ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
Trung Tâm VNIT

Giáo Trình

Lập Trình Ứng Dụng Với Ngôn Ngữ Visual Basic

Biên soạn : GV Bùi Tiến Trường

Http ://dontruongbt.spaces.live.com/

- 2009 -

Chương I : Tổng Quan Về Visual Basic

Microsoft Visual Basic 6.0 là môi trường phát triển ứng dụng tích hợp (Integrated Development Environment – IDE) của Micorosoft dành cho lập trình viên sử dụng ngôn ngữ Visual Basic để xây dựng các ứng dụng.
Visual Basic 6.0 là một thành phần của bộ công cụ phát triển ứng dụng Visual Studio 98

Các phiên bản chính:

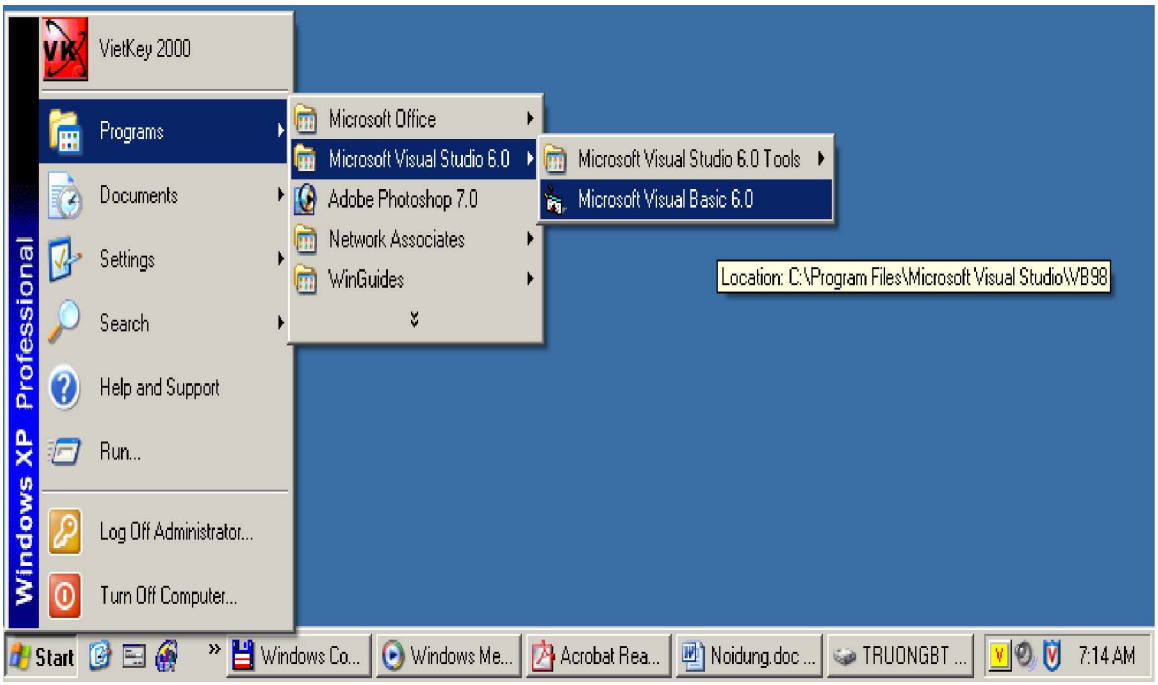
- MS Visual Basic 6.0 Learning Edition
- MS Visual Basic 6.0 Professional Edition
- MS Visual Basic 6.0 Enterprise Edition
-

Cách cài đặt Visual Basic 6.0

- Các yêu cầu về cấu hình phần cứng: CPU, Ổ cứng, RAM,...
- Cài đặt Visual Basic 6.0 từ bộ cài đặt Visual Studio 98
- Cài đặt Visual Basic 6.0 từ bộ cài đặt dành riêng cho Visual Basic
- Cài đặt tài liệu tham khảo Microsoft Develop Network – MSDN

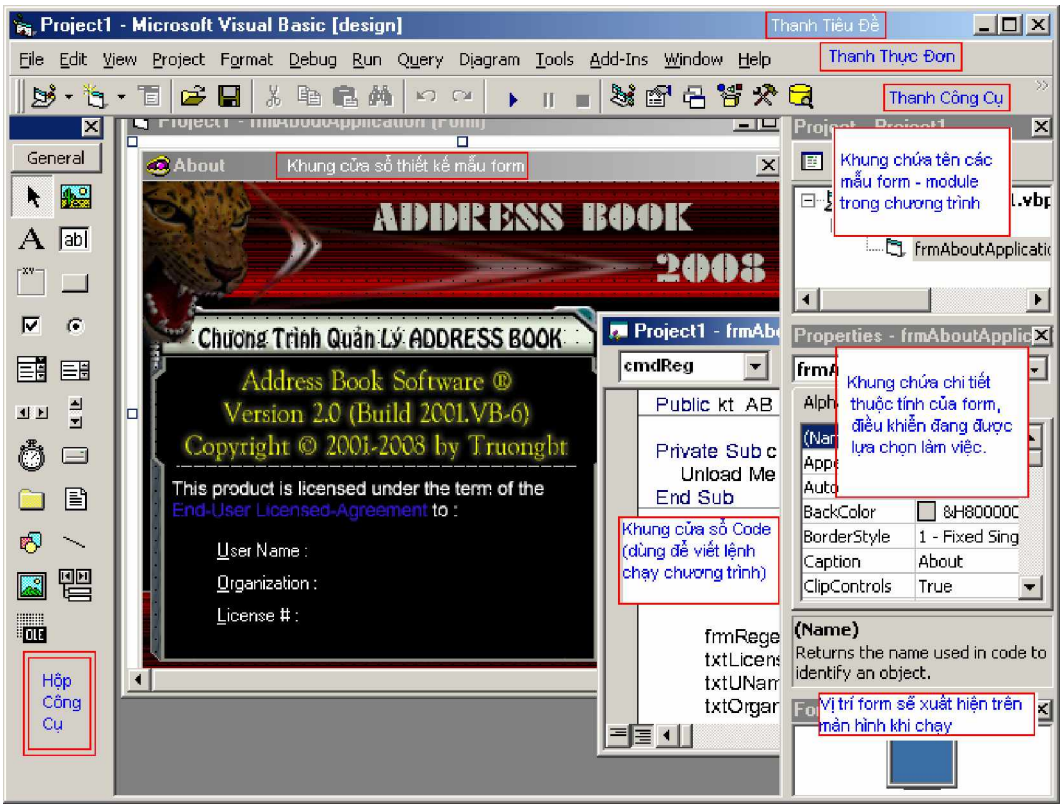
I, Khởi động chương trình VB

Start à Programs à Microsoft Visual Studio 6.0 à Microsoft Visual Basic 6.0



Đường dẫn : C:\Program Files\Microsoft Visual Studio\VB98\VB6.EXE

II, Giới thiệu giao diện cửa sổ làm việc của VB

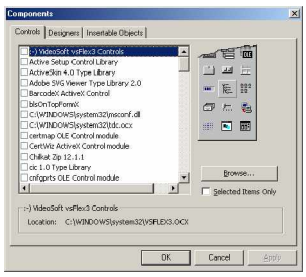


II.1, Thanh công cụ (Tool Box)

(Menu View / Toolbox)

Toolbox là cửa sổ chứa các nút công cụ với các điều khiển (Control) hay còn gọi là các đối tượng (Object) của VB.

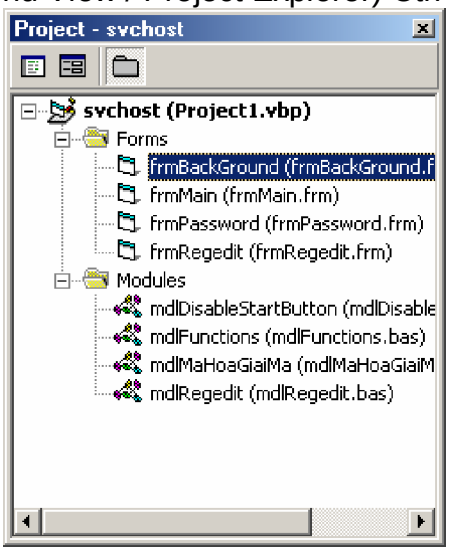
Có 2 loại nhóm công cụ :
- Công cụ hỗ trợ sẵn ngay khi khởi động chương trình
- Công cụ mở rộng, được lấy thêm theo yêu cầu của chương trình. Việc lấy thêm công cụ tiến hành bằng cách vào menu Project / chọn mục Components.. (phím tắt là Ctrl + T).



Để lấy công cụ nào thì bạn có thể tích chọn vào nó rồi ấn OK

II.2, Cửa sổ Project Explorer

(menu View / Project Explorer) Ctrl + R



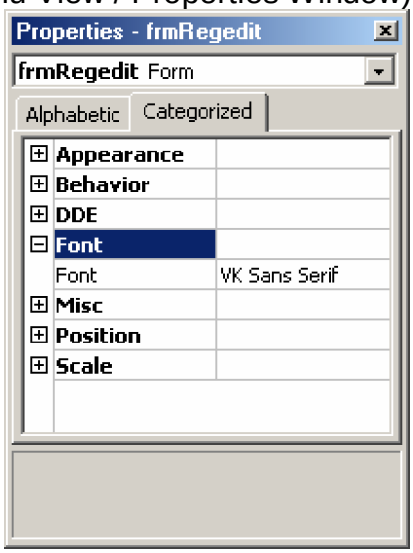
Cửa sổ Project Explorer dùng để quan sát và quản lý toàn bộ dự án mà bạn đang thiết kế. Trên cửa số này sẽ liệt kê tên dự án và tất cả các form mà bạn đã thiết kế.

Việc sắp xếp có thể theo nhóm thư mục hoặc theo vần tên form

Hỗ trợ chức năng giúp bạn có thể gọi xem thiết kế hoặc mã nguồn của form được chọn trong danh sách

II.3, Thanh thuộc tính (Properties)

(mnu View / Properties Window) F4



Properties là thanh chứa những tính chất quan trọng của các đối tượng mà chương trình Visual Basic đã thiết kế sẵn. Còn việc thiết lập và sử dung thuộc tính như thế nào thì tuỳ thuộc vào người lập trình.

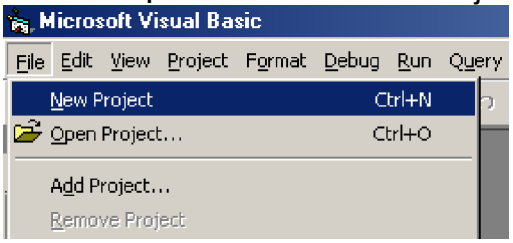
Mỗi khi thay đổi việc lựa chọn công cụ thì các thuộc tính trên thanh Properties cũng sẽ tự động thay đổi theo.

Việc sắp xếp các thuộc tính có thể theo vần Alpha hoặc theo nhóm chức năng của thuộc tính.

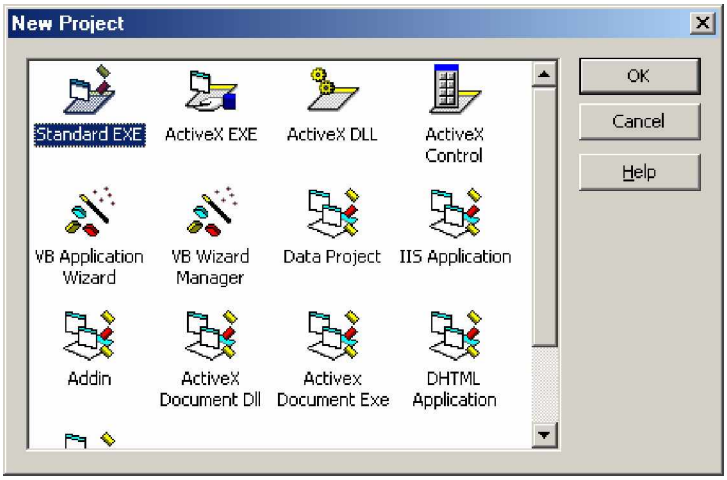
III, Tạo/Lưu Project làm việc

III.1, Tạo Project

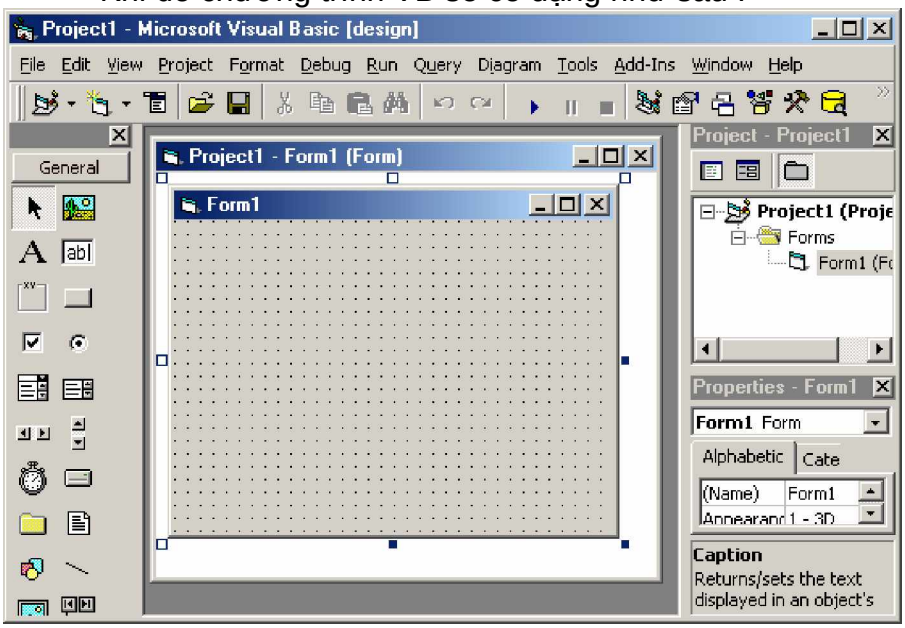
- Khởi động chương trình Visual Basic
- Chọn menu File / New Project (Ctrl + N)



- Trong hộp thoại New Project thì ta sẽ lựa chọn loại Project muốn tạo. Thông thường là lựa chọn Standard EXE, còn nếu muốn lập trình các dự án với cơ sở dữ liệu thì chọn Data Project.



- Khi đó chương trình VB sẽ có dạng như sau :



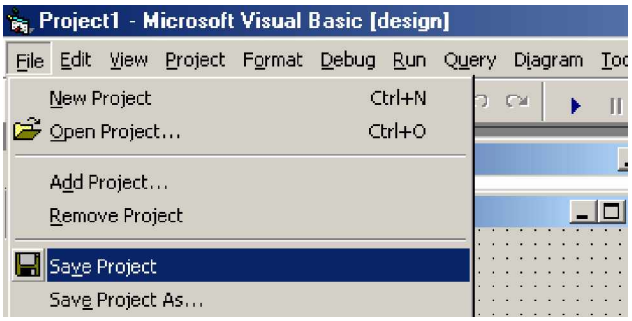
III.2, Lưu Project

Một trong những vấn đề quan trọng khi viết chương trình là lưu lại những gì mình đã làm. Mặc dù chưa có thao tác gì thay đổi trong Form trống, ra cũng nên lưu Form lên đĩa. Khi ra lưu Project, có 2 loại tập tin được lưu :

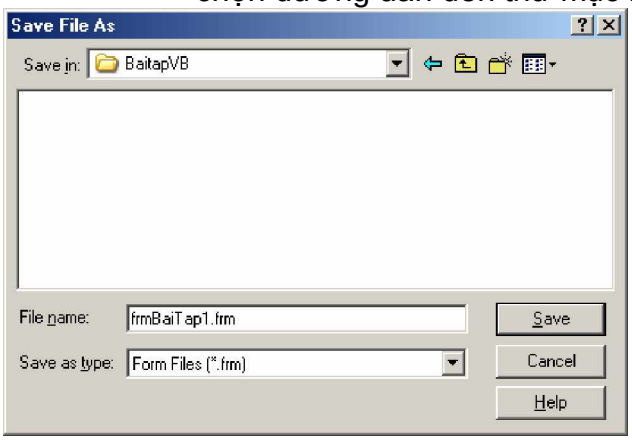
1. Tập tin chứa các thông tin VB cần trong việc xây dựng Project. Tập tin này có phần mở rộng là VBP
2. Tập tin chứa thông tin về Form. Tập tin này có phần mở rộng là FRM

Các bước thực hiện :

- Vào menu File / Save Project



- VB sẽ hiện lên hộp thoại cho phép ta lựa chọn đường dẫn để lưu bài. Ta chọn đường dẫn đến thư mục riêng của mình trong đĩa, sau đó ấn Save



Lưu ý : Ta nên đổi tên form trước khi thực hiện việc lưu. Tên đặt cho form tuỳ theo chức năng xử lý của chương trình, và thường bắt đầu bằng frm

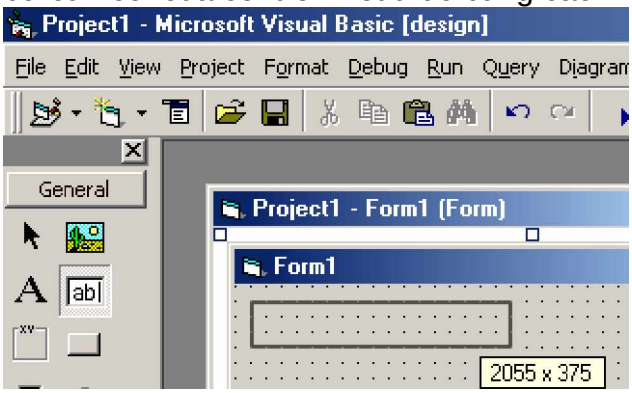
IV, Các bước cơ bản xây dựng 1 chương trình (Hello)

IV.1, Tạo giao diện : gồm 1 form , 1 textbox và 1 command button

Texbox

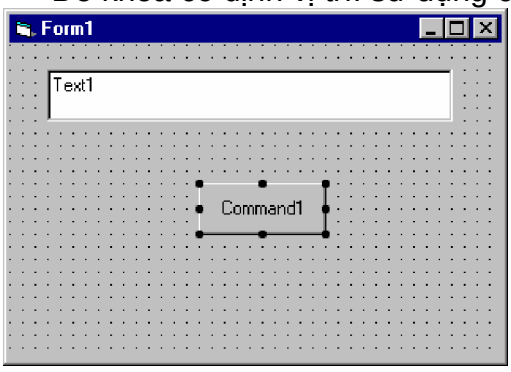
Command button

Để vẽ 1 control (điều khiển) ta Nhấn chọn điều khiển trên toolbox – trong bài này chọn textbox. Chuyển con trỏ lên form tới vị trí muốn đặt control, Nhấn và Kéo chuột tới kích cỡ cần có của control. Kết thúc bằng cách Nhả chuột



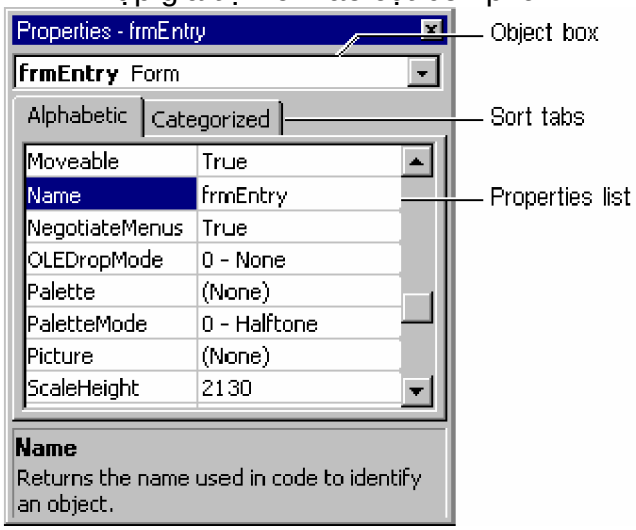
Thay đổi kích cỡ, di chuyển, khoá 1 control :

- Để thay đổi kích cỡ: nhấn chọn control, đặt chuột tới góc (phải, dưới) và kéo chuột tới kích cỡ mong muốn.
- Để thay đổi vị trí: kéo control tới vị trí bằng chuột, rồi thả chuột.
- Để khoá cố định vị trí: sử dụng chức năng *Format* | *Lock Controls*



IV.2, Đặt giá trị thuộc tính cho điều khiển :

- Nhấn chọn control
- Hiển thị cửa sổ thuộc tính: *View* | *Properties*
- Chọn thuộc tính cần đặt từ : *Properites List*
- Nhập giá trị mới vào cột bên phải

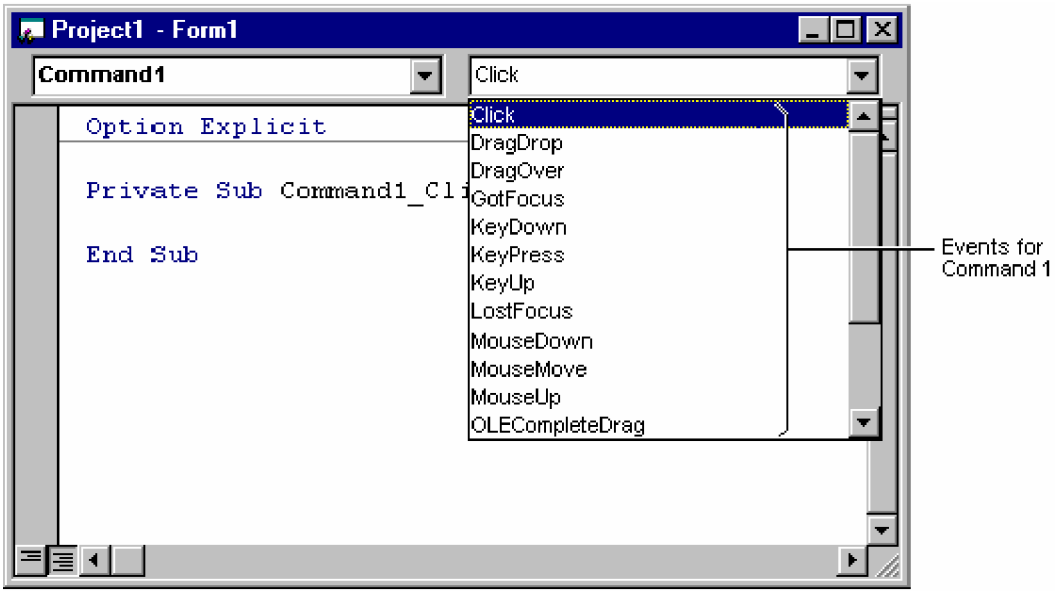


Ghi chú:

- Object box: Danh sách form và các control trên form
- Sort tabs: Kiểu sắp xếp danh sách các thuộc tính: theo bảng chữ cái (Alphabetic) hay theo phân loại (Categorized)
- Properties list: Danh sách các thuộc tính gắn với đối tượng được chọn (trên form hay từ object box)

IV.3, Viết mã lệnh

Mở Code Editor bằng cách: Nhấn đúp lên control hoặc chọn *View Code* từ cửa sổ *Project Explorer*.



Tạo thủ tục đáp ứng sự kiện :
+ Chọn control từ hộp danh sách bên trái (chứa form và các control)
+ Chọn tên sự kiện từ hộp bên phải (chứa danh sách sự kiện gắn với đối tượng vừa được chọn)

```
Private Sub Command1_Click ()
        Text1.Text = "Hello, world!"
End Sub
```

Để chuyển đổi giữa hiển thị tất các thủ tục trên cùng 1 cửa sổ và hiển thị mỗi thủ tục tại 1 thời điểm:
+ Chọn *Tools* | *Options*
+ Thay đổi các giá trị tương ứng trong tab: *Editor*

IV.4, Chạy chương trình :

- Chọn *Run* | *Start* hay nhấn *F5* để chạy chương trình
- Chọn *Run* | *End* để dừng chương trình đang chạy
- Chọn *Run* | *Break* hay nhấn *<Ctrl+Break>* để kết thúc chương trình bất thường

Chương II : Form và Control

I, Thuộc tính, sự kiện, phương thức :

Form và control (điều khiển) của Visual Basic là các đối tượng với thuộc tính (property), phương thức (method) và sự kiện (event).

- Thuộc tính : là các đặc điểm quy định đối tượng, tập hợp các thuộc tính của control có thể thiết lập khi thiết kế hay khi chạy chương trình.
- Phương thức : là các hành động mà control có thể thực hiện được

- Sự kiện : là sự đáp ứng tác động gắn với mỗi đối tượng. Khi 1 sự kiện xảy ra với control thì chương trình sẽ tự động xử lý 1 hàm sự kiện (Event_Handle). Ví dụ khi bạn kích vào 1 nút lệnh trên form thì sẽ tự động thực hiện hành động Command_Click

Chúng ta có thể coi FORM như 1 control đặc biệt.Sau đây là một số sự kiện thông dụng trên form :

- **Form_Initialize()** : sự kiện này xảy ra trước nhất và chỉ xảy ra 1 lần duy nhất. Trong quá trình thực hiện chương trình ta đóng mở form nhiều lần thì sự kiện này chỉ xảy ra ở lần mở form đầu tiên
- **Form_Load()** : sự kiện này luôn xảy ra mỗi khi ta gọi mở đến form
- **Form_Activate()** : sau khi form xử lý xong sự kiện Form_Load(), nếu không có gì thay đổi, chương trình sẽ phát sinh ra sự kiện này
- **Form_QueryUnload(Cancel As Integer, UnloadMode As Integer) :** Khi người sử dụng ấn nút X để đóng form thì sẽ tạo ra sự kiện này, tham số UnloadMode cho biết ai hay tác vụ nào đóng form. Do đó ta có thể bắt sự kiện này để hỏi người sử dụng có chắc muốn đóng form hay không, nếu không thì thiết lập tham số **Cancel=1**.

II, Các điều khiển cơ sở :

Các điều khiển trong môi trường Visual Basic được chia thành 2 loại, điều khiển cơ sở và điều khiển mở rộng.

- Điều khiển cơ sở gồm những điều khiển có thể được sử dụng ngay trong bất kỳ màn hình thiết kế nào, đó là những điều khiển mặc định do môi trường cung cấp.
- Điều khiển mở rộng còn gọi là những Component độc lập, ta có thể thêm hoặc bớt những Component khi thiết kế giao diện. Nhờ các điều khiển mở rộng mà các giao diện của ứng dụng trở nên đa dạng và và tiện dụng hơn với người dùng.

Trong phạm vi của bài này, chúng ta sẽ tìm hiểu ý nghĩa và cách sử dụng của các điều khiển cơ sở.

II.1, Các thuộc tính chung của điều khiển :

- Name : Tên của công cụ
- Caption, Text : Chú Thích, Chú Giải, Nội Dung
- BackColor (Màu Nền)
- ForeColor (Màu Chữ)
- Font : Chọn Phông Chữ
- Top (Trên), Left (Trái)
- Height (Chiều Cao), Width (Chiều Ngang)
- Enabled : True : hiện công cụ cho sử dụng
  False : làm mờ chức năng không cho sử dụng
- Visible : True : Hiển thị công cụ
  False : ẩn công cụ không cho nhìn thấy

II.2, Công cụ điều khiển TEXTBOX

Một trong những thành phần quen thuộc đối với những người làm việc trên môi trường Windows đó là TextBox, một điều khiển có dạng hình hộp và có thể nhập văn bản vào nó. Văn bản trong TextBox có thể nhập ở nhiều dòng khác nhau, có hệ thống thanh cuộn để di chuyển qua các dòng, màu của văn bản, … và có nhiều tính chất khác ta sẽ xem xét trong phần này. Các thuộc tính và hành động của TextBox tập trung vào việc xử lý dữ liệu văn bản được nhập vào trong TextBox.

Sau đây là một số các thuộc tính và hành động thông thường của TextBox

Name

Trong môi trường lập trình, định danh để phân biệt các biến, hàm, ... là thành phần không thể thiếu và giá trị của không được trùng trong cùng một phạm vi sử dụng. Khi làm việc với các điều khiển của Visual Basic thì định danh là thành phần có ảnh hưởng nhiều đến cách thức viết chương trình, nó có thể tạo ra mảng các điều khiển.
Định danh của điều khiển sẽ được lưu giữ trong thuộc tính Name của từng điều khiển, không chỉ riêng gì điều khiển TextBox mà nó có ý nghĩa cho tất cả các điều khiển trong Visual Basic. Khi tạo giá trị cho thuộc tính Name phải tuân thủ các qui tắc đặt tên tương tự như tên biến.

Text

Thuộc tính lưu trữ nội dung của điều khiển. Có 2 cách để đưa nội dung vào cho thuộc tính Text của TextBox : trên màn hình giao tiếp, nhập trực tiếp vào vùng nhập liệu của đối tượng TextBox hoặc là sử dụng phép gán trong cửa sổ lệnh
VD : *Text1.Text = "Bạn gán giá trị cho thuộc tính Text của điều khiển TextBox"*

MultiLine

Với giá trị mặc định của các thuộc tính trong điều khiển khi được tạo, ta sẽ có một TextBox chỉ có thể nhập văn bản trên 1 dòng, với số ký tự tối đa là 65.535 ký tự. Khi thiết kế TextBox, chiều dài hiển thị của TextBox sẽ có thể nhỏ hơn số ký tự mà nó có thể lưu trữ, do đó khi văn bản hiển thị quá dài thì sẽ gây khó khăn cho người sử dụng khi làm việc với văn bản.
Ta có thể thiết lập thuộc tính MultiLine của TextBox là True để TextBox có thể hiển thị văn bản ở nhiều dòng và chuỗi văn bản sẽ tự động xuống dòng khi dài hơn chiều dài của TextBox.

Alignment

Nội dung văn bản của TextBox sẽ được canh theo lề bên trái, ở chế độ mặc định. Nếu bạn muốn văn bản của mình hiển thị ở giữa chứ không phải là trái thì thuộc tính Alignment của TextBox sẽ giúp ta thực hiện được điều này.
Thuộc tính Alignment có thể nhận một trong ba giá trị sau:

- 0 – Left Justify : dùng để canh trái
- 1 – Right Justify : dùng để canh phải
- 2 – Center : dùng để canh giữa.

Thuộc tính canh lề của TextBox ảnh hưởng đến toàn bộ khối văn bản có trong TextBox

MaxLength
Như đã trình bày ở trên, số ký tối đa có thể nhập vào TextBox là 65.535 ký tự, tuy nhiên nếu ta muốn giới hạn số ký tự tối đa mà người dùng có thể nhập vào TextBox thì khi đó ta dùng thuộc tính MaxLength của TextBox. Mặc định giá trị của MaxLength bằng 0 có nghĩa là TextBox có thể nhận tối đa số ký tự do chương trình qui định

PasswordChar
Nếu bạn muốn tạo một ô để cho người dùng nhập mật mã vào (các ký tự nhập vào sẽ tự động được hiển thị thành một ký tự khác) thì ta phải sử dụng đến thuộc tính PasswordChar. Thuộc tính này sẽ nhận là một ký tự bất kỳ và chỉ duy nhất một ký tự, khi người dùng nhập nội dung vào TextBox, dữ liệu hiển thị trên TextBox là những ký tự của thuộc tính PasswordChar, nhưng nội dung được nhập trong thuộc tính Text thì không thay đổi.

Locked và Enabled
Nếu bạn định viết một chương trình tính tổng 2 số, trên hộp thoại bạn sử dụng 2 TextBox dùng để nhập 2 số cần tính tổng và một TextBox để chứa kết quả của phép tính. Khi này, TextBox kết quả chỉ nên để cho người dùng xem kết quả và không cho họ thay đổi kết quả tính được. Locked và Enabled là 2 giải pháp bạn có thể chọn để tạo ra những TextBox chỉ xem, không cho chỉnh sửa.
Khi giá trị của Locked của TextBox được thiết lập là True thì TextBox sẽ bị khoá, người dùng có thể đưa con trỏ vào vùng nhập liệu của điều khiển, nhưng sẽ không thực hiện được thao tác nhập trực tiếp lên TextBox. TextBox không nhận Focus khi thuộc tính Enabled của nó là False. Người sử dụng sẽ không thể nhập dữ liệu vào TextBox, nội dung của văn bản trong TextBox sẽ bị mờ. Đây là điểm khác biệt giữa thuộc tính Locked và Enabled. Người sử dụng sẽ không nhận biết được một TextBox có Locked hay không cho đến khi họ thực hiện thao tác nhập liệu cho TextBox, nhưng họ có thể dễ dàng biết được TextBox nào không thể nhập liệu khi nó bị làm mờ bằng thuộc tính Enabled.
Khi ta sử dụng một trong hai thuộc tính trên để vô hiệu hoá các TextBox thì nội dung của chúng sẽ không thể thay đổi trực tiếp nhưng vẫn có thể thay đổi bằng lệnh của chương trình.
Ví dụ : *Text1.Text = "Điều khiển bị khoá"*

SetFocus
Focus là vị trí của con trỏ xuất hiện trên điều khiển. Khi con trỏ nằm ở điều khiển nào thì điều khiển đó được gọi là đang nhận Focus. Trong khi chương trình đang chạy, ta có thể đặt con trỏ vào bất kỳ điều khiển nào có trên hộp thoại làm việc, tất nhiên điều khiển đó phải có khả năng nhận Focus. Việc đặt Focus vào một điều khiển có thể thực hiện bằng chuột, bàn phím, và có thể sử dụng lệnh.
Hành động SetFocus dùng để lấy Focus về cho điều khiển khi cần, hành động này gần như có ở tất cả các điều khiển trong môi trường của Visual Basic. Ví dụ sau đây sẽ chuyển Focus đến cho điều khiển có tên Text1
Ví dụ : *Text1.SetFocus*

Sau đây là một vài sự kiện thường gặp khi làm việc với TextBox

Change
Sự kiện này sẽ phát sinh khi trong vùng nhập liệu của điều khiển có bất kỳ sự thay đổi nào

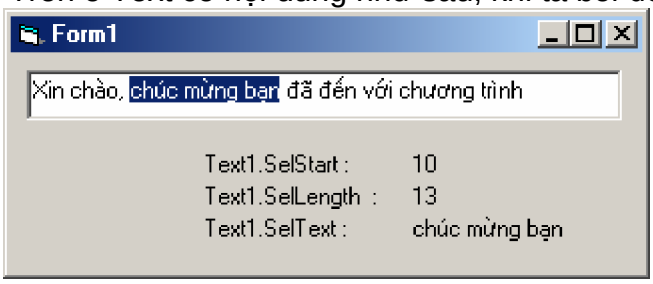
Validate
Sau khi nhập nội dung cho TextBox, bạn có thể kiểm tra giá trị nhập có hợp lệ hay không. Nếu giá trị không hợp lệ ta có thể yêu cầu nhập lại giá trị phù hợp.
Sự kiện Validate sẽ phát sinh trước khi ta chuyển Focus sang điều khiển khác, ta có thể gán giá trị True cho tham số Cancel của thủ tục xử lý sự kiện để ngăn không cho Focus di chuyển sang điều khiển khác

Sau đây là một vài xử lý cơ bản trên TextBox

Chọn và thay thế nội dung trong TextBox
Ta có thể thực hiện thao tác chọn nội dung có trong TextBox bằng chuột, bàn phím giống với bất kỳ môi trường soạn thảo văn bản khác. Bên cạnh đó, ta còn có thể thực hiện thao tác chọn nội dung bằng lệnh thông qua của thuộc tính của điều khiển: SelLength, SelStart, SelText

- **SelLength**: trả về hay thiết lập chiều dài của số ký tự được chọn trên TextBox
- **SelStart**: trả về hay thiết lập vị trí bắt đầu của nội dung được chọn, nếu không có ký tự nào được chọn thì SelStart chính là vị trí mà nội dung có trong thuộc tính SelText sẽ được chèn vào
- **SelText**: trả về chuỗi đang được chọn hay gán chuỗi nội dung vào thuộc tính Text tại vị trí SelStart

Ví dụ : Trên ô Text có nội dung như sau, khi ta bôi đen 1 đoạn chữ thì cho kết quả :



Điều khiển việc nhập liệu vào TextBox

Trong quá trình nhập liệu trên TextBox, người sử dụng có thể vô tình nhập sai kiểu dữ liệu, chẳng hạn như nhập chữ cái vào vùng của số điện thoại. Khi đó, chương trình phải phát sinh ra lỗi về kiểu dữ liệu mà người dùng không biết. Để hạn chế việc có thể nhập sai dữ liệu ta có thể hạn chế bằng cách dùng các sự kiện về bàn phím để kiểm tra dữ liệu khi họ nhập.

Sự kiện KeyPress, KeyUp, KeyDown thường được dùng để giới hạn việc nhập liệu cho người dùng. Trong đó, KeyPress sử dụng để kiểm tra những ký tự bình thường, không có các phím tổ hợp.

Ví dụ sau đây sẽ giới hạn chỉ cho phép người dùng nhập số, sử dụng sự kiện KeyPress của điều khiển TextBox Text1

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
   If (KeyAscii < Asc("0") Or KeyAscii > Asc("9")) And KeyAscii <> 8 Then
      KeyAscii = 0
   End If
End Sub
```

II.3, Command Button, CheckBox và Option Group :

Trong khi giao tiếp giữa ứng dụng và người dùng, ngoài việc tạo những vùng nhập liệu còn một thao tác quan trọng không thể thiếu đó là ghi nhận lại những chọn lựa của họ. Cách ghi nhận các chọn lựa của người dùng một cách trực quan nhất thông qua hệ thống các nút bấm.

Có 3 loại nút trong môi trường của Visual Basic, đó là Command Button, CheckBox, Option Button.

- Command Button: là những nút bấm có dạng hình chữ nhật.
- CheckBox: trạng thái thông thường của điều khiển là một ô vuông. Click vào checkbox để chọn nó, click nó một lần nữa để bỏ chọn.
- Option Button: có trạng thái là chọn và không chọn giống như checkbox, nhưng nó ở dạng hình tròn. Option Button thường phải sử dụng trong một nhóm và đây cũng là điểm khác biệt giữa CheckBox và Option Button: CheckBox có thể làm việc một mình nhưng Option Button thì phải làm việc theo nhóm. Khi ta click chọn một option button thì tất cả những option button khác có trong nhóm sẽ tự động bỏ chọn

Các thuộc tính, hành động và sự kiện của điều khiển

Các thuộc tính của điều khiển có thể được thiết lập giá trị bằng cửa sổ thuộc tính của điều khiển khi thiết kế, hay giá trị được gán khi chương trình đang chạy (run time). Bên cạnh đó, có những thuộc tính chỉ nhận giá trị lúc chương trình đang thực thi.

Caption

Các điều khiển khi được tạo đều mang một ý nghĩa nhất định, Caption chính là thuộc tính để người sử dụng có thể biết được ý nghĩa sử dụng của nút điều khiển. Thuộc tính này có thể thay đổi lúc thiết kế hay lúc chương trình đang chạy.

Trong môi trường Windows, ta có thể di chuyển đến một điều khiển nhanh bằng phím tắt nếu có. Phím tắt là sự kết hợp của phím Alt và một ký tự bất kỳ. Ta có thể tạo phím tắt để di chuyển đến các nút điều khiển bằng cách thêm dấu & trước ký tự cần tạo phím kết hợp ở thuộc tính Caption.
Ví dụ : Tạo phím tắt Atl + T để di chuyển đến nút điều khiển có Caption Thực hiện Caption: &Thực hiện à Thực hiện

TabIndex
Ta có thể di chuyển qua lại giữa các điều khiển theo một thứ tự tuần tự thông qua thuộc tính TabIndex. Nhưng trước tiên ta phải thiết lập thuộc tính TabStop của điều khiển là True, cho phép điều khiển nhận Focus khi người dùng tab đến nó.
TabIndex có giá trị nhỏ nhất là 0, chỉ ra vị trí hiện hành của điều khiển so với thứ tự Tab

Enabled
Khác với TextBox, các điều khiển nút chỉ có thuộc tính mờ, không có thuộc tính Locked. Khi giá trị của thuộc tính Enabled là False thì nút điều khiển sẽ bị mờ và người sử dụng không thể tương tác với nó.

Visible
Một kỹ thuật khi thiết kế thành phần giao tiếp là cho ẩn những điều khiển chưa cần sử dụng nhằm tạo cho hộp thoại giao tiếp có vẽ thoáng và tiết kiệm được không gian lưu trữ. Khi cần sử dụng thì sẽ cho xuất hiện điều khiển cần thiết.
Giá trị của Visible bằng False sẽ làm cho các điều khiển không hiển thị khi chương trình đang chạy. Ta có thể thay đổi giá trị của thuộc tính khi thiết kế cũng như khi thực thi.

Click
Đây là sự kiện xử lý chính của các nút điều khiển. Sự kiện sẽ phát sinh khi người dùng thực hiện thao tác click chuột lên điều khiển

Một vài xử lý cơ bản trên các nút điều khiển

Xác định trạng thái của điều khiển CheckBox
Khi trên màn hình giao tiếp có nhiều CheckBox khác nhau thì lúc xử lý ta phải xác định được điều khiển nào được chọn. Thuộc tính Value của điều khiển sẽ giúp ta xác định được trạng thái chính xác của từng CheckBox, Value có thể chứa 1 trong 3 giá trị sau: 0 là không có chọn, 1 thì có chọn và 2 là mờ

Ví dụ :

```
If Check1.Value = 1 then
        Check1.Caption  = "Check 1 được chọn"
End if
```

Ta cũng có thể sử dụng thuộc tính Value để thiết lập trạng thái cho CheckBox khi chương trình đang chạy. Ví dụ sau sẽ xác định trạng thái có chọn cho điều khiển Check1
Ví dụ : *Check1.Value = 1*

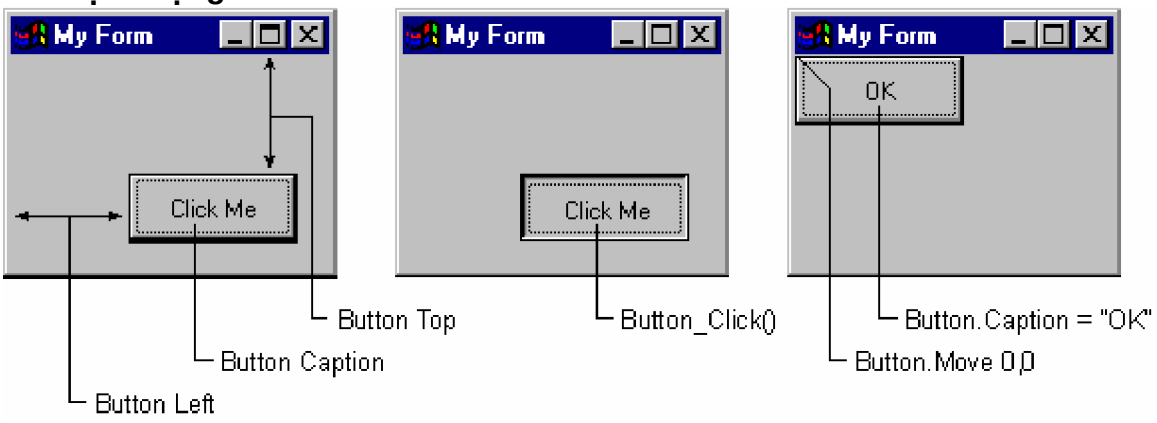
Xác định trạng thái của điều khiển Option Group

Khi làm việc với Option Group ta phải tạo nhóm cho các điều khiển loại trừ nhau. Để biết được trạng thái của từng Option Group, ta phải dựa vào thuộc tính Value. Nếu giá trị của Value là True thì điều khiển được chọn. Tương tự như CheckBox, ta cũng có thể sử dụng thuộc tính này để chọn hay bỏ chọn của Option Group.

Ví dụ :

```
If Option1.Value=True then
        MsgBox "Nút điều khiển đang được chọn"
Else
        MsgBox "Nút điều khiển không được chọn"
End if
```

Xác định trạng thái của điều khiển Command Button



Ví dụ : Bài toán tính tổng dựa trên phép tính được lựa chọn từ OptionBox

Thiết kế giao diện có dạng như sau :


```
Private Sub cmdtinh_Click()
   If Option1.Value Then
      Text3 = Val(Text1) + Val(Text2)
   End If
   If Option2.Value Then
      Text3 = Val(Text1) - Val(Text2)
   End If
   If Option3.Value Then
      Text3 = Val(Text1) * Val(Text2)
   End If
   If Option4.Value Then
      If Val(Text2) <> 0 Then
         Text3 = Val(Text1) / Val(Text2)
      Else
         Text3 = "E"
      End If
   End If
End Sub
```

II.4, List Box và Combo Box

List Box và Combo Box cũng là hai điều khiển cơ sở của môi trường Visual Basic nhưng chúng có nhiều tính năng hỗ trợ người sử dụng trong việc nhập dữ liệu và giao tiếp với chương trình ứng dụng.

List Box có dạng là một danh sách liệt kê các phần tử đã được đưa vào trong khi thiết kế, hay bằng lệnh. Người sử dụng chỉ có thể chọn những phần tử có trong danh sách của List Box. List Box có hỗ trợ thanh cuộn nên ta có thể hiển thị dữ liệu ở nhiều dòng mà không làm tốn nhiều không gian hiển thị.

Combo Box là sự kết hợp của một List Box và một Text Box. Bạn có thể cho phép người sử dụng chọn một phần tử trên List Box hoặc gõ trực tiếp nội dung họ cần vào vùng Text Box.

Các thuộc tính, hành động và sự kiện của điều khiển

Các thuộc tính của điều khiển có thể được thiết lập giá trị bằng cửa sổ thuộc tính của điều khiển khi thiết kế, hay giá trị được gán khi chương trình đang chạy (run time). Bên cạnh đó, có những thuộc tính chỉ nhận giá trị lúc chương trình đang thực thi.

AddItem

Khi làm việc với List Box và Combo Box thì việc đầu tiên cần thực hiện là phải tạo nguồn dữ liệu cho điều khiển. Nguồn dữ liệu có thể được tạo khi thiết kế, hay tạo bằng lệnh lúc thực thi.

Khi thiết kế, ta có thể tạo nguồn dữ liệu cho List Box, Combo Box thông qua thuộc tính List trong cửa sổ thuộc tính của điều khiển.

Khi thực thi, ta có thể dùng hành động AddItem để tạo dữ liệu nguồn cho các điều khiển, có dạng như sau:

VD : *List1.AddItem “Chuỗi nội dung cần thêm vào danh sách”*
Combo1.AddItem “Chuỗi nội dung cần thêm vào danh sách”

List, ListCount, ListIndex

Nguồn dữ liệu sau khi thêm vào cho điều khiển sẽ được chứa trong thuộc tính List của điều khiển. Thuộc tính List được tổ chức là một mảng các phần tử, để truy xuất từng phần trong List ta sử dụng chỉ số của mảng.

Chỉ số của mảng từ 0 đến ListCount-1. ListCount cho biết tổng số phần tử có trong mảng. Khi ta click vào một phần tử trên điều khiển, để lấy giá trị của phần tử đó ta dùng thuộc tính ListIndex và List. ListIndex là chỉ số của phần tử được click.

Ví dụ : cho biết vị trí và nội dung của phần tử được chọn trong List

```
Private Sub List1_Click()
        MsgBox "Phần tử thứ " & List1.ListIndex & " có giá trị : " &  List1(List1.ListIndex)
End Sub
```

ItemData

Khi bạn cần lưu trữ một giá trị đặc biệt cùng với phần tử được thêm vào trong danh sách thì ItemData là cách giải quyết nhẹ nhàng và hiệu quả. ItemData là một mảng các số nguyên, chỉ số của các phần tử trong ItemData ứng với chỉ số của List. Ta có thể sử dụng thuộc tính NewIndex, chỉ số của phần tử mới thêm vào bằng lệnh AddItem, để đồng bộ giữa chỉ số của List với chỉ số của ItemData

Ví dụ :
```
Combo1.AddItem "donTRUONGBT"
Combo1.ItemData(Combo1.NewIndex) = 28121982
```

RemoveItem, Clear

Trong khi thực thi, nếu muốn xoá một phần tử của List Box ta sử dụng hành động RemoveItem. Nếu muốn xoá hết tất cả các phần tử có trong danh sách, dùng hành động Clear

Ví dụ :

List1.RemoveItem 1 à *xoá phần tử ở vị trí thứ 2 trong listbox*

Combo1.Clear à *xoá toàn bộ nội dung trong listbox*

Multiselect, Selected của List Box

Điều khiển List Box cho phép ta chọn nhiều phần tử cùng một lúc. Để có thể chọn nhiều phần tử cùng một lúc, ta phải thiết lập lại giá trị cho thuộc tính Multiselect

0	chỉ cho phép chọn một phần tử
1	dùng chuột hay thanh Space bar để chọn nhiều phần tử
2	nhấn giữ phím Shift và click chuột để chọn nhiều phần tử có trong List

Khi chương trình đang thực thi, để biết được những phần tử nào được chọn ta phải duyệt qua danh sách các phần tử có trong List Box, sau đó dùng thuộc tính Selected để kiểm tra. Thuộc tính Selected cũng được tổ chức ở dạng mảng. Một phần tử được chọn nếu giá trị của Selected bằng True

Ví dụ :

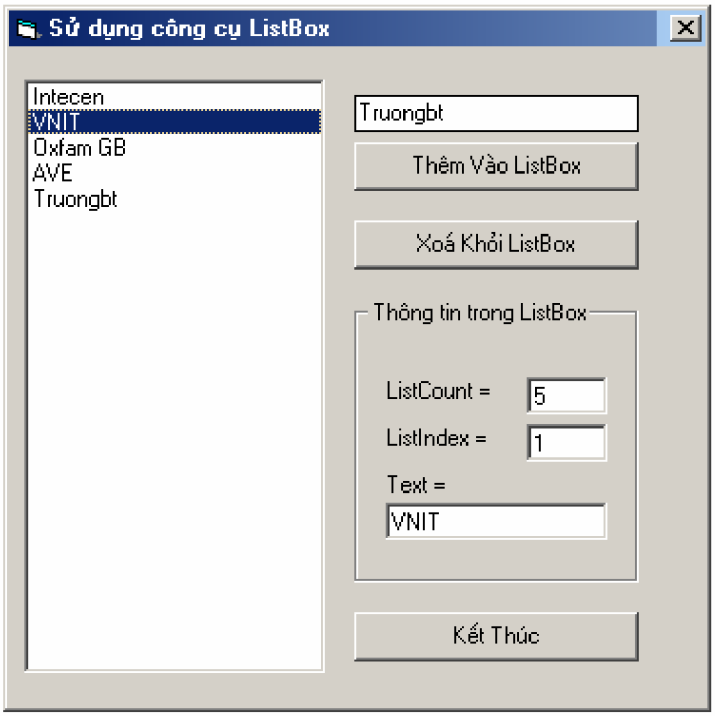
```
For i  = 0 To List1.ListCount -  1
        If List1.Selected( i) = True  Then
                MsgBox "Phần tử thứ  " & i & " được chọn"

        End If
Next
```

Text của Combo Box

Combo Box được tạo thành bởi Text Box và List Box, để truy xuất giá trị của List Box ta dùng thuộc tính List như đã trình bày ở trên. Text Box sẽ chứa giá trị hiện hành của phần tử được chọn, ta có thể truy xuất giá trị này thông qua thuộc tính Text của điều khiển.

Ví dụ : thực hiện các chức năng cơ bản trên ListBox

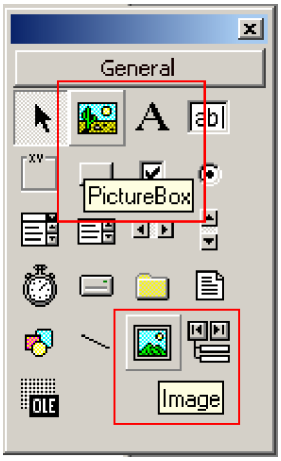
```
Private Sub Form_Load()
    List1.AddItem "Intecen"
    List1.AddItem "VNIT"
    List1.AddItem "Oxfam GB"
    List1.AddItem "AVE"
    List1.ListIndex = 2
End Sub

Private Sub Command1_Click()
    If txtNhap.Text <> "" Then
        List1.AddItem txtNhap.Text
    End If
End Sub

Private Sub Command2_Click()
If List1.ListIndex >= 0 Then
 List1.RemoveItem List1.ListIndex
End If
End Sub

Private Sub List1_Click()
    Text1 = List1.ListCount
    Text2 = List1.ListIndex
    Text3 = List1.Text
End Sub
```



II.5, Picture Box và Image

Image là một điều khiển chỉ dùng để hiển thị hình ảnh lên màn hình giao tiếp của ứng dụng. Trong khi Picture Box có thể hiển thị hình ảnh và thực hiện các thao tác xử lý trên đối tượng ảnh như là lưu ảnh vào đĩa, thực hiện các thao tác vẽ thông qua các hàm đồ họa của hệ thống, …

Ta có thể sử dụng điều khiển Picture Box như là Image nhưng phải quan tâm một điều là điều khiển Image chiếm ít vùng nhớ và có thao tác tự vẽ lại nhanh hơn nhiều so với Picture Box. Picture Box chiếm nhiều vùng nhớ và tốn nhiều thời gian xử lý nhiều hơn Image.



Điểm đặc biệt của điều khiển Picture Box là có thể làm vật chứa có các điều khiển khác và là điều khiển có thể được tạo trực tiếp trên màn hình MDI.

Trong cửa sổ thiết kế, Picture Box và Image được thể hiện dưới dạng biểu tượng đồ hoạ trên thanh Toolbox như hình bên, Picture Box là biểu tượng nằm bên trong hình vuông ở trên, Image là biểu tượng nằm bên trong hình vuông ở dưới.

Các thuộc tính, hành động và sự kiện của điều khiển

Các thuộc tính của điều khiển có thể được thiết lập giá trị bằng cửa sổ thuộc tính của điều khiển khi thiết kế, hay giá trị được gán khi chương trình đang chạy (runtime). Bên cạnh đó, có những thuộc tính chỉ nhận giá trị lúc chương trình đang thực thi.

Picture

Khi bạn cần đưa hình ảnh lên màn hình giao tiếp bạn có thể tạo Image hoặc Picture Box chứa ảnh cần hiển thị. Nội dung của ảnh sẽ được chứa trong thuộc tính Picture của điều khiển. Giá trị của thuộc tính là đường dẫn đến tập tin ảnh cần hiển thị. Giá trị này có thể được thiết lập khi thiết kế hay trong lúc thực thi.

Khi thực thi, ta không thể gán trực tiếp đường dẫn đến tập tin ảnh cho thuộc tính Picture mà phải thông qua một hàm xử lý để chuyển ảnh từ tập tin vào thuộc tính Picture. Hàm xử lý cần sử dụng là LoadPicture, hàm này sẽ nhận tham số vào là đường dẫn đến tập tin ảnh

Ví dụ :

Picture1.Picture = LoadPicture("C:\ Sicily.bmp")
Image1.Picture = LoadPicture("C:\ Omertà.bmp")

Stretch của Image và AutoSize của Picture Box

Thuộc tính Stretch của Image khi thiết lập là True sẽ làm thay đổi kích thước của của ảnh theo kích thước của điều khiển nhưng không làm thay đổi cấu trúc của ảnh nguồn.

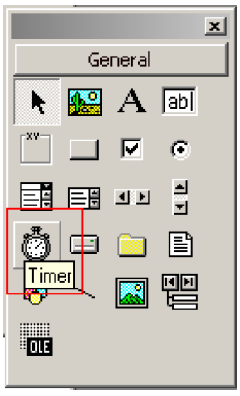
Đối tượng Picture Box không thay đổi kích thước của ảnh nguồn, nhưng nó có thể tự động thay đổi kích thước bằng với kích thước của ảnh thông qua thuộc tính AutoSize với giá trị được gán là True.

II.6, Timer

Ý nghĩa

Trong các những ứng dụng luôn có một sự kiện tự động phát sinh và tồn tại ở bên dưới chỉ có người lập trình mới có thể can thiệp xử lý sự kiện này, đó chính là sự kiện phát sinh theo thời gian. Môi trường phát triển của Visual Basic cung cấp khả năng xử lý sự kiện thời gian thông qua Timer. Điều khiển Timer giúp ta thực thi một đoạn lệnh của chương trình theo chu kỳ thời gian được chỉ định.

Sử dụng



Để làm việc với Timer, trước tiên ta phải thêm điều khiển vào trong chương trình, trên Toolbox điều khiển có dạng như hình bên, và thiết lập thuộc tính Interval của điều khiển. Giá trị của Interval sẽ xác định chu kỳ thời gian phát sự kiện của điều khiển được phát sinh. Đơn vị tính của thuộc tính này là Mili giây (1 phần ngàn của giây). Ví dụ muốn điều khiển Timer sau 1 giây phát sinh ra 1 sự kiện thì giá trị của Interval sẽ bằng 1000.

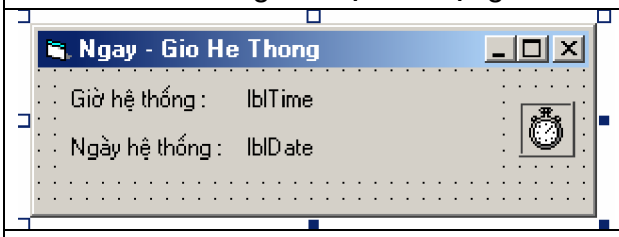
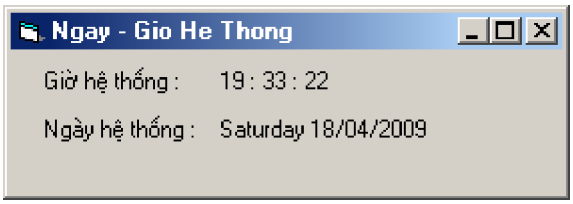
Sau khi điều khiển được thêm vào chương trình, ta có thể kích hoạt hoặc là vô hiệu hoá điều khiển thông qua thuộc tính Enabled. Khi giá trị của Enabled là True thì nó sẽ phát sinh ra sự kiện Timer sau một

chu kỳ thời gian được xác định bởi thuộc tính Interval. Ta sẽ viết các lệnh cần thực hiện trong thủ tục của sự kiện phát sinh, chẳng hạn như là Timer1_Timer().

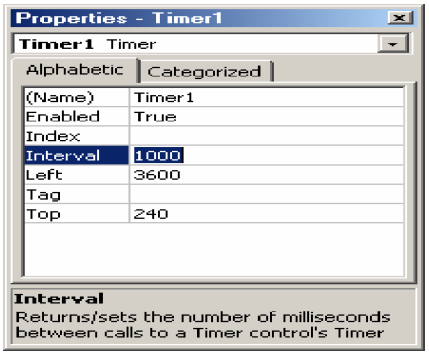
Ví dụ, nếu ta muốn sau mỗi 5 giây chương trình hiện một hộp thoại có nội dung "Xin chao cac ban" thì ta sẽ thêm một điều khiển Timer có tên là Timer1 vào chương trình với các thuộc tính sau: Interval = 5000, Enabled = True. Các thuộc tính này đều có thể được thiết lập giá trị trong lúc thực thi. Khi đó, ta sẽ viết lệnh xử lý trong sự kiện của điều khiển Timer như sau:

```
Private Sub Timer1_Timer()
        MsgBox "Xin chao cac  ban"
End Sub
```

Ví dụ đưa ra ngày giờ của hệ thống :

Thiết kế giao diện có dạng	Kết quả khi thực hiện chương trình
	

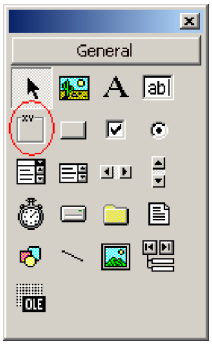
Thiết lập thông tin cho Timer và đoạn mã lệnh thực hiện




```
Private Sub Timer1_Timer()
    lblTime.Caption = Format(Time, "hh : mm : ss")
    lblDate.Caption = Format(Date, "dddd dd/mm/yyyy")
End Sub
```

II.7, Frame, Label, Shape và Line

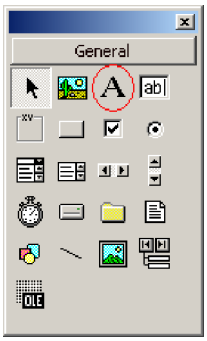
Frame



Điều khiển Frame thường được dùng để nhóm các điều khiển trên màn hình thành một nhóm có ý nghĩa sử dụng như nhau. Frame xuất hiện trên màn hình như là một hình hộp, có chứa điều khiển Lable ở góc trên bên trái của đối tượng.
Ta có thể làm cho các điều khiển có trong Frame trở thành một chức năng duy nhất, chẳng hạn như nhóm các Option Button thành một nhóm. Khi đó, nếu ta chọn một điều khiển Otion Button bất kỳ trong nhóm thì điề u khiển khác trong nhóm sẽ tự động bỏ chọn.

Để tạo nhóm cho các điều khiển ta phải tạo ra điều khiển Frame trước tiên, sau đó vẽ điều khiển của nhóm cần tạo trong phạm vi Frame chứa nhóm. Nếu một điều khiển được vẽ bên ngoài Frame hay đã được tạo trước đó thì sẽ không thuộc về nhóm của các điều khiển khác có trong Frame đang tạo. Ta có thể ghi chú ý nghĩa của nhóm thông qua thuộc tính Caption của Frame.

Label

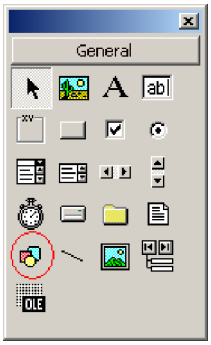


Bạn có thể sử dụng điều khiển Label để hiển thị những văn bản mà không có phép người sử dụng thay đổi giá trị trực tiếp khi chương trình đang chạy. Nhưng ta vẫn có thể thay đổi nội dung của Lable lúc thực thi thông qua thuộc tính Caption của nó. Ngoài ra Label còn dùng để hiển thị tiêu đề cho những điều khiển không có thuộc tính biểu diễn tiêu đề riêng, như TextBox, Image…

Nhìn chung thì điều khiển Label có thể hiện thị nội dung văn bản tương tự như một TextBox nhưng điểm khác biệt cơ bản là Label không cho phép nhập nội dung trực tiếp lên nó. Ngoài ra Label có thể thực hiện các thao tác định dạng giống với TextBox về canh lề, font chữ..

Tuy Label không thể nhận Focus nhưng nó vẫn có khả năng nhận sự kiện Click, DlbClick và xử lý được việc di chuyển đến điều khiển bằng tổ hợp phím tắt (Alt + phím) đến điều khiển được phép nhận Focus có TabIndex lớn hơn và mang giá trị gần nó nhất

Shape



Shape là một điều khiển gồm các đối tượng đồ hoạ. Ta có thể tạo các hình như hình tròn, chữ nhật, vuông, oval và thực hiện các thao tác tô màu nền, màu viền, kiểu tô cho hình thông qua điều khiển này

Việc tạo ra điều khiển Shape được thực hiện chủ yếu trong giai đoạn thiết kế, tuy nhiên ta có thể thay đổi hình dạng, màu sắc của điều khiển khi thực thi

Điều khiển Shape có thể được hiển thị ở một số hình khác nhau thông qua thuộc tính Shape của điều khiển. Thuộc tính Shape có thể nhận một trong những giá trị từ 0 đến 5 tương ứng với hình chữ nhật, hình vuộng, hình oval, hình tròn, hình chữ nhật có góc tròn, hình vuông có góc tròn

Ta có thể thiết lập màu tô, màu nền, kiểu tô cho điều khiển thông qua các thuộc tính FillColor, BackColor, FilStyle,

Line

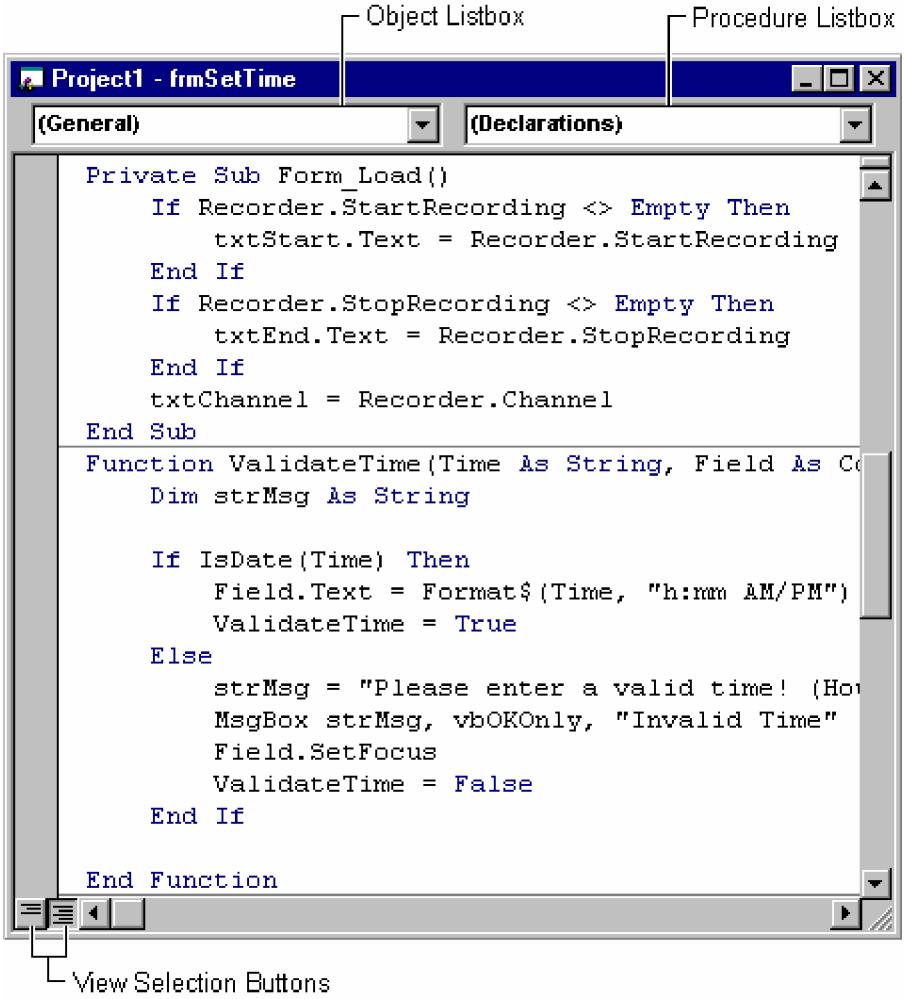
Giống với điều khiển Shape, Line cũng là một điều khiển thuộc về hình học. Ta có thể sử dụng điều khiển này để vẽ những đường ngang, đường dọc tại bất kỳ vị trí nào trên màn hình giao tiếp.

Ta có thể thiết lập các kiểu đường khác nhau cho điều khiển Line thông qua thuộc tính BorderStyle với tập giá trị từ 0 đến 6 tương ứng với các kiểu đường khác nhau.

Chương III : Ngôn Ngữ Lập Trình Visual Basic

I, Sử dụng Code Editor :

Code Editor của Visual Basic là một cửa sổ nơi bạn viết phần lớn mã chương trình. Nó giống một trình soạn thảo được chuyên hoá cao với nhiều tính năng tạo thuận lợi cho việc viết mã Visual Basic.



Một cửa sổ riêng được mở cho mỗi module bạn chọn từ Project Explorer. Mã nằm trong mỗi module được chia thành các phần riêng cho từng đối tượng trong module.

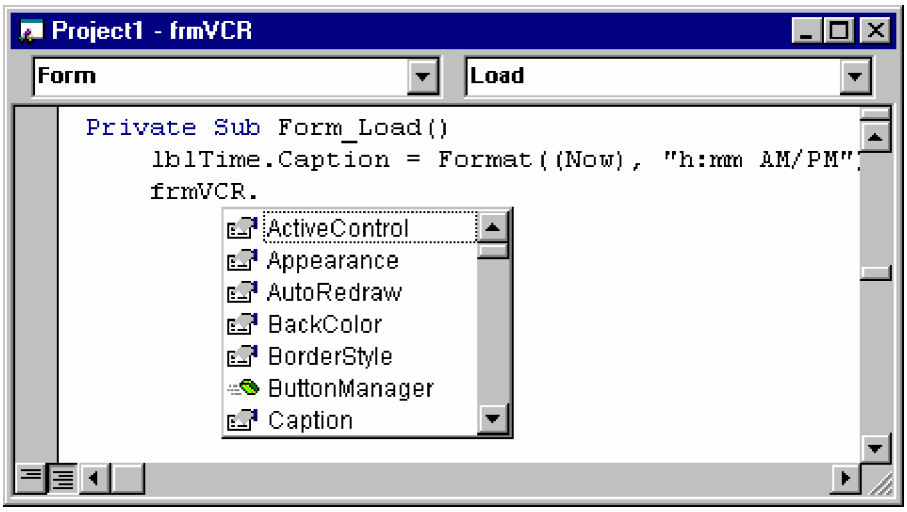
Mỗi module form gồm:

Phần **general**: chứa phần khai báo và các thủ tục dùng chung.

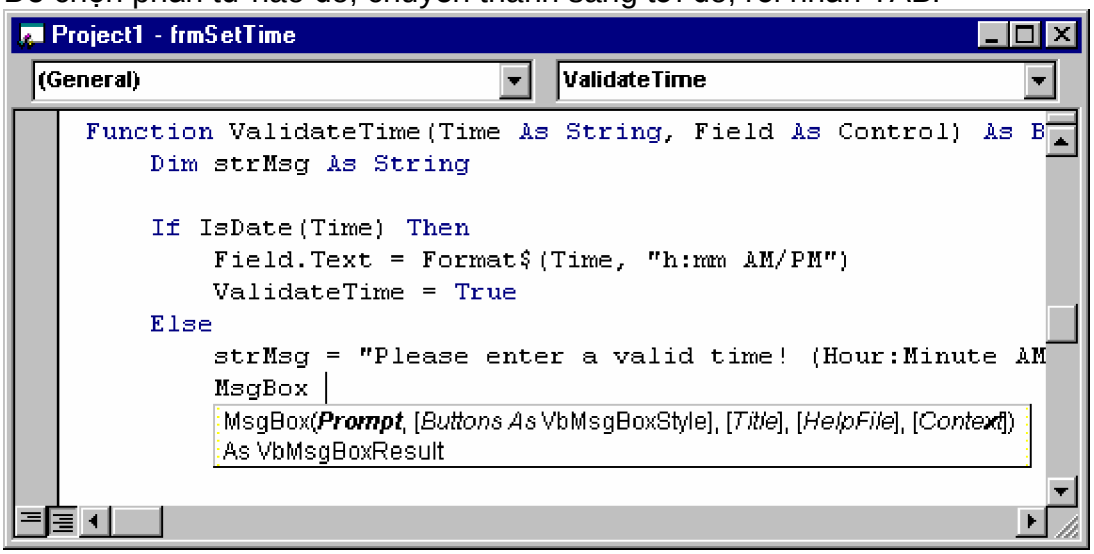
Phần **declarations**: chứa thủ tục sự kiện cho từng đối tượng.

Để truy nhập tới phần mã lệnh cần viết, nên sử dụng hộp combo box trái rồi hộp combo box phải.

Visual Basic có chức năng Auto Code Completion giúp viết mã lệnh trong Code Editor được dễ dàng. (gọi bằng Ctrl + SpaceBar)



Để chọn phần từ nào đó, chuyển thanh sáng tới đó, rồi nhấn TAB.



Ngoài ra, chức năng Bookmarks giúp đánh dấu 1 vị trí mã lệnh để có thể quay lại được nhanh chóng (Edit | Bookmarks).

II, Qui ước viết lệnh trong Visual Basic

II.1, Chia 1 lệnh thành nhiều dòng : Bằng cách sử dụng ký tự: [_]

```
adoJT.RecordSource = _
        "SELECT * FROM Bang1, Bang2" _
        & "WHERE Bang1.ID = Bang2.ID" _
        & "AND Bang1.NickName = 'Truongbt' "
```

Trong một số trường hợp cách chia 1 câu lệnh này không thực hiện được : nối các dòng chú giải, ...

II.2, Nối nhiều lệnh vào 1 dòng : Bằng cách sử dụng kí tự: [:]

```
txtMsg.Text = "Hello" : Red = 255 : txtMsg.BackColor = Red
```

II.3, Thêm chú giải vào mã lệnh : Bằng cách sử dụng kí tự: [']

```
' đây là dòng chú thích cho lệnh đưa chuỗi "Hi!" vào ô textbox.
txtMsg.Text = "Hi!"   ' chú thích trên cùng dòng lệnh
```

II.4, Hệ thống số

Cơ số 10 (Decimal)	Cơ số 8 (Octal)	Cơ số 16 (Hexadecimal)
9	&O11	&H9
15	&O17	&HF
16	&O20	&H10
20	&O24	&H14
255	&O377	&HFF

II.5, Qui tắc đặt tên : Tên thủ tục, hàm, biến, hằng phải tuân theo qui tắc:

- Bắt đầu bằng kí tự chữ
- Không chứa ký tự phân cách (như [.] hay kí tự rỗng), ký tự khai báo kiểu.
- Không dài hơn 255 kí tự. Tên của control, forms, class, và module không vượt quá 40 kí tự.
- Không trùng với các từ khoá của ngôn ngữ.

Bảng quy cách đặt tên cho các điều khiển (theo thông thường)

Điều khiển	Tên	Quy cách	Điều khiển	Tên	Quy cách
	Label	lbl		Image	img
	TextBox	txt		Adodc	ado
	Frame	fra		DataGrid	dg – grd
	Command	cmd		DataList	dlst
	Checkbox	chk		DataCombo	dcbo
	OptionBox	opt		TreeView	tvw
	ComboBox	cbo		ListView	lvw
	ListBox	lst		RichTextBox	rtb
	Timer	tmr		Adodc	ado
	PictureBox	pic		DataGrid	dg – grd

Bảng quy cách đặt tên cho các kiểu dữ liệu (theo thông thường)

Kiểu dữ liệu	Tên	Byte	Phạm vi	Quy cách
Số nguyên	Integer	2	-32.768 đến 32.767	int
	Long	4	-2,147,483,648 đến 2,147,483,647	lng
Số thực	Single	4	-3.402823E38 đến - 1.401298E -45 với các trị âm	
	Double	8	-1.79769313486231E308 đến -4.94065645841247E-324 cho trị âm 4.94065645841247E-324 đến 1.79769313486232E308 cho trị dương	dbl
	Byte	1	0 – 255	
Chuỗi	String	(1byte/1ký tự). Khoảng 2 tỷ ký tự		str
	Char	1		chr
Logic	Boolean	2	True hoặc False	
Ngày giờ	Date	8	1/1/100 đến 31/12/9999 00:00:00 đến 23:59:59	
Mặc định	Variant			

II.6, Một số khái niệm :

Giá trị :
Các giá trị dạng chuỗi (text) nằm trong cặp dấu nháy kép “”. Ví dụ: “Xin chào”. Các giá trị khi đặt trong dấu nháy kép có phân biệt chữ hoa và chữ thường.
Các giá trị số thập phân ghi ở dạng số như bình thường. Ví dụ: 10 hay 100.9 … Các giá trị số thập lục phân bắt đầu bằng &H.

Ví dụ ghi giá trị thập lục phân của 255: &Hff

Có hai giá trị đặc biệt là giá trị rỗng (Empty) và giá trị không xác định (Null). Giá trị rỗng chỉ dùng cho các biến chuỗi hay biến kiểu Variant và được thể hiện bằng “”. Giá trị Null chỉ dùng cho các biến kiểu Variant.
Với các biến kiểu đối tượng, khi chưa được xác định thì giá trị của đối tượng là Nothing.

Biểu thức :
Biểu thức là thành phần của một câu lệnh có thể tính ra giá trị. Biểu thức có thể chứa các biến, hàm và các toán tử (số học, logic, so sánh)

Các toán tử : Visual Basic có các bộ toán tử sau:

- Toán tử so sánh: =, <>, >, <, >=,<=
- Toán tử số học: +, -, *, /, \, ^
- Toán tử gán = và toán tử nối chuỗi &
- Toán tử logic: AND, OR, NOT (Toán tử một ngôi)
- Toán tử bitwise: AND, OR, XOR, NOT

Các từ khoá :
Visual Basic tương tự như các ngôn ngữ lập trình khác sử dụng một bộ từ khoá giúp xác định cú pháp của các câu lệnh. Theo quy ước, tên của các đối tượng lập trình trong chương trình như biến, thủ tục, hàm,… không được trùng với các từ khoá.

- Dim, Private, Public, Friend, Static, As, WithEvents, Declare
- If, Then, Else, End, Select, Case, For, Do, While, Until, Loop, Next, Step, To
- Let, Set, Get, Property Error, GoTo, On, Resume, Option, Explicit, Time, Date, Seek, Len, Mid, Print, Lock
- True, False, Is, Null,Nothing, Empty, New
- ByVal, ByRef, Optional,ParamArray
- Binary, String

III, Biến - Hằng - Kiểu Dữ Liệu

III.1, Biến số :

Biến số là nơi lưu trữ tạm thời giá trị. Biến thường được định nghĩa bởi người sử dụng. Chúng ta có thể xem các thuộc tính của 1 đối tượng như biến.

```
intSoNguyen = 10
intSoNguyen = intSoNguyen + 1
txtKetqua.text = intSoNguyen * 28250
```

Khai báo biến

Cú pháp : **Dim[Public] TênBiến [As KiểuDữLiệu]**
VD : Dim intSO As Integer ‘ khai báo biến số kiểu số nguyên

Biến khai báo với **Dim** trong 1 thủ tục tồn tại khi thủ tục đó được chạy : Khi thủ tục kết thúcthì giá trị của biến đó cũng mất; Tức là biến này có phạm vi địa phương (local) trong thủ tục mà nó được khai báo.

Phạm vi cơ bản của biến:
Biến khai báo trong phần Declarations của 1 module form, standard, hay class, chứ không phải bên trong 1 thủ tục có thể được sử dụng bởi mọi thủ tục trong module.
Biến khai báo với từ khoá **Public** sử dụng được trong toàn chương trình.
Biến khai báo với từ khoá **Static** vẫn lưu tiếp giá trị của nó ngay cả khi thủ tục chứa khai báo của nó đã kết thúc.
Chú ý: Trong 1 phạm vi, không được có 2 biến khai báo trùng tên.

Khai báo không tường minh : Trong Visual Basic, biến có thể không khai báo trước khi sử dụng:

```
Function BinhPhuong(so As Long) As Double
    GiaTri = Abs(so) ‘ ta sử dụng biến GiaTri mà không cần khai báo
    BinhPhuong = Sqr(GiaTri)
End Function
```

Tuy nhiên, điều này dễ dẫn sinh lỗi trong chương trình vì trường hợp trùng biến hoặc trùng hàm.

Khai báo tường minh : Để yêu cầu Visual Basic luôn kiểm tra biến phải được khai báo trước khai sử dụng:
Đặt câu lệnh sau vào phần Declarations của 1 module class, form, hay standard:

Option Explicit

Hay từ menu Tools, chọn Options, nhấn chọn tab Editor rồi đánh dấu chọn Require Variable Declaration. Sau đó, câu lệnh Option Explicit sẽ luôn được chèn tự động.

III.2, Hằng số : Có 2 kiểu hằng số

Hằng số cơ sở hay định nghĩa trước bởi hệ thống
Hằng số do người dùng định nghĩa

Tự khai báo 1 hằng số:

[Public|Private] Const TênHằngSố[As KiểuDL] = GiáTrị

Ví dụ:

```
Const conPi = 3.14159265358979
Public Const conMaxPlanets As Integer = 9
Const conReleaseDate = #1/1/95#
```

Phạm vi hằng số do người dùng định nghĩa :

- Hằng khai báo trong 1 thủ tục có phạm vi chỉ trong thủ tục đó
- Hằng khai báo trong phần Declarations của 1 module có phạm vi trong toàn module.
- Hằng khai báo trong phần Declarations của 1 module với từ khoá Public có phạm vi trong toàn chương trình.

III.3, Kiểu dữ liệu : Khai báo biến với kiểu dữ liệu

Biến thường được khai báo trước với kiểu dữ liệu định sẵn. Ví dụ:

```
Private I As Integer
Dim HTT As Double
Static YourName As String
Public ThapPhan As Currency
Private JT As Integer, HTT As Double
```

Nếu kiểu biến không nêu rõ, biến sẽ có kiểu là **Variant**.

```
Dim v
```

Dữ liệu kiểu số : Integer, Long, Single, Double, Currency

Kiểu Byte : Khi biến số chứa dữ liệu nhị phân, nên khai báo nó thành mảng các dữ liệu kiểu byte. Điều đó giúp giữ được khuôn dạng dữ liệu trong quá trình chuyển đổi.

Kiểu dữ liệu chuỗi : String

Ví dụ:

```
Dim S as String
S = "Database"
S = Left(S, 4)
```

Mặc định, biến chuỗi có độ dài biến đổi phù hợp với giá trị nó lưu trữ. Để khai báo 1 biến chuỗi có độ dài cố định, sử dụng cú pháp: String * size
Ví dụ:

```
Dim EmpName As String * 50
```

Visual Basic có thể tự động chuyển đổi dữ liệu giữa kiểu số và chuỗi. Tuy nhiên, chúng ta phải luôn cẩn thận, bởi khi trao đổi giữa chuỗi và số, lỗi có thể nảy sinh khi chuỗi số chứa giá trị không ở khuôn dạng số.

Kiểu dữ liệu logic (Boolean) : True/False

Kiểu dữ liệu thời gian (Date) : Giá trị về ngày tháng, hay thời gian có thể được lưu trữ trong kiểu dữ liệu Date hay trong Variant.

Kiểu dữ liệu đối tượng : Biến đối tượng được lưu trữ thành địa chỉ 32-bit. Địa chỉ này tham trỏ tới đối tượng trong chương trình, hay tới đối tượng trong chương trình khác.

```
Dim objDb As Object
Set objDb = OpenDatabase("c:\Vb6\Biblio.mdb")
```

Kiểu dữ liệu Variant : Biến kiểu Variant có thể lưu trữ dữ liệu có kiểu bất kỳ. Chúng ta không phải chuyển đổi giữa các kiểu dữ liệu nếu chúng ta gán chúng cho 1 biến kiểu Variant.

Kiểu dữ liệu do người dùng tự định nghĩa (Type) : Chúng ta có thể định nghĩa những kiểu dữ liệu mới, tương tự như khái niệm Record trong Pascal.

IV, Các cấu trúc điều khiển :

IV.1, Các lệnh rẽ nhánh :

A, If … Then … Else

Trước khi đi vào cấu trúc đầy đủ, ta hãy tiếp cận trước qua cấu trúc **If … End If**
Cú Pháp :

```
If <biểu thức điều kiện> Then
        Các câu lệnh
End If
```

Mô Tả :

- Sử dụng cú pháp này, người lập trình muốn khai báo với trình biên dịch rằng : các câu lệnh nằm trong vùng If chỉ được thực hiện nếu như điều kiện chỉ ra trong mệnh đề If là đúng (True)
- Điều kiện trong mệnh đề If là một biểu thức trả về giá tị True/False hoặc một giá trị số. Khi đó các giá trị số khác 0 là True, ngược lại là False

Ví dụ :

```
'Kiểm tra xem i có phải số chẵn không?
    If i Mod 2 = 0 Then
        MsgBox i & " là số chẵn"
```

```
    End If

'Kiểm tra xem bạn đã nhập họ tên hay chưa?
    Str = Input("Nhập vào tên của bạn: ")
    If Str = "" Then
        MsgBox "Bạn chưa nhập vào tên của mình."
    End If
```

Cấu trúc If … Then … End If còn thiếu sót vì đôi khi ta muốn thực hiện các lệnh này nếu điều kiện đúng nhưng nếu điều kiện sai thì thực hiện các lệnh kia. Khi đó, nếu sử dụng cấu trúc **If … Then … End If** ta sẽ viết như sau:

```
    If điều_kiện Then
        Các lệnh sẽ thực hiện nếu điều kiện đúng
    End If
    If Not điều_kiện Then
        Các lệnh sẽ thực hiện nếu điều kiện sai
    End If
```

Để thay thế cách viết trên, Visual Basic (và hầu hết các ngôn ngữ khác) cung cấp cấu trúc **If … Then … Else … End If**

Cú Pháp :

```
    If <biểu thức điều kiện> Then
        Các câu lệnh khi btđk đúng
    Else
        Các câu lệnh khi btđk sai
    End If
```

Mô Tả :

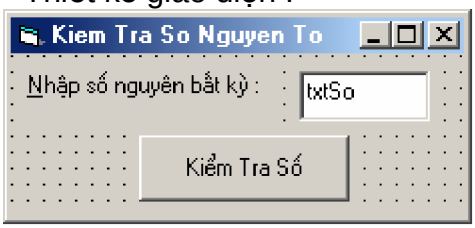
- Điều kiện trong mệnh đề If là một biểu thức trả về giá tị True/False hoặc một giá trị số. Khi đó các giá trị số khác 0 là True, ngược lại là False

Ví dụ :

```
'Kiểm tra xem i có phải số chẵn không?
    If i Mod 2 = 0 Then
        MsgBox i & " là số chẵn"
    Else
        MsgBox i & " là số lẻ"
    End If
'Kiểm tra xem bạn đã nhập họ tên hay chưa?
    Str = Input("Nhập vào tên của bạn: ")
    If Str = "" Then
        MsgBox "Bạn chưa nhập vào tên của mình."
    Else
        MsgBox "Chào bạn, " & Str
    End If
```

Ví dụ : Kiểm tra số nhập có phải số nguyên tố hay không?

- Thiết kế giao diện :



- Code :

```
Private Sub cmdKiemTra_Click()
    Dim kt As Boolean, so As Long, i As Long
    kt = True
    so = Val(txtSo)
    For i = 2 To so / 2
        If so Mod i = 0 Then kt = False
    Next
    If kt = True Then
        MsgBox so & " là số nguyên tố"
    Else
        MsgBox so & " không phải số nguyên tố"
    End If
End Sub
```

B, Select … Case

Cấu trúc If … Then … Else hạn chế giá trị trả về của biểu thức điều kiện là True/False hoặc là giá trị số và thường chỉ dùng khi số trường hợp cần xét là ít. Trường hợp số trường hợp cần xét nhiều, người lập trình có thể sử dụng cú pháp **Select … Case**

Cú pháp :

```
Select Case<Biểu thức>
    Case <Giá trị 1>
        Các câu lệnh thực hiện khi giá trị của biểu thức bằng Giá trị 1
    Case <Giá trị 2>
        Các câu lệnh thực hiện khi giá trị của biểu thức bằng Giá trị 2
    ...
    Case Else
        Các câu lệnh thực hiện khi giá trị của biểu thức không bằng các giá trị đã
    xét ở trên
End Select
```

Mô tả :

Khối lệnh Case Else có thể không cần viết. Tuy nhiên, người lập trình được khuyên dùng khối lệnh này trong cấu trúc Select … Case để giảm bớt các lỗi logic.
Các giá trị dùng đề so sánh có thể gồm nhiều giá trị phân biệt bởi dấu phẩy (,)
hoặc là một phần của biểu thức so sánh.

Ví dụ :

```
Select Case thang
        Case 1,3,5,7,8,10,12
                MsgBox “Thang nay co 31 ngay.”
        Case 4,6,9,11
                MsgBox “Thang nay co 30 ngay.”
        Case 2
                If (nam mod  400  = 0) and _
                ( nam mod 4 = 0 and nam mod 100 <> 0)  Then
                        MsgBox “Thang nay co 29 ngay.”
                Else
                        MsgBox “Thang nay co 28 ngay.”
                End If
        Case Else
                MsgBox “Bạn nhập tháng không tồn tại.”
End Select

Select Case DiemTB
        Case Is < 5
                xeploai = “Yếu”
        Case Is <= 6.5
                xeploai = “ Trung bình  yếu”
        Case Is <= 7
                xeploai = “Trung bình”
        Case Is <= 8
                xeploai = “Khá”
        Case Is <= 9
                xeploai = “Giỏi”
        Case Else
                xeploai = “Xuất sắc”
End Select
```

IV.2, Cú pháp vòng lặp :

Cấu trúc lặp cho phép thực hiện lặp đi lặp lại nhiều lần một tập hợp lệnh nào đó của chương trình. Các cấu trúc lệnh lặp thường được sử dụng trong Visual Basic gồm:

A, For … Next

Cú pháp :

```
For < biến đếm >  = < giá trị đầu>   To  <giá t rị cuối>  [Step <bước> ]
        Các câu lệnh của vòng lặp For
Next [biến_đếm]
```

Mô tả :

- Sử dụng cú pháp này, người lập trình muốn khai báo với trình biên dịch rằng: các câu lệnh đặt trong vùng For … Next chỉ được thực hiện nếu như < biến đếm > có giá trị trong đoạn [<giá trị đầu> , <giá trị cuối>]

- Sau mỗi lần thực hiện các lệnh trong vùng For … Next ,< biến đếm > sẽ tự động tăng thêm <bước>. Nếu không chỉ ra <bước> có giá trị là 1
- Nếu <bước> có trị lớn hơn 0, cấu trúc lặp chỉ thực hiện khi <giá trị đầu> <= <giá trị cuối>
- Nếu <bước> có trị nhỏ hơn 0, cấu trúc lặp chỉ thực hiện khi <giá trị đầu> >= <giá trị cuối>

Ví dụ :

```
Dim i As Long, tong As Long
tong  = 0
For i= 1 To 10
        tong = tong + i
Next
MsgBox " Tong cac so nguyen trong khoang [1-10] co gia tri = " & tong

' cấu trúc lặp sau sẽ không thực hiện
For i= 1 To 10 Step -1
        tong = tong + i
Next
```

B, For Each … Next

Cú pháp :

```
For Each <phần tử>  In  < tập hợp>
        Các câu lệnh của vòng lặp For
Next [ phần tử ]
```

Mô tả :

Với cú pháp này, chương trình sẽ duyệt qua từng phần tử trong tập hợp các phần tử đang duyệt.

Phải khai báo biến <phần tử> là kiểu của phần tử trong tập hợp đang duyệt

Ví dụ :

- Duyệt các form đang mở trong ứng dụng và hiển thị tên của form

```
Dim frm  As Form
For each frm In Forms
        MsgBox frm.Name
Next
```

- Lấy ra toàn bộ font của hệ thống và đưa vào listbox

```
Sub ShowFonts()
        Dim i As Long
        lstFont.Clear
        For i = 1 To Screen.FontCount - 1
                lstFont.AddItem Screen.Fonts(i)
        Next
End Sub
```

Chúng ta có thể chấm dứt lặp khi đang ở giữa chừng vòng lặp bằng lệnh **Exit For**

C, Do While … Loop hoặc While … Wend

Cú pháp :

```
Do While <biểu thức logic>
        Các lệnh
Loop

While <biểu thức logic>
        Các lệnh
Wend
```

Mô tả :

- Sử dụng cú pháp này, người lập trình muốn khai báo với trình biên dịch rằng: các câu lệnh đặt trong vùng lặp chỉ được thực hiện khi < biểu thức logic > có giá trị True
- Sau mỗi lần thực hiện các lệnh trong vùng lặp, < biểu thức logic > sẽ được kiểm tra lại :
    - Nếu có trị True, thực hiện lại vòng lặp
    - Nếu < biểu thức logic > có trị False, chấm dứt vòng lặp
- Cấu trúc này kiểm tra < biểu thức logic > trước khi thực hiện các lệnh nên có thể không xảy ra lần lặp nào nếu ngay lần đầu tiên < biểu thức logic > có giá trị False

Ví dụ : Tính tổng các số nguyên từ 1 đến 10

```
Dim i As Long, Tong As Long
i = 1
Do While i<= 10
        Tong = Tong + i
        i = i + 1
Loop
MsgBox “Tổng từ 1 đến 10 có giá trị là : “ & Tong
```

D, Do … Loop While

Cú pháp :

```
Do
        Các lệnh
Loop While <biểu thức logic>
```

Mô tả :

- Sử dụng cú pháp này, người lập trình muốn khai báo với trình biên dịch rằng: các câu lệnh đặt trong vùng Do ...Loop While sẽ tiếp tục được thực hiện khi <biểu thức logic > có giá trị True
- Sau mỗi lần thực hiện các lệnh trong vùng Do While ...Loop, < biểu thức logic > sẽ được kiểm tra lại :
    - Nếu có trị True, thực hiện lại vòng lặp

- Nếu < biểu thức logic > có trị False, chấm dứt vòng lặp
- Cấu trúc này kiểm tra < biểu thức logic > sau khi thực hiện các lệnh lần đầu tiên nên lặp xảy ra ít nhất một lần nếu ngay lần đầu tiên < biểu thức logic > có giá trị False

Chúng ta có thể chấm dứt lặp khi đang ở giữa chừng vòng lặp bằng lệnh **Exit Do**

Ví dụ :
Tính tổng các số nguyên từ 1 đến 10

```
Dim i As Long, Tong As Long
i = 0 : Tong = 0
Do
        i = i + 1
        Tong = Tong + i
Loop While (i< 10)
```

Kiểm tra một số M có sẵn có phải là số nguyên tố hay không ?

```
Dim i As Integer
i = 2
Do While i <= (M \2)
        If M mod i = 0 Then Exit Do
        i= i + 1
Loop
If i > M\2 Then
        MsgBox M & " là số nguyên tố"
Else
        MsgBox M & " là không số nguyên tố"
End If
```

V, Hàm - Thủ Tục

Đa số các ngôn ngữ khác coi thủ tục và hàm như nhau, Visual Basic phân biệt rõ ràng thủ tục và hàm bởi đặc điểm hàm có giá trị trả về còn thủ tục thì không

V.1, Thủ Tục :

Khai báo thủ tục

Cú pháp

```
[Private|Public][Static] Sub TênThủTục (tham số)
        Các lệnh trong thủ tục
End Sub
```

Mô tả :

- Từ khoá Public chỉ ra thủ tục có phạm vi sử dụng toàn cục trong chương trình
- Từ khoá Private chỉ ra thủ tục chỉ có thể sử dụng trong phạm vi của module khaibáo
- Nếu không chỉ rõ Public hay Private thì mặc định Visual Basic sử dụng Public
- Các tham số có thể có hay không tuỳ theo chức năng của thủ tục

Ví dụ :

```
Sub MoCSDL()
```

Thủ tục xử lý sự kiện

Để xử lý các sự kiện xảy ra cho một đối tượng, Visual Basic 6.0 sử dụng thủ tục xử lý sự kiện để cung cấp cho người lập trình nơi viết lệnh đáp ứng lại sự kiện.

Cấu trúc của thủ tục xử lý sự kiện:

```
Private Sub Tên_đối_tượng_Tên_sự_kiện(các tham số về thông tin của sự kiện)
    Các lệnh trong thủ tục
End Sub
```

Ví dụ : Khi người dùng click lên một nút lệnh có tên là cmdThoat

```
Private Sub cmdThoat_Click()
    End
End Sub
```

Khi người dùng di chuyển chuột trên form thì hiện vị trí chuột lên TitleBar

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Me.Caption = “X: “ & X & “ Y:” & Y
End Sub
```

V.2, Hàm :

Cú pháp :

```
[Private|Public][Static] Functiion TênHàm (tham số) As KiểuDữLiệu
    Các lệnh trong hàm
End Function
```

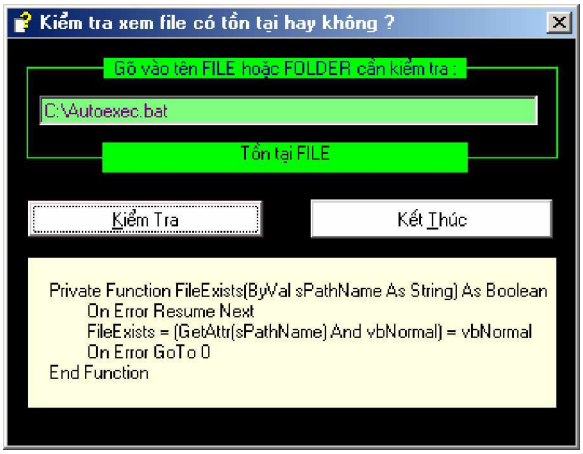
Mô tả :

- Từ khoá Public chỉ ra thủ tục có phạm vi sử dụng toàn cục trong chương trình
- Từ khoá Private chỉ ra thủ tục chỉ có thể sử dụng trong phạm vi của module khai báo
- Nếu không chỉ rõ Public hay Private thì mặc định Visual Basic sử dụng Public
- Hàm phải có giá trị trả về, giá trị đó phải thuộc kiểu dữ liệu của VB. Nếu không khai báo rõ thì sẽ trả về kiểu Variant

Ví dụ :

Kiểm tra xem 1 thư mục hay 1 tập tin có tồn tại trong máy tính không?

```
Private Function FileExists(ByVal sPathName As String) As Boolean
  On Error Resume Next
    FileExists = (GetAttr(sPathName) And vbNormal) = vbNormal
    On Error GoTo 0
End Function
```




```
Private Sub cmdCheck_Click()
   strFileName = Trim(txtFileName.Text)
   If FileExists(strFileName) = True Then
      If Left(Right(strFileName, 4), 1) = "." Then
         txtResuilt.Text = "Tồn tại FILE"
      Else
         txtResuilt.Text = "Tồn tại FOLDER"
      End If
   Else
      If Left(Right(strFileName, 4), 1) = "." Then
         txtResuilt.Text = "Không tồn tại FILE"
      Else
         txtResuilt.Text = "Không tồn tại FOLDER"
      End If
   End If
End Sub
```

Tìm USNLN và BSCNN của 2 số (đưa vào qua tham số)

```
Public Function TimUSCLN(intA As Long, intB As Long) As Long
   While (intA <> intB)
      If intA > intB Then
         intA = intA - intB
      Else
         intB = intB - intA
      End If
   Wend
   TimUSCLN = intA
End Function

Public Function TimBSCNN(intC As Long, intD As Long) As Long
   Dim Tuso As Double,Mauso As Double
   Tuso = intC * intD
   Mauso = TimUSCLN(intC, intD)
   TimBSCNN = Tuso / Mauso
End Function
```

Một số hàm thông dụng :

- Các hàm chuyển đổi kiểu dữ liệu : Val; CLng; CInt; CDate
- Các hàm về ngày giờ : Date; Now; DateSerial; Day; Month; Year; DateDiff; WeekDay
- Các hàm xử lý chuỗi : Len; Left; Mid; Right; InStr; Replace; StrConv; LCase; UCase; Format
- Các hàm kiểm tra giá trị : IsNull; IsEmpty; IsNumeric; IsDate

Chương IV : Mảng - Record

I, Tổng quan về mảng

I.1, Khái niệm, cấu trúc và ý nghĩa sử dụng mảng

Khái niệm về mảng :

Là một tập hợp hay danh sách các phần tử có các đặc điểm giống nhau.

Ví dụ :

- Các học sinh trong một lớp học tạo thành một mảng trong đó mỗi phần tử là một học sinh. Với danh sách các học sinh này chúng ta có thể thực hiện các công việc như sắp xếp tên các học sinh theo thứ tự alphabet, tìm trong danh sách một học sinh có một tên cho trước,…

- Tập hợp các số nguyên 15, 16, 3, 18, 46 lập thành một mảng các số nguyên có 5 phần tử. Phần tử thứ nhất là 15, thứ 2 là 16,…. Ta có thể thực hiện các công việc như tìm số lớn nhất, nhỏ nhất, sắp xếp các số theo thứ tự tăng dần,…

Vị Trí	1	2	3	4	5
Giá Trị	15	16	3	18	46

Cấu trúc của một mảng

Mỗi phần tử được phân biệt với phần tử khác dựa vào số thứ tự hay vị trí của nó trong mảng. Các phần tử được đánh số thứ tự liên tục, thường bắt đầu từ 1 hoặc 0.

Ví dụ :

- Các học sinh trong danh sách lớp sẽ được đánh số thứ tự 1, 2, 3,...

- Trong mảng các số nguyên có 5 phần tử ở trên, phần tử thứ nhất là 15, thứ 2 là 16, ...Tuỳ theo cách khai báo mà chỉ số của phần tử thứ nhất có thể là 0 hay 1.

Với cấu trúc như vậy, mảng được dùng khi nào?

Ý nghĩa của việc sử dụng mảng

Mảng là công cụ giúp nhóm các đối tượng giống nhau lại để xử lý như một đối tượng duy nhất. Thay vì sử dụng tên để đề cập tới một đối tượng, ta sử dụng tên mảng và chỉ ra vị trí của đối tượng muốn truy cập tới.

Ví dụ :

Thay vì khai báo 100 biến khác nhau A1, A2,… A100 để chứa giá trị của 100 số nguyên, ta có thể khai báo một mảng A là tập hợp các số nguyên, có 100 phần tử. Ít nhất, có thể thấy rằng chỉ cần 1 câu khai báo biến thay vì sử dụng 100 câu khai báo. Vì có thể truy cập tới một đối tượng thông qua vị trí của nó trong mảng, ta có thể sử dụng các vòng lặp với một biến đếm để duyệt qua từng phần tử của A.

I.2, Khai báo mảng và cách truy xuất đến phần tử mảng :

Khai báo mảng :

Cú pháp :

```
Dim TenMang (Chỉ số cuối) As Kiểu
```

Hoặc
Dim TenMang (Chỉ số đầu To chỉ số cuối) As Kiểu

Mô tả :
- Nếu dùng cách thứ nhất, chỉ số của phần tử đầu tiên trong mảng là 0, số phần tử của mảng bằng Chỉ số cuối + 1.
- Nếu dùng cách thứ hai, chỉ số của phần tử đầu tiên và cuối cùng được chỉ rõ trong câu khai báo, số phần tử của mảng bằng (Chỉ số cuối – Chỉ số đầu + 1). Chú ý rằng chỉ số cuối phải lớn hơn chỉ số đầu.

Ví dụ : Khai báo mảng A chứa 10 số nguyên
Dim A(9) As Integer
Hoặc
Dim A(1 to 10) As Interger

Truy cập một phần tử của mảng :
Cú pháp :
TenMang(Vị trí của phần tử muốn truy xuất)

Mô tả :
- Nếu vị trí cần truy xuất nằm ngoài chỉ số mảng thì sẽ báo lỗi .
- Sử dụng hàm LBound(Tên mảng), UBound(Tên mảng) để xác định chỉ số nhỏ nhất, lớn nhất của mảng.

Ví dụ : Hiển thị vị trí và giá trị của từng phần tử trong mảng A(5)
```
Private Sub cmdHienThi_Click()
    For i = LBound(A) To UBound(A)
        MsgBox “Vị trí " & i & ", có giá trị là " & A(i)
    Next
End Sub
```

I.3, Một số giải thuật trên mảng :

A, Giải thuật duyệt mảng
Giải thuật duyệt mảng đơn giản sử dụng một vòng lặp For để duyệt từng phần tử của mảng dựa trên chỉ số. Đây là giải thuật cơ bản nhất cần nắm vững để mở rộng trong nhiều bài toán khác.
```
For i = LBound(A) To UBound(A)
    ‘Thực hiện các xử lý với phần tử A(i).
Next
```
Ví dụ :
Tăng giá trị của mỗi phần tử trong mảng A lên 1
```
For i = LBound(A) To UBound(A)
    A(i) = A(i) + 1
Next
```
Tính tổng các phần tử trong mảng

```
Tong = 0
For i = LBound(A) To UBound(A)
        Tong  = Tong + A(i)
Next
MsgBox "Tổng các phần tử trong mảng = " & Tong
```

B, Tìm kiếm các phần tử thỏa một điều kiện cho trước

Bản chất của việc tìm kiếm là duyệt qua từng phần tử trong mảng và so sánh mỗi phần tử với điều kiện tìm kiếm do đó, giải thuật này có thể coi là một sự mở rộng của giải thuật ở trên

Duyệt mảng có điều kiện.

```
For i = LBound(A) To UBound(A)
        If KiemTra(A(i)) = True Then
                'A(i) là phần tử thoả điều kiện tìm kiếm
                'Thực hiện các bước xử lý tiếp theo
        End If
Next
```

Trong giải thuật này, KiemTra() là một hàm dùng để so sánh giá trị của phần tử đang xét với điều kiện tìm kiếm.

Ví dụ : Liệt kê tất cả các phần tử của A là số nguyên tố

```
For i = LBound(A) To UBound(A)
        If LaSoNT(A(i)) Then
                MsgBox "Số nguyên tố  : " & A(i) & " ở vị trí " & i
        End if
Next
```

Với hàm LaSoNT() như sau:

```
Function LaSoNT (pt As Integer) As Boolean
        'Điều kiện 1: Số nguyên tố là số nguyên dương lớn hơn 1
        LaSoNT = (pt > 1)
        'Điều kiện 2: Số nguyên tố là số chỉ chia hết cho 1 và chính no
        For i=2 to pt \ 2
                If pt  Mod i = 0 Then
                        LaSoNT = False
                        Exit Function
                End If
        Next
End Function
```

C, Tìm giá trị nhỏ nhất của mảng

Giải thuật này sử dụng một kỹ thuật gọi là kỹ thuật xoá vết :

```
Dim gtnn  As Kiểu của một phần tử trong mảng A
gtnn  = A(LBound(A))
For i = LBound(A)  To UBound(A)  'Để đơn giản, không nên dùng i=LBound(A)+1
        If A(i) < gtnn Then    gthh = A(i)
Next
MsgBox "Giá trị nhỏ nhất trong mảng là : " & gtnn
```

Đôi khi, ta cần tìm ra vị trí của phần tử chứa giá trị nhỏ nhất thay vì giá trị nhỏ nhất của mảng, giải thuật sẽ thay đổi như sau

```
gtnn  = LBound(A)
For i = LBound(A) To UBound(A) If A(i) < A(gtnn) Then
        gtnn  = i
Next
MsgBox "Vị trí của phần tử mang giá trị nhỏ nhất là : " & gtnn
```

D, Tính tổng của các phần tử :

Giải thuật này sử dụng kỹ thuật lưu vết. Lấy giá trị cũ để tìm ra giá trị mới

```
TongChan = 0
For i = LBound(A) To UBound(A)
        If A(i) mod 2 = 0 Then TongChan = TongChan + A(i)
Next
MsgBox "Tổng các phần tử chẵn trong mảng = " & TongChan
```

E, Sắp xếp các phần tử theo thứ tự tăng dần :

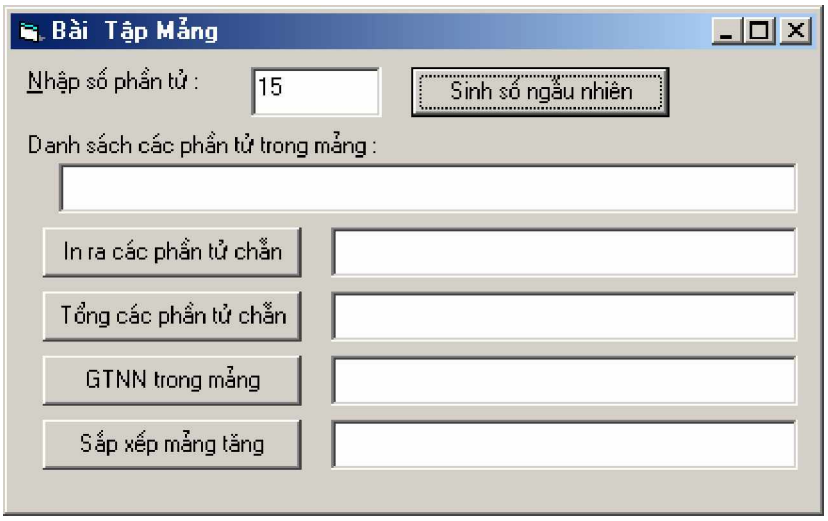
```
For i = 0 To UBound(A) – 1
        For j = i + 1 To UBound(A)
                If A(i) > A(j) Then
                        tam = A(i)
                        A(i) = A(j)
                        A(j) = tam
                End If
        Next
Next
```

In mảng sau khi sắp xếp :

```
strChuoi = ""
For i = 0 To UBound(A) – 1
        strChuoi = strChuoi & " " & A(i)
Next
MsgBox "Mảng đã sắp xếp : " & strChuoi
```

F, Hướng dẫn bài tập :

Thiết kế giao diện chương trình như sau :

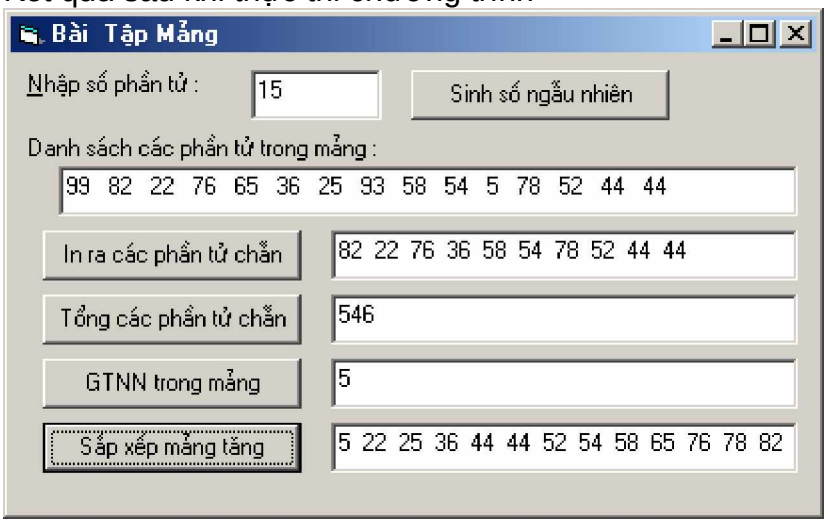


Mã nguồn của chương trình

```
Dim A(100) As Long, intSoPT As Integer, i As Integer, j As Integer

Private Sub cmdSinhSo_Click()
    intSoPT = Val(txtSoPhanTu.Text)
    txtHienPhanTu.Text = ""
    Randomize
    For i = 1 To intSoPT
        A(i) = Int(100 * Rnd)
        txtHienPhanTu = txtHienPhanTu & A(i) & "   "
    Next
End Sub

Private Sub cmdInPTChan_Click()
    txtPTChan.Text = ""
    For i = 1 To intSoPT
        If A(i) Mod 2 = 0 Then txtPTChan = txtPTChan & A(i) & "  "
    Next
End Sub

Private Sub cmdTongPTChan_Click()
    Dim intTongChan As Long
    intTongChan = 0
    For i = 1 To intSoPT
        If A(i) Mod 2 = 0 Then intTongChan = intTongChan + A(i)
    Next
    txtTongPTChan.Text = intTongChan
End Sub

Private Sub cmdGTNN_Click()
    Dim gtnn As Integer
    gtnn = A(1)
```

```
    For i = 2 To intSoPT
        If A(i) < gtnn Then gtnn = A(i)
    Next
    txtGTNN.Text = gtnn
End Sub

Private Sub cmdSapxeptang_Click()
    For i = 1 To intSoPT - 1
        For j = i + 1 To intSoPT
            If A(i) > A(j) Then
                tg = A(i): A(i) = A(j): A(j) = tg
            End If
        Next
    Next
    txtSapxeptang.Text = ""
    For i = 1 To intSoPT
        txtSapxeptang.Text = txtSapxeptang.Text & A(i) & "  "
    Next
End Sub
```

Kết quả sau khi thực thi chương trình



II, Các hàm thao tác trên mảng :

II.1, Hàm Array()

Hàm Array() tạo ra một biến kiểu Variant chứa một mảng. Ta có thể sử dụng biến này như một biến mảng bình thường.

Cú pháp :

Array(Danh sách các phần tử của mảng)

Mô tả :

Các phần tử trong danh sách được phân cách nhau bởi dầu phảy (,)

Ví dụ :

```
Dim A 'Khai báo một biến kiểu Variant để chứa mảng
```

```
A = Array(1,3,6)
For i = 0 To UBound(A)
        Print A(i) & " "
Next
'Kết quả: 1 3 6
```

II.2, Hàm Choose()

Hàm Choose() trả về giá trị của một phần tử trong mảng là một danh sách các tham số dựa trên thứ tự của phần tử đó.

Cú pháp :

Choose(chỉ số, phần tử 1[, phần tử 2, ... [, phần tử n]])

Mô tả :

- Chỉ số là vị trí của phần tử muốn cho ïn từ danh sách. Các phần tử trong danh sách được đánh số bắt đầu từ 1
- Các phần tử trong danh sách được phân cách nhau bởi dầu phảy (,)

Ví dụ : Liệt kê tên các tháng trong năm

```
Dim i As Integer
For i = 1 To 12
        Print Choose(i,"Tháng 1","Tháng 2","Tháng 3", _
        "Tháng 4","Tháng 5",  Tháng 6","Tháng 7",_
        "Tháng 8",  "Tháng 9","Tháng 10","Tháng 11","Tháng 12")
Next
```

II.3, Hàm Split()

Hàm Split() cắt một chuỗi thành một mảng các chuỗi con

Cú pháp :

Split(Chuỗi muốn cắt [, dấu phân cách[, số phần tử kết quả]])

Mô tả :

- Chuỗi muốn cắt là chuỗi sẽ được chia thành các chuỗi con
- Dấu phân cách dùng để xác định cách chia chuỗi muốn cắt thành các chuỗi con. Nếu không chỉ rõ thành dấu phân cách mặc định là " ".
- Số phần tử kết quả giới hạn số chuỗi con có trong mảng kết quả. Mặc định số phần tử kết quả là - 1 nghĩa là không giới hạn số chuỗi con trả về.

Ví dụ : Cắt một câu thành các từ

```
Dim Str As String
Str = InputBox("Nhập vào một câu")
Dim A
A = Split(Str)
'Lần lượt in ra các từ trong câu
For i = 0 To UBound(A)
        Print A(i) & " "
Next
```

II.4, Hàm Join()

Hàm Join() trả về một chuỗi là kết hợp của các phần tử trong một mảng kiểu chuỗi.

Cú pháp

Join(Mảng nguồn[, dấu phân cách])

Mô tả

Mảng nguồn là mảng chứa các phần tử sẽ nối lại thành một chuỗi.

Dấu phân cách nếu không được chỉ rõ thì mặc định là " "

Ví dụ : Nối các từ đã cắt ở ví dụ trên thành một chuỗi cách nhau bởi dấu phảy

Str = Join(A)

III, Kiểu Dữ Liệu Record

III.1, Khái niệm

Kiểu Record hay còn gọi là kiểu dữ liệu do người dùng định nghĩa là một kiểu dữ liệu hỗn hợp gