm mới có cùng màu với BackColor của form.

Method Line

Method Line vẽ một đường thẳng từ một tọa độ này đến một tọa độ khác trong màu do ta chỉ định. Với hai methods PSet và Line ta có thể làm được rất nhiều chuyện. Thí dụ muốn cho một vật di động, ta xóa vật ấy bằng cách vẽ lại nó với cùng màu của BackColor của form, rồi vẽ vật ấy ở vị trí mới. Muốn vẽ một đa giác như tam giác hay chữ nhật ta ráp nhiều đường thẳng lại với nhau, đầu của mỗi đường thẳng là cuối của đường thẳng vừa mới được vẽ trước. Muốn sơn Shade bên trong một hình chữ nhật ta dùng PSet..v.v.

Có ba cách để chỉ định tọa độ của hai đầu của một đường thẳng ta muốn vẽ:

1. Cho biết tọa độ của đầu và cuối đường thẳng:
thí dụ: **Line (50, 100)-(3000, 4000)**
Khi đường này được vẽ xong thì vị trí của graphic cursor có tọa độ là vị trí của cuối đường, tức là CurrentX=3000 và CurrentY=4000 trong trường hợp này.
2. Chỉ cho biết tọa độ cuối đường thẳng:
thí dụ: **Line -(3600, 4500), vbMagenta**
Trong trường hợp này vị trí của graphic cursor (CurrentX, CurrentY) được lấy làm tọa độ của đầu đường thẳng khi vẽ. Tức là nếu trước khi execute dòng code này CurrentX=3000 và CurrentY=4000 thì dòng code tương đương với:
Line (3000,4000)-(3600,4500), vbMagenta
3. Dùng chữ **Step** để nói sự khác biệt từ CurrentX và CurrentY:
thí dụ: **Line Step(400, 600)-Step(800, -500), vbGreen**
Nếu trước khi execute dòng code này CurrentX=3600 và CurrentY=4500 thì dòng code tương đương với:

Line (4000,5100)-(4800,4600), vbGreen

Trong thí dụ dưới đây, một hình tam giác được vẽ bằng hai cách coding khác nhau. Khi chạy program để thử, bạn hãy lần lượt click **Triangle METHOD I** và **Triangle METHOD II** để thấy cả hai cách vẽ đều y như nhau, chỉ khác màu thôi.

```
Private Sub CmdTrianI_Click()  
    ' Drawing a black triangle: METHOD I  
    Line (700, 500)-(2800, 2400)  
    Line (2800, 2400)-(1800, 900)  
    Line (1800, 900)-(700, 500)  
End Sub  
  
Private Sub CmdTrianII_Click()  
    ' Drawing a red triangle: METHOD II  
    ' Draw a red line from Location(700, 500) to Location (2800, 24000)  
    Line (700, 500)-(2800, 2400), vbRed  
    ' Draw a red line from Location(2800,2400) to Location (1800,900)  
    Line -(1800, 900), vbRed  
    ' Draw a red line from Location(1800,900) to Location (700,500)  
    Line -(700, 500), vbRed  
End Sub
```

Để vẽ một hình chữ nhật, cách tiện nhất là dùng Step như dưới đây:

```
Private Sub Rectangle(ByVal X1 As Integer, ByVal Y1 As Integer, ByVal X2 As Integer,  
    ByVal Y2 As Integer)
```

```

' Draw a rectangle
Line (X1, Y1)-(X2, Y1)
Line -(X2, Y2)
Line -(X1, Y2)
Line -(X1, Y1)
End Sub

```

Ta còn có thể vẽ một hình chữ nhật với bốn góc tròn như sau:

```

Private Sub RoundCornerRectangle(ByVal X1 As Integer, ByVal Y1 As Integer, ByVal X2 As Integer, ByVal Y2 As Integer)
    Const Delta = 50
    ' Draw a rectangle with round corner
    Line (X1 + Delta, Y1)-(X2 - Delta, Y1)
    Line -Step(Delta, Delta)
    Line -(X2, Y2 - Delta)
    Line -Step(-Delta, Delta)
    Line -(X1 + Delta, Y2)
    Line -Step(-Delta, -Delta)
    Line -(X1, Y1 + Delta)
    Line -Step(Delta, -Delta)
End Sub

```

Ta cũng có thể sơn Shade bên trong hình chữ nhật bằng cách dùng method PSet để chấm các đốm cách nhau chừng 50 pixels như sau:

```

Private Sub Shade(ByVal X1 As Integer, ByVal Y1 As Integer, ByVal X2 As Integer, ByVal Y2 As Integer)
    ' Shade a roundcorner rectangle by plotting dots using method Pset
    Const Delta = 50
    Dim i As Integer
    Dim j As Integer
    ' Make sure that X1 is less than X2
    ' Swap values of X1, X2 if X1 > X2
    If X2 < X1 Then
        Temp = X1
        X1 = X2
        X2 = Temp
    End If
    ' Make sure that Y1 is less than Y2
    ' Swap values of Y1, Y2 if Y1 > Y2

```

```

If Y2 < Y1 Then
    Temp = Y1
    Y1 = Y2
    Y2 = Temp
End If
' Plotting dots inside the rectangle at 50 pixels apart
For i = X1 + Delta To X2 - Delta Step 50
    For j = Y1 + Delta To Y2 - Delta Step 50
        PSet (i, j)
    Next
Next
End Sub

```

Muốn Shade đậm hơn, bạn có thể chấm các đốm gần nhau hơn, thí dụ cho cách nhau 30 pixels thay vì 50 pixels. Có một cách khác là tăng trị số của **DrawWidth**, độ dày của đường vẽ hay đốm.

Bây giờ phối hợp cách vẽ hình chữ nhật với method Shade nói trên và **method Print** ta có thể viết chữ bên trong một khung màu nhạt như sau:

```

Private Sub CmdDrawFrame_Click()
    Dim X1 As Integer
    Dim Y1 As Integer
    Dim X2 As Integer
    Dim Y2 As Integer
    ' Initialise Coordinates of rectangle
    X1 = 4200: Y1 = 1000
    X2 = 6200: Y2 = 2000
    ' Draw a roundcorner rectangle
    RoundCornerRectangle X1, Y1, X2, Y2
    ' Shade the rectangle
    Shade X1, Y1, X2, Y2
    ' Position cursor to Print some text
    CurrentX = X1 + 50
    CurrentY = Y1 + 50
    ' Define Font Size
    Font.Size = 18
    ' Print the text at cursor location
    Print "Hello there!"
End Sub

```

Khi chạy chương trình này và click tất cả các buttons trên form, bạn sẽ có hình dưới đây:



Hãy nhớ set property **AutoDraw** của form ra **True** để các graphic chương trình vẽ không bị mất khi user minimises form.

Bạn cũng có thể dùng những kỹ thuật nói trên với **Object Printer** để in các mẫu giấy điền chi tiết.

Method Circle

Ta dùng **Method Circle** để vẽ hình tròn, hình bầu dục và đường cung, với bên trong trống rỗng hay được sơn đầy bằng một màu ta chỉ định. Ta phải cho biết tọa độ của tâm điểm vòng tròn và bán kính của nó.

Bạn hãy khởi động một dự án VB6 mới, đặt lên form một button với tên **frmCircle** và caption **Circle & Lines**. DoubleClick lên button ấy và viết code sau đây:

```
Private Sub CmdCircleLine_Click()  
    ' Draw a circle centered at 2000,1500 with radius equal 800  
    Circle (2000, 1500), 800  
    ' Draw a vertical line from center  
    Line (2000, 1500)-Step(0, 800)  
    ' Draw a horizontal line from center  
    Line (2000, 1500)-Step(800, 0)  
End Sub
```

Bây giờ hãy đặt lên form một button khác tên **CmdArc** và caption **Draw Arc**. Thay vì vẽ nguyên một vòng tròn, ta sẽ chỉ vẽ một hình vòng cung bằng màu đỏ.

Để chỉ định rằng ta sẽ vẽ từ vị trí nào trên vòng tròn đến vị trí nào khác, thí dụ từ 45độ đến 230độ, ta cần phải đổi degree ra đơn vị Radian bằng cách dùng **Function Rads** như sau:

```

Private Function Rads(ByVal Degree As Single) As Single
    ' Convert Degrees to Radian
    Const PI = 22 / 7
    Rads = Degree / 180 * PI
End Function

```

Vòng cung luôn luôn được vẽ ngược chiều kim đồng hồ. Dưới đây là code để vẽ một đường vòng cung màu đỏ bán kính 800, tâm điểm ở (4000, 2000), từ 45độ đến 230độ:

```

Private Sub CmdArc_Click()
    Circle (4000, 2000), 800, vbRed, Rads(45), Rads(230)
End Sub

```

Ta có thể cho sơn bên trong các hình tròn, hay Pie Slices (một phần của hình tròn) bằng cách set **FillStyle** bằng 0 và chỉ định màu **FillColor**. Một Pie Slice là một vòng cung đồng kính bởi hai đường thẳng bán kính ở hai đầu. Muốn vẽ một Pie Slice ta đánh thêm dấu trừ ("-") trước hai trị số Radian, tức là dùng **-Rads(45)**, **-Rads(230)** thay vì **Rads(45)**, **Rads(230)**.

Dưới đây là code vẽ hai Pie Slices, có tâm điểm lệch nhau một tí, đồng thời thêm chú thích 87.5% và 12.5%.

```

Private Sub CmdPie_Click()
    FillStyle = 0 ' Fill inside any closed shaped
    FillColor = vbYellow
    ' Draw a Pie Slice from 90deg to 45deg in Yellow
    Circle (3000, 4000), 800, , -Rads(90), -Rads(45)
    ' Position the graphic cursor to Print some text
    CurrentX = 2800: CurrentY = 4400
    Print "87.5%"
    FillColor = vbBlue
    ' Draw a Pie Slice from 45deg to 90deg in Blue
    Circle (3050, 3900), 800, , -Rads(45), -Rads(90)
    ' Position the graphic cursor to Print some text
    CurrentX = 3400: CurrentY = 3000
    Print "12.5%"
    FillStyle = 1 ' No fill
End Sub

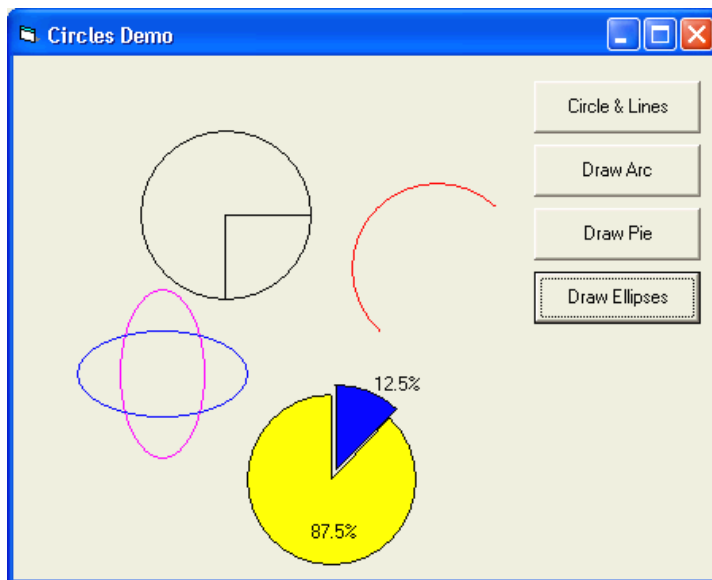
```

Cách dùng cuối cùng của method Circle là để vẽ một **hình bầu dục (Ellipse)**. Vẽ hình bầu dục giống như vẽ một hình tròn nhưng ta cần cho thêm một parameter gọi là **Aspect**. Aspect là sự liên hệ giữa bán kính vertical và bán kính horizontal. Thí dụ nếu Aspect=2 thì chiều cao của hình bầu dục gấp đôi chiều ngang, ngược lại, nếu Aspect=0.5 thì chiều ngang sẽ gấp đôi chiều cao.

Dưới đây là code ta dùng để vẽ hai hình bầu dục cùng cỡ, một cái màu tím nằm thẳng đứng và một cái màu xanh nằm ngang.

```
Private Sub CmdEllipse_Click()  
    Circle (1400, 3000), 800, vbMagenta, , , 2  
    Circle (1400, 3000), 800, vbBlue, , , 0.5  
End Sub
```

Nếu bạn khởi động chương trình và click cả bốn buttons bạn sẽ thấy hình sau đây:



Property DrawMode

Thông thường khi ta vẽ, trị số default của **property DrawMode** là **13- Copy Pen**. Có một trị số DrawMode rất thích hợp cho áp dụng hoạt họa là **7- Xor Pen**. Muốn xóa một hình vừa vẽ xong ta chỉ cần vẽ lại hình ấy trong DrawMode Xor Pen, không cần biết trước đó background như thế nào, nó sẽ hiện ra trở lại.

Chương Mười Ba - Cơ sở dữ liệu (Database)

Table, Record và Field

Nói đến cơ sở dữ liệu, ta lập tức nghĩ đến SQLServer, Access hay Oracle .v.v., những nơi chứa rất nhiều dữ liệu để ta có thể lưu trữ hay lấy chúng ra một cách tiện lợi và nhanh chóng. Hầu hết các chương trình ta viết đều có truy cập cơ sở dữ liệu, và ta dùng nó như một công cụ để làm việc với rất nhiều dữ liệu trong khi tập trung vào việc lập trình phần giao diện với người dùng (users).

Do đó ta cần có một kiến thức căn bản về kiến trúc của cơ sở dữ liệu để hiểu lý do tạo sao ta thiết kế hay truy cập nó theo những cách nhất định.

Ta sẽ dùng Access Database **biblio.mdb**, nằm ở **C:\Program Files\Microsoft Visual Studio\VB98\biblio.mdb** để minh họa các ý niệm cần biết về cơ sở dữ liệu.

Trong database này có 4 **tables**: **Authors** (tác giả), **Publishers** (nhà xuất bản), **Titles** (đề mục) và **Title Author**.

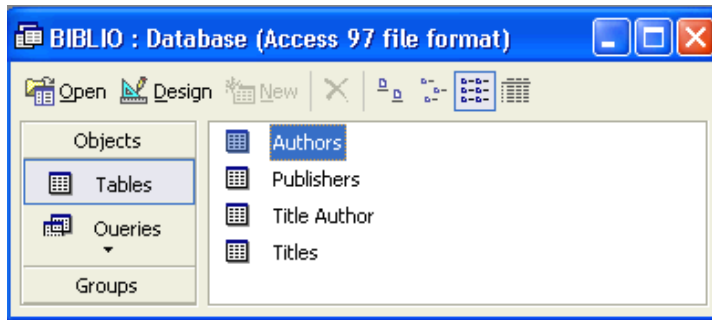


Table Authors chứa nhiều **records**. Mỗi record trong table Authors chứa 3 **fields**: **Au_ID**, **Author** và **Year Born** (năm sanh). Ta có thể trình bày Table Authors dưới dạng một spreadsheet như sau:

	Au_ID	Author	Year Born
+	1	Jacobs, Russell	1950
+	2	Metzger, Philip W.	1962
+	3	Boddie, John	1954
+	4	Sydow, Dan Parks	1963
+	6	Lloyd, John	1948
+	8	Thiel, James R.	1955
+	10	Ingham, Kenneth	1945
+	12	Wellin, Paul	1939
+	13	Kamin, Sam	1947
+	14	Gaylord, Richard	1963
+	15	Curry, Dave	1958
+	17	Gardner, Juanita Mercado	1957

Record: 1 of 6246

Vì cùng một field của các records hiển thị trong cùng một cột của spreadsheet, nên ta cũng nói đến một field như một **column** (cột). Và vì mỗi data record chiếm một row (hàng) của spreadsheet, nên có khi ta cũng nói đến một record như một **row**.

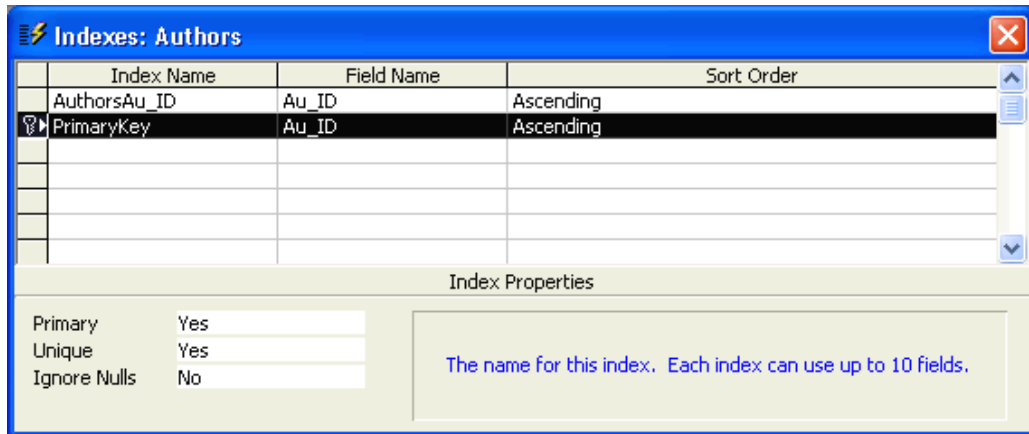
Thật tình mà nói, ta không cần phải có một computer để lưu trữ hay làm việc với một table như Authors này. Ta đã có thể dùng một hộp cạt, trên mỗi cạt ta ghi các chi tiết Au_ID, Author và Year Born của một Author. Như thế mỗi tám cạt tương đương với một record và nguyên cái hộp là tương đương với Table Authors.

Ta sẽ sắp các cạt trong hộp theo thứ tự của số Au_ID để có thể truy cập record nhanh chóng khi biết Au_ID. Chỉ khổ một nỗi, nếu muốn biết có bao nhiêu tác giả, trong số 300 cạt trong hộp, gần hơn 50

tuổi thì phải mất vài phút mới có thể trả lời được. Database trong computer nhanh hơn một hệ thống bằng tay (Manual) là ở chỗ đó.

Primary Key và Index

Để tránh sự trùng hợp, thường thường có một field của record, thí dụ như Au_ID trong Table Authors, được dành ra để chứa một trị số độc đáo (unique). Tức là trong Table Authors chỉ có một record với field Au_ID có trị số ấy mà thôi. Ta gọi nó là **Primary Key**.

