# MỤC LỤC

# Chương Một - 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 y 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 hễ 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 ngươi giựt chuông và người đàn. Anh Tư Thông có người bạn làm chủ tiệm hình, nên anh nhận trách nhiệm mướn 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 nầy 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 **thầu** của bạn. Chắc bạn đã lưu ý rằng cái thí dụ tổ chức đám cưới nầy cho thấy nói chung muốn triển khai dự án lớn nào ta cần phải nhờ những người thầu giúp đở. 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 **thầu** 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 thầu không?

Người thầu 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 Tầm 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
```

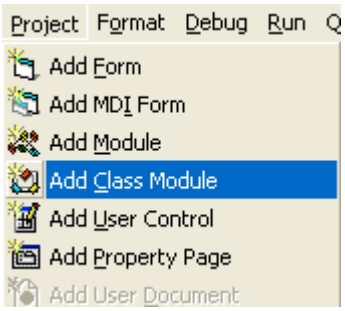
Trong trường hợp nầy thật ra frmMyForm tuy là một Class nhưng được dùng y như một Object. Chớ nếu muốn, ta có thể tạo ra hai, ba Objects của Class frmMyForm cùng một lúc như trong thí dụ sau:

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

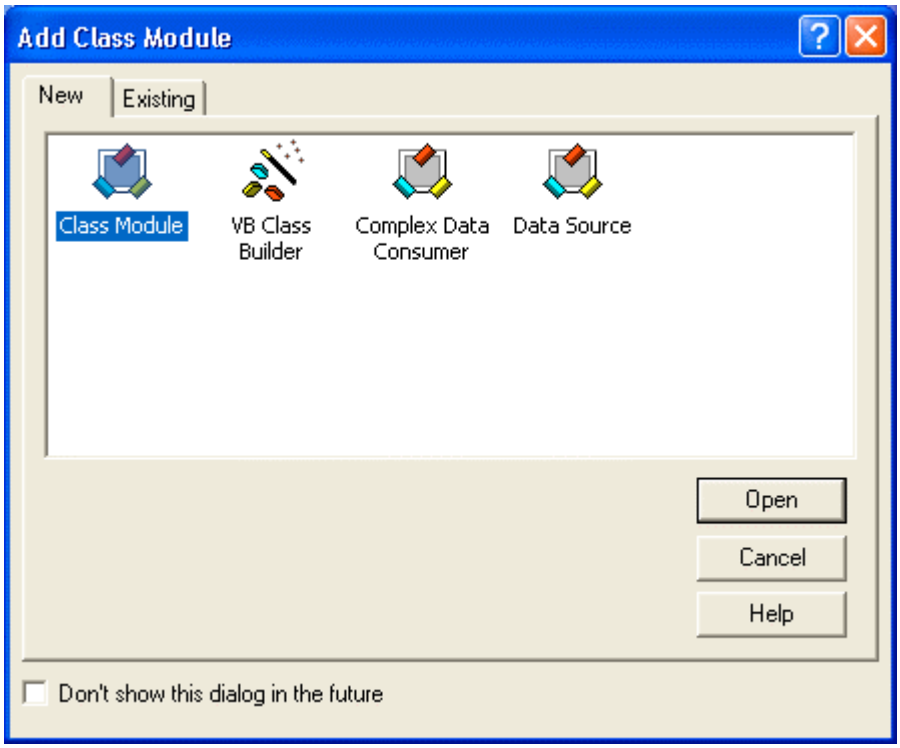
Trong thí dụ trên ta declare firstForm và secondForm là những Objects của Class frmMyForm. Sau đó ta làm nên (**instantiate**) các Objects firstForm và secondForm bằng statements **Set... = New...** firstForm và secondForm còn được gọi là các **instances** của Class frmMyForm. Class giống như cái khuôn, 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. Đều nầy 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 xãy 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. Classtrong VB6 không có hổ 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.

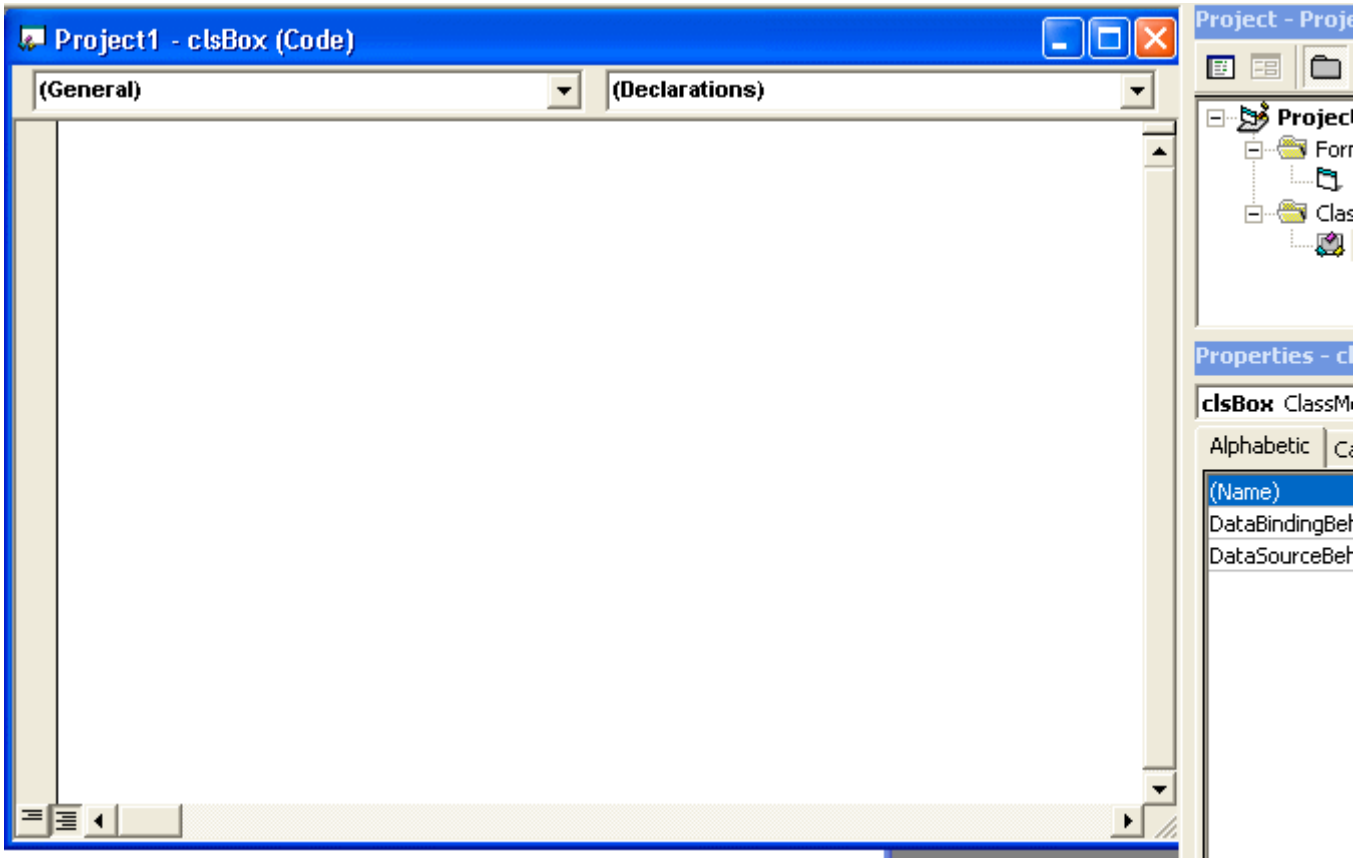
Bây giờ chúng ta hãy bắt đầu viết một Class. Bạn hãy mở một Project mới loại Standard EXE Visual Basic. Sau đó dùng Menu Command chọn **Add Class Module**:




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,

```
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 nầy 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:

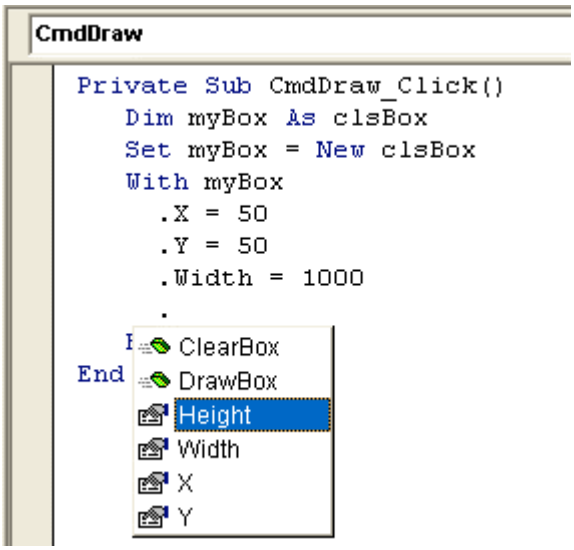
```
' Declare an object of Class StdFont of VB6
Dim myFont As StdFont
Set myFont = New StdFont
myFont.Name = "Arial"
myFont.Bold = True
Dim myBox As clsBox
Set myBox = New clsBox
Set myBox.Font = myFont  ' Call the Property Set method
```

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:

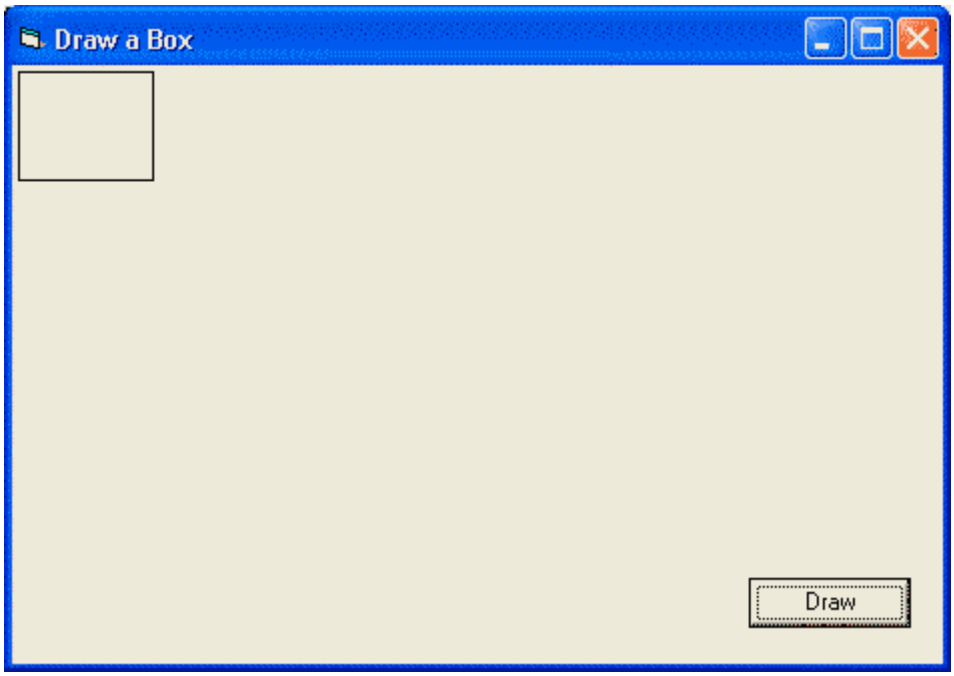
```
Public Sub DrawBox(Canvas As Object, Optional fColor As Long)
   If IsMissing(fColor) Then
      Canvas.Line (mX, mY)-(mX + mWidth, mY + mHeight), , B
   Else
      Canvas.Line (mX, mY)-(mX + mWidth, mY + mHeight), fColor, B
   End If
End Sub
```

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:

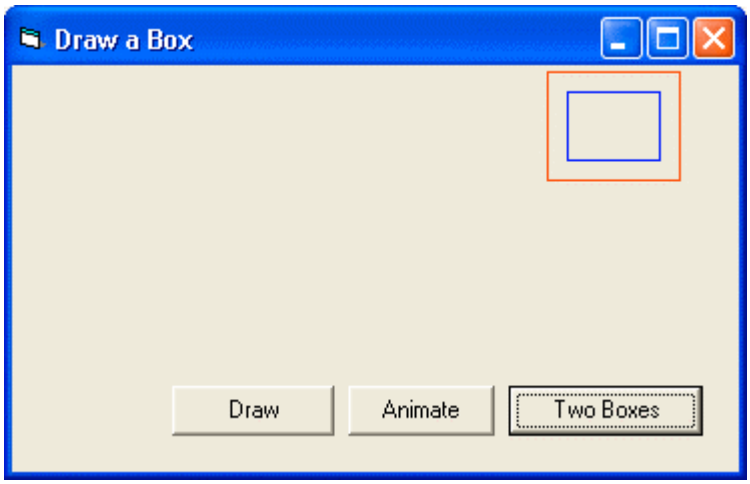



Trong chương trình nầy, mỗi khi ta click nút **Draw** thì một Box được instantiate, cho tọa độ X,Y và kích thước Width, Height, rồi được vẽ ra ngay trên form. Chữ **Me** trong code nói đến chính cái form **frmClass**.



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

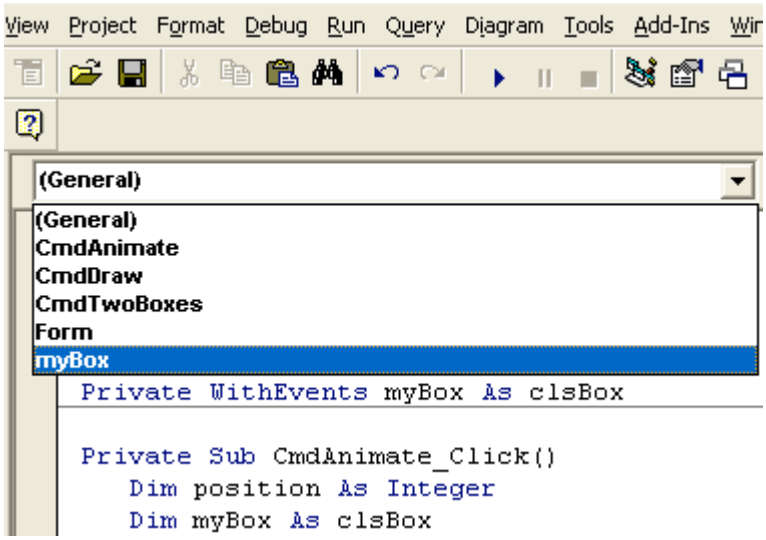
Khi user clicks nút **Two Boxes** ta sẽ vẽ hai boxes, hộp trong màu xanh, hộp ngoài màu đỏ, và cho chúng chạy từ trái sang phải. Ở đây ta biểu diễn cho thấy mình muốn instantiate bao nhiêu boxes từ clsBox cũng được, và dĩ nhiên mỗi box có một bộ properties với giá trị riêng của nó.



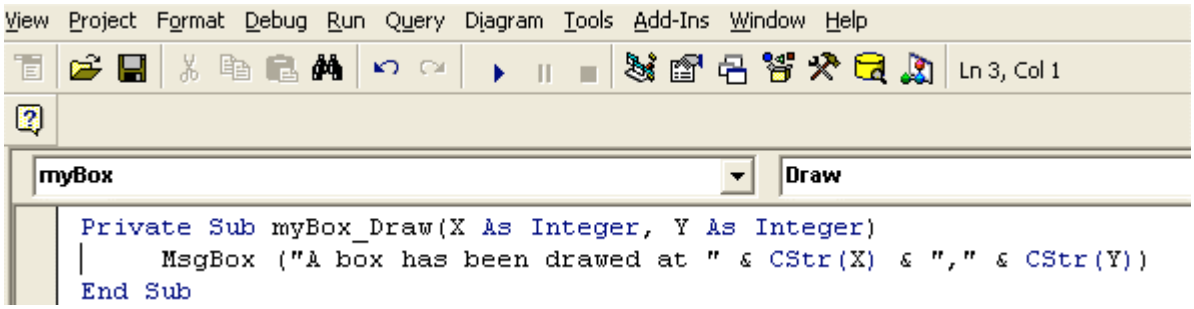

Ta có thể lập trình để cho Object báo cáo program chủ của nó khi có một biến cố (Event) xãy ra bên trong Class.

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.

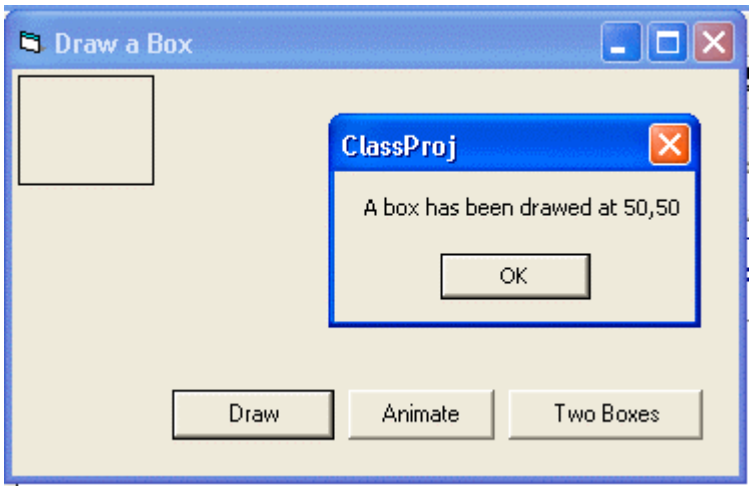
```
Public Event Draw(X As Integer, Y As Integer)
Public Sub DrawBox(Canvas As Object, Optional fColor As Long)
   If IsMissing(fColor) Then
      Canvas.Line (mX, mY)-(mX + mWidth, mY + mHeight), , B
   Else
      Canvas.Line (mX, mY)-(mX + mWidth, mY + mHeight), fColor, B
   End If
   RaiseEvent Draw(mX, 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 sứ 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 nầy 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 Hai – 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 nầy 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 nầy 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 lầm gì cả. Để đạt mục tiêu ấy, chẳng những cần có người kiểm phẩm mà chính các nhân viên lấp ráp thận trọng để công việc chính của người kiểm phẩm là xác nhận kết quả tốt chớ không phải tìm lỗi lầm.

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. Khổ nhất là lúc gần giao hàng mới khám phá ra có nhiều điểm trong chương trình khách muốn một đàng 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 nầy 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 xẩy ra trước, việc gì xẩy 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 nầy 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 nầy 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 sữa 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 sữa 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 sữa đổi. Điều nầy 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 nầy.

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 mách 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 hễ có điểm nào khác thường, bí hiểm thì bạn cần thông báo và giải thích tại sao bạn làm cách ấy. Có thể sau nầy ta khám phá ra đoạn code có bugs; lúc đọc lại có thể ta sẽ thấy dầu 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 chớ không phải dựa vào những tiểu 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 đậy 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 nầy 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 xẩy 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 tắc. 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 sở, nếu tiện thì trong vài tuần đầu, thay gì chạy EXE của chương trình, bạn chạy source code trong VB6 IDE. Nếu có bug nào xẩy ra, bạn có thể cho program ngừng trong source code để debug.

Khi bạn dùng statement:

**On Error Resume Next**

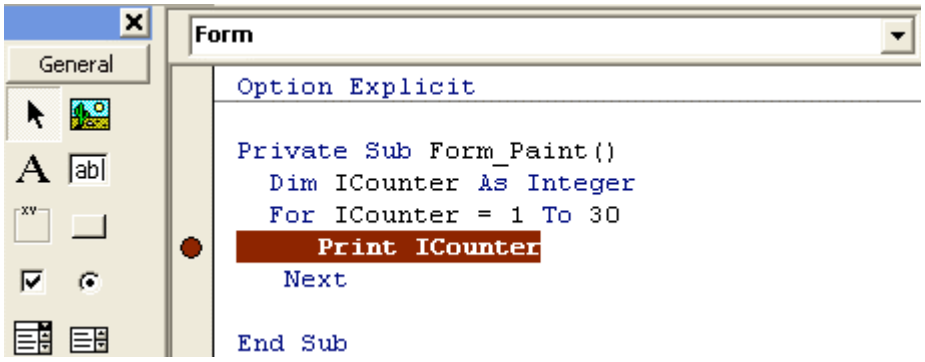
thì từ chỗ đó trở đi, nếu chương trình gặp Error, nó sẽ bỏ qua (ignore) hoàn toàn. Điểm nầy tiện ở chỗ giúp chương trình EXE của ta tránh bị té cái ạch rồi biến mất, rất là "quê" với khách hàng. Nhưng nó cũng bất lợi

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 nầy 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 break point.

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

```
Private Sub Form_Paint()
  Dim ICounter As Integer
  For ICounter = 1 To 30
     Print ICounter
   Next
End Sub
```
(ICounter = 3)

Có một số chuyện khác bạn có thể làm trong lúc nầy. 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
```

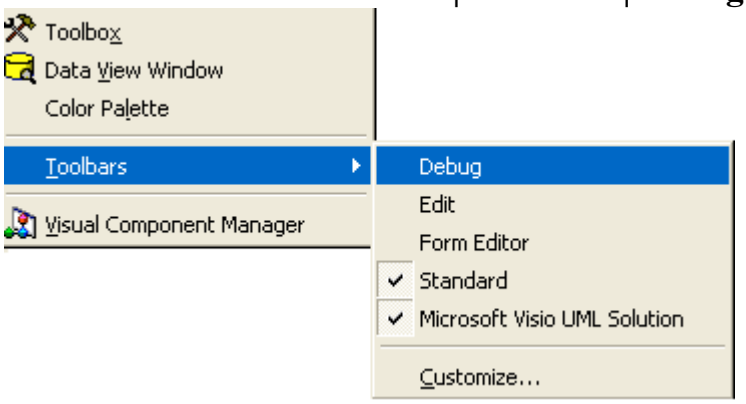
Nhớ là để cho program chạy lại bạn bấm **F5**, tương đương với Menu command **Run | Continue**.

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 mánh lới 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:

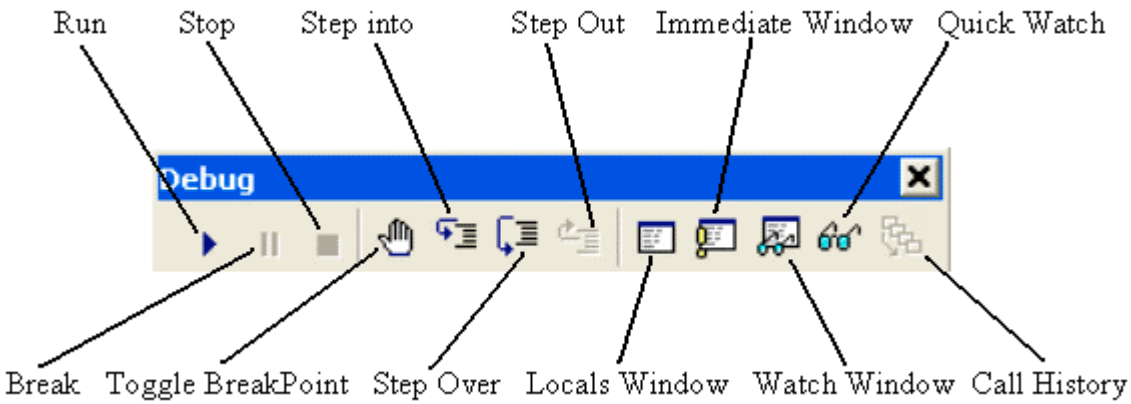
```
Private Sub Form_Paint()
  Dim ICounter As Integer
  For ICounter = 1 To 100
     If ICounter = 30 Then
        Beep
     End If
     Print ICounter
   Next

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**



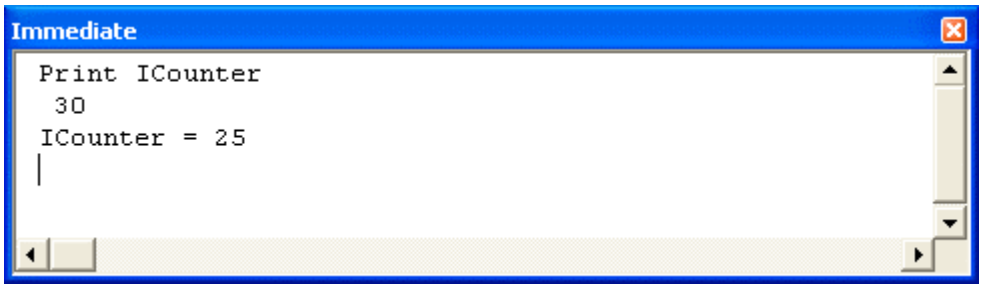

VB6 IDE sẽ hiển thị Debug Toolbar như sau:



## Dùng Immediate Window

**Immediate Window** cho phép ta execute những VB statement strong 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.

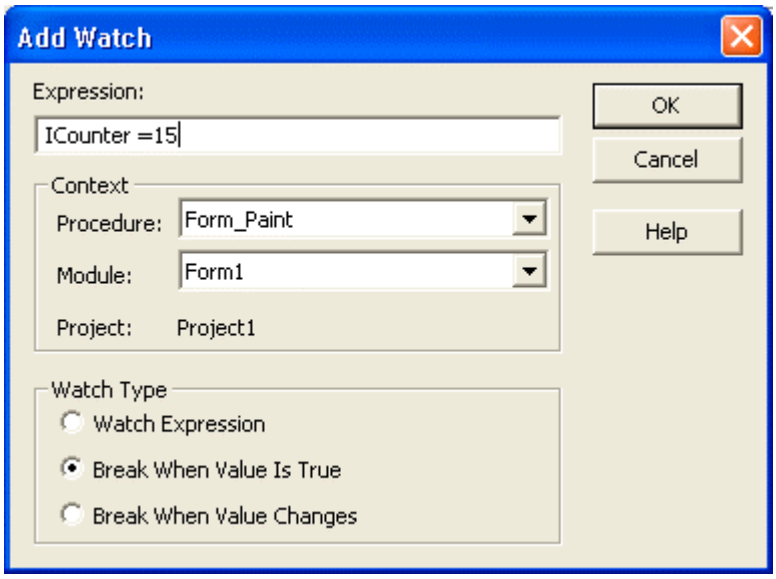
Có một cách khác là thay vì cho hiển thị trong Immediate Window bạn cho viết xuống (**Log**) vào trong một text file. Dưới đây là một Sub điển hình bạn có thể dùng để Log một Event message:

```
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
```

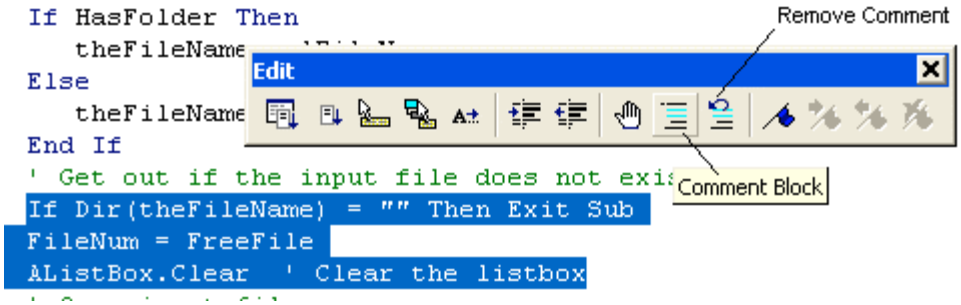
## Dùng Watch Window

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.

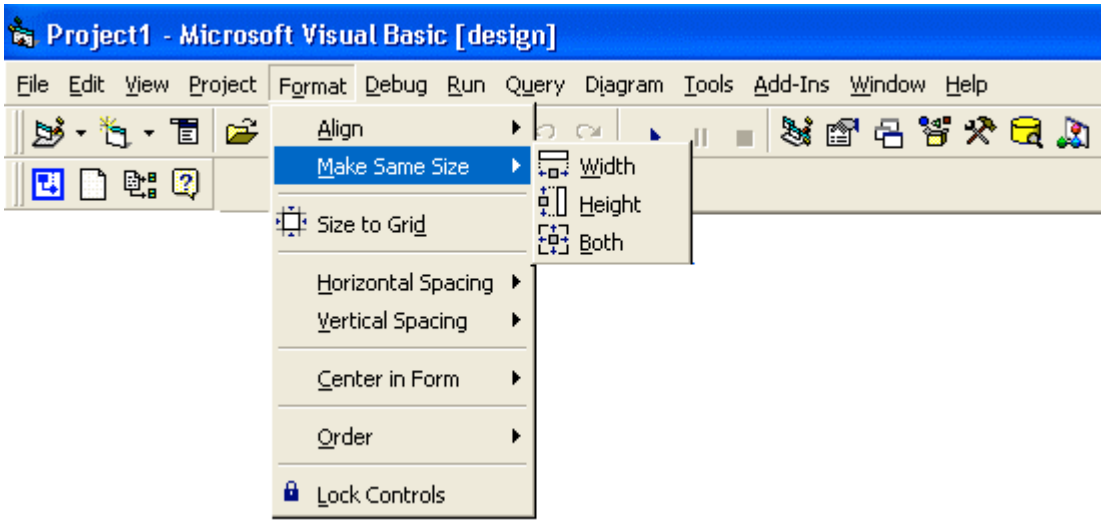



Khi dùng Elimination Method bạn phải cân nhắc Logic của code bạn trong khi quyết định Comment Out những hàng nào, nếu không, đó là một phương pháp khá nguy hiểm.

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.

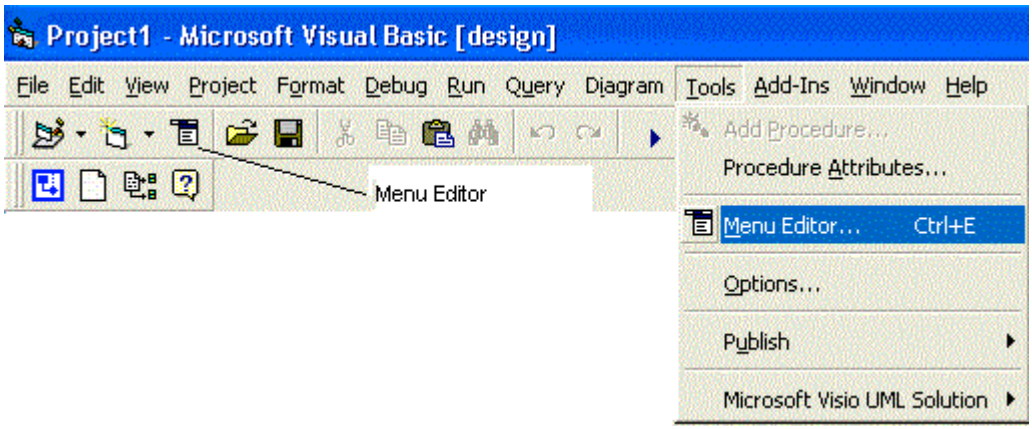
# Chương Ba - Dùng Menu

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**):

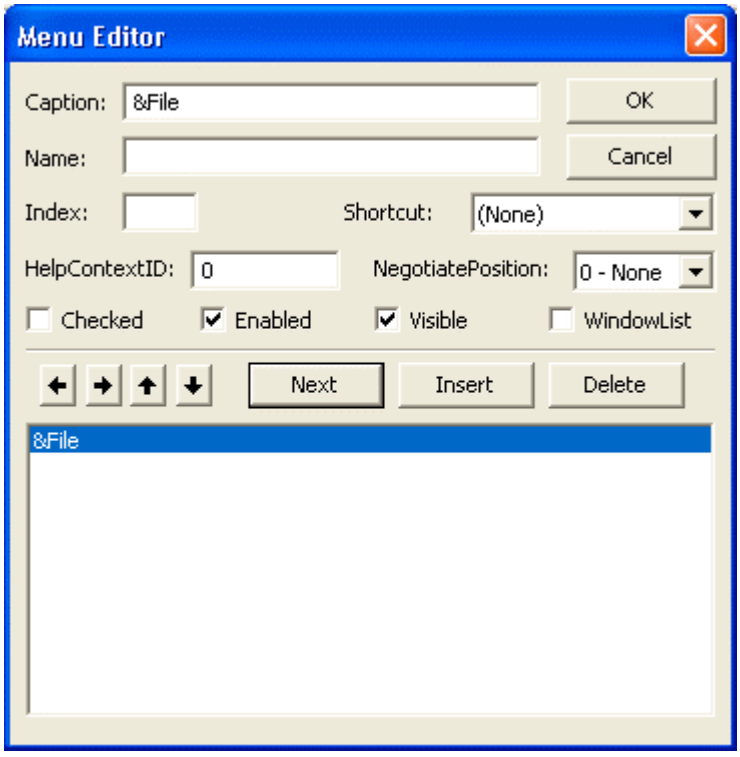



## Main Menu

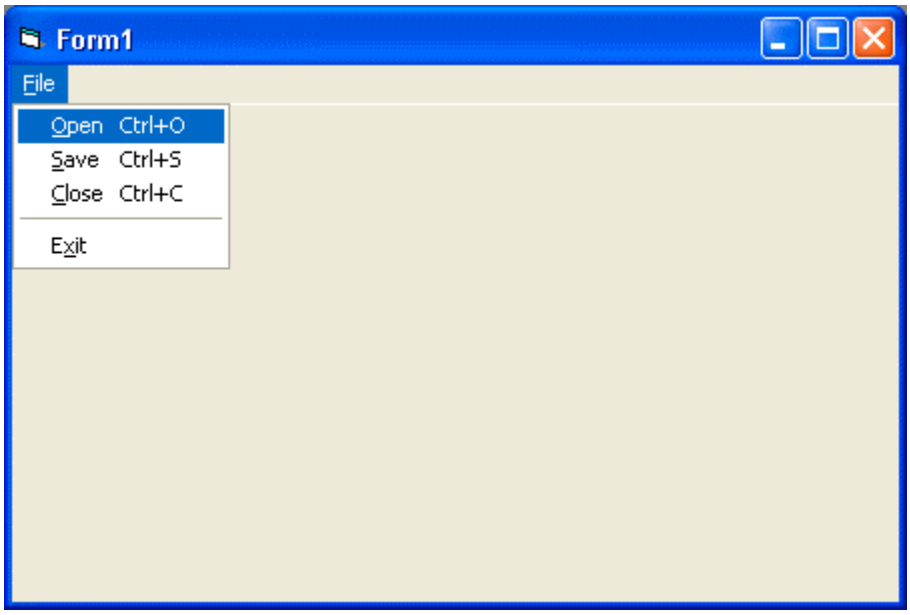
Ta dùng **Menu Editor** để tạo hoặc sữa một Menu cho program. Menu thuộc về một Form. Do đó, trước hết ta select một Form để làm việc với Designer của nó (chớ không phải code của Form). Kế đó ta dùng Menu Command **Tools | Menu Editor** hay click lên icon của Menu Editor trên Toolbar để làm cho Menu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



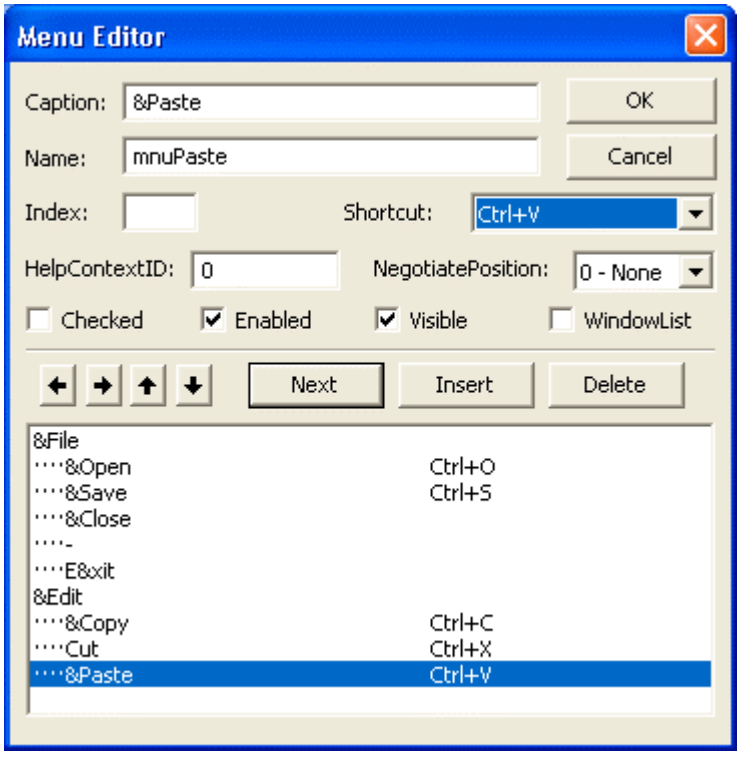

Đầu tiên có một vệt màu xanh nằm trong khung trắng của Menu Editor, nơi sẽ hiển thị Caption của Menu Command đầu tiên của Form. Khi ta đánh chữ **&File** vào Textbox **Caption**, nó cũng hiện ra trên vệt xanh nói trên. Kế đó, bạn có thể đánh tên của Menu Command vào Textbox **Name**. Dù ta cho Menu Command 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 MenuItems nhiều hơn.




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




Hình dưới đây cho thấy tất cả các MenuItems 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 trồi 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.

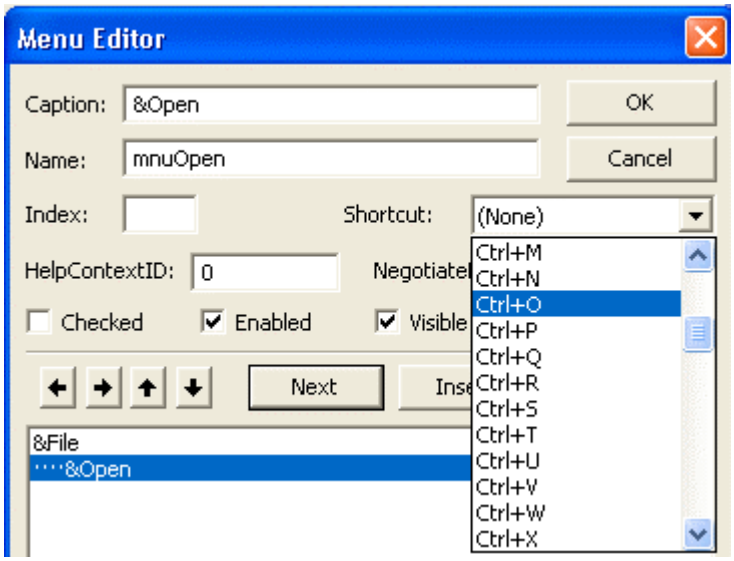
Nếu bạn đặt cho MenuItem **&Open** tên **mnuOpen**, thì khi bạn Click lên Caption nó trên Form trong lúc thiết kế, VB6 IDE sẽ hiển thị cái vỏ của **Sub mnuOpen_Click()**, giống như Sub cmdButton_Click() của một CommandButton:

```
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.



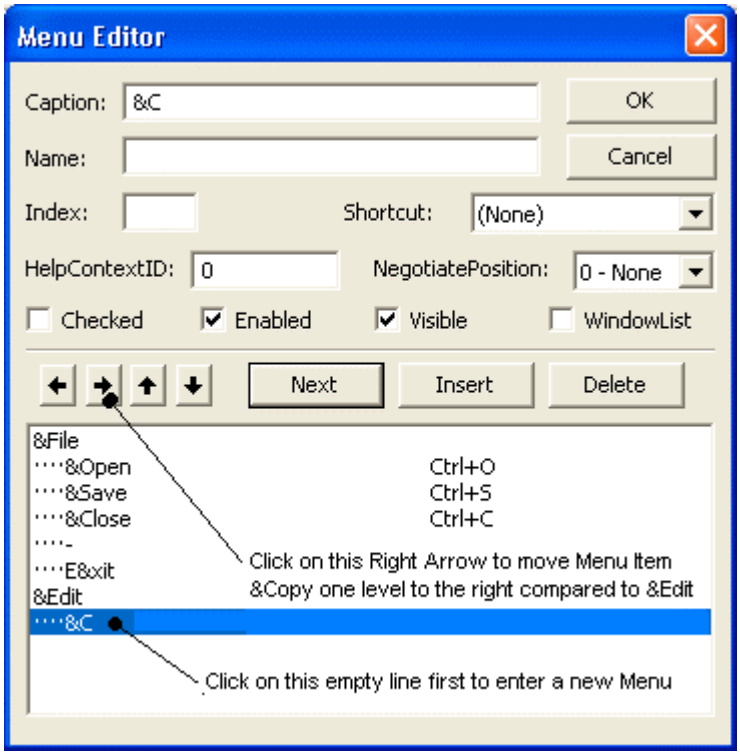
By default, menuItem được Enabled và Visible. Lúc thiết kế bạn có thể cho MenuItem giá trị khởi đầu của Enabled và Visible bằng cách dùng Checkboxes Enabled và Visible.

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.

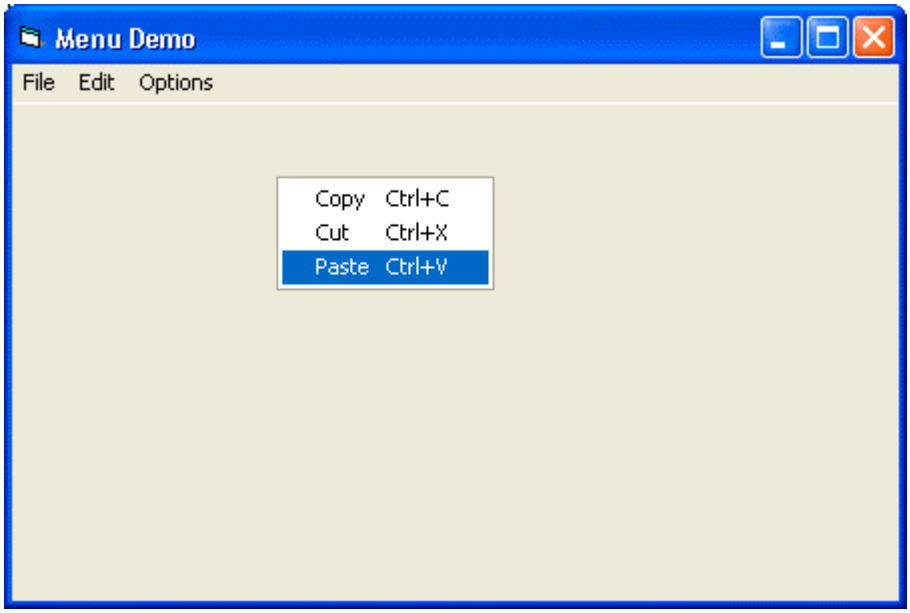



## Pop-up Menu

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:



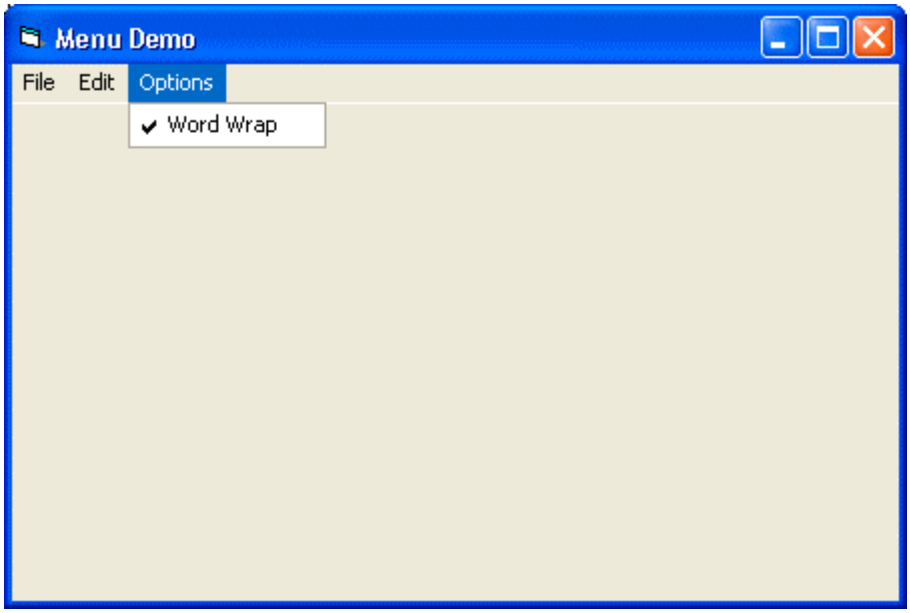

Code làm cho Popup menu hiện lên được viết trong Event Mousedown của một Object mà tình cờ ở đây là của chính cái Form:

```
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.

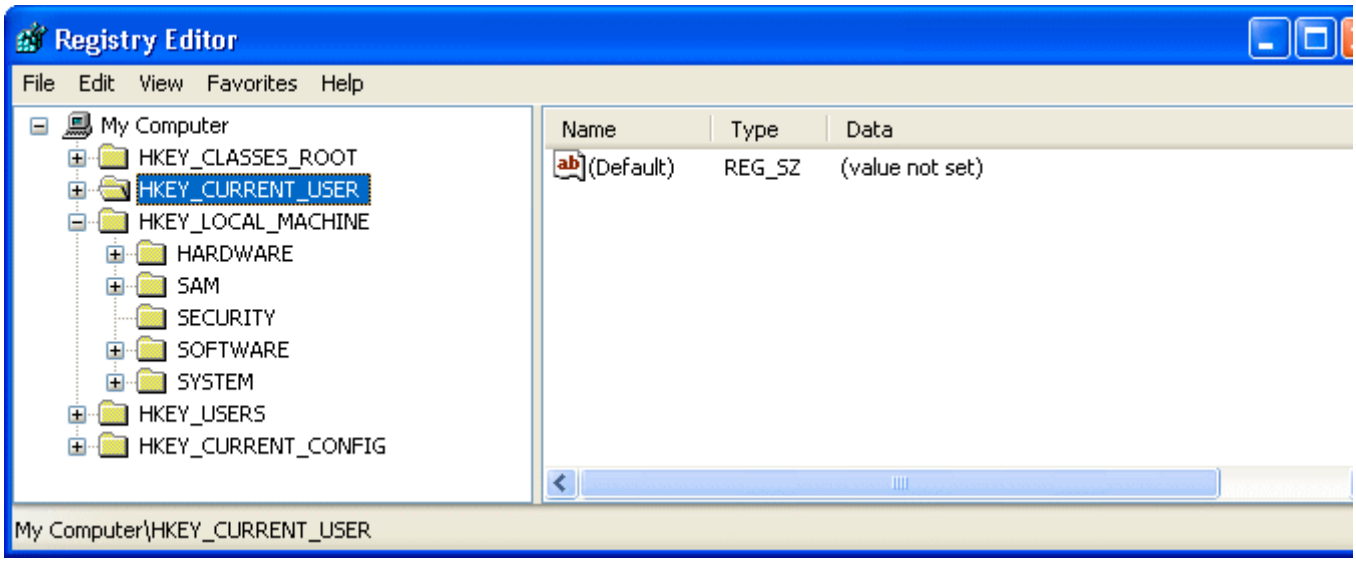
## Chứa menu Settings trong Registry

Giả tỉ 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**.

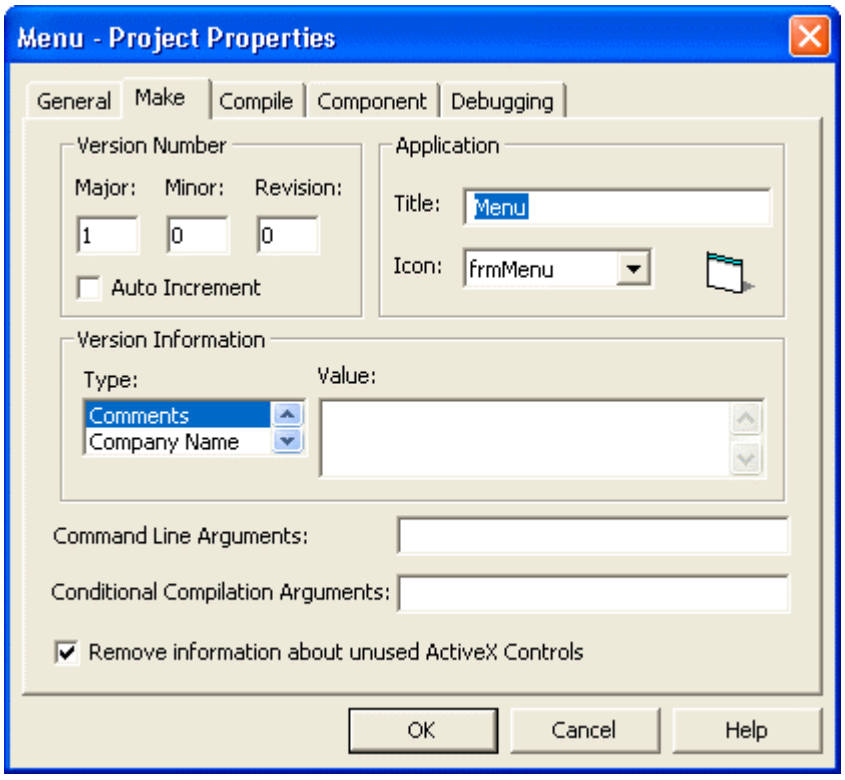



Trong program nầy ta cũng nhân tiện bắt program nhớ luôn vị trí của Form khi program ngừng lại, để lần tới khi User khởi động program thì program sẽ có vị trí lúc đầu giống y như trước.

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 sữa 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
```

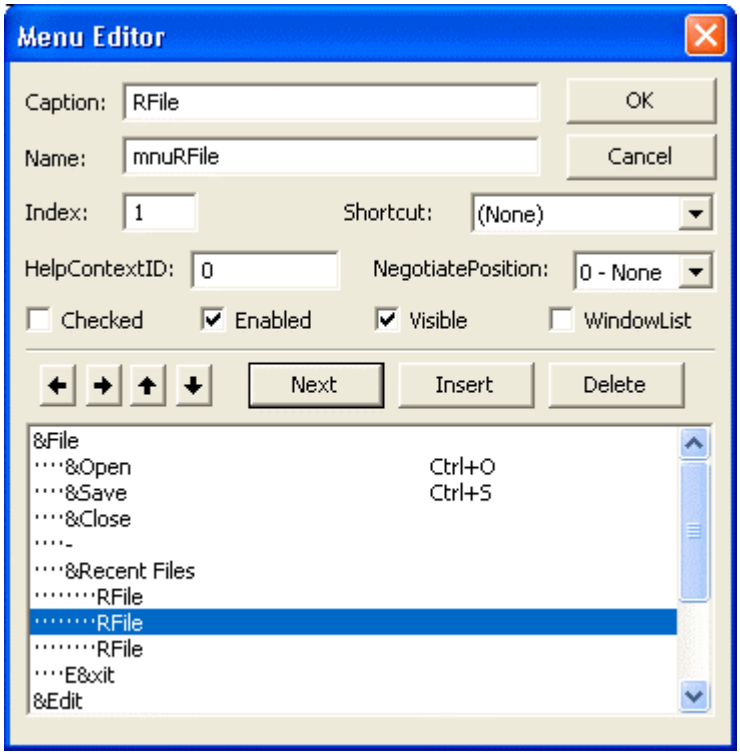
Lúc đầu khi chưa có gì trong Registry thì **"0"** (string "0" được converted bởi Val ra 0) là default value cho Left và Top, còn **"False"** là default value của mnuWordWrap.Checked.

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.

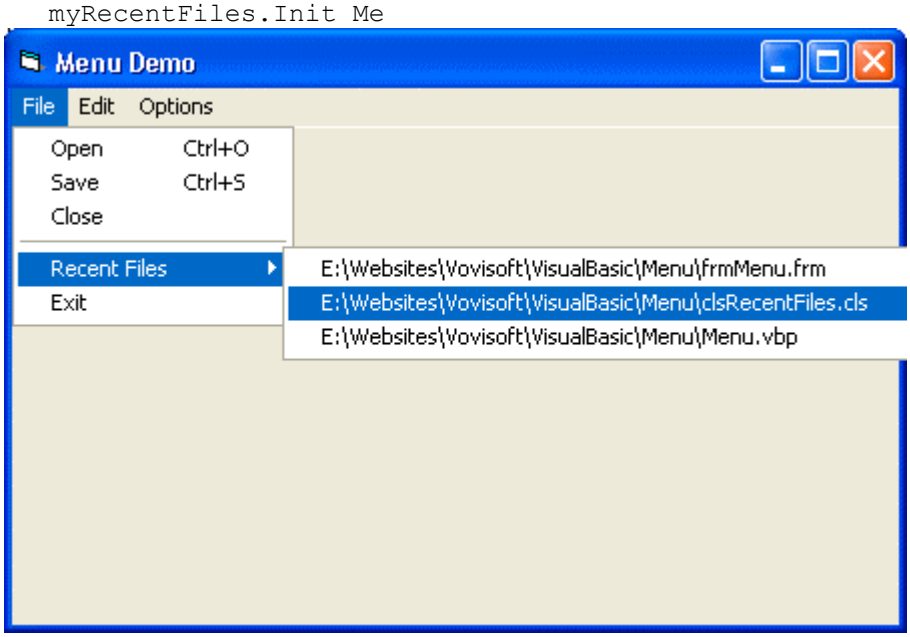



Dưới đây là code dùng để thêm tên File mới dùng nhất vào Registry:

```
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
   myRecentFiles.Init Me
```

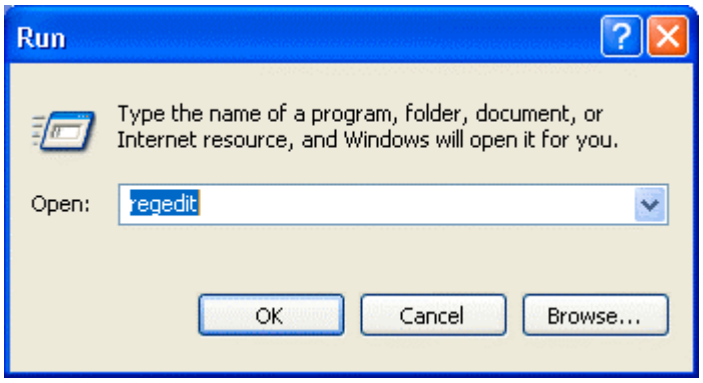



Ta sẽ dùng một Class tên **clsRecentFiles** để đặc biệt lo việc chứa tên Files vào Registry và hiển thị tên các Files ấy trong Menu. Bên trong clsRecentFiles ta cũng dùng clsString, là một Class giúp ta ngắt khúc String trong Registry ra tên của các Files dựa vào chỗ các delimiter character |.

```
' 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",
```

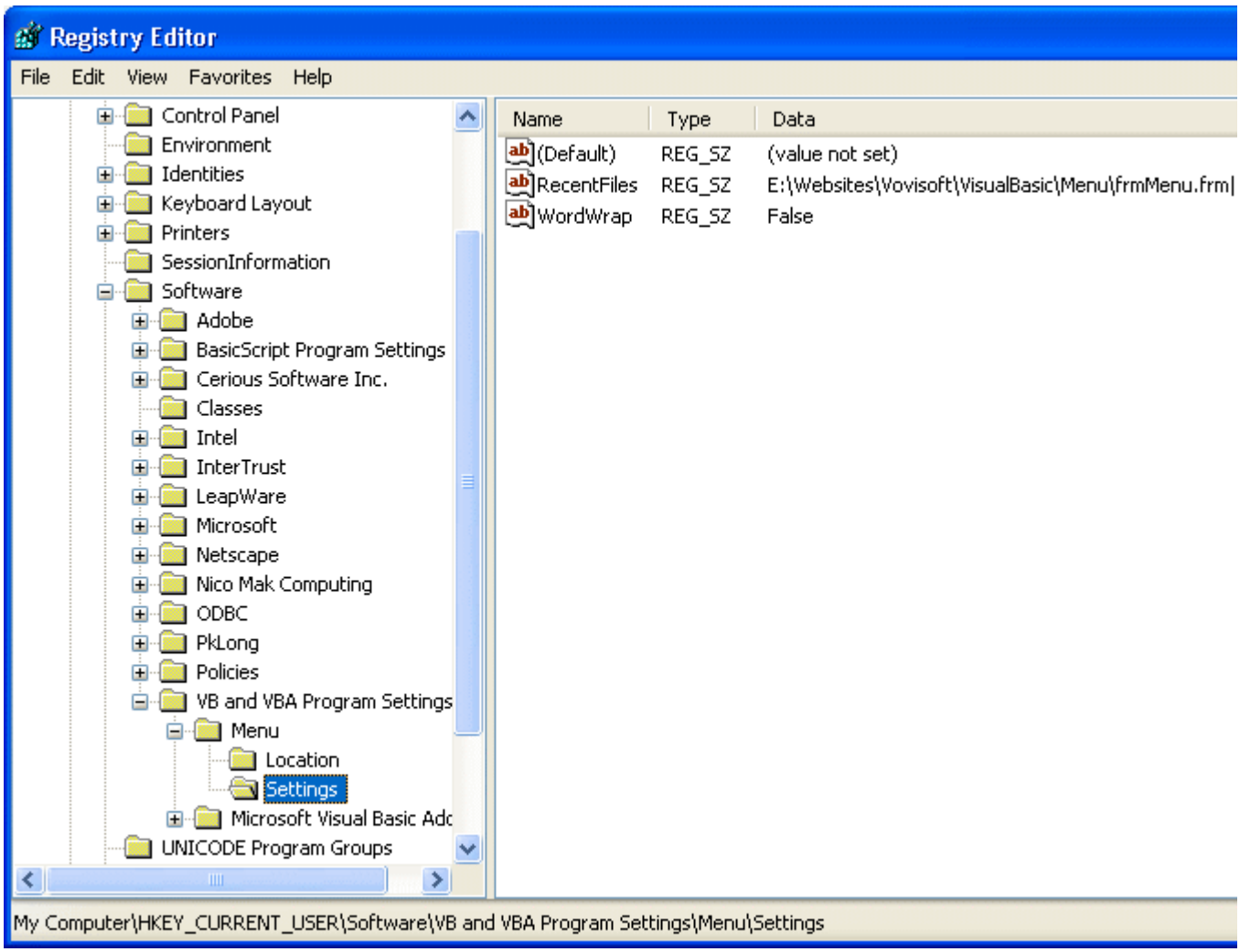
```
"RecentFiles", "")
   ' Assign the Delimiter character and tokennise the String (i.e.
split it) into FileNames
   RecentFiles.Delimiter = "|"
   UpdateMenu
End Sub

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
```

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 Bốn - Dùng Dialogs

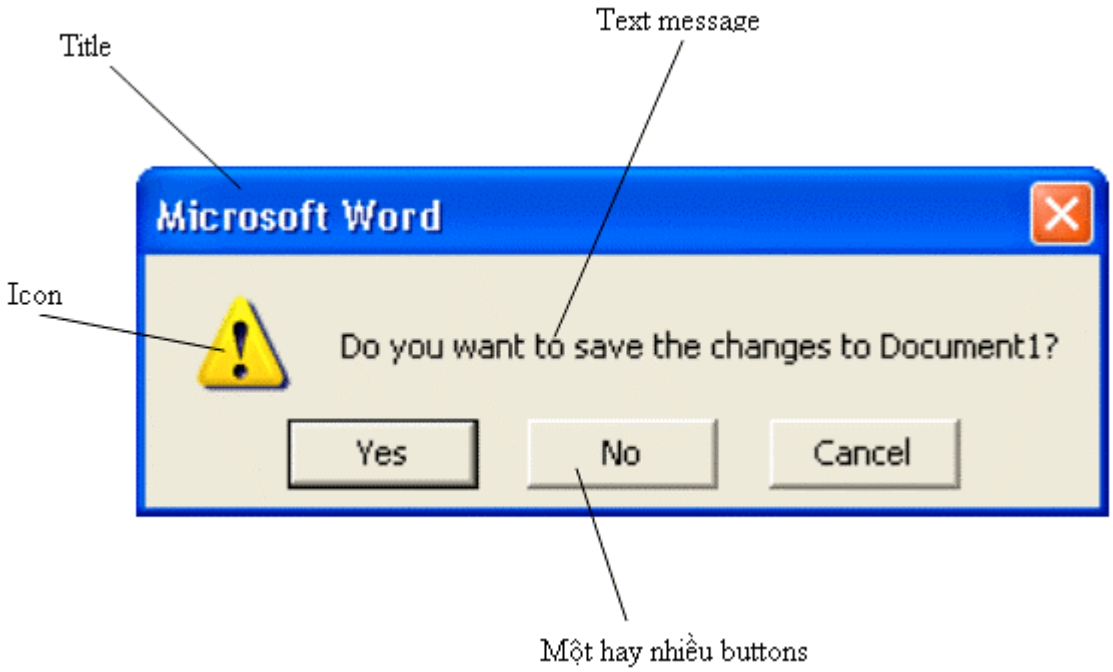
Dialogs (giao thoại) được dùng để hiển thị tin tức và nhận mouse hay keyboard input từ users tùy theo tình huống. Chúng được dùng để tập trung sự chú ý của users vào công tác đương thời của program nên rất hữu dụng trong các chương trình của Windows.

Có nhiều dạng Dialogs, mỗi thứ áp dụng cho một hoàn cảnh riêng biệt. Trong chương nầy 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

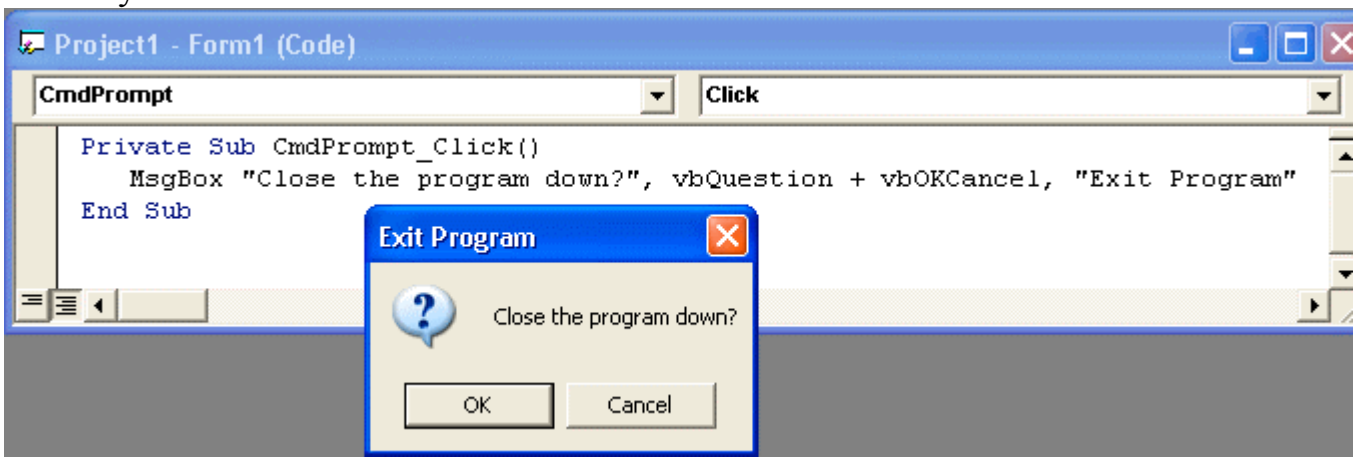
**Message Boxes** được dùng để nhắc nhở user một chuyện gì, và đòi hỏi một phản ứng nào đó từ user. Thí dụ như khi ta chấm dứt program MSWord mà chưa lưu trử hồ sơ thì MSWord sẽ nhắc ta lưu trử nó bằng Dialog dưới đây:




Trong trường hợp nầy user có thể click một trong 3 buttons. Nếu click **Yes** thì sẽ xúc tiến việc lưu trử 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 thi 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 nầy muốn đổi cách hiển thị Error message chỉ cần edit ở một chỗ. Nếu user muốn bạn lưu trữ tất cả mọi errors xẩy ra lúc run-time, bạn chỉ cần thêm vài hàng code trong Sub DisplayError để viết Error message vào một text file.

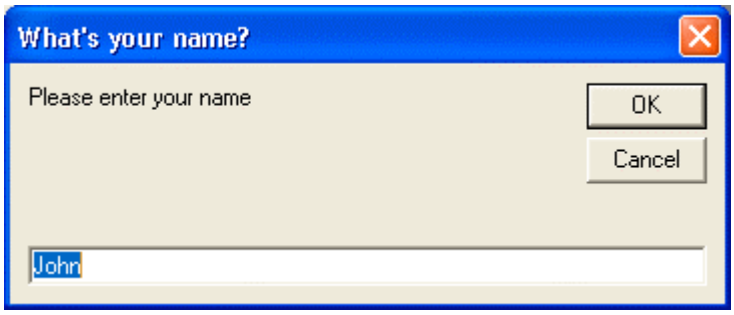
## Input Boxes

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 đàng 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 nầy.

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
   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:

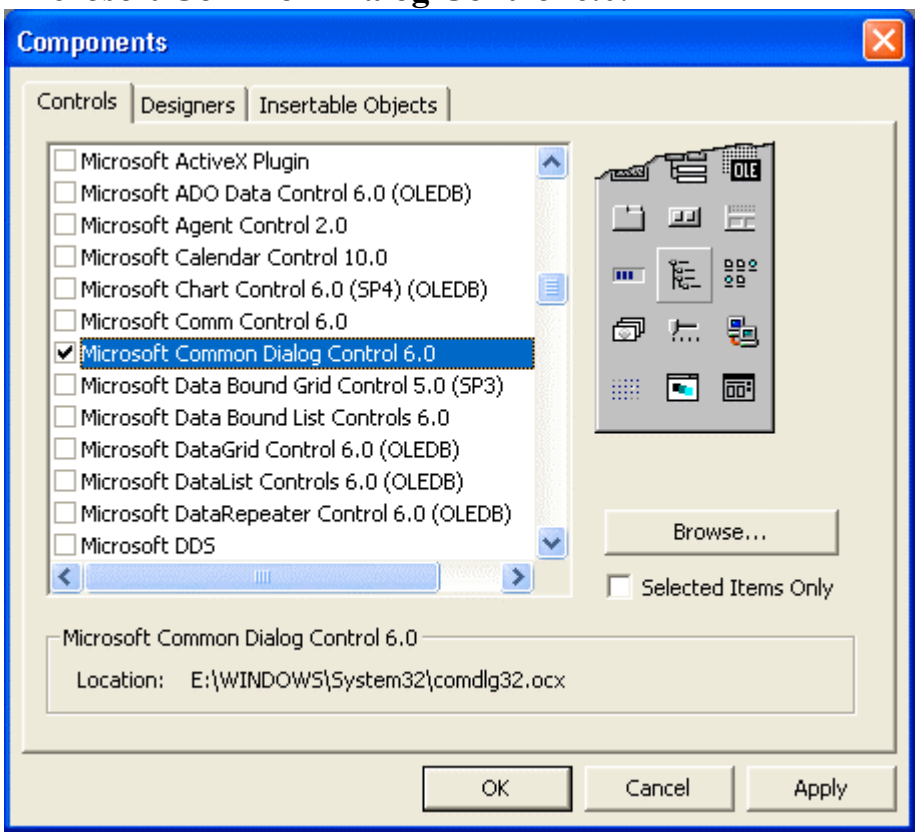
- Ta không thể làm gì được trong lúc user input data, phải đợi sau khi user click OK thì mới bắt đầu xử lý input textstring. Ngược lại nếu ta dùng một Textbox trong một Form thông thường, ta có thể code trong các Event handlers của Events **KeyPress** hay **Change** để kiểm soát các keystrokes của user.

- 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

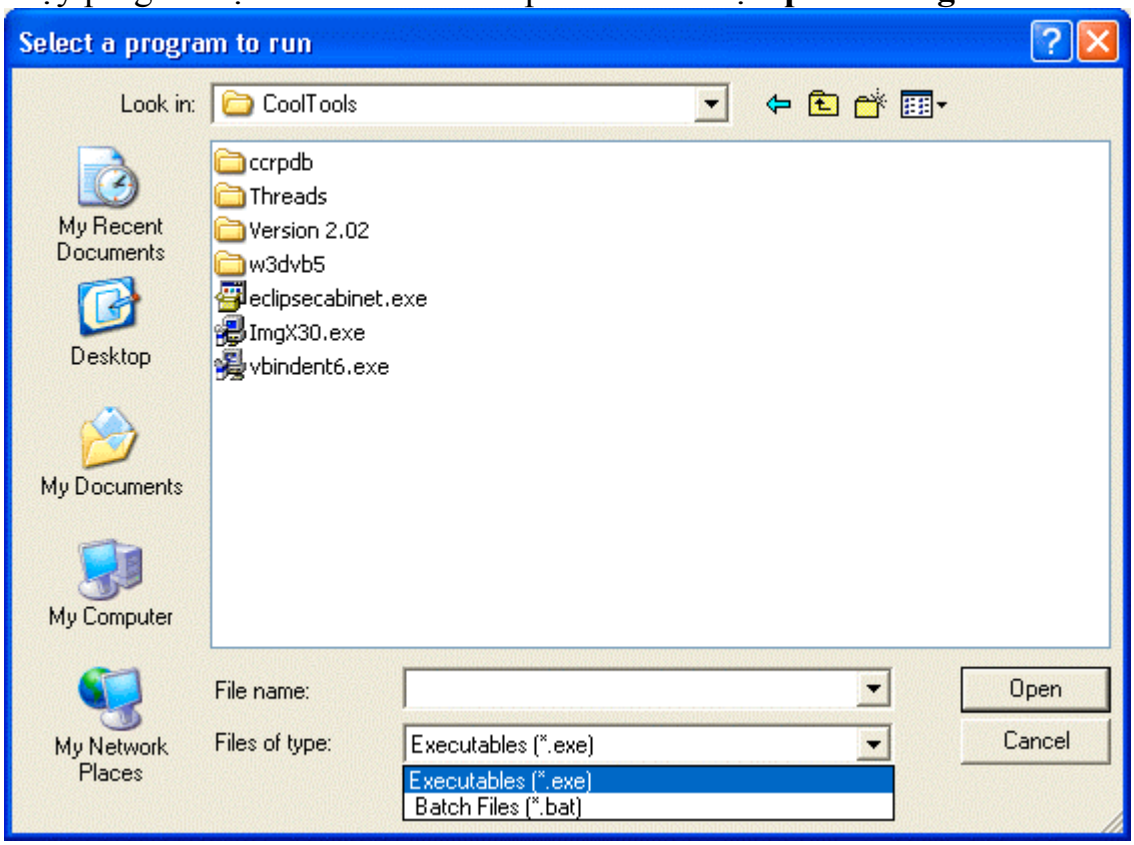
Bạn hãy mở một Project mới với một button tên CmdOpen trong Form1 và đánh vào code sau đây cho Sub CmdOpen_Click:

```
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   ' Lauch 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

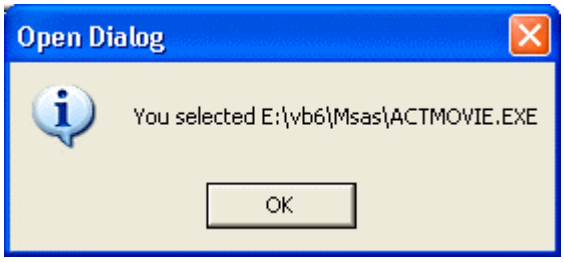
Form1. Vậy bạn hãy doubleclick icon của nó trong ToolBox. Bây giờ hãy chạy program lại và click button Open để hiển thị **Open Dialog**.



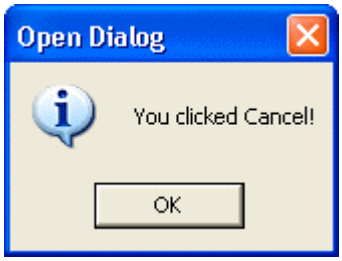

Bạn có thể chọn folder nào tùy ý bằng cách di chuyển từ folder nầy qua folder khác hay thay đổi disk drive. Nếu bạn click vào bên phải của combobox **File of type**, nó sẽ dropdown để cho thấy bạn có thể chọn một trong hai loại Files như liệt kê trong statement:

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

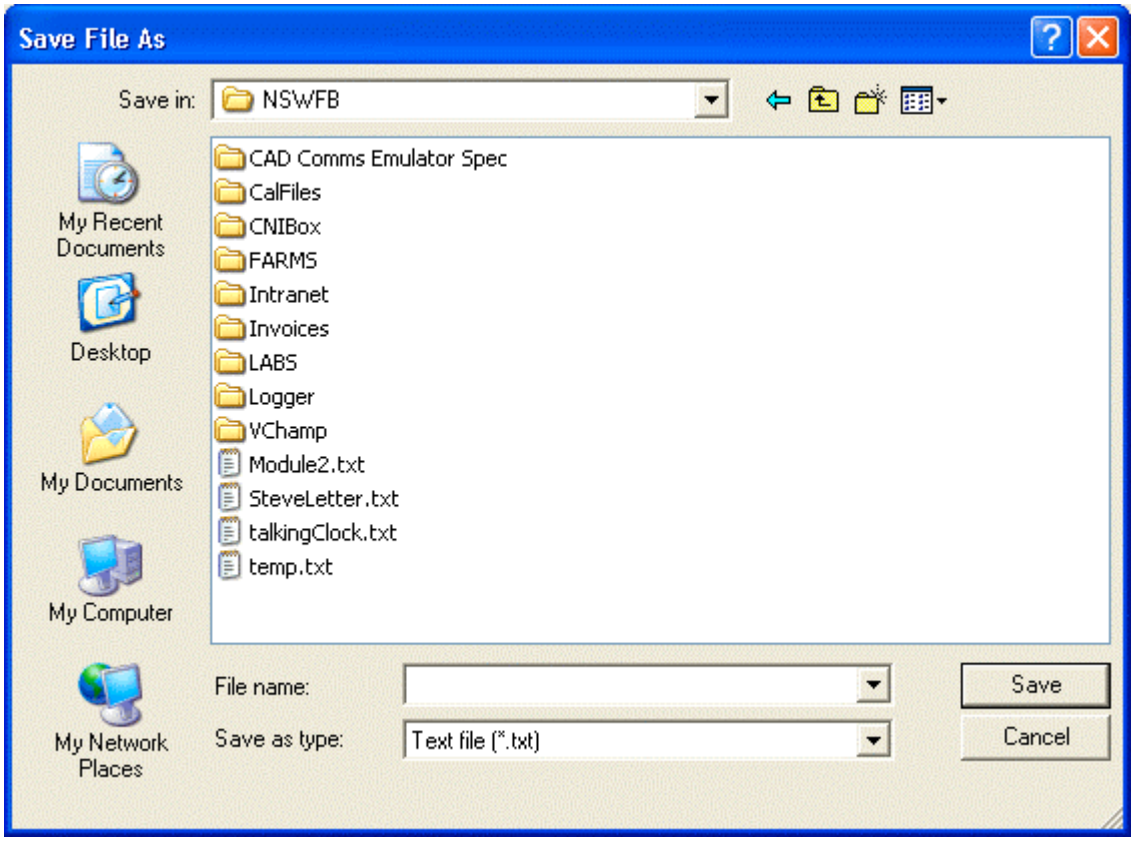
Sau khi chọn một Filename có sẵn hay đánh một tên vào **File name** textbox, bạn click **Open**. Sau đó, CommonDialog1.Filename sẽ chứa tên file bạn đã chọn hay đánh vào.



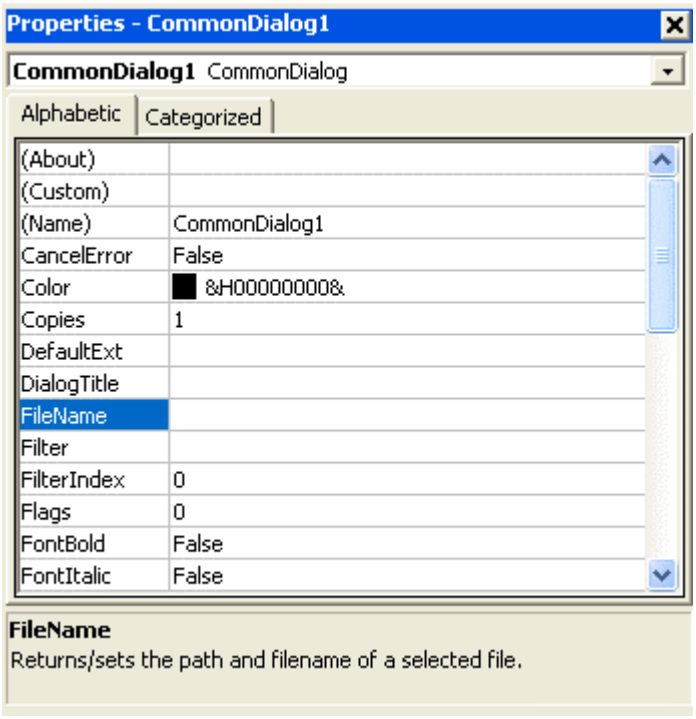
Vì ta cho **.CancelError = True** nên nếu user click Cancel program sẽ generate một Error số **32755 (cdlCancel)**. Ở đây ta bắt Error ấy bằng cách dùng **On Error GoTo DialogError** và thử Err.Number= cdlCancel để hiển thị Error message dưới đây:



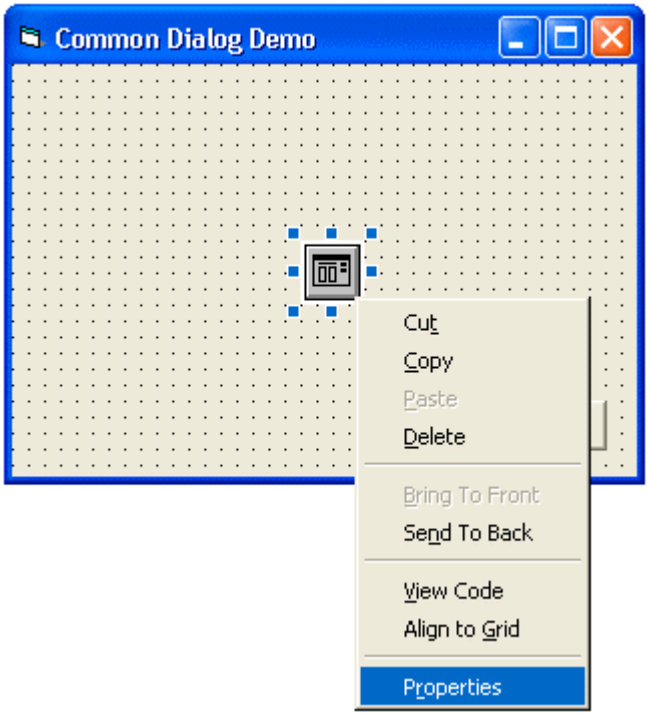

**Save Dialog** cũng tương tự như Open Dialog, ta dùng method **ShowSave** để hiển thị nó.



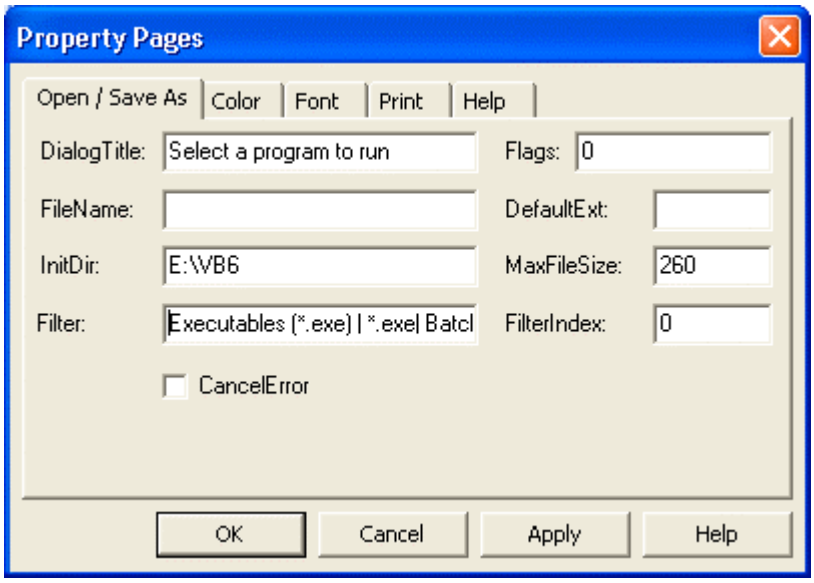

Trong thí dụ trên ta định nghĩa các properties của CommonDialog1 bằng code. Bạn cũng có thể dùng Properties Windows để định nghĩa chúng như dưới đây:




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**:

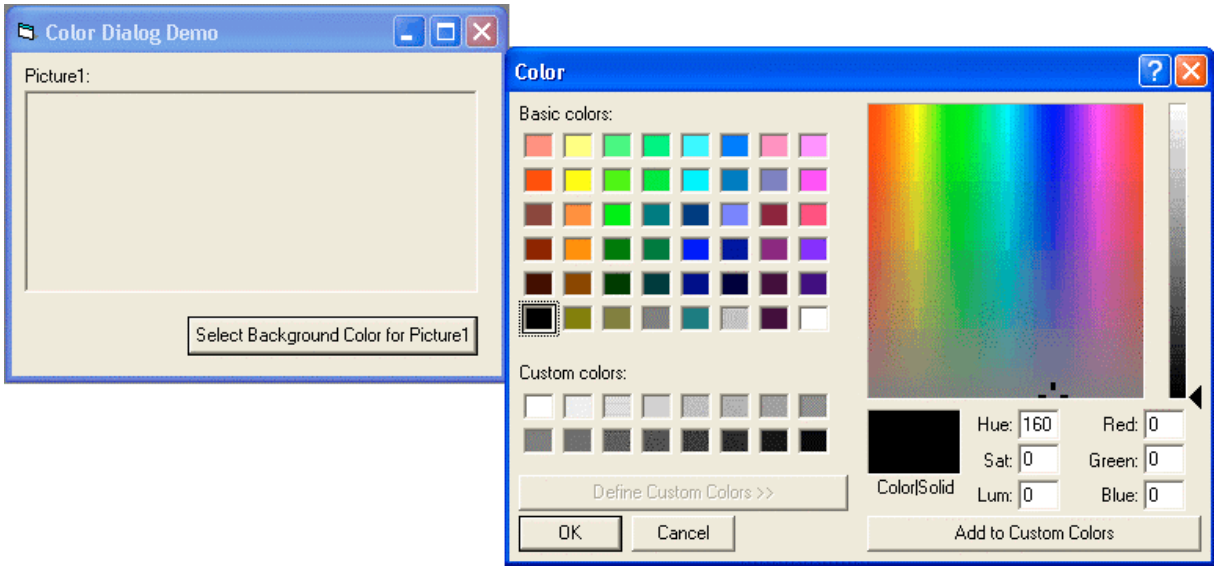



Properties Pages Dialog sẽ hiển thị với Tab **Open/Save As** có sẵn lúc đầu, bạn có thể đánh các tin tức như sau:

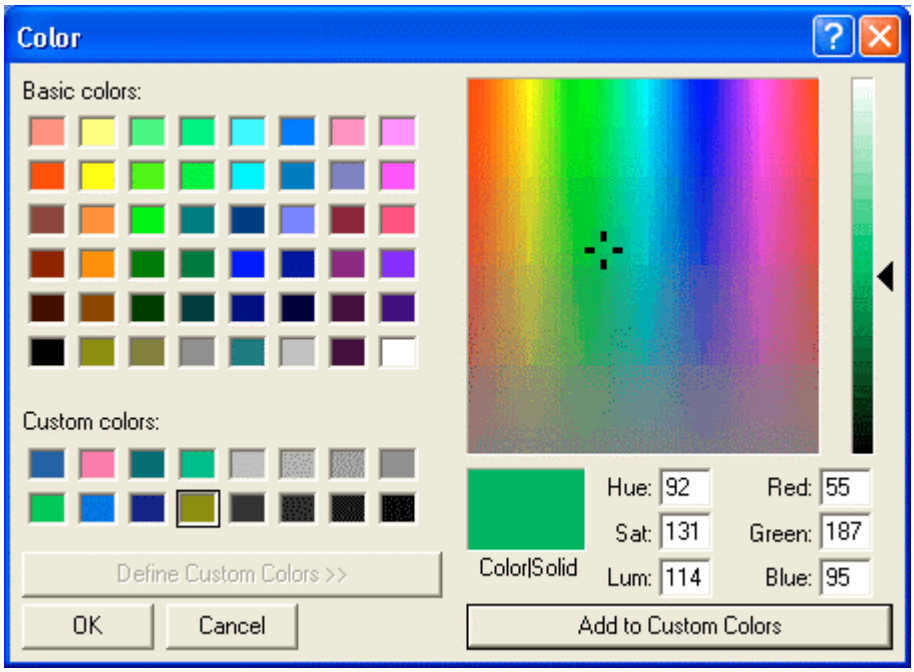



## Color Dialog

**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 **Color|Solid**. 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
      Exit Sub
   End With
NoColorChosen:
   ' Get here if user clicks the Cancel button
   MsgBox "You did not select a color!", vbInformation, "Cancelled"
   Exit Sub
End Sub
```

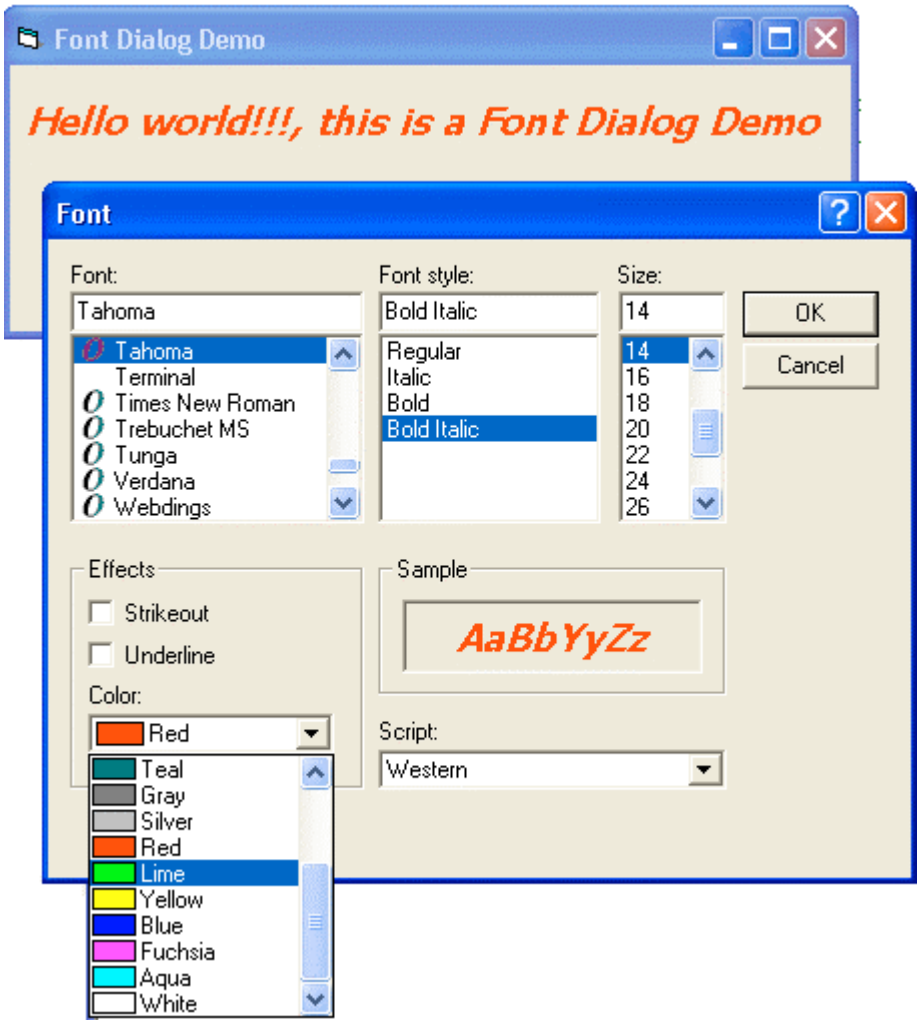
## Font Dialog

**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 hổ trợ** |
| **cdlCFPrinterFonts** | **2** | **Chỉ hiển thị các Fonts của màn ảnh, chưa chắc tất cả đều được printer hổ trợ** |
| **cdlCFBoth** | **3** | **Hiiển 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
      .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
```

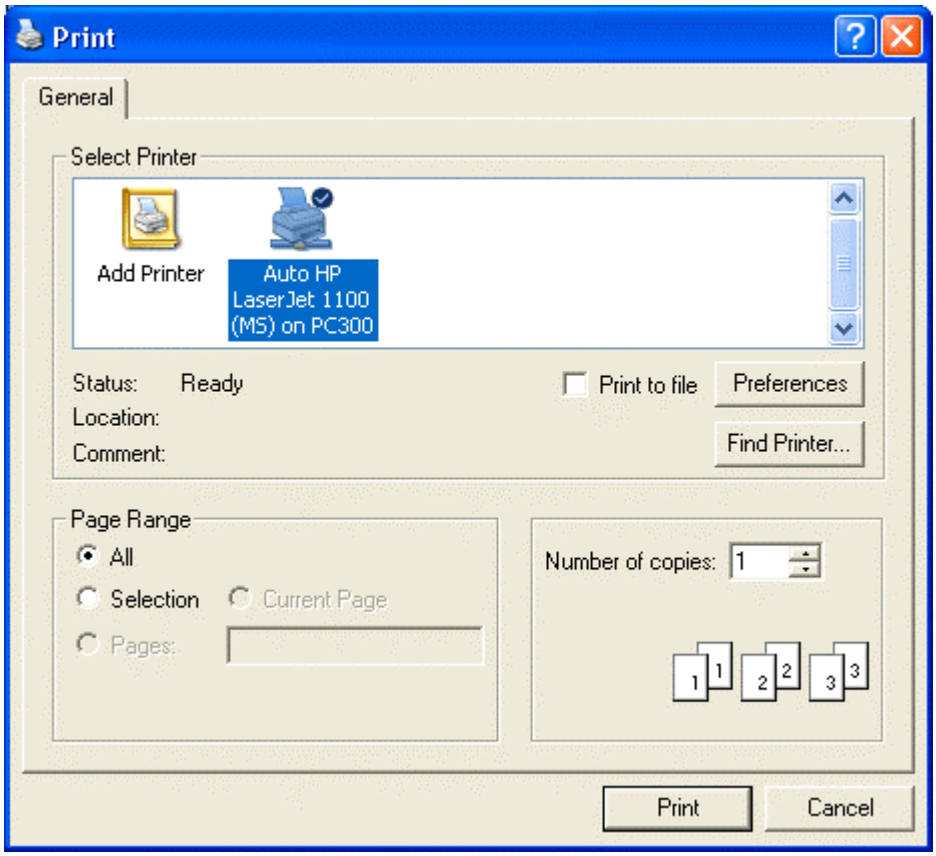
Chú ý: Nếu bạn quên cho Flags một trong những hằng số nói trên program sẽ cho một Error message như sau:


## 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 nầy, 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.

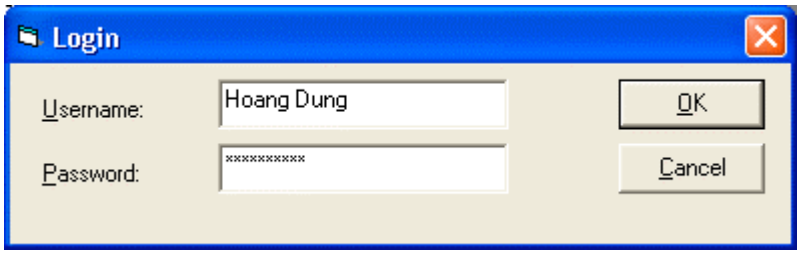
Sau đây ta thử triển khai một Login Form tổng quát, có thể dùng trong nhiều trường hợp. Khi khởi động, program nầy sẽ hiển thị một Login form yêu cầu user đánh vào tên và mật khẩu. Sau đó, nếu tên và mật khẩu hợp lệ thì cái Form chính của program mới hiện ra. Cách ta thực hiện là cho program khởi động 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:

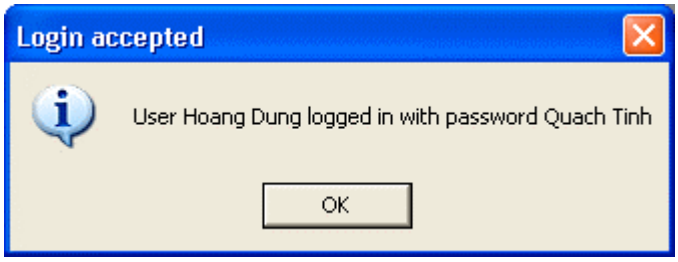
```
Sub Main()
   Dim strUserName As String
   Dim strPassword As String
   ' Call local Sub getUserInfo to obtain UserName and Password
   GetUserInfo strUserName, strPassword
   If strUserName = "" Then
      MsgBox "Login failed or aborted", vbInformation, "login
Aborted"
   Else
```

```
      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 tợ 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
End Sub
```

# Chương Năm - Dùng Đồ Họa (Phần I)

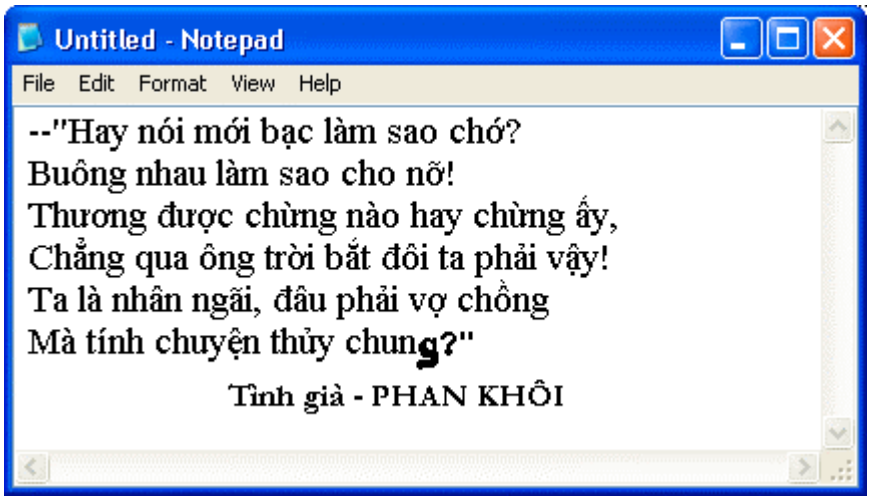
Tục ngữ Anh có câu: "Một hình ảnh đáng giá một ngàn chữ (a picture is worth a thousand words)", ý nói khi ta dùng hình ảnh để diễn tả sẽ giúp người xem hiểu nhanh hơn khi ta chỉ có nói thôi.

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 nầy 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 nầy 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 đóm nhỏ, mỗi đóm, được gọi là một **pixel**, có khả nằng 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 nầy, 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 1024x768=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 nầy có khả năng hiển thị $2^{16}$ = 65536 màu khác nhau. Muốn dùng 16 bit color

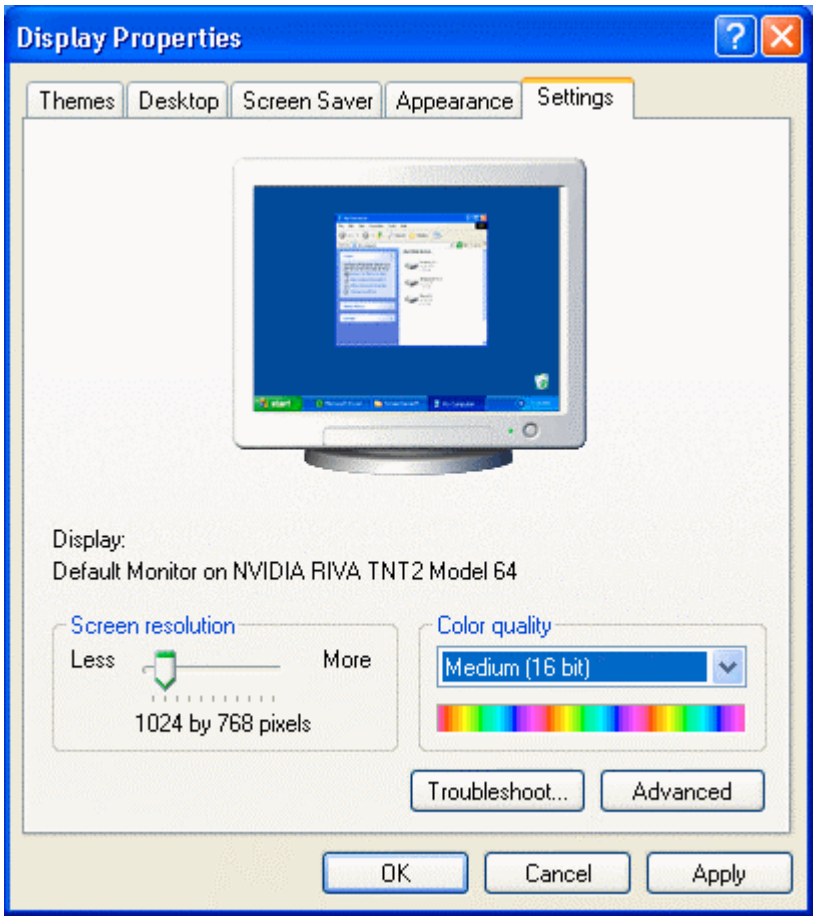
cho SuperVGA, ta cần phải có 1024x768x2 =1572864 bytes, tức là gần 1,6 MB. Đó là lý do tại sao ta cần 2MB RAM. Lưu ý là RAM của VGA (Vector Graphic Adapter) card không liên hệ gì với RAM của bộ nhớ computer.

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 nhờ bà con ngồi xích 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:



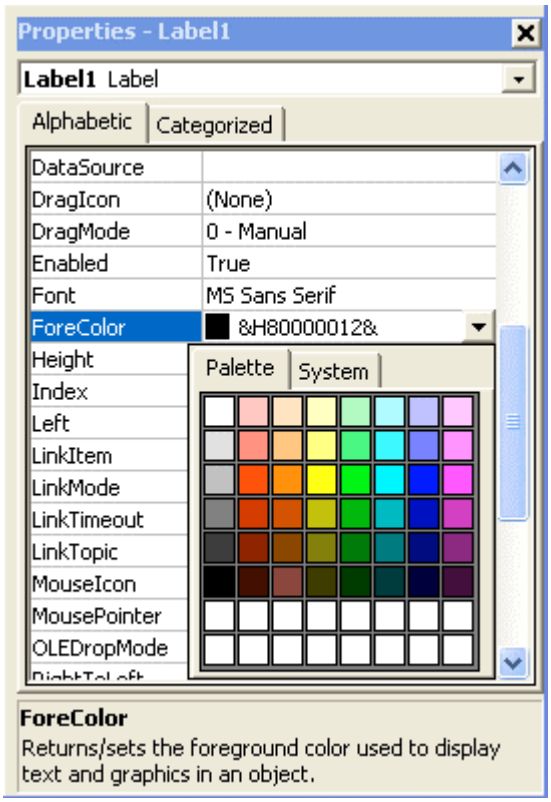

Khi ta tăng độ mịn của màn ảnh, các hình ảnh sẽ nhỏ lại vì kích thước của pixel được thu nhỏ lại. Do đó, ta có thể cho hiển thị nhiều thứ hơn trên desktop. Phẩm chất của các graphic vẫn không thay đổi, mặc dầu hình nhỏ hơn. Nhớ là muốn hình rõ hơn thì khi cấu tạo và chứa graphic, ta phải dùng một độ 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 rẽ 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 cở 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** | 0xFFFFFF | White |

- Dùng **Function QBColor** để chọn một trong 16 màu. Function QBColor xuất phát từ thời Quick Basic (QBasic) của Microsoft. QBsic 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.

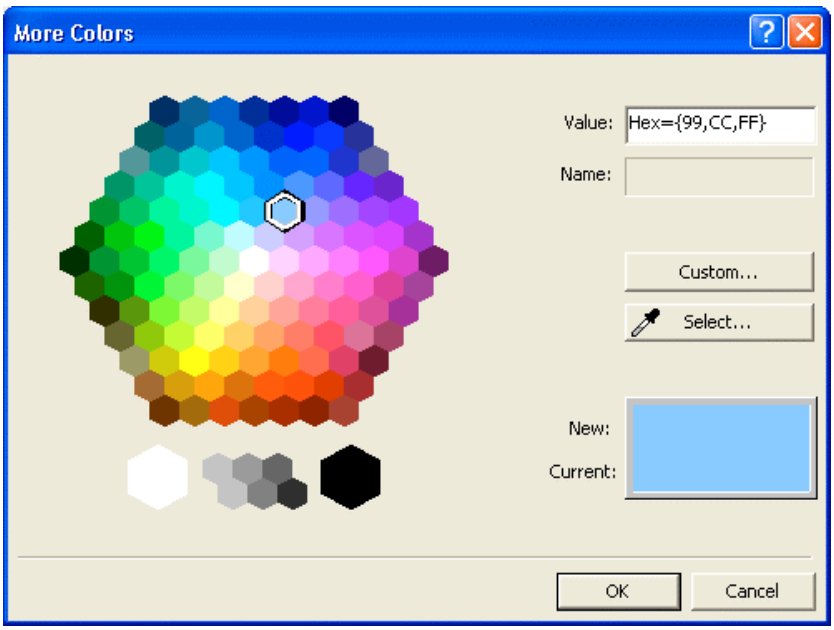
| Trị số | Màu | Trị số | Màu |
|---|---|---|---|
| 0 | Black | 8 | Gray |
| 1 | Blue | 9 | Light Blue |
| 2 | Green | 10 | Light Green |
| 3 | Cyan | 11 | Light Cyan |
| 4 | Red | 12 | Light Red |
| 5 | Magenta | 13 | Light Magenta |
| 6 | Yellow | 14 | Light Yellow |
| 7 | White | 15 | Bright White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

- Dùng **Function RGB** để trộn ba màu **R**ed, **G**reen và **B**lue. Trong cái bảng liệt kê các intrinsic color constants phía trên, nếu để ý bạn sẽ thấy vbWhite(0xFFFFFF) 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.

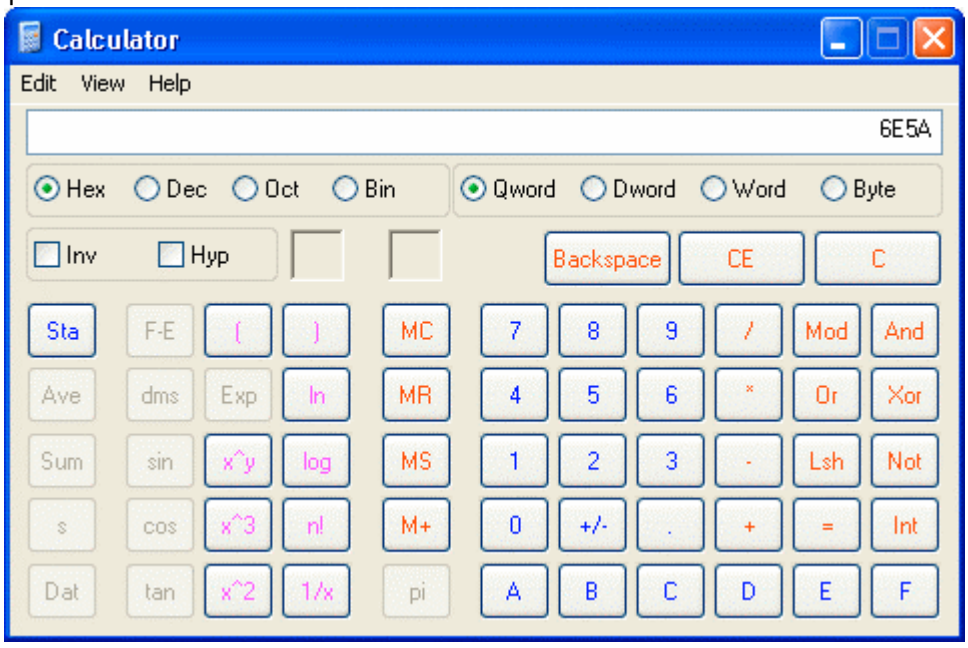
  Hệ thống số ta dùng hằng ngày là Thập Phân. Trị số 0xFF của vbRed là con số 255 viết dưới dạng Thập lục phân (Hexadecimal hay **Hex** cho gọn và ở đây được đánh dấu bằng **0x** trước con số để phân biệt với số Thập phân). Trong hệ thống số Hex ta đếm từ 0 đến 9 rồi A,B,C,D,E,F rồi qua số hàng thập lục 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):



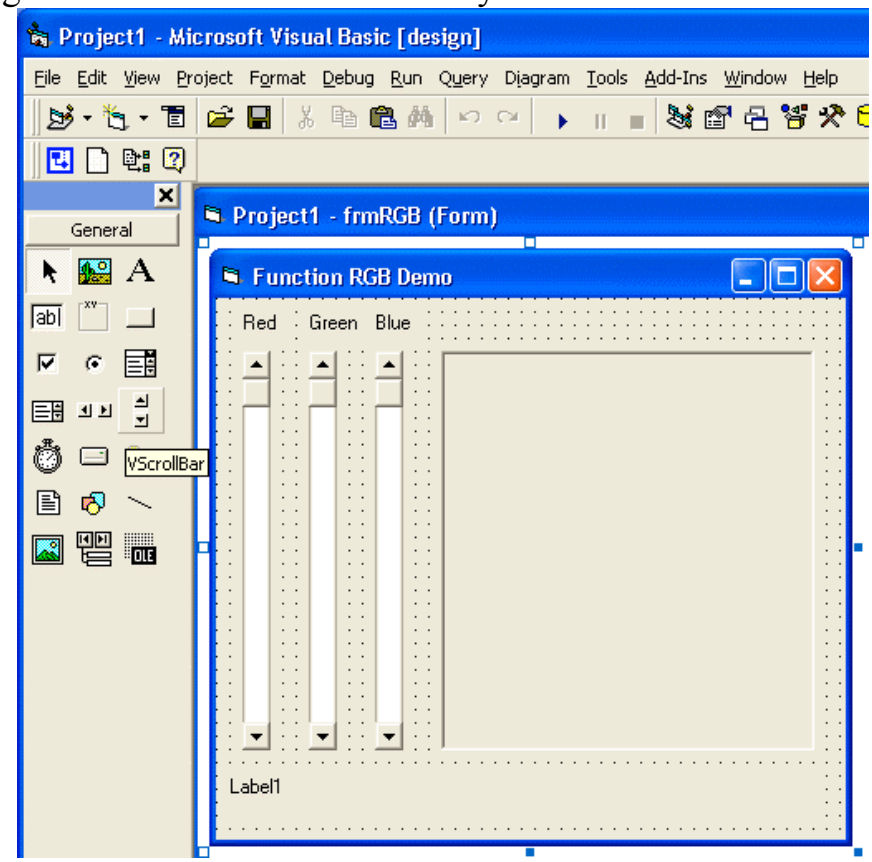

***Ghi chú:*** *Bạn có thể dùng Windows Calculator để hoán chuyển số giữa các dạng Decimal, Binary và Hexadecimal. Chọn* ***View|Scientific*** *thay vì* ***View|Standard****.*



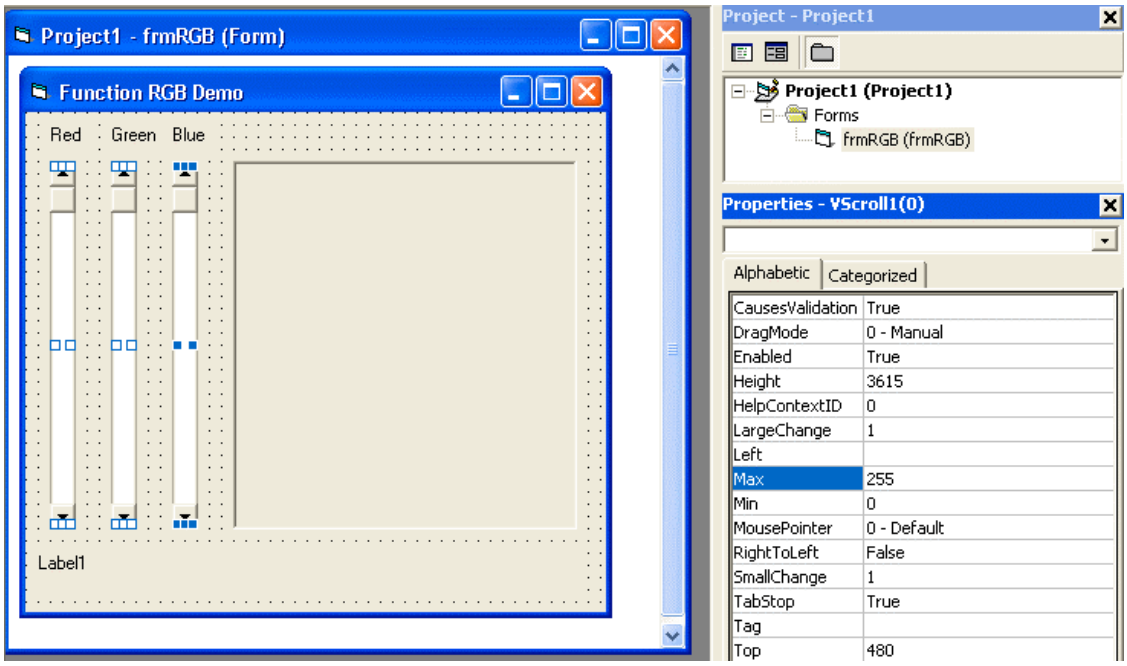
## Function RGB

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 nầy 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:



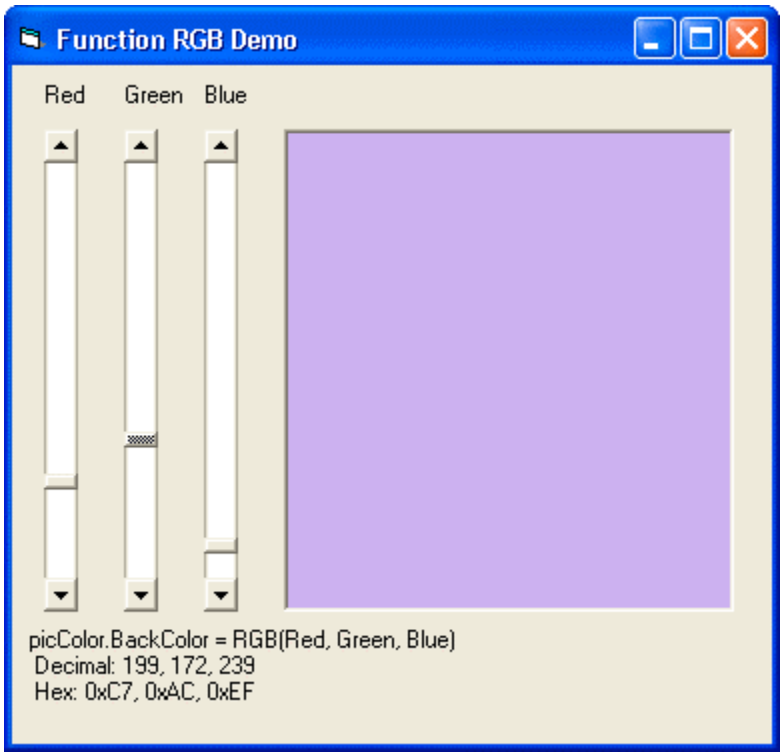

Bây giờ select cả ba VScrolls và edit value của **property Max** trong cửa sổ Properties thành **255**, ý nói khi kéo cái bar của một VScroll1 lên xuống ta giới hạn trị số của nó từ Min là 0 đến Max là 255.




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:



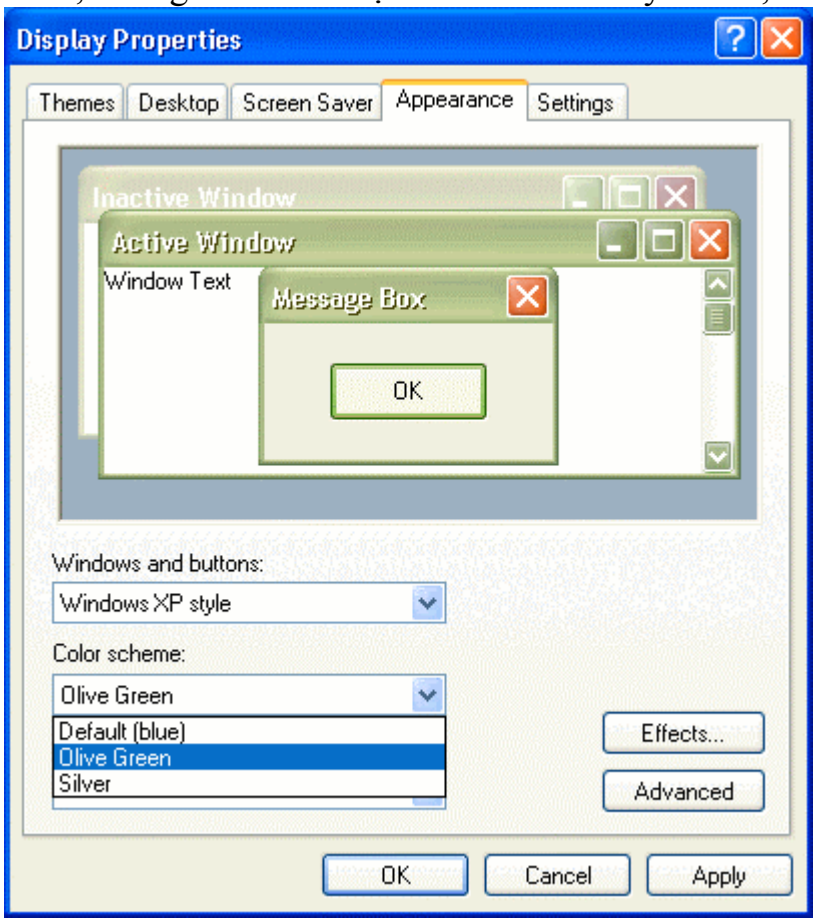

## Color Mapping

Nếu dùng Hex Calculator đổi con số 0xFFFFFF ra decimal ta sẽ được 16777215, nếu kể cả số 0 ta sẽ có tổng cộng 16777216 màu. Lúc nãy 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 hổ trợ đến 8 bits. Nó sẽ cung cấp 256 màu khác nhau. Nếu Function RGB đói 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 nầy được gọi là **Color Mapping** và cái màu được làm ra được gọi là **custom color**.

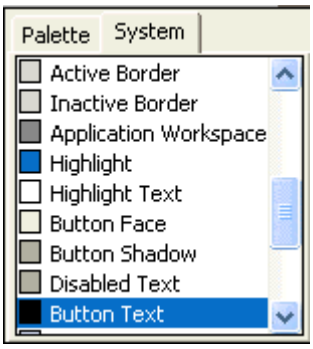
## Dùng Intrinsic Color Constants

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 nổi 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 trử 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 trử kiểu nầy 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.