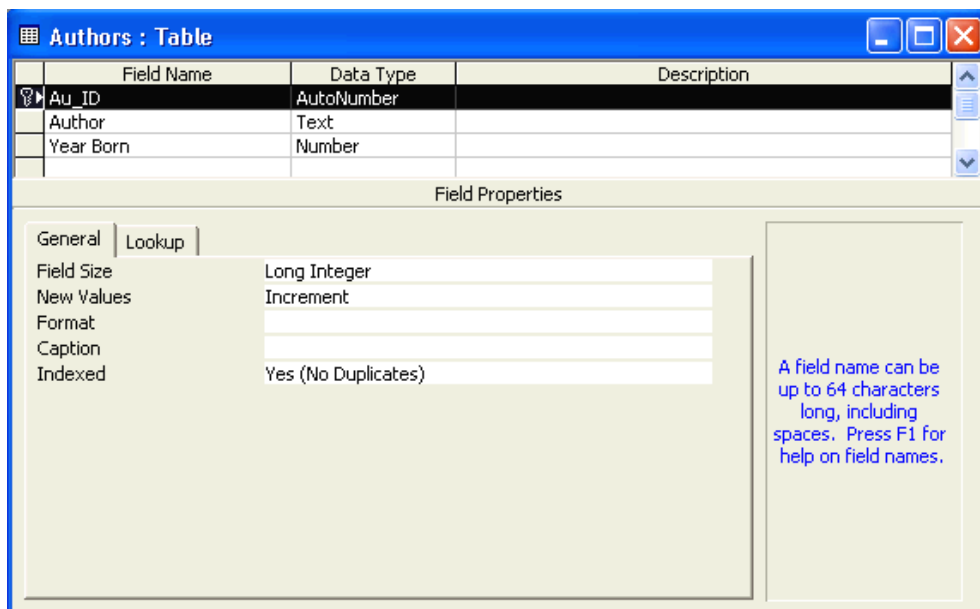


Không phải lúc nào ta cũng muốn truy cập một record Author dựa vào Au_ID. Nhiều khi ta muốn dùng chính tên của Author để truy cập, do đó ta cũng cần phải sort sẵn các records theo thứ tự alphabet. Ta cũng có thể hợp nhiều fields lại để sort các records. Thật ra, chính các records không cần phải được dời đi để nằm đúng vị trí thứ tự. Ta chỉ cần nhớ vị trí của nó ở đâu trong table là đủ rồi.

Cái field hay tập hợp của nhiều fields (thí dụ surname và firstname) để dùng vào việc sorting này được gọi là **Index** (ngón tay chỉ). Một Table có thể có một hay nhiều Index. Mỗi Index sẽ là một table nhỏ của những **pointers**, chứa vị trí của các records trong Table Authors. Nó giống như mục lục index ở cuối một cuốn sách chứa trang số để chỉ ta đến đúng phần ta muốn tìm trong quyển sách.

Khi thiết kế một Table ta chỉ định **Datatype** của mỗi field để có thể kiểm tra data cho vào có hợp lệ hay không. Các Datatypes thông dụng là Number, String (để chứa Text), Boolean (Yes/No), Currency (để chứa trị số tiền) và Date (để chứa date/time). Datatype Number lại gồm có nhiều loại datatypes về con số như Integer, Long (integer chiếm 32 bits), Single, Double, .v.v.

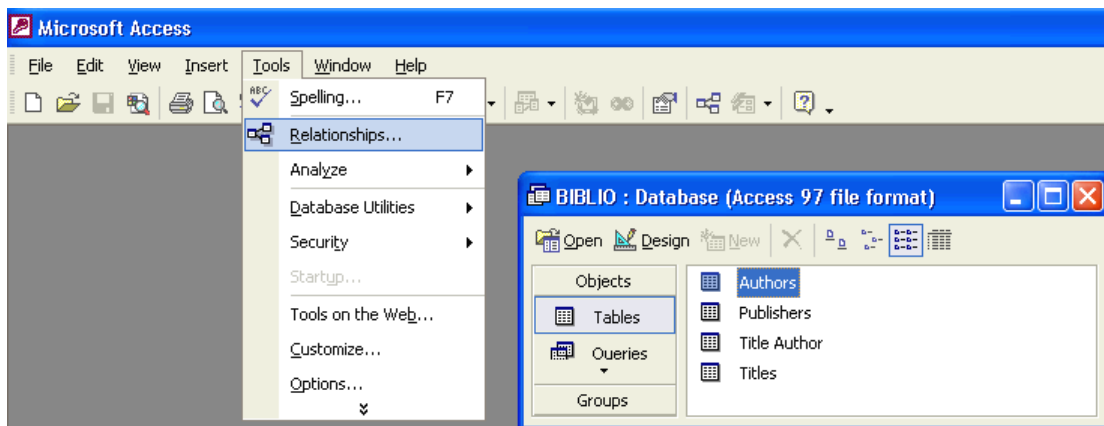
Dưới đây là Datatypes của các fields trong record Author:



Có loại Datatype đặc biệt tên là **AutoNumber**. Thật ra nó là Long nhưng trị số được phát sinh tự động mỗi khi ta thêm một record mới vào Table. Ta không làm gì hơn là phải chấp nhận con số ấy.

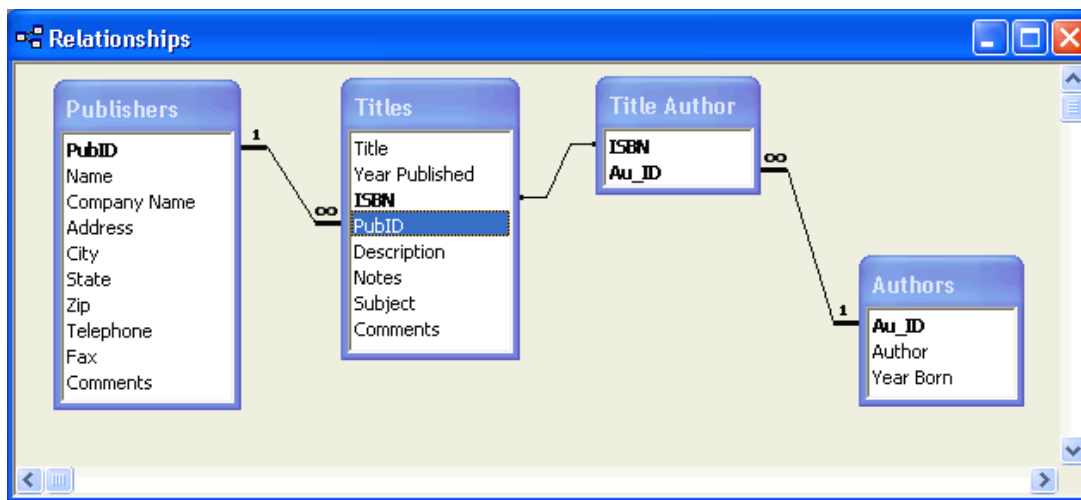
Relationship và Foreign Key

Bây giờ, nếu bạn đang chạy Microsoft Access để quan sát database biblio.mdb, bạn có thể dùng Menu Command **Tools | Relationships** như sau để xem sự liên hệ (relationships) giữa các tables.



Access sẽ hiển thị giao thoại Relationships, trong đó mỗi table có chứa tên các fields. Mỗi table lại có một hay hai sợi dây nối qua các tables khác. Mỗi sợi dây là một mối liên hệ (relationship), nó nối một field trong một table với một field có cùng tên trong table kia.

Thí dụ như giữa hai tables **Publishers** và **Titles** có mối liên hệ dựa trên field **PubID** (**P**ublisher **I**dentification - số lý lịch của nhà xuất bản). Hơn nữa, nếu để ý bạn sẽ thấy ở đầu dây phía table Publishers có con số **1**, còn ở đầu dây bên phía table Titles có dấu vô cực (∞). Ta gọi mối liên hệ (**1- ∞**) là **one-to-many**, ý nói **một** nhà xuất bản có thể phát hành **nhiều** đề mục sách/CD.



Tương tự như vậy, trong mỗi liên hệ one-to-many giữa table Authors và Title Author, ta thấy một tác giả (bên đầu có con số 1) có thể sáng tác nhiều tác phẩm được đại diện bởi các record Title Author.

Trong khi đó giữa hai tables Titles và Title Author, ta có một mối liên hệ **one-to-one**, tức là tương ứng với mỗi record Title chỉ có một record Title Author. Câu hỏi đặt ra là các mối liên hệ one-to-many có cái gì quan trọng.

Tưởng tượng khi ta làm việc với table Titles (tạm gọi là Tác phẩm), nhiều khi ta muốn biết chi tiết của nhà xuất bản của tác phẩm ấy. Thật ra ta đã có thể chứa chi tiết của nhà xuất bản của mỗi tác phẩm ngay trong table Titles. Tuy nhiên, làm như thế có điểm bất lợi là records của các tác phẩm có cùng nhà xuất bản sẽ chứa những dữ liệu giống nhau. Mỗi lần muốn sửa đổi chi tiết của một nhà xuất bản ta phải sửa chúng trong mỗi record Title thuộc nhà xuất bản ấy. Vì muốn chứa chi tiết của mỗi nhà xuất bản ở một chỗ duy nhất, tránh sự lặp lại, nên ta đã chứa chúng trong một table riêng, tức là table Publishers.

Nếu giả sử ta bắt đầu thiết kế database với Table Titles, rồi quyết định tách các chi tiết về nhà xuất bản để vào một table mới, tên Publishers, thì kỹ thuật ấy được gọi là **normalization**. Nói một cách khác, normalization là thiết kế các tables trong database làm sao để mỗi loại mảnh dữ kiện (không phải là Key) chỉ xuất hiện ở một chỗ.

Trong mỗi liên hệ one-to-many giữa tables Publishers và Titles, field PubID là Primary Key trong table Publishers. Trong table Titles, field PubID được gọi là **Foreign Key**, có nghĩa rằng đây là Primary Key của một table lạ (foreign). Hay nói một cách khác, trong khi làm việc với table Titles, lúc nào cần chi tiết một nhà xuất bản, ta sẽ lấy chìa khóa lạ (Foreign Key) dùng làm Primary Key của Table Publishers để truy cập record ta muốn. Để ý là chính Table Titles có Primary Key ISBN của nó.

Relational Database

Một database có nhiều tables và hỗ trợ các liên hệ, nhất là one-to-many, được gọi là **Relational Database**. Khi thiết kế một database, ta sẽ tìm cách sắp đặt các dữ liệu từ thế giới thật bên ngoài vào trong các tables. Ta sẽ quyết định chọn các cột (columns/fields) nào, chọn Primary Key, Index và thiết lập các mối liên hệ, tức là đặt các Foreign Key ở đâu.

Các lợi ích

Trong số các lợi ích của một thiết kế Relational Database có:

- Sửa đổi dữ kiện, cho vào records mới hay delete (gạch bỏ) records có sẵn rất hiệu quả (nhanh).

- Truy cập dữ kiện, làm báo cáo (Reports) cũng rất hiệu quả.
- Vì dữ kiện được sắp đặt thứ tự và có quy củ nên ta có thể tin cậy tính tình của database (không có ba trộn, khi thì thế này, khi thì thế khác - giựt giựt).
- Vì hầu hết dữ kiện nằm trong database, thay vì trong chương trình ứng dụng, nên database tự có documentation (tài liệu cắt nghĩa).
- Dễ sửa đổi chính cấu trúc của các tables.

Integrity Rules (các quy luật liên chính)

Integrity Rules được dùng để nói về những qui luật cần phải tuân theo trong khi làm việc với database để đảm bảo là database còn tốt. Có hai loại quy luật: luật tổng quát (General Integrity Rules) và luật riêng cho database (Database-Specific Integrity Rules). Các luật riêng này thường tùy thuộc vào các quy luật về mậu dịch (Business Rules).

General Integrity Rules

Có hai quy luật liên chính liên hệ hoàn toàn vào database: Entity (bản thể) Integrity Rule và Referential (chỉ đến) Integrity Rule.

Entity Integrity Rule nói rằng **Primary Key** không thể thiếu được, tức là không thể có trị số **NULL**. Quy luật này xác nhận là vì mỗi Primary Key đưa đến một row độc đáo trong table, nên dĩ nhiên nó phải có một trị số đăng hoàng.

Lưu ý là Primary Key có thể là một **Composite Key**, tức là tập hợp của một số keys (columns/fields), nên nhất định không có key nào trong số các columns là NULL được.

Referential Integrity Rule nói rằng database không thể chứa một Foreign Key mà không có Primary Key tương ứng của nó trong một table khác. Điều ấy hàm ý rằng:

- Ta không thể thêm một Row vào trong một Table với trị số Foreign Key trong Row ấy không tìm thấy trong danh sách Primary Key của table bên phía **one** (1) mà nó liên hệ.
- Nếu có thay đổi trị số của Primary Key của một Row hay delete một Row trong table bên phía **one** (1) thì ta không thể để các records trong table bên phía **many** (∞) chứa những rows trở thành mồ côi (orphans).

Nói chung, có ba nhiệm ý (options) ta có thể chọn khi thay đổi trị số của Primary Key của một Row hay delete một Row trong table bên phía **one** (1):

1. **Disallow** (không cho làm): Hoàn toàn không cho phép chuyện này xảy ra.
2. **Cascade** (ảnh hưởng dây chuyền): Nếu trị số Primary Key bị thay đổi thì trị số Foreign Key tương ứng trong các records của table bên phía **many** (∞) được thay đổi theo. Nếu Row chứa Primary Key bị deleted thì các records tương ứng trong table bên phía **many** (∞) bị deleted theo.

3. **Nullify** (cho thành NULL): Nếu Row chứa Primary Key bị deleted thì trị số Foreign Key tương ứng trong các records của table bên phía **many** (∞) được đổi thành NULL, để hàm ý đừng có đi tìm thêm chi tiết ở đâu cả.

Database-Specific Integrity Rules

Những quy luật liên chính nào khác không phải là Entity Integrity Rule hay Referential Integrity Rule thì được gọi là Database-Specific Integrity Rules. Những quy luật này dựa vào chính loại database và nhất là tùy thuộc vào các quy luật về mậu dịch (Business Rules) ta dùng cho database, thí dụ như mỗi record về tiền lương của công nhân phải có một field Số Thuế (Tax Number) do sở Thuế Vụ phát hành cho công dân. Lưu ý là các quy luật này cũng quan trọng không kém các quy luật tổng quát về liên chính. Nếu ta không áp dụng các Database-Specific Integrity Rules nghiêm chỉnh thì database có thể bị hư và không còn dùng được.

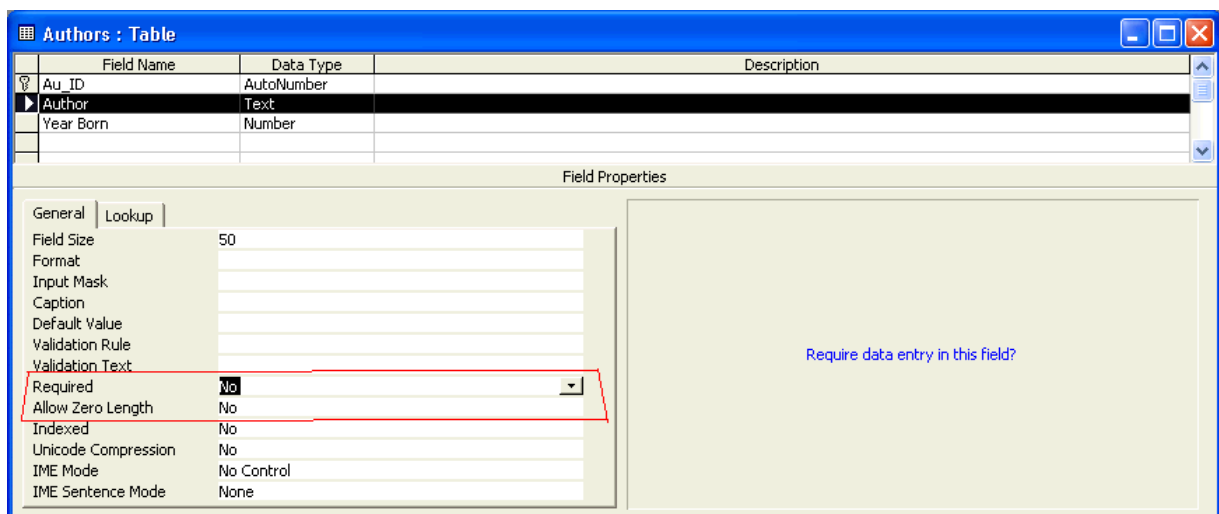
Microsoft Access Database Management System (MSAccess DBMS)

Microsoft Access Database Management System gồm có Database Engine và những công cụ đi chung để cung cấp cho users một môi trường làm việc thân thiện với database, như Database Design (thiết kế các tables và mối liên hệ), Data entry và báo cáo (reports). Kèm theo với Visual Basic 6.0 khi ta mua là một copy của Database Engine của MSAccess. Tên nó là **Jet Database Engine**, cái lõi của MSAccess DBMS. Các chương trình VB6 có thể truy cập database qua Jet Database Engine.

Nếu trên computer của bạn có cài sẵn MSAccess, thì bạn có thể dùng đó để thiết kế các tables của database hay cho data vào các tables.

Properties Required và Allow Zero Length

Khi thiết kế một table field, lưu ý property **Required** và nhất là property **Allow Zero Length** của Text. Nếu property **Required** của một field là **Yes** thì ta không thể update (viết) một record với field ấy có trị số NULL. Nếu một Text field có property **Allow Zero Length** là **No** thì ta không thể update một record khi field ấy chứa một empty string.

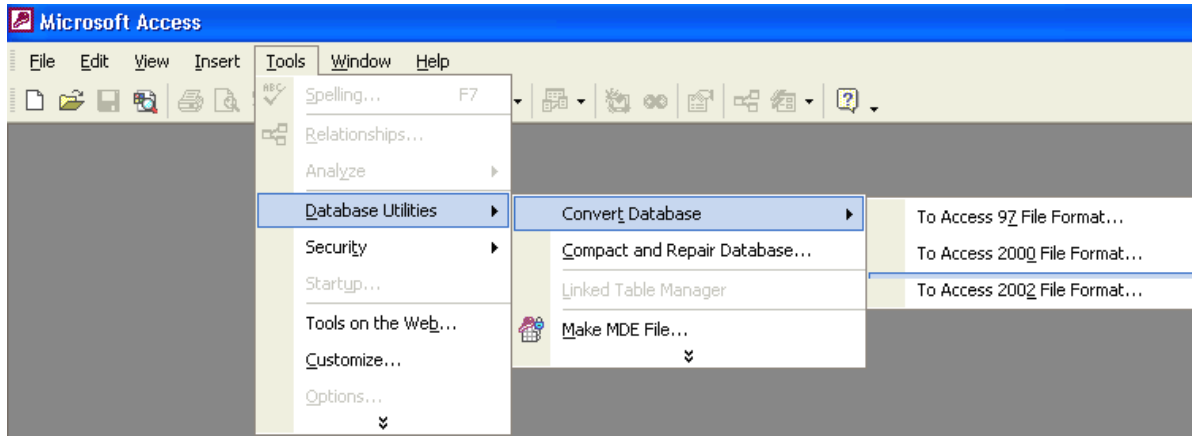


Khi ta tạo một record lần đầu, nếu không cho trị số của một field, thì field ấy có trị số là **NULL**.

Thường thường, Visual Basic 6.0 không thích NULL value nên ta phải thử một field với **Function IsNULL()** để đảm bảo nó không có trị số NULL trước khi làm việc với nó. Nếu IsNULL trả về trị số False thì ta có thể làm việc với field ấy. Nhớ là khi trị số NULL được dùng trong một expression, ngay cả khi chương trình không cho Error, kết quả cũng là NULL.

Làm việc với các versions khác nhau

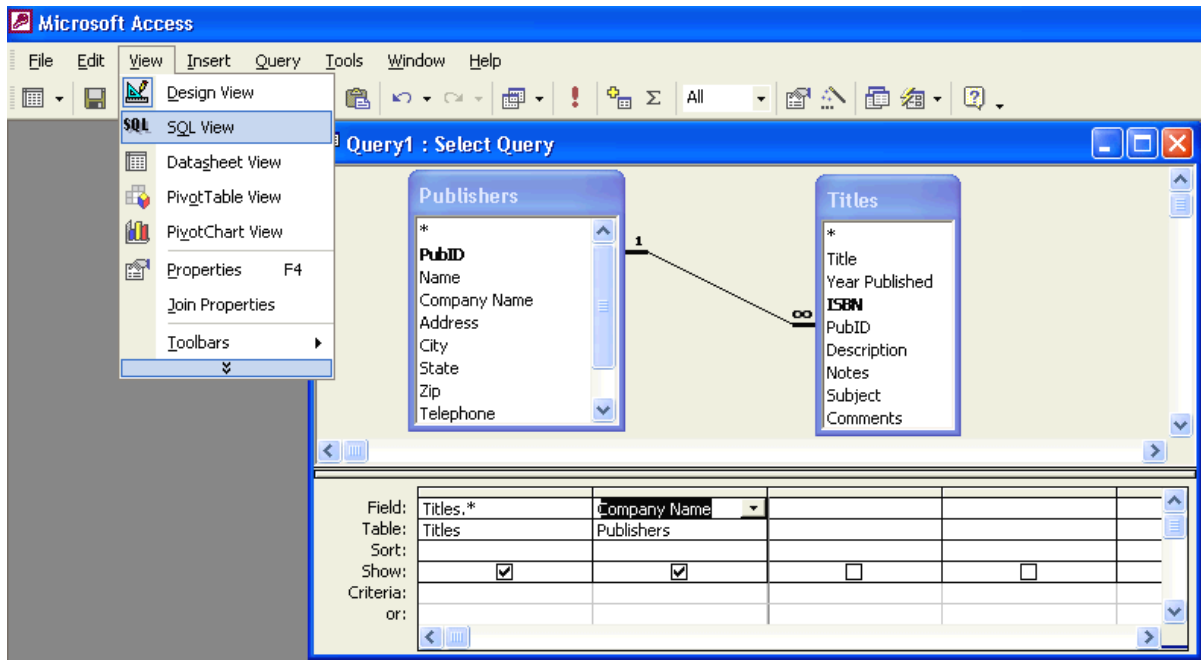
Nếu máy bạn đang chạy MSAccess2002 thì bạn có thể làm việc với Access database file version 97, 2000 và 2002. Nếu cần phải convert từ version này qua version khác, bạn có thể dùng Access DBMS Menu Command **Tools | Database Utilities | Convert Database | To Access 2002 File Format...**. Nếu muốn giữ nguyên version, bạn có thể convert database qua File Format 2002 để sửa đổi, rồi sau đó convert trở lại File Format cũ.



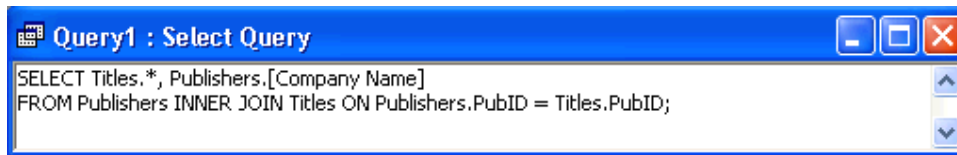
Access database file lớn lên rất nhanh, vì các records đã bị deleted vẫn còn nằm nguyên, nên mỗi tuần bạn nên nhớ nén nó lại để bỏ hết các records đã bị deleted bằng cách dùng Access DBMS Menu Command **Tools | Database Utilities | Compact and Repair Database...** hay dùng **function DBEngine.CompactDatabase** trong VB6.

Dùng Query để viết SQL

Một cách để truy cập database là dùng ngôn ngữ **Structured Query Language (SQL)** theo chuẩn do ISO/IEC phát hành năm 1992, gọi tắt là **SQL92**. Tất cả mọi database thông dụng đều hỗ trợ SQL, mặc dầu nhiều khi chúng còn cho thêm nhiều chức năng rất hay nhưng không nằm trong chuẩn. Các lệnh SQL thông dụng là **SELECT**, **UPDATE**, **INSERT** và **DELETE**. Ta có thể dùng phương tiện thiết kế Query của MSAccess để viết SQL. Sau khi thiết kế Query bằng cách drag drop các fields, bạn có thể dùng Menu Command **View | View SQL** như sau:

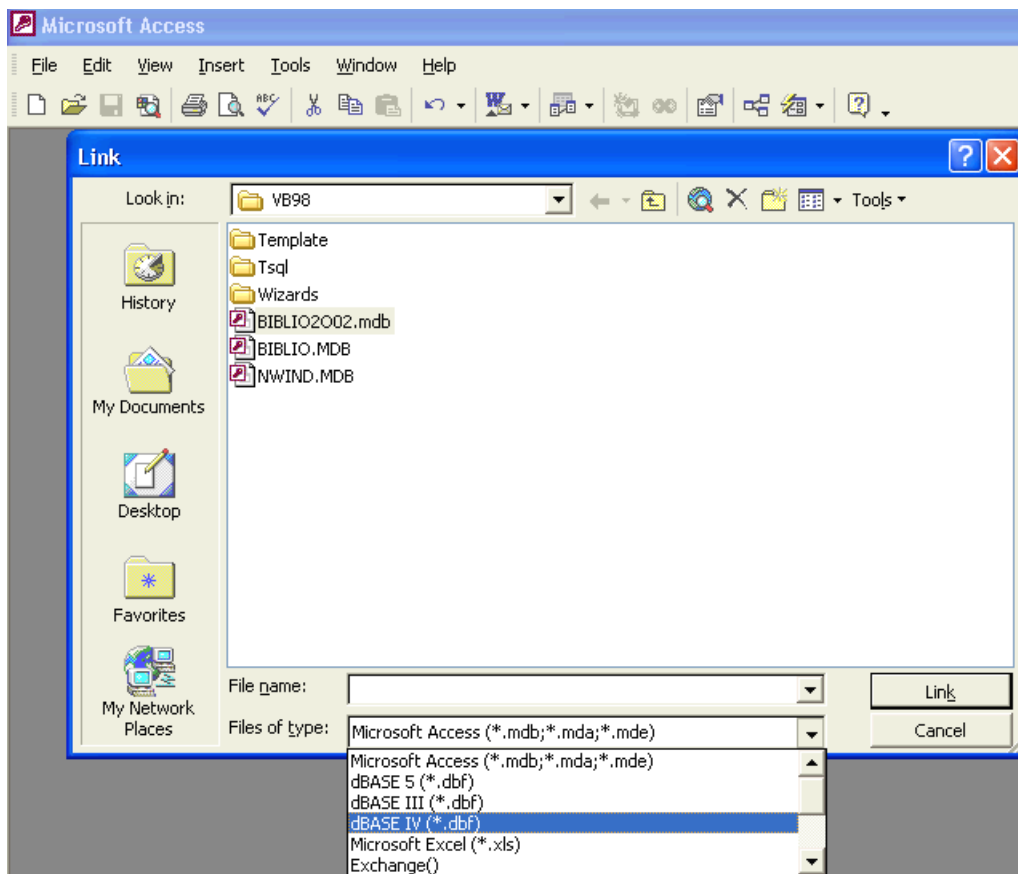


Tiếp theo đây là SQL statement của Query bên trên mà bạn có thể copy để paste vào trong code VB6:



Dùng Link Table để làm việc trực tiếp với database loại khác

Ta có thể dùng một database loại khác, như DBase, trực tiếp trong VB6 như dùng một Access database bình thường. Muốn thiết lập mối nối ấy, bạn dùng Menu Command **File | Get External Data | Link Tables...** rồi chọn loại DBase và chính file của table mà bạn muốn dùng để nhét nó vào Access database đang mở:



Database Server và một số ý niệm

Dù Jet Database Engine là một relational database rất tốt và hiệu năng, nó thuộc loại **File Based database**, tức là nó thụ động, không chạy một mình nhưng phải tùy thuộc vào chương trình dùng nó. File Based database không thích hợp với những ứng dụng có nhiều người dùng cùng một lúc.

Trong khi đó, một Database Server như **SQLServer** chạy riêng để phục vụ bất cứ chương trình khách (client) nào cần. Database Server thích hợp cho các ứng dụng có nhiều users vì chỉ có một mình nó chịu trách nhiệm truy cập dữ liệu cho mọi clients. Nó có thể chứa nhiều routines địa phương, gọi là **Stored Procedures**, để thực hiện các công tác client yêu cầu rất hiệu năng. Database Server thường có cách đối phó hữu hiệu khi có sự cố về phần cứng như đĩa hư hay cúp điện. Ngoài ra, Database Server có sẵn các phương tiện về an ninh và backup. Nó cũng có thêm các chức năng để dùng cho mạng.

Ngày nay ta thu thập dữ liệu dưới nhiều hình thức như Email, Word documents, Spreadsheets. Không nhất thiết dữ liệu luôn luôn được chứa dưới dạng table của những records và không nhất thiết dữ liệu luôn luôn được lưu trữ trong một database đang hoạt động. Dù vậy, chúng vẫn được xem như database dưới mắt một chương trình ứng dụng. Do đó, ta dùng từ **Data Store** (Kho dữ liệu) thay thế cho database để nói đến nơi chứa dữ liệu. Và đối với chương trình tiêu thụ dữ liệu, ta nói đến **Data Source** (Nguồn dữ liệu) thay vì database.

Khi lập trình bằng VB6 để truy cập database, ta nhìn database một cách trừu tượng, tức là dầu nó là Access, dBase, SQLServer hay Oracle ta cũng xem như nhau. Nếu có thay đổi loại database bên dưới, cách lập trình của ta cũng không thay đổi bao nhiêu.

Trong tương lai, một **XML file** cũng có thể được xem như một database nho nhỏ. Nó có thể đứng một mình hay là một table trích ra từ một database chính huy. XML là một chuẩn mà ta có thể dùng để import/export dữ liệu với tất cả mọi loại database hỗ trợ XML. Ta có thể trao đổi dữ liệu trên mạng

Intenet dưới dạng XML. Ngoài ra, thay vì làm việc trực tiếp với một database lớn, ta có thể trích ra vài tables từ database ấy thành một XML file. Kế đó ta chỉ lập trình với XML file cho đến khi kết thúc sẽ hòa (merge/reconcile) XML file với database lớn. Nếu phần lớn các chương trình áp dụng được thiết kế để làm việc cách này, thì trong tương lai ta không cần một Database Server thật mạnh.

Chương Mười Bốn - Dùng Control Data

Control Data

Từ VB5, Visual Basic cho lập trình viên một control để truy cập cơ sở dữ liệu, tên nó chỉ đơn sơ là **Data**. Như ta biết, có một cơ sở dữ liệu Microsoft gói kèm khi ta mua VB6 - đó là **Jet Database Engine**. Jet Database Engine là cái "**phòng máy**" của chính MS Access Database Management System.

Cho đến thời VB5, Microsoft cho ta ba kỹ thuật chính:

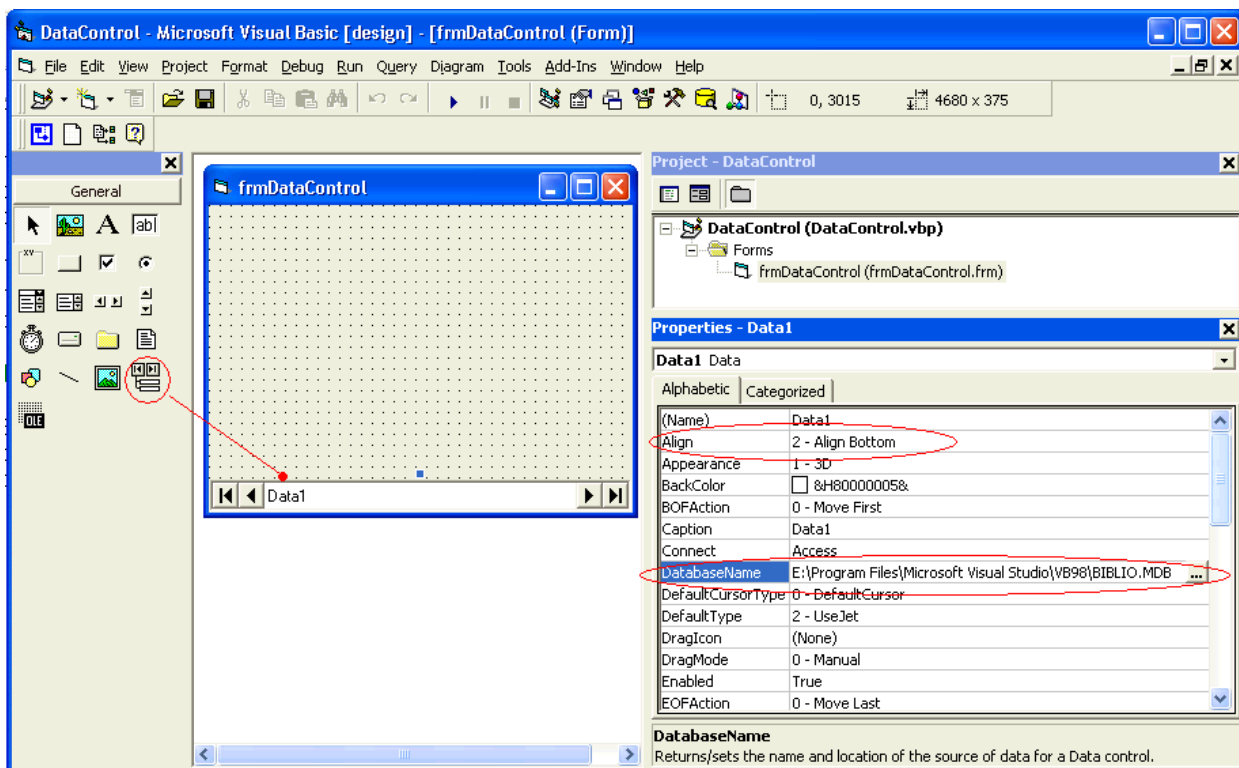
- **DAO (Data Access Objects)**: DAO là kỹ thuật bí truyền của Microsoft, chỉ để dùng với Jet Database Engine. Nó rất dễ dùng, hiệu năng và tiện, nhưng bị giới hạn trong phạm vi MS Access. Dầu vậy, nó rất thịnh hành vì có lợi ích thực tiễn.
- **ODBC (Open Database Connectivity)**: ODBC được thiết kế để cho phép users nối với đủ loại databases mà chỉ dùng một method duy nhất. Điều này cất bớt gánh nặng cho lập trình viên, để chỉ cần học một kỹ thuật lập trình duy nhất mà có thể làm việc với bất cứ loại database nào. Nhất là khi sau này nếu cần phải thay đổi loại database, như nâng cấp từ Access lên SQLServer chẳng hạn, thì sự sửa đổi về coding rất ít. Khi dùng ODBC chung với DAO, ta có thể cho Access Database nối với các databases khác. Có một bất lợi của ODBC là nó rắc rối.
- **RDO (Remote Data Object)**: Một trong những lý do chính để RDO được thiết kế là giải quyết khó khăn về sự rắc rối của ODBC. Cách lập trình với RDO đơn giản như DAO, nhưng thật ra nó dùng ODBC nên cho phép users nối với nhiều databases. Tuy nhiên, RDO không được thịnh hành lắm.

VB6 tiếp tục hỗ trợ các kỹ thuật nói trên, và cho thêm một kỹ thuật truy cập database mới, rất quan trọng, đó là **ADO (ActiveX Data Objects)**. Trong một bài tới ta sẽ học về ADO với những ưu điểm của nó. Tuy nhiên, vì DAO rất đơn giản và hiệu năng nên ta vẫn có thể tiếp tục dùng nó rất hữu hiệu trong hầu hết các áp dụng. Do đó bài này và bài kế sẽ tập trung vào những kỹ thuật lập trình phổ biến với DAO.

Cách dùng giản tiện của control Data là đặt nó lên một Form rồi làm việc với những Properties của nó. Bạn hãy bắt đầu một dự án VB6 mới, cho nó tên **DataControl** bằng cách click tên project trong Project Explorer bên phải rồi edit property Name trong Properties Window.

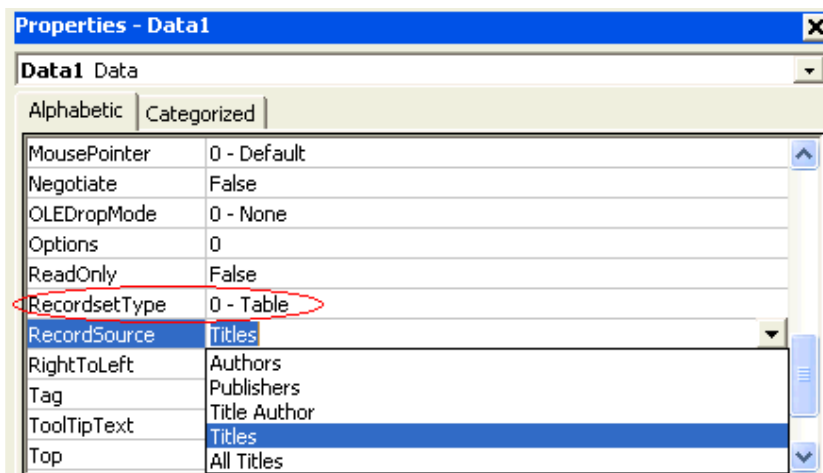
DoubleClick lên Icon của Control Data trong Toolbox. Một Control Data tên **Data1** sẽ hiện ra trên Form. Muốn cho nó nằm bên dưới Form, giống như một StatusBar, hãy set **property Align** của nó trong Properties Window thành **2 - Align Bottom**.

Click bên phải hàng **property DatabaseName**, kế đó click lên nút browse có ba chấm để chọn một file Access database từ giao thoại cho Data1. Ở đây ta chọn **E:\Program Files\Microsoft Visual Studio\VB98\BIBLIO.MDB**, trong computer của bạn có thể nó nằm trên disk C hay D.



Trong chương trình này ta muốn làm việc với **table Titles** của database BIBLIO.MDB, để xem và edit các records. Để ý **property DefaultType** của Data1 có trị số **2- UseJet**, tức là dùng kỹ thuật DAO, thay vì dùng kỹ thuật ODBC.

Khi bạn click lên **property Recordsource** của Data1, rồi click lên cái tam giác nhỏ bên phải, một ComboBox sẽ mở ra cho ta thấy danh sách các tables trong database. Bạn hãy chọn **Titles**. Để ý **property RecordsetType** của Data1 có trị số là **0 - Table**:



Cái từ mới mà ta sẽ dùng thường xuyên khi truy cập dữ liệu trong VB6 là **Recordset** (bộ records). Recordset là một **Set of records**, nó có thể chứa một số records hay không có record nào cả. Một record trong Recordset có thể là một record lấy từ một Table. Trong trường hợp ấy có thể ta lấy về tất cả records trong table hay chỉ những records thỏa đúng một điều kiện, thí dụ như ta chỉ muốn lấy các records của những sách xuất bản trước năm 1990 (Year Published < 1990).

Một Record trong Recordset cũng có thể là tập hợp các cột (columns) từ hai (hay ba)

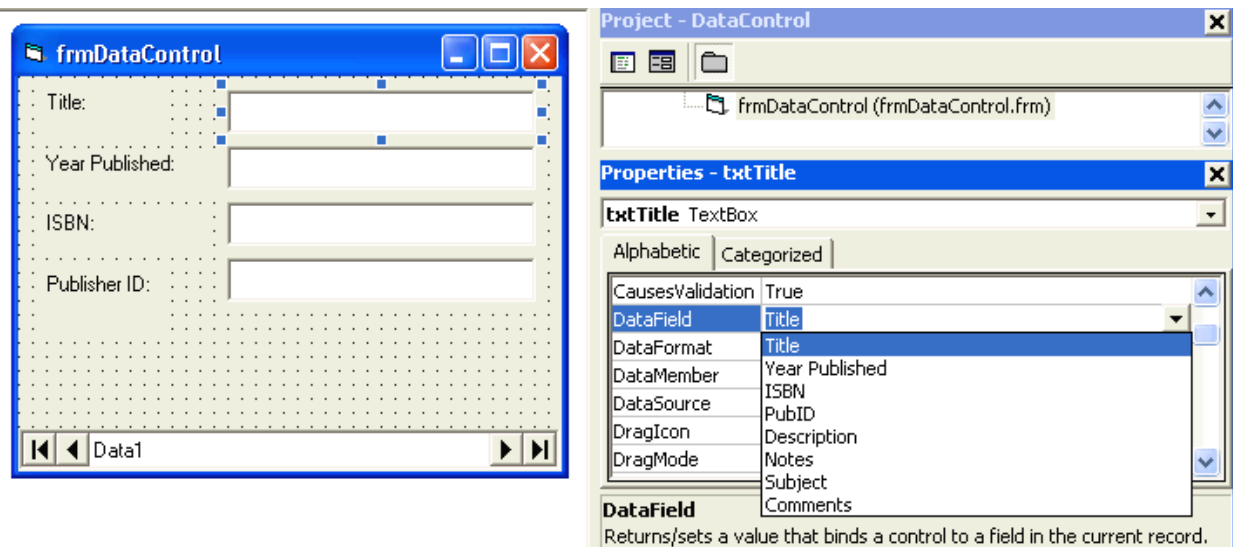
tables qua các mối liên hệ one-to-one và one-to-many. Thí dụ như khi lấy các records từ table Titles, ta muốn có thêm chi tiết tên công ty (Company Name) và điện thoại (Telephone) của nhà xuất bản (table Publishers) bằng cách dùng **Foreign Key PubID** trong table Titles làm **Primary Key** trong table Publishers để lấy các chi tiết ấy. Nếu bạn chưa nắm vững ý niệm Foreign Key thì hãy đọc lại bài Database.

Trong trường hợp ấy ta có thể xem như có một **virtual (ảo) table** là tập hợp của hai tables Titles và Publishers.

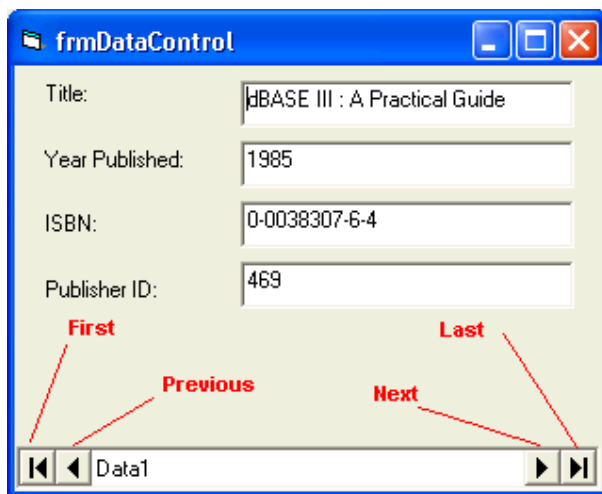
Bây giờ bạn hãy đặt lên Form 4 labels với captions: **Title, Year Published, ISBN** và **Publisher ID**. Kế đó cho thêm 4 textboxes tương ứng và đặt tên chúng là **txtTitle, txtYearPublished, txtISBN** và **txtPublisherID**.

Chọn textbox txtTitle, rồi set **property Datasource** của nó trong Properties Window thành **Data1**. Khi click lên **property Datafield** của txtTitle và mở ComboBox ra bạn sẽ thấy liệt kê tên các Fields trong table Titles. Đó là vì Data1 được coi như trung gian lấy table Titles từ database. Ở đây ta sẽ chọn cột Title.

Lập lại công tác này cho 3 textboxes kia, và chọn các cột Year Published (năm xuất bản), ISBN (số lý lịch trong thư viện quốc tế), và PubID (số lý lịch nhà xuất bản) làm Datafield cho chúng.



Tới đây, mặc dầu chưa viết một hàng code, ta có thể chạy chương trình được rồi. Nó sẽ hiển thị chi tiết của record đầu tiên trong table Titles như dưới đây:



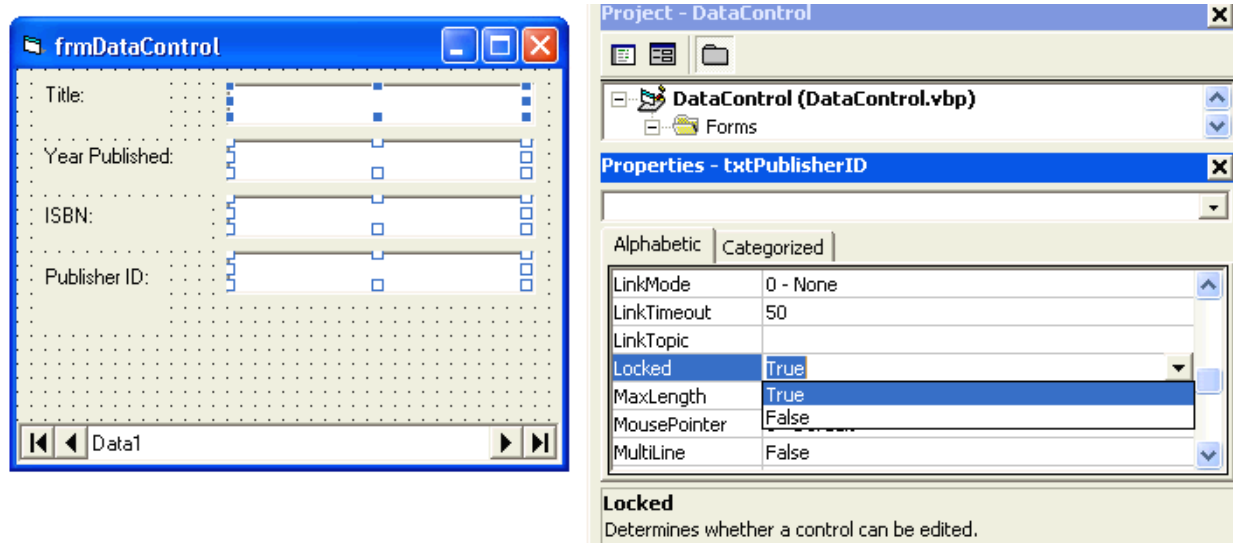
Bạn có thể bấm các nút di chuyển **Navigator Buttons** để đi đến các record **đầu (first)**, **trước (previous)**, **kế (next)** và **cuối (last)**. Mỗi lần bạn di chuyển đến một record mới là chi tiết của record ấy sẽ hiển thị. Nếu không dùng các Navigator Buttons, ta cũng có thể code để làm công tác tương đương bằng cách gọi các Recordset **methods MoveFirst, MovePrevious, MoveNext** và **MoveLast**.

Khi record cuối của Recordset đang hiển thị, nếu ta gọi method MoveLast thì **property EOF (End-Of-File)** của Recordset trở thành True. Tương tự như vậy, khi record thứ nhất của Recordset đang hiển thị, nếu ta gọi method MovePrevious thì **property BOF (Begin-Of-File)** của Recordset trở thành True. Nếu một Recordset không có chứa một record nào cả thì cả hai properties EOF và BOF đều là True.

Đặc tính hiển thị dữ liệu trong các textboxex theo đúng record hiện thời (**current record**) được gọi là **data binding** hay **data bound** (buộc vào dữ liệu) và control TextBox hỗ trợ chức năng này được nói là **Data Aware** (biết bà con dữ liệu).

Khi record đầu tiên đang hiển thị, nếu bạn edit **Year Published** để đổi từ 1985 thành **1983** rồi click Navigator button Next để hiển thị record thứ nhì, kế đó click Navigator button Previous để hiển thị lại record đầu tiên thì bạn sẽ thấy là field Year Published của record đầu tiên đã thật sự được thay đổi (updated) thành 1983.

Điều này có nghĩa rằng khi Data1 navigates từ record này đến record khác thì nếu record này đã có sự thay đổi vì user edited, nó lưu trữ sự thay đổi đó trước khi di chuyển. Chưa chắc là bạn muốn điều này, do đó, nếu bạn không muốn user tình cờ edit một record thì bạn có thể set **property Locked** của các textboxes ấy thành True để user không thể edit các textboxes như trong hình dưới đây:



Chỉ định vị trí Database lúc chạy chương trình

Cách chỉ định tên DatabaseName trong giai đoạn thiết kế (at design time) ta đã dùng trước đây tuy tiện lợi nhưng hơi nguy hiểm, vì khi ta cài chương trình này lên computer của khách, chưa chắc file database ấy nằm trong một folder có cùng tên. Thí dụ trên computer mình thì database nằm trong folder E:\Program Files\Microsoft Visual Studio\VB98, nhưng trên computer của khách thì database nằm trong folder C:\VB6\DataControl chẳng hạn. Do đó, khi chương trình khởi động ta nên xác định lại vị trí của database. Giả dụ ta muốn để database trong cùng một folder với chương trình đang chạy, ta có thể dùng **property Path** của Application Object **App** như sau:

```
Dim AppFolder As String
Private Sub Form_Load()
    ' Fetch Folder where this program EXE resides
    AppFolder = App.Path
    ' make sure it ends with a back slash
    If Right(AppFolder, 1) <> "\" Then AppFolder =
AppFolder & "\"
    ' Assign Full path database filename to Data1
    Data1.DatabaseName = AppFolder &
"BIBLIO.MDB"
End Sub
```

Với cách code nói trên ta sẽ đảm bảo chương trình tìm thấy file database đúng chỗ, không cần biết người ta cài chương trình bạn ở đâu trong hard disk của computer khách.

Nếu bạn đang học VB6 từ xa, khi nộp bài database cho giám thị VB6 mà bạn hardcode

(viết chết cứng) vị trí của file database trong lúc thiết kế thì giám thị (tutor) cũng gặp cùng sự khó khăn này vì chưa chắc giám thị sẽ chứa database trong một folder có cùng tên như trong harddisk của bạn.

Thêm bớt các Records

Chương trình trên dùng cũng tạm được, nhưng nó không cho ta phương tiện để thêm (add), bớt (delete) các records. Bây giờ bạn hãy để vào Form 5 buttons tên: **cmdEdit**, **cmdNew**, **cmdDelete**, **cmdUpdate** và **cmdCancel**.

Mặc dầu bạn không thấy, nhưng thật ra Control Data **Data1** có một **property Recordset** và khi ta dùng Navigator buttons là di chuyển từ record này đến record khác trong Recordset ấy. Ta có thể nói đến nó bằng Notation (cách viết) **Data1.Recordset**, và mỗi lần muốn lấy Recordset mới nhất từ database ta dùng **method Refresh** như **Data1.Recordset.Refresh**.

Lúc chương trình mới khởi động, user đang xem (browsing) các records thì hai buttons **Update** và **Cancel** không cần phải làm việc. Do đó ta sẽ nhân tiện Lock (khóa) các textboxes và disable (làm cho bất lực) hai buttons này vì không cần dùng chúng.

Trong **Sub SetControls** dưới đây, ta dùng một parameter gọi là **Editing** với trị số False hay True tùy theo user đang Browse hay Edit, ta gọi là **Browse mode** và **Edit mode**. Trong **Edit mode**, các Textboxes được unlocked (mở khóa) và các nút **cmdNew**, **cmdDelete** và **cmdEdit** trở nên bất lực:

```
Sub SetControls(ByVal Editing As Boolean)
    ' Lock/Unlock textboxes
    txtTitle.Locked = Not Editing
    txtYearPublished.Locked = Not Editing
    txtISBN.Locked = Not Editing
    txtPublisherID.Locked = Not Editing
    ' Enable/Disable buttons
    CmdUpdate.Enabled = Editing
    CmdCancel.Enabled = Editing
    CmdDelete.Enabled = Not Editing
    cmdNew.Enabled = Not Editing
    CmdEdit.Enabled = Not Editing
End Sub
```

Trong **Browse mode**, Form có dạng như sau:

Sub SetControls được gọi trong **Sub Form_Load** khi chương trình khởi động và trong **Sub CmdEdit** khi user click nút **Edit** như sau:

```
Private Sub Form_Load()
    ' Fetch Folder where this program EXE resides
    AppFolder = App.Path
    ' make sure it ends with a back slash
    If Right(AppFolder, 1) <> "\" Then AppFolder = AppFolder & "\"
    ' Assign Full path database filename to Data1
    Data1.DatabaseName = AppFolder & "BIBLIO.MDB"
    ' Place controls in Browse Mode
    SetControls (False)
End Sub

Private Sub CmdEdit_Click()
    ' Place controls in Edit Mode
    SetControls (True)
End Sub
```

Khi ta Delete một record trong recordset, vị trí của record hiện tại (current record) vẫn không thay đổi. Do đó, sau khi delete một record ta phải **MoveNext**. Khổ nỗi, nếu ta vừa delete record cuối của Recordset thì sau khi MoveNext, **property EOF** của Recordset sẽ thành True. Thành ra ta phải kiểm tra điều đó, nếu đúng vậy thì lại phải **MoveLast** để hiển thị record cuối của Recordset như trong code của **Sub cmdDelete_Click** dưới đây:

```
Private Sub CmdDelete_Click()
```

```

On Error GoTo DeleteErr
With Data1.Recordset
    ' Delete new record
    .Delete
    ' Move to next record
    .MoveNext
    If .EOF Then .MoveLast
Exit Sub
End With
DeleteErr:
    MsgBox Err.Description
Exit Sub
End Sub

```

Trong lúc code, ta Update (cập nhật hóa) một record trong Recordset bằng method **Update**. Nhưng ta chỉ có thể gọi method Update của một Recordset khi Recordset đang ở trong **Edit hay AddNew mode**. Ta đặt một Recordset vào Edit mode bằng cách gọi **method Edit** của Recordset, thí dụ như **Data1.Recordset.Edit**. Tương tự như vậy, ta đặt một Recordset vào AddNew mode bằng cách gọi **method AddNew** của Recordset, thí dụ như **Data1.Recordset.AddNew**.

```

Private Sub cmdNew_Click()
    ' Place Recordset into Recordset AddNew mode
    Data1.Recordset.AddNew
    ' Place controls in Edit Mode
    SetControls (True)
End Sub

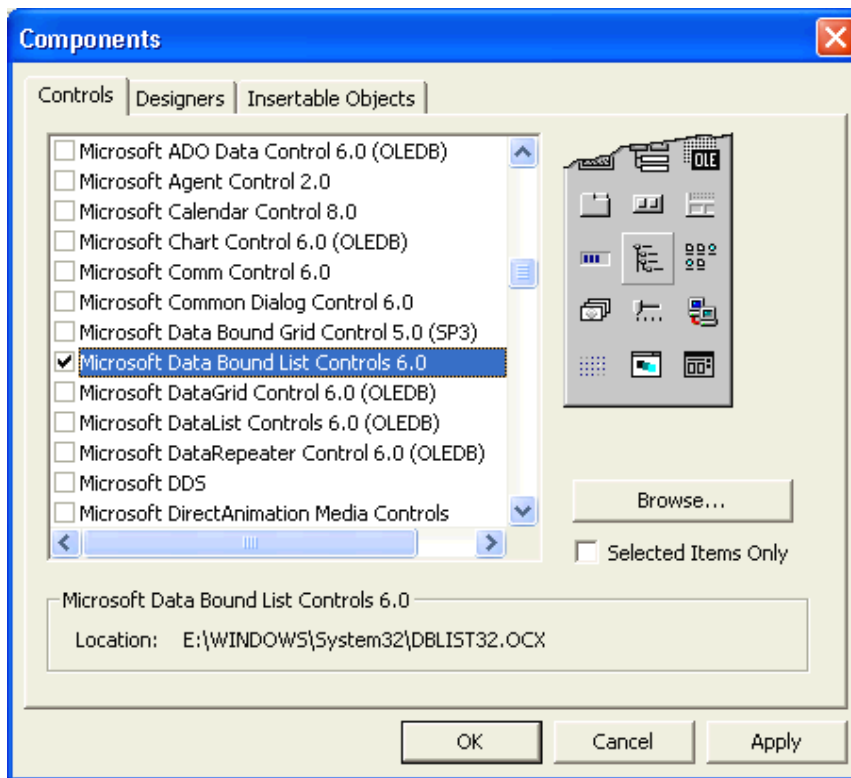
```

Sau khi Recordset gọi method Update thì Recordset ấy ra khỏi AddNew hay Edit modes. Ta cũng có thể tự thoát ra khỏi AddNew hay Edit modes, hay nói cho đúng hơn là hủy bỏ mọi pending (đang chờ đợi) Update bằng cách gọi **method CancelUpdate**, thí dụ như **Data1.Recordset.CancelUpdate**.

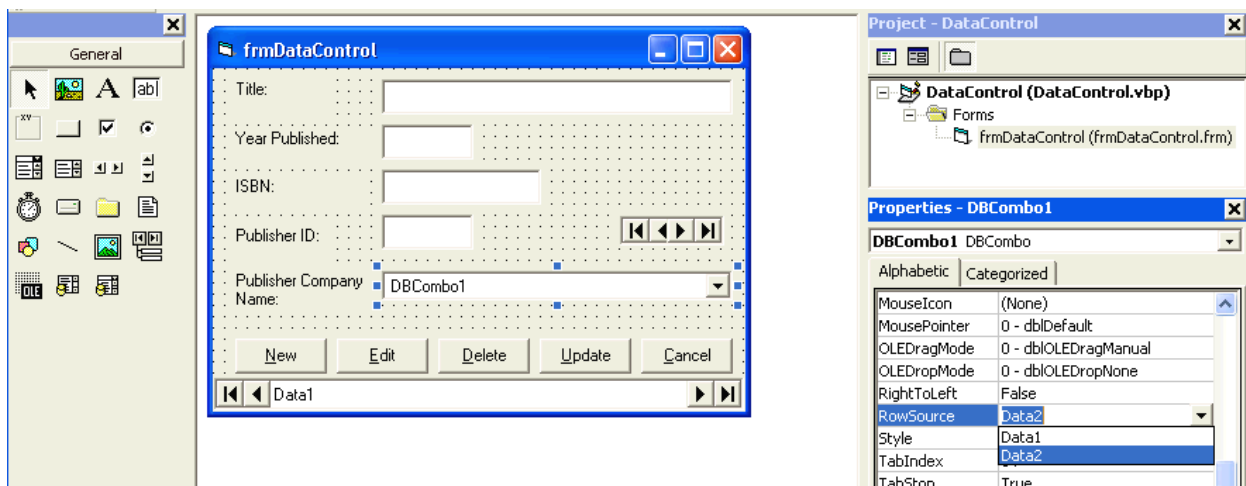
Dùng DataBound Combo

Trong chương trình hiện tại ta chỉ hiển thị lý lịch nhà xuất bản (PubID) của Title, chứ không có thêm chi tiết. Phải chi mặc dầu chương trình lưu trữ **PubID**, nhưng hiển thị được **Company Name** của nhà xuất bản cho ta làm việc để khỏi phải nhớ các con số thì

hay quá. Ta có thể thực hiện điều đó bằng cách dùng Control **DBCombo (Data Bound Combo)**. Bạn hãy dùng IDE Menu Command **Project | Components...** để chọn **Microsoft Data Bound List Controls 6.0** rồi click **Apply**.



Kế đó, thêm một DBCombo tên **DBCombo1** vào Form. Vì ta cần một Recordset khác để cung cấp Table Publisher cho DBCombo1, nên bạn hãy thêm một control Data thứ nhì tên **Data2** vào Form. Cho Data2, hãy set property DatabaseName thành **E:\Program Files\Microsoft Visual Studio\VB98\BIBLIO.MDB** và property RecordSource thành **Publishers**. Để không cho người ta thấy hình Data2 lúc run-time, bạn hãy set property **Visible** nó thành False.



Cái mục đích của chúng ta khi dùng DBCombo1 là hiển thị Company Name của nhà xuất bản, nhưng đằng sau lưng thì không có gì thay đổi, tức là ta vẫn làm việc với PubID cho các record Title của Data1. Khi user click lên DBCombo1 để chọn một nhà

xuất bản, thì ta theo Company Name đó mà chứa PubID tương ứng trong record Title của Data1. Do đó có nhiều thứ ta phải sắp đặt cho DBCombo1 như sau:

Property	Value	Chú thích
RowSource	Data2	Đây là datasource của chính DBCombo1. Nó cung cấp table Publishers.
Listfield	Company Name	Khi RowSource phía trên đã được chọn rồi, Combo của property Listfield này sẽ hiển thị các fields của table Publishers. Company Name là field của RowSource mà ta muốn hiển thị trên DBCombo1.
DataSource	Data1	Đây là datasource của record mà ta muốn. edit, tức là record của table Titles
Datafield	PubID	Field (của record Title) sẽ được thay đổi.
BoundColumn	PubID	Field trong RowSource (table Publishers) tương ứng với item user chọn trong DBCombo1 (Company Name).

Khi trong Edit mode user chọn một Company Name khác trong DBCombo1 rồi click nút Update bạn sẽ thấy Textbox txtPublisherID cũng đổi theo và hiển thị con số lý lịch PubID mới. Nếu trước khi Update bạn muốn thấy PubID mới hiển thị trong Textbox txtPublisherID thì bạn có thể dùng Event Click của DBCombo1 như sau:

```
Private Sub DBCombo1_Click(Area As Integer)
    ' Display new PuBID
    txtPublisherID.Text = DBCombo1.BoundText
End Sub
```

Property BoundText của DBCombo1 là trị số của BoundColumn mà ta có thể truy cập (viết hay đọc) được. Thí dụ như bạn muốn mỗi khi thêm một record Title mới thì default PubID là 324, tức là Company Name= "GLOBAL ENGINEERING". Bạn có thể assign trị số 324 vào property BoundText của DBCombo1 trong Sub cmdNew_Click như sau:

```
Private Sub cmdNew_Click()
    ' Place Recordset into Recordset AddNew mode
```

```

Data1.Recordset.AddNew
' Default Publisher is "GLOBAL ENGINEERING", i.e. PubID=324
DBCombo1.BoundsText = 324
' Place controls in Edit Mode
SetControls (True)
End Sub

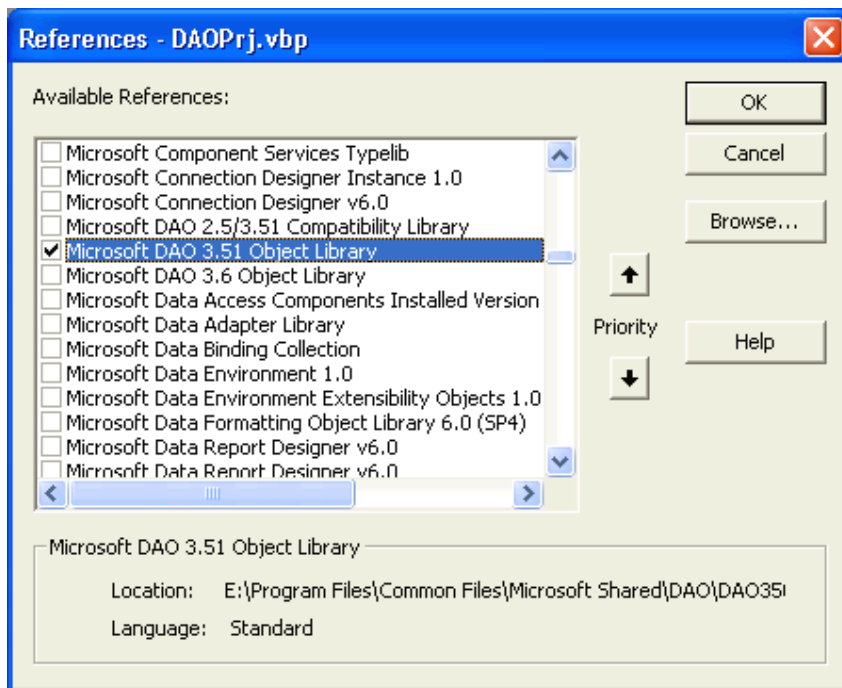
```

Trong bài tới ta sẽ học thêm về cách coding để dùng Recordset trong kỹ thuật DAO.

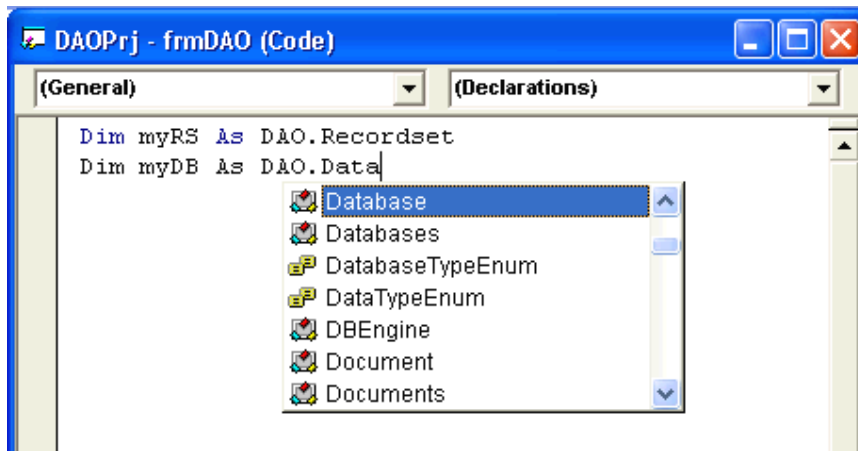
Chương Mười Lăm - Lập trình với kỹ thuật DAO

Reference DAO

Trong bài này ta sẽ học những cách lập trình căn bản với MS Access database qua kỹ thuật DAO mà không cần dùng đến **Control Data** như trong bài trước. Ta sẽ cần đến vài Objects trong thư viện DAO, do đó nếu bạn mở một dự án VB6 mới thì hãy dùng Menu Command **Project | References...** để chọn **Microsoft DAO 3.51 Object Library** bằng cách click cái checkbox bên trái như trong hình dưới đây. (Một cách để nhớ tên của Object này là nhớ câu "thằng cha của ĐÀO 35 con dê").



Sau đó trong code của Form chính ta sẽ declare variable **myDatabase** cho một instance của **DAO database** và variable **myRS** cho một **DAO recordset**. Ở đây ta nói rõ Database và Recordset là thuộc loại **DAO** để phân biệt với Database và Recordset thuộc loại **ADO (ActiveX Data Object)** sau này. Để ý là Intellisense giúp ta trong lúc viết code:



Bây giờ bạn hãy đặt lên Form chính, tên **frmDAO**, 4 labels với captions: **Title**, **Year Published**, **ISBN** và **Publisher ID**. Kế đó cho thêm 4 textboxes tương ứng và đặt tên chúng là **txtTitle**, **txtYearPublished**, **txtISBN** và **txtPublisherID**.

Điều ta muốn làm là khi Form mới được loaded, nó sẽ lấy về từ database một Recordset chứa tất cả records trong **table Titles** theo thứ tự về mẫu tự (alphabetical order) của **field Title** và hiển thị record đầu tiên.

Dùng keyword SET

Chuyện trước hết là mở một Database Object dựa vào tên đầy đủ (full path name) của Access database:

```
' Open main database
Set myDB = OpenDatabase(AppFolder & "BIBLIO.MDB")
```

Để ý chữ **Set** trong câu code trên. Đó là vì myDB là một **Pointer** đến một Object. Mặc dầu từ rày về sau ta sẽ dùng myDB như một Database theo cách giống như bất cứ variable thuộc data type nào khác, nhưng khi chỉ định lần đầu là nó từ đâu đến thì ta dùng chữ Set, để nói rằng thật ra myDB không phải là Object Database, nhưng là Pointer đến Object Database. Điểm này càng nói đến càng khó hiểu.

Đại khái là VB6 runtime dynamically allocates (dành ra cho khi cần) một phần trong bộ nhớ (memory) để chứa Object Database khi ta nhận được nó từ execution của **Method OpenDatabase**. Dầu vị trí chỗ chứa Object Database trong bộ nhớ không nhất định, nhưng vì ta nắm cái cán chỉ đến vị trí ấy nên ta vẫn có thể làm việc với nó một cách bình thường. Cái cán ấy là value (trị số) của variable myDB. Vì value này không

phải là Object, nhưng nó chứa **memory address** chỉ đến (**point to** hay **refer to**) Object Database, nên ta gọi nó là Pointer.

Lập trình dùng Pointer nói chung rất linh động là hiệu năng trong các ngôn ngữ như C, Pascal, C++ ,v.v.. Tuy nhiên, lập trình viên phải nhớ trả lại Operating System phần memory mình dùng khi không còn cần nó nữa để Operating System lại allocate cho Object khác. Nếu công việc quản lý dùng lại memory không ổn thỏa thì có những mảnh memory nằm lang bang mà Operating Sytem không biết. Lăn lăn Operating System sẽ không còn memory dư nữa. Ta gọi hiện tượng ấy là **memory leakage (rỉ)**. Các ngôn ngữ sau này như Java, C# đều không dùng Pointer nữa. Visual Basic không muốn lập trình viên dùng Pointer. Chỉ trong vài trường hợp đặc biệt VB6 mới lộ ra cho ta thấy thật ra ở trong hậu trường VB6 Runtime dùng Pointer, như trong trường hợp này.

Tương tự như vậy, vì Recordset là một Pointer đến một Object, ta cũng dùng **Set** khi chỉ định một DAO Recordset lấy về từ **Method OpenRecordset** của database myDB.

```
'Open recordset
Set myRS = myDB.OpenRecordset("Select * from Titles ORDER BY Title")
```

Cái parameter loại String ta dùng cho method OpenRecordset là một **Lệnh (Statement) SQL**. Nó chỉ định cho database lấy tất cả mọi fields (columns) (**Select ***) của mỗi record từ Table Titles (**from Titles**) làm một Recordset và sort các records trong Recordset ấy theo alphabetical order của field Title (**ORDER BY Title**).

Nhớ là Recordset này cũng giống như **property Recordset** của một Control Data mà ta dùng trong bài trước. Bây giờ có Recordset rồi, ta có thể hiển thị chi tiết của record đầu tiên nếu Recordset ấy có ít nhất một record. Ta kiểm tra điều ấy dựa vào **property RecordCount** của Recordset như trong code dưới đây:

```
Private Sub Form_Load()
    ' Fetch Folder where this program EXE resides
    AppFolder = App.Path
    ' make sure it ends with a back slash
    If Right(AppFolder, 1) <> "\" Then AppFolder = AppFolder & "\"
    ' Open main database
    Set myDB = OpenDatabase(AppFolder & "BIBLIO.MDB")
    'Open recordset
    Set myRS = myDB.OpenRecordset("Select * from Titles ORDER BY Title")
    ' if Recordset is not empty then display the first record
    If myRS.RecordCount > 0 Then
```

```

myRS.MoveFirst ' move to first record
Displayrecord ' display details of current record
End If
End Sub

```

Sau khi dùng **method MoveFirst** của Recordset để position **current record** ở Record đầu tiên, ta hiển thị trị số các fields của record bằng cách assign chúng vào các textboxes của Form như sau:

```

Private Sub Displayrecord()
' Assign record fields to the appropriate textboxes
With myRS
' Assign field Title to textbox txtTitle
txtTitle.Text = .Fields("Title")
txtYearPublished.Text = .Fields("[Year Published]")
txtISBN.Text = .Fields("ISBN")
txtPublisherID.Text = .Fields("PubID")
End With
End Sub

```

Đề ý vì field **Year Published** gồm có hai chữ nên ta phải đặt tên của field ấy giữa hai dấu ngoặc vuông ([]). Để tránh bị phiền phức như trong trường hợp này, khi bạn đặt tên database field trong lúc thiết kế một table hãy dán dính các chữ lại với nhau, đừng để rời ra. Thí dụ như dùng **YearPublished** thay vì **Year Published**.

Các nút di chuyển

Muốn có các nút Navigators tương đương với của một Control Data, bạn hãy đặt lên Form 4 buttons mang tên **CmdFirst**, **CmdPrevious**, **CmdNext** và **CmdLast** với captions: <<, <, >, >>.

Code cho các nút này cũng đơn giản, nhưng ta phải coi chừng khi user muốn di chuyển quá record cuối cùng hay record đầu tiên. Ta phải kiểm tra xem **EOF** có trở thành True khi user click CmdNext, hay **BOF** có trở thành True khi user click CmdPrevious:

```

Private Sub CmdNext_Click()
myRS.MoveNext ' Move to next record
' Display record details if has not gone past the last record
If Not myRS.EOF Then
Displayrecord ' display details of current record
Else
myRS.MoveLast ' Move back to last record

```

```

End If
End Sub

Private Sub CmdPrevious_Click()
    myRS.MovePrevious ' Move to previous record
    ' Display record details if has not gone past the first record
    If Not myRS.BOF Then
        Displayrecord ' display details of current record
    Else
        myRS.MoveFirst ' Move back to first record
    End If
End Sub

Private Sub CmdFirst_Click()
    myRS.MoveFirst ' Move back to first record
    Displayrecord ' display details of current record
End Sub

Private Sub CmdLast_Click()
    myRS.MoveLast ' Move back to last record
    Displayrecord ' display details of current record
End Sub

```

Khi chạy chương trình bạn sẽ thấy nó hiển thị chi tiết của Record đầu tiên khác với trong bài trước đây vì các records đã được sorted:

Bạn hãy thử dùng các Navigator buttons cây nhà, lá vườn của mình xem chúng làm việc có đúng không.

Tới đây, không biết bạn có để ý là dù user có vô tình sửa đổi một chi tiết nào trong các textboxes, không có record nào bị cập nhật hóa trong database khi user di chuyển từ record này đến record khác. Lý do là các Textboxes không có Data Bound với các Fields của Recordset.

Thêm bớt các Records

Giống như chương trình trong bài rồi, ta sẽ thêm phương tiện để thêm (add), bớt (delete) các records. Bây giờ bạn hãy để vào Form 5 buttons tên: **cmdEdit**, **cmdNew**, **cmdDelete**, **cmdUpdate** và **cmdCancel**.

Chỗ nào trong chương trình trước ta dùng **Data1.Recordset** thì bây giờ ta dùng **myRS**.

Ta sẽ dùng lại **Sub SetControls** với parameter **Editing** có trị số False hay True tùy theo user đang Browse hay Edit. Trong **Browse mode**, các Textboxes bị Locked (khóa) và các nút **cmdUpdate** và **cmdCancel** trở nên bất lực. Trong **Edit mode**, các Textboxes được unlocked (mở khóa) và các nút **cmdNew**, **cmdDelete** và **cmdEdit** trở nên bất lực.

Vì ở đây không có Data Binding nên đợi cho đến khi **Update (cập nhật hóa)** ta mới đặt Recordset vào **AddNew** hay **Edit mode**. Do đó ta chỉ cần nhớ là khi user edits là đang Edit một record hiện hữu hay thêm một Record mới. Ta chứa trị số Boolean ấy trong variable **AddNewRecord**. Nếu user sắp thêm một record mới thì **AddNewRecord = True**, nếu User sắp Edit một record hiện hữu thì **AddNewRecord = False**.

Ngoài ra, khi User sắp thêm một record mới bằng cách click nút New thì ta phải tự clear (làm trắng) hết các textboxes bằng cách assign Empty string vào text property của chúng như sau:

```
' If Editing existing record then AddNewRecord = False
' Else AddNewRecord = true
Dim AddNewRecord As Boolean

Private Sub ClearAllFields()
    ' Clear all the textboxes
    txtTitle.Text = ""
    txtYearPublished.Text = ""
    txtISBN.Text = ""
    txtPublisherID.Text = ""
End Sub

Private Sub cmdNew_Click()
    ' Remember that this is Adding a new record
    AddNewRecord = True
    ' Clear all textboxes
    ClearAllFields
    ' Place controls in Edit Mode
    SetControls (True)
End Sub

Private Sub CmdEdit_Click()
    ' Place controls in Edit Mode
```



```

SetControls (True)
' Remember that this is Editing an existing record
AddNewRecord = False
End Sub

```

Nếu user clicks Cancel trong khi đang edit các textboxes, ta không cần gọi **method CancelUpdate** vì Recordset chưa bị đặt vào AddNew hay Edit mode. Ở đây ta chỉ cần hiển thị lại chi tiết của current record, tức là hủy bỏ những gì user đang đánh vào:

```

Private Sub CmdCancel_Click()
' Cancel update
SetControls (False)
' Redisplay details or current record
Displayrecord
End Sub

```

Lúc user clicks Update, bạn có dịp để kiểm tra data xem có field nào bị bỏ trống (nhất là **Primary Key ISBN** bắt buộc phải có trị số) hay có gì không valid bằng cách gọi **Function GoodData**. Nếu GoodData trả lại một trị số False thì ta không xúc tiến với việc Update. Nếu GoodData trả về trị số True thì ta đặt Recordset vào AddNew hay Edit mode tùy theo trị số của Boolean variable AddNewRecord.

Giống như khi hiển thị chi tiết của một Record ta phải assign từng Field vào textbox, thì bây giờ khi Update ta phải làm ngược lại, tức là assign property Text của từng textbox vào Record Field tương ứng. Sau cùng ta gọi **method Update** của recordset và cho các controls trở lại Browse mode:

```

Private Function GoodData() As Boolean
' Check Data here. If Invalid Data then GoodData = False
GoodData = True
End Function

Private Sub CmdUpdate_Click()
' Verify all data, if Bad then do not Update
If Not GoodData Then Exit Sub
' Assign record fields to the appropriate textboxes

```

```

With myRS
  If AddNewRecord Then
    .AddNew ' Place Recordset in AddNew Mode
  Else
    .Edit ' Place Recordset in Edit Mode
  End If
  ' Assign text of txtTitle to field Title
  .Fields("Title") = txtTitle.Text
  .Fields("[Year Published]") = txtYearPublished.Text
  .Fields("ISBN") = txtISBN.Text
  .Fields("PubID") = txtPublisherID.Text
  ' Update data
  .Update
End With
' Return controls to Browse Mode
SetControls (False)
End Sub

```

Cũng vì không có Data Binding, nên khi User Delete một record, sau khi di chuyển qua record kế tiếp ta phải tự hiển thị chi tiết của record đó như sau:

```

Private Sub CmdDelete_Click()
  On Error GoTo DeleteErr
  With myRS
    ' Delete new record
    .Delete
    ' Move to next record
    .MoveNext
    If .EOF Then .MoveLast
    ' Display details of current record
    Displayrecord
  Exit Sub
End With
DeleteErr:
  MsgBox Err.Description
Exit Sub
End Sub

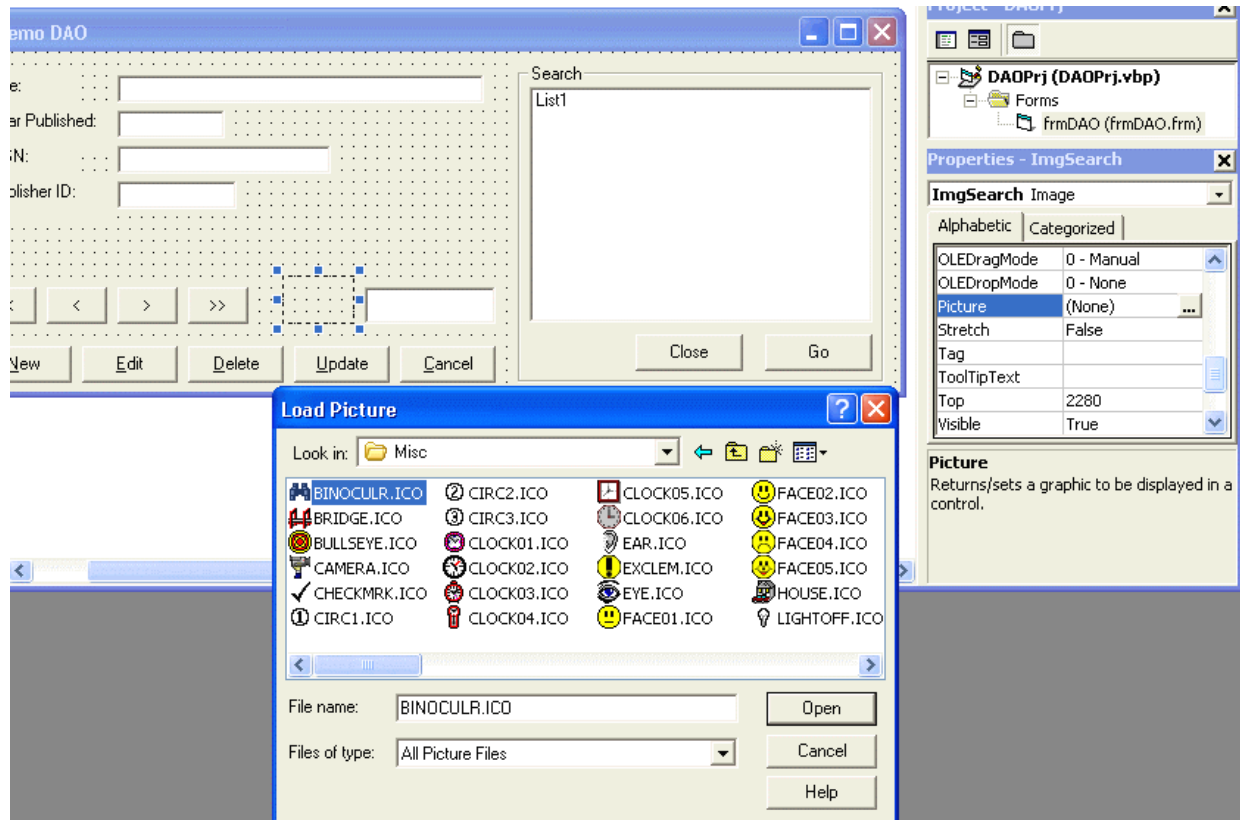
```

Tìm một record

Tiếp theo đây, ta muốn liệt kê các sách có tiêu đề chứa một chữ hay câu nào đó, thí dụ như chữ "**Guide**". Kế đó user có thể chọn một sách bằng cách select tiêu đề sách ấy và click nút **Go**. Chương trình sẽ locate (tìm ra) record của sách ấy và hiển thị chi tiết của nó.

Bây giờ bạn hãy cho vào Form một textbox tên **txtSearch** và một Image tên **ImgSearch**. Kế đó đặt một frame tên **fraSearch** vào Form. Để lên frame này một listbox tên **List1** để hiển thị tiêu đề các sách, và hai buttons tên **CmdClose** và **CmdGo**, với caption Close và Go. Sau khi select một sách trong List1, user sẽ click nút **Go** để hiển thị chi tiết sách ấy. Nếu đổi ý, user sẽ click nút **Close** để làm biến mất frame fraSearch.

Bình thường frame fraSearch chỉ hiện ra khi cần, nên lúc đầu hãy set **property Visible** của nó thành False. Ta sẽ cho ImgSearch hiển thị hình một ống dòm nên bạn hãy click vào bên phải **property Picture** trong Properties Window để chọn Icon BINOCULR.ICO từ folder E:\Program Files\Microsoft Visual Studio\Common\Graphics\Icons\Misc:



Cái Primary Key của table Titles là **ISBN**. Khi user select một sách ta muốn biết ISBN của sách ấy để locate (định chỗ) nó trong Recordset myRS. Do đó trong khi thêm tiêu đề của một sách vào List1, ta đồng thời thêm ISBN của sách ấy vào một Listbox thứ hai tên List2. Ta chỉ sẽ dùng List2 sau hậu trường, nên hãy set property Visible của nó thành False. Dưới đây là code để load tiêu đề sách và ISBN vào các Listboxes:

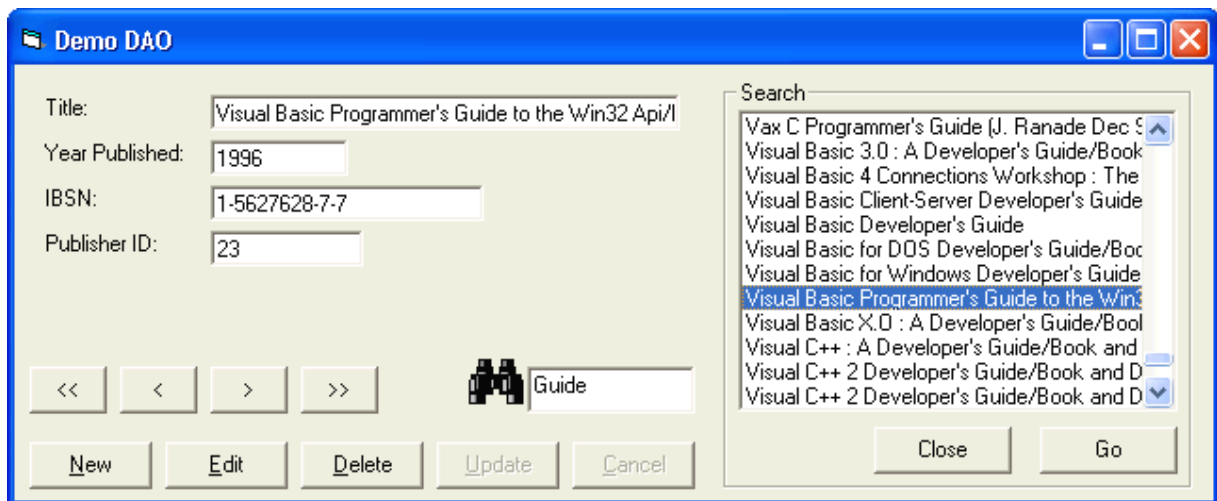
```
Private Sub ImgSearch_Click()
    ' Show Search Frame
    fraSearch.Visible = True
    Dim SrchRS As DAO.Recordset
    Dim SQLCommand As String
    ' Define SQL statement
    SQLCommand = "Select * from Titles where Title LIKE '" & "*" & txtSearch & "*" & "'"
```

```

ORDER BY Title"
' Fetch all records having Title containing the text pattern given by txtSearch
Set SrchRS = myDB.OpenRecordset(SQLCommand)
' If Recordset is not Empty then list the books' titles in List1
If SrchRS.RecordCount > 0 Then
    List1.Clear ' Clear List1
    ' We use List2 to contain the Primary Key ISBN corresponding to the books in List1
    List2.Clear ' Clear List2
    With SrchRS
        ' Iterate through the Recordset until EOF
        Do While Not SrchRS.EOF
            ' Display Title in List1
            List1.AddItem .Fields("Title")
            ' Store corresponding ISBN in List2
            List2.AddItem .Fields("ISBN")
            .MoveNext ' Move to next record in the Recordset
        Loop
    End With
End If
End Sub

```

Khi user Click ImgSearch với text pattern là chữ **Guide**, ta sẽ thấy hình dưới đây:



Trong SELECT statement bên trên ta dùng operator **LIKE** trên text pattern, chữ **Guide**, có **wildcard character (*)** ở hai bên. Wildcard character là chỗ có (hay không có) chữ gì cũng được. Trong trường hợp này có nghĩa là hề có chữ Guide trong tiêu đề sách là được, không cần biết nó nằm ở đâu. Ngoài ra sự chọn lựa này **Không có Case Sensitive**, tức là chữ **guide**, **Guide** hay **GUIDE** đều được cả.

Khi user clicks nút Go, ta sẽ dùng **method FindFirst** của Recordset myRS để định chỗ của record có trị số Primary Key là hàng text trong List2 tương ứng với tiêu đề được chọn trong List1 như sau:

```

Private Sub CmdGo_Click()
    Dim SelectedISBN As String
    Dim SelectedIndex As Integer
    Dim Criteria As String
    ' Index of line selected by user in List1
    SelectedIndex = List1.ListIndex
    ' Obtain corresponding ISBN in List2
    SelectedISBN = List2.List(SelectedIndex)
    ' Define Search criteria - use single quotes for selected text
    Criteria = "ISBN = '" & SelectedISBN & "'"
    ' Locate the record, it will become the current record
    myRS.FindFirst Criteria
    ' Display details of current record
    Displayrecord
    ' Make fraSearch disappeared
    fraSearch.Visible = False
End Sub

```

Lưu ý là trong string Criteria, vì ISBN thuộc loại text, chứ không phải là một con số, nên ta phải kẹp nó giữa hai dấu ngoặc đơn.

Bookmark

Khi di chuyển từ record này đến record khác trong Recordset, đôi khi ta muốn đánh dấu vị trí của một record để có dịp sẽ trở lại. Ta có thể thực hiện điều ấy bằng cách ghi nhớ **Bookmark** của Recordset.

Thí dụ khi user clicks nút Go, ta muốn nhớ vị trí của record lúc ấy để sau này quay trở lại khi User clicks nút **Go Back**. Bạn hãy thêm vào Form một button tên **CmdGoBack** với Caption **Go Back**. Ta sẽ thêm một variable tên **LastBookmark** loại data type **Variant**:

```
Dim LastBookMark As Variant
```

Lúc đầu button CmdGoBack invisible, và chỉ trở nên visible sau khi user clicks nút Go. Ta thêm các hàng codes sau vào Sub CmdGo_Click() như sau:

```

' Remember location of current record
LastBookMark = myRS.BookMark
CmdGoback.Visible = True

```

Dưới đây là code để quay trở lại vị trí current record trước đây trong Recordset:

```
Private Sub CmdGoback_Click()
```

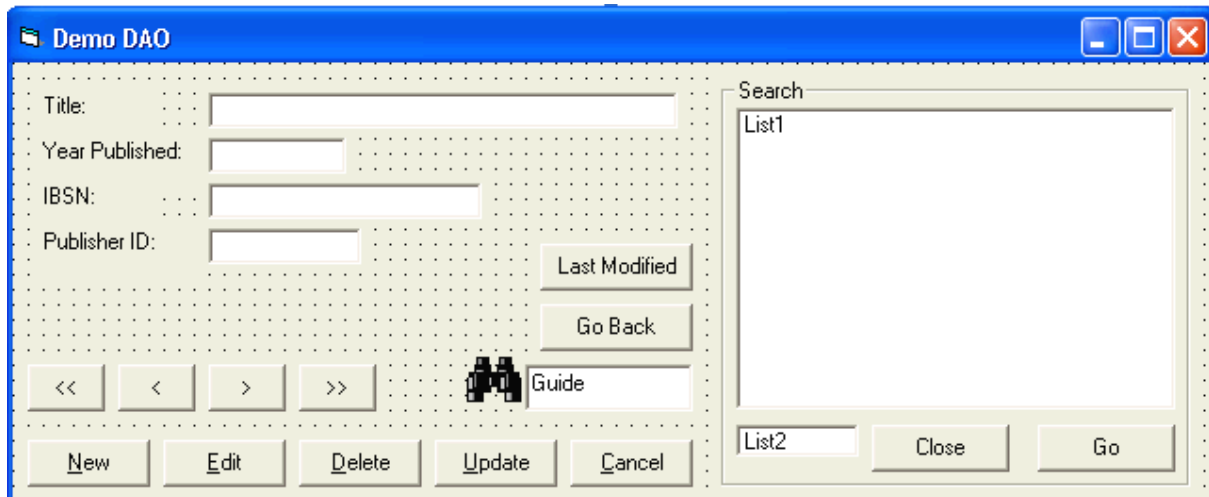
```
' Reposition record to last position
myRS.BookMark = LastBookMark
' Redisplay details or current record
Displayrecord
End Sub
```

LastModified

LastModified là vị trí của record vừa mới được sửa đổi hay thêm vào trong Recordset. Để thử điều này bạn hãy thêm một button invisible tên **CmdLastModified** với caption là **Last Modified**. Button này chỉ hiện ra sau khi user clicks Update. Bất cứ lúc nào bạn Click nút CmdLastModified, record mới vừa được sửa đổi hay thêm vào sẽ hiển thị:

```
Private Sub CmdLastModified_Click()
' Reposition record to last position
myRS.BookMark = myRS.LastModified
' Redisplay details or current record
Displayrecord
End Sub
```

Dưới đây là hình của Form lúc đang được thiết kế:



Ta sẽ học kỹ thuật ADO (ActiveX Data Object) trong bài tới.

Chương Mười Sáu - Lập trình với ADO (phần I)

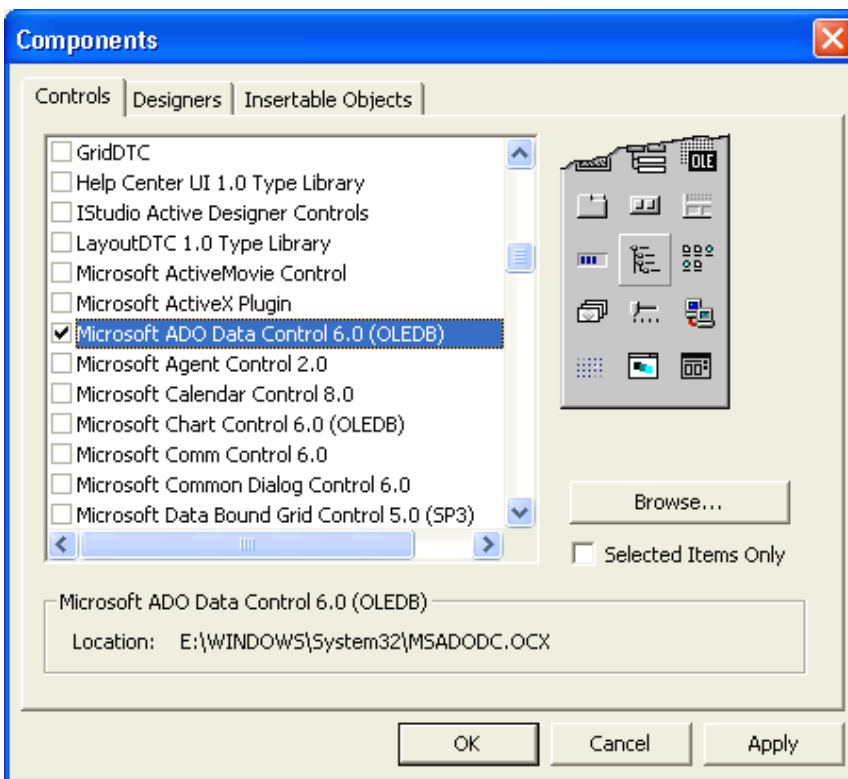
Control Data ADO

Visual Basic 6 cho ta sự lựa chọn về kỹ thuật khi lập trình với database, hoặc là dùng **DAO** như trong hai bài trước, hoặc là dùng **ADO (ActiveX Data Objects)**.

Sự khác biệt chính giữa ADO và DAO là ADO cho phép ta làm việc với mọi loại nguồn dữ kiện (data sources), không nhất thiết phải là Access database hay ODBC. Nguồn dữ kiện có thể là danh sách các địa chỉ Email, hay một file text string, trong đó mỗi hàng là một record gồm những fields ngăn cách bởi các dấu phẩy (**comma separated values**).

Nếu trong DAO ta dùng thẳng tên của MSAccess Database thì trong ADO cho ta **nối với (connect)** một database qua một Connection bằng cách chỉ định một **Connection String**. Trong Connection String có **Database Provider** (thí dụ như Jet, ISAM, Oracle, SQLServer..v.v.), tên Database, **UserName/Password** để logon một database .v.v.. Sau đó ta có thể **lấy về (extract)** những recordsets, và cập nhật hóa các records bằng cách dùng những **lệnh SQL** trên các tables hay dùng những **stored procedures** bên trong database.

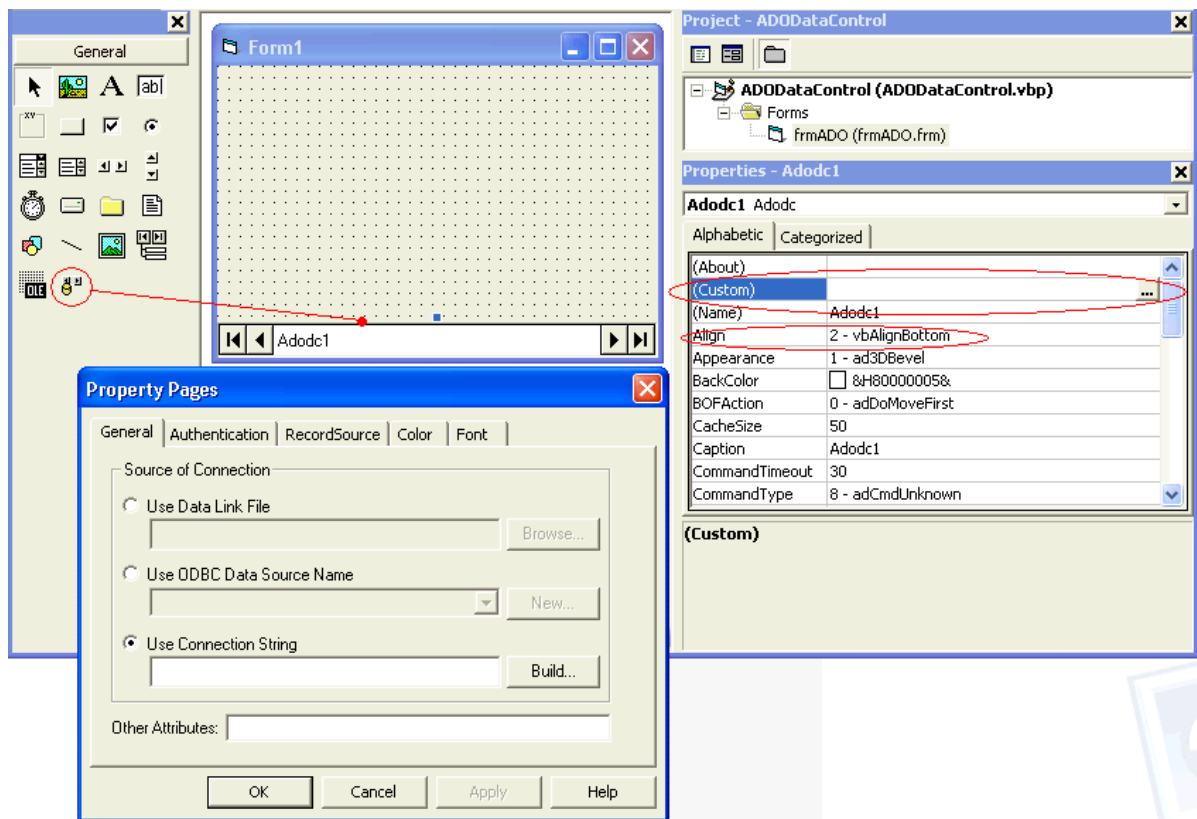
Bình thường, khi ta mới khởi động một project VB6 mới, Control **Data ADO** không có sẵn trong IDE. Muốn có nó, bạn hãy dùng Menu Command **Project | Components...**, rồi chọn **Microsoft ADO Data Control 6.0 (OLEDB)** từ giao diện Components như dưới đây:



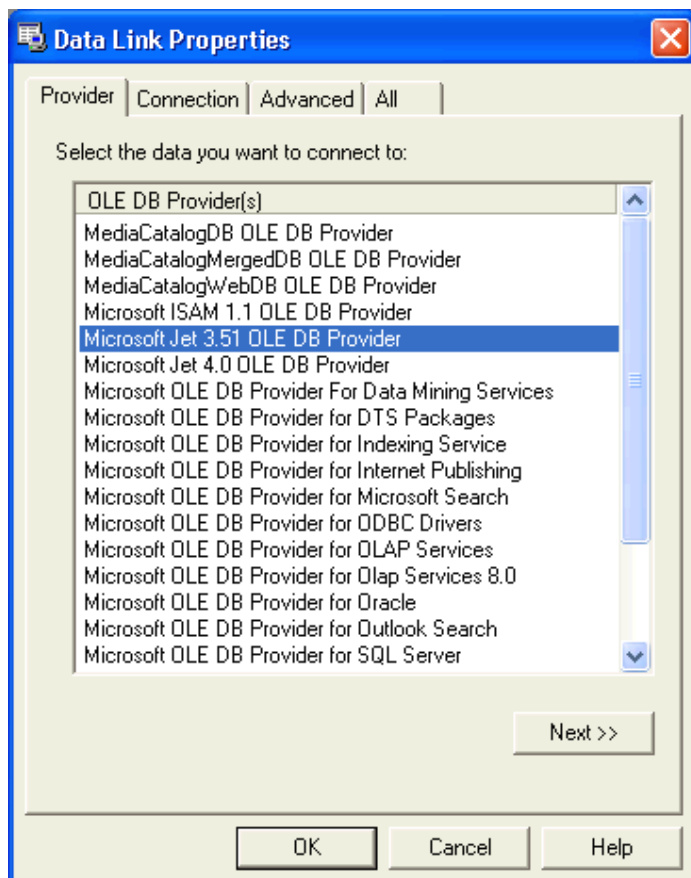
Bạn hãy bắt đầu một dự án VB6 mới, cho nó tên **ADODataControl** bằng cách click tên project trong Project Explorer bên phải rồi edit property Name trong Properties Window. Sửa tên của form chính thành **frmADO**, và đánh câu **ADO DataControl Demo** vào Caption của nó.

DoubleClick lên Icon của Control Data ADO trong Toolbox. Một Control Data ADO tên **Adodc1** sẽ hiện ra trên Form. Muốn cho nó nằm bên dưới Form, giống như một StatusBar, hãy set **property Align** của nó trong Properties Window thành **2 - vbAlignBottom**.

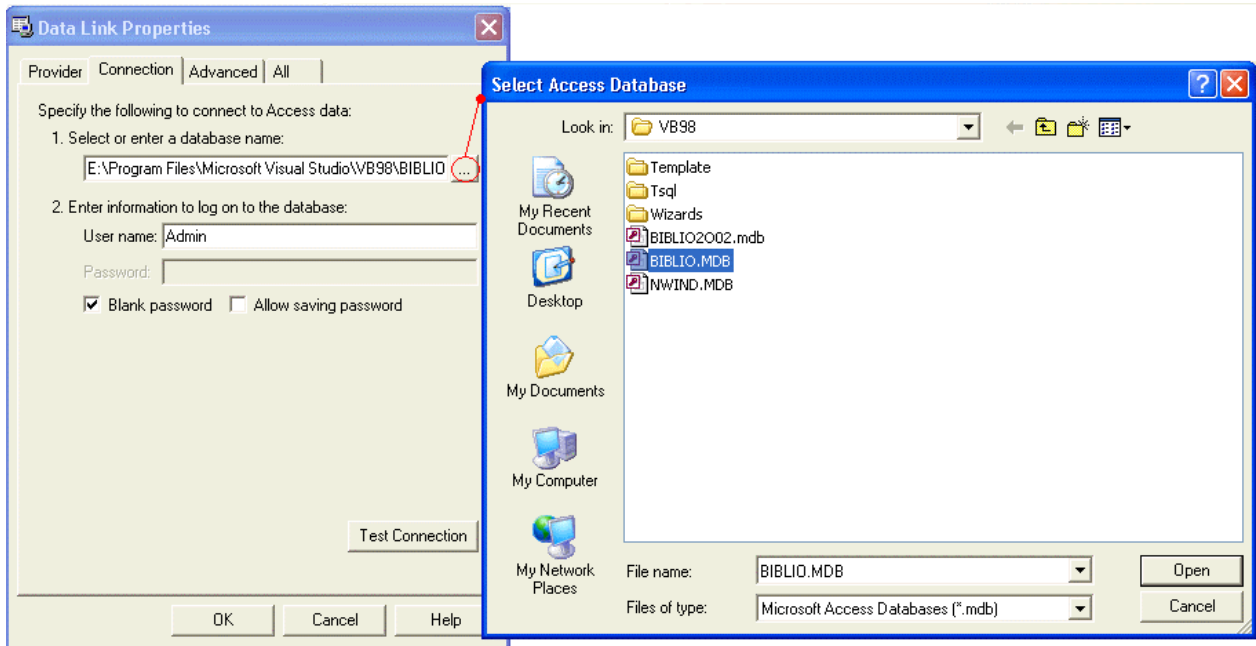
Click bên phải hàng **property (Custom)**, kế đó click lên nút browse có ba chấm để giao thoại **Property Pages** hiện ra. Trong giao thoại này, trên **Tab General** chọn Radio (Option) Button **Use Connection String** rồi click nút **Build....**



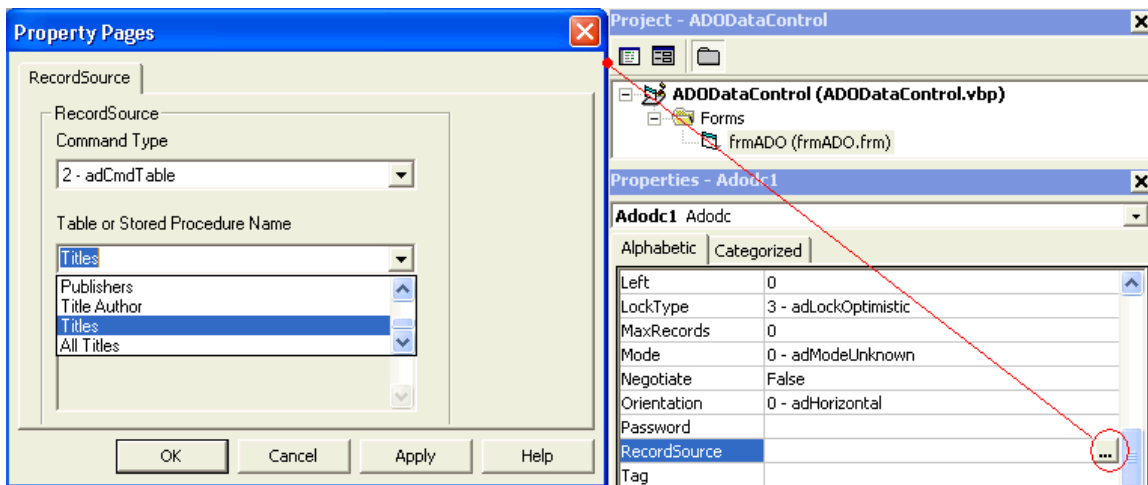
Trong giao thoại **Data Link Properties**, Tab **Provider**, chọn **Microsoft Jet 3.51 OLE DB Provider**, rồi click nút **Next >>** hay **Tab Connection**.



Ở chỗ **Select or enter a database name** ta chọn **E:\Program Files\Microsoft Visual Studio\VB98\BIBLIO.MDB**, trong computer của bạn có thể file ấy nằm trên disk **C** hay **D**. Sau đó, bạn có thể click nút **Test Connection** phía dưới để thử xem connection có được thiết lập tốt không.



Lập connection xong rồi, ta chỉ định muốn lấy gì về làm Recordset bằng cách click **property RecordSource** của **Adodc1**. Trong giao diện Property Pages của nó chọn **2-adCmdTable** làm **Command Type**, kế đó mở Combo box cho **Table or Stored Procedure Name** để chọn **table Titles**.

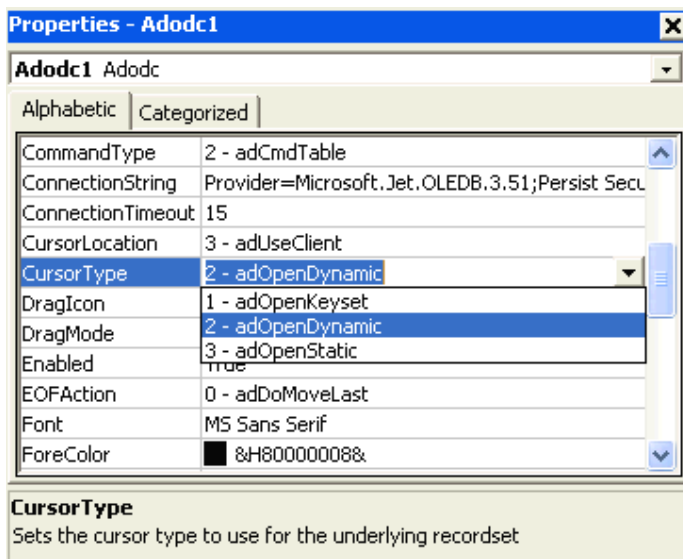


Tùy theo cách ta dùng Recordset trong ADO, nó có ba loại và được gọi là **Cursor Type**. Cursor chẳng qua là một tên khác của Recordset:

- **Static Cursor:** Static Cursor cho bạn một static copy (bản sao cứng ngắt) của các records. Trong lúc bạn dùng Static Cursor, nếu có ai khác sửa đổi hay thêm, bớt gì vào recordset bạn sẽ không thấy.
- **Keyset Cursor:** Keyset Cursor hơn Static Cursor ở chỗ trong lúc bạn dùng nó, nếu có ai sửa đổi record nào bạn sẽ biết. Nếu ai delete record nào, bạn sẽ không thấy nó nữa. Tuy nhiên bạn sẽ không biết nếu có ai thêm một record nào vào recordset.

- **Dynamic Cursor:** Như chữ **sống động (dynamic)** hàm ý, trong lúc bạn đang dùng một Dynamic Cursor, nếu có ai khác sửa đổi hay thêm, bớt gì vào recordset bạn sẽ thấy hết.

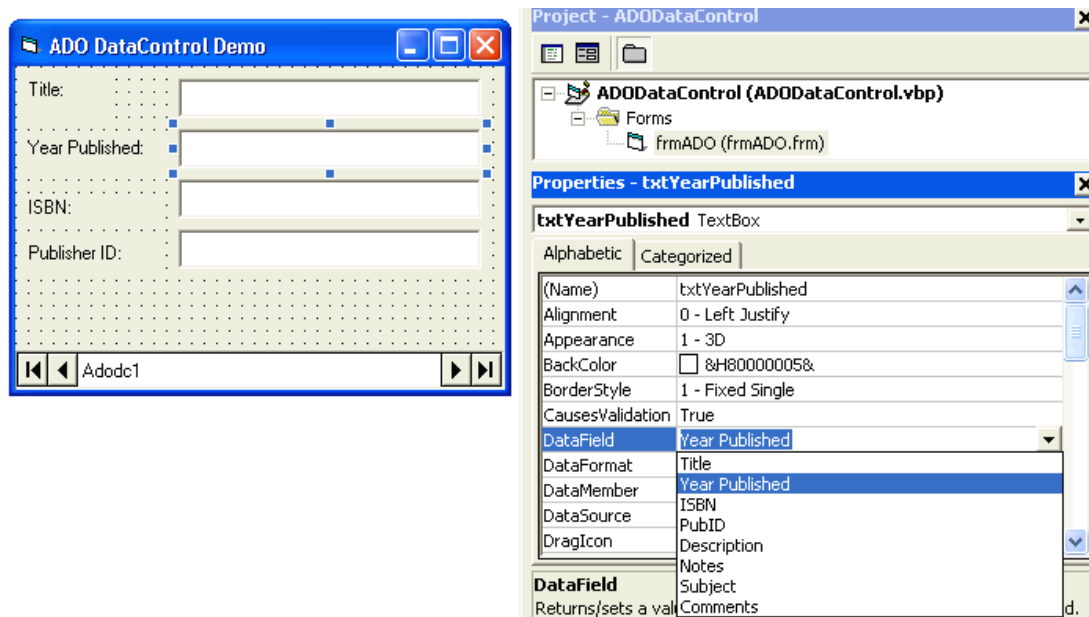
Bạn hãy chọn trị số **2-adOpenDynamic** cho property Cursor Type của Adodc1:



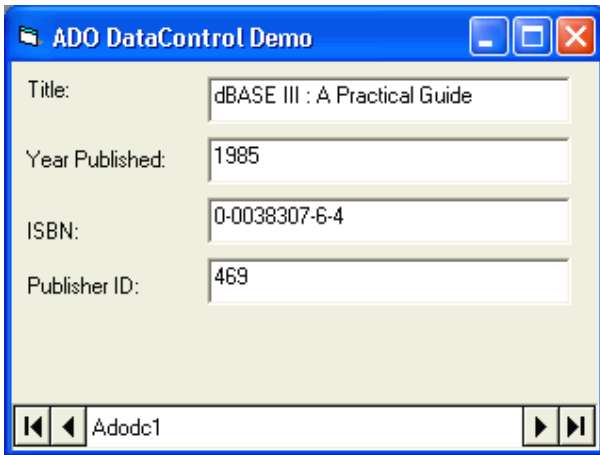
Bây giờ bạn hãy đặt lên Form 4 labels với captions: **Title, Year Published, ISBN** và **Publisher ID**. Kế đó cho thêm 4 textboxes tương ứng và đặt tên chúng là **txtTitle**, **txtYearPublished**, **txtISBN** và **txtPublisherID**.

Để thực hiện Data Binding, bạn hãy chọn textbox **txtYearPublished** (năm xuất bản), rồi set **property Datasource** của nó trong Properties Window thành **Adodc1**. Khi click lên **property DataField** của txtYearPublished và mở ComboBox ra bạn sẽ thấy liệt kê tên các Fields trong table Titles. Đó là vì Adodc1 được coi như trung gian lấy table Titles từ database. Ở đây ta sẽ chọn cột Year Published.

Lập lại công tác này cho 3 textboxes kia, và chọn các cột Title (Tiêu đề), ISBN (số lý lịch trong thư viện quốc tế), và PubID (số lý lịch nhà xuất bản) làm DataField cho chúng.



Đến đây, mặc dầu chưa viết một hàng code nào, bạn có thể chạy chương trình và nó sẽ hiển thị như dưới đây:



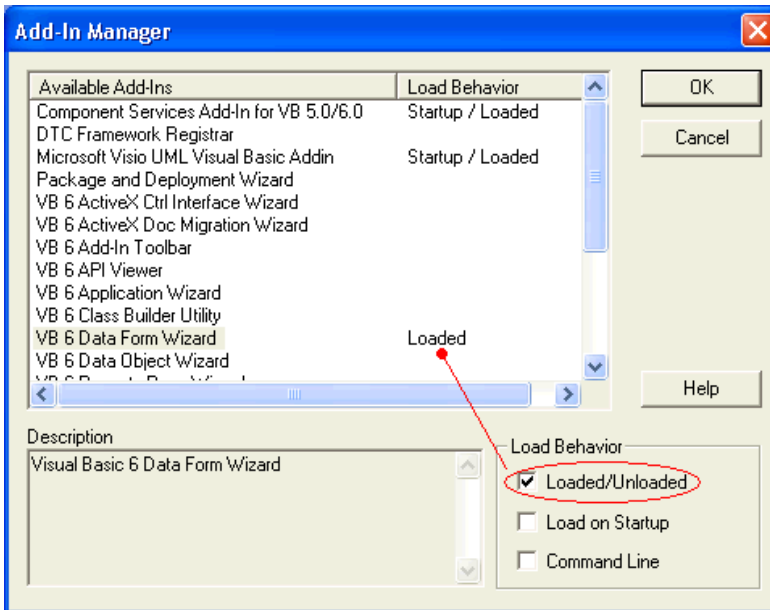
Bạn có thể tải về chương trình dùng Control Data ADO này từ đây ADODatacontrol.zip.

Data Form Wizard

Để giúp lập trình viên thiết kế các data forms nhanh hơn, VB6 cho ta **Data Form Wizard** để generate (phát sinh) ra một form có hỗ trợ Edit, Add và Delete records.

Bây giờ bạn hãy khởi động một standard project VB6 mới, tên **ADOCClass** và copy MS Access file BIBLIO.MDB, tức là database, vào trong cùng folder của dự án mới này.

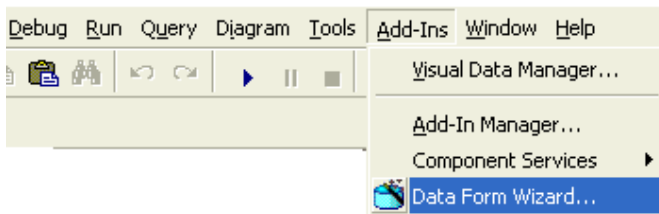
Muốn dùng Data Form Wizard, trước hết ta phải thêm nó vào môi trường phát triển (IDE) của VB6. Bạn hãy dùng IDE Menu Command **Add-Ins | Add-In Manager....** Chọn **VB6 Data Form Wizard** trong giao thoại, rồi click Checkbox **Loaded/Unloaded** để chữ Loaded hiện bên phải hàng "VB6 Data Form Wizard" như trong hình dưới đây:



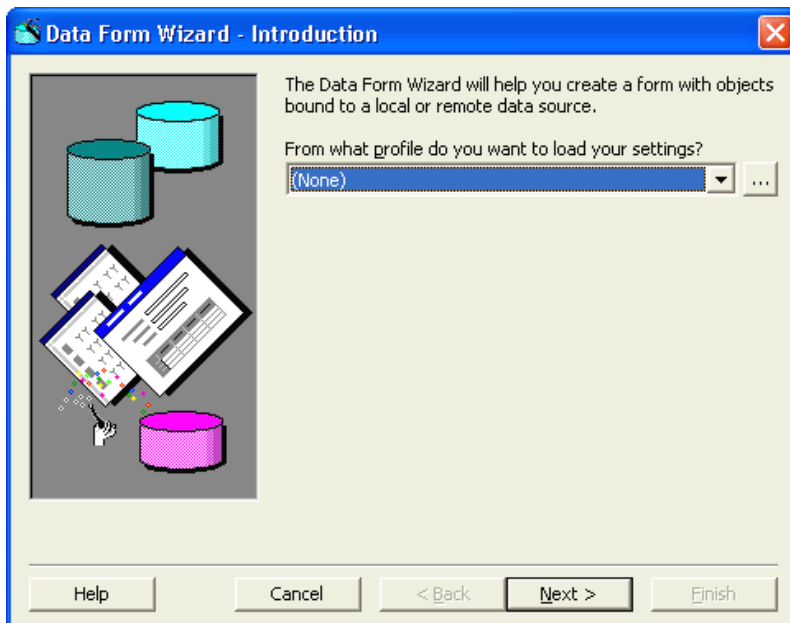
Nếu bạn muốn mỗi lần khởi động VB6 IDE là có sẵn Data Form Wizard trong menu Add-Ins thì ngoài option Loaded, bạn click thêm check box **Load on Startup**.

Một **Add-In** là một menu Item mới mà ta có thể thêm vào một chương trình ứng dụng có sẵn.

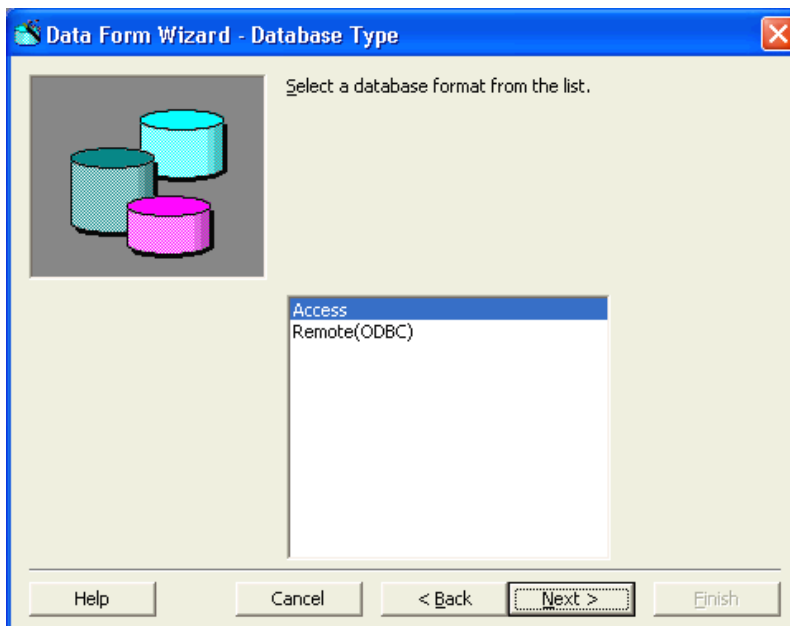
Thường thường, người ta dùng Add-Ins để thêm chức năng cho một chương trình, làm như là chương trình đã có sẵn chức năng ấy từ đầu. Bạn hãy khởi động Data Form Wizard từ IDE Menu Command mới **Add-Ins | Data Form Wizard...**



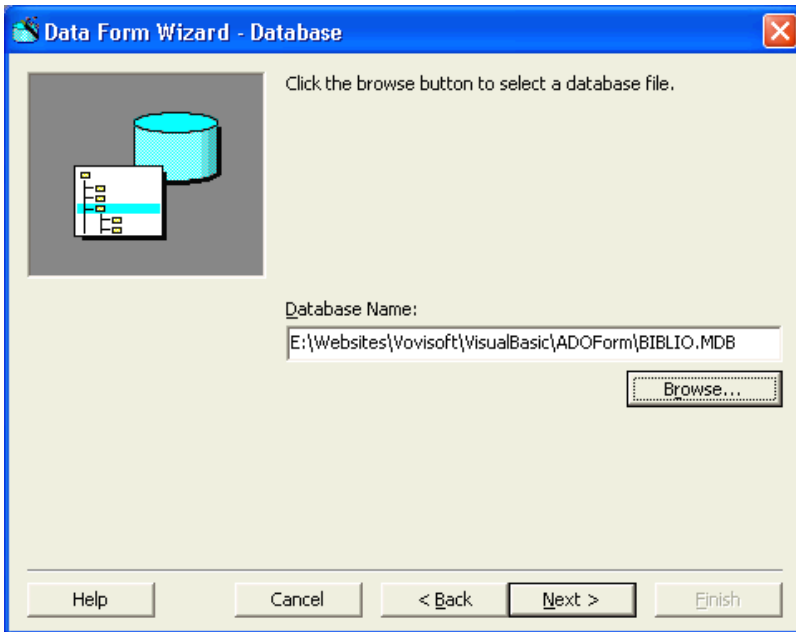
Khi trang **Data Form Wizard - Introduction** hiện ra, click **Next**



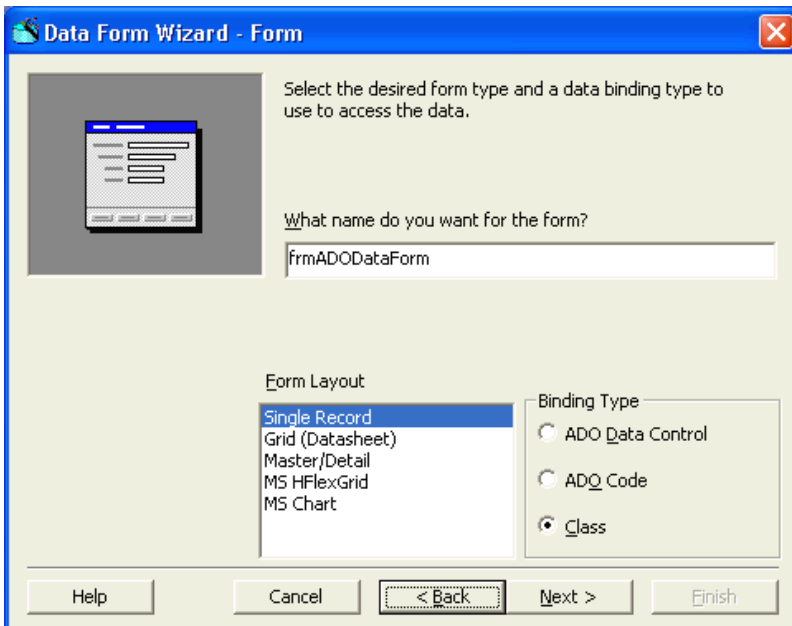
Trong trang kế đó chọn **Access** làm **Database Type**.



Trong trang Database, click **Browse** để chọn một MS Access database file. Ở đây ta chọn file **BIBLIO.MDB** từ chính folder của chương trình này. Đoạn click **Next**.



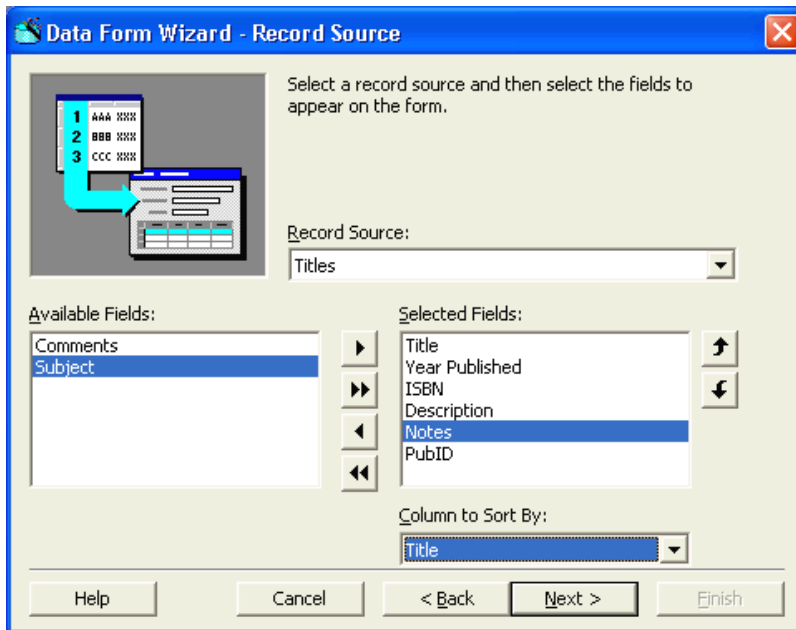
Trong trang Form, ta chọn **Single Record** cho **Form Layout** và **Class** cho **Binding Type**. Đoạn click **Next**. Nếu ta chọn **ADO Data Control** thì kết quả sẽ giống giống như khi ta dùng Control Data DAO như trong một bài trước.



Trong trang Record Source ta chọn **table Titles**. Listbox của **Available Fields** sẽ hiển thị các fields của table Titles. Sau khi chọn một field bằng cách click lên tên field ấy trong Listbox, nếu bạn click hình tam giác chỉ qua phải thì tên field ấy sẽ được dời qua nằm dưới cùng trong Listbox **Selected Fields** bên phải.

Nếu bạn click hình hai tam giác chỉ qua bên phải thì tất cả mọi fields còn lại bên trái sẽ được dời qua bên phải. Bạn cũng có thể sắp đặt vị trí của các selected fields bằng cách click lên tên field ấy rồi click hình mũi tên chỉ lên hay xuống để di chuyển field ấy lên hay xuống trong danh sách các fields.

Ngoài ra, bạn hãy chọn Title làm **Column to Sort By** trong cái Combobox của nó để các records trong Recordset được sắp xếp theo thứ tự ABC (alphabetical order) của field Tiêu đề (Title).



Trong trang Control Selection, ta sẽ để ý nguyên để có đủ mọi buttons. Bạn hãy click **Next**.



Khi Data Form Wizard chấm dứt, nó sẽ generate form **frmADODataForm**. Bạn hãy remove Form1 và dùng Menu Command **Project | ADODataControl Properties...** để đổi **Startup Object** thành frmADODataForm. Thế là tạm xong chương trình để Edit các records của table Titles.

Chúng ta hãy quan sát cái Form và phần code được Data Form Wizard generated. Trong frmADODataForm, các textboxes làm thành một array tên txtFields. Mọi textbox đều có property **DataField** định sẵn tên field của table Titles. Thí dụ như **txtFields(2)** có DataField là **ISBN**. Form chính không dùng Control Data ADO nhưng dùng một Object của **class clsTitles**.

Phần Initialisation của class clsTitles là Open một Connection và lấy về một Dataset có tên **DataMember** là **Primary** như sau:

```
Private Sub Class_Initialize()
    Dim db As Connection
    Set db = New Connection
    db.CursorLocation = adUseClient
    ' Open connection
    db.Open "PROVIDER=Microsoft.Jet.OLEDB.3.51;Data
Source=E:\Websites\Vovisoft\VisualBasic\ADODForm\BIBLIO.MDB;"
    ' Instantiate ADO recordset
    Set adoPrimaryRS = New Recordset
    ' Retrieve data for Recordset
    adoPrimaryRS.Open "select Title,[Year Published],ISBN,Description,Notes,PubID from
Titles Order by Title", _
                        db, adOpenStatic, adLockOptimistic
    ' Define the only data member, named Primary
    DataMembers.Add "Primary"
End Sub
```

Về vị trí của database, nếu bạn không muốn nó chết cứng ở một folder nào thì dùng **App.Path** để xác định mối liên hệ giữa vị trí của database và folder của chính chương trình đang chạy, thí dụ như:

```
db.Open "PROVIDER=Microsoft.Jet.OLEDB.3.51;Data Source=" & App.Path &
"\BIBLIO.MDB;"
```

Trong **Sub Form_Load**, ta có thể dùng **For Each** để đi qua hết các textboxes trong array txtFields. Vì property Datasource của textbox là một Object nên ta dùng keyword **Set** để point

nó đến Object **PrimaryCLS**. Đồng thời ta cũng phải chỉ định tên của DataMember của mỗi textbox là Primary:

```
Private Sub Form_Load()  
    ' Instantiate an Object of class clsTitles  
    Set PrimaryCLS = New clsTitles  
    Dim oText As TextBox  
    ' Iterate through each textbox in the array txtFields  
    ' Bind the text boxes to the data source, i.e. PrimaryCLS  
    For Each oText In Me.txtFields  
        oText.DataMember = "Primary"  
        ' Use Set because property Datasource is an Object  
        Set oText.DataSource = PrimaryCLS  
    Next  
End Sub
```

Khi sự di chuyển từ record này đến record khác chấm dứt, chính Recordset có raise **Event MoveComplete**. Event ấy được handled (giải quyết) trong class clsTitles bằng cách lại raise **Event MoveComplete** để nó được handled trong Form.

Muốn handle Event trong clsTitles ta phải declare recordset adoPrimaryRS với WithEvents:

```
Dim WithEvents adoPrimaryRS As Recordset  
Private Sub adoPrimaryRS_MoveComplete(ByVal adReason As ADODB.EventReasonEnum,  
    —  
    ByVal pError As ADODB.Error, adStatus As ADODB.EventStatusEnum, ByVal pRecordset  
    As ADODB.Recordset)  
    ' Raise event to be handled by main form  
    RaiseEvent MoveComplete  
End Sub
```

Và trong Form ta cũng phải declare (object clsTitles) PrimaryCLS với WithEvents:

```
Private WithEvents PrimaryCLS As clsTitles
```

Trong Form, Event MoveComplete sẽ làm hiển thị vị trí tuyệt đối (Absolute Position) của record bằng code dưới đây:

```
Private Sub PrimaryCLS_MoveComplete()  
    ' This will display the current record position for this recordset  
    lblStatus.Caption = "Record: " & CStr(PrimaryCLS.AbsolutePosition)  
End Sub
```


Khi user clicks **Refresh**, các textboxes sẽ được hiển thị lại với chi tiết mới nhất của record từ trong recordset, nhờ khi có ai khác đã sửa đổi record. **Method Requery** của clsTitles lại gọi method Requery của Recordset như sau:

```
Private Sub cmdRefresh_Click()  
    'This is only needed for multi user applications  
    On Error GoTo RefreshErr  
    ' fetch the latest copy of Recordset  
    PrimaryCLS.Requery  
    Exit Sub  
RefreshErr:  
    MsgBox Err.Description  
End Sub  
  
'In Class clsTitles  
Public Sub Requery()  
    ' Fetch latest copy of record  
    adoPrimaryRS.Requery  
    DataMemberChanged "Primary"  
End Sub
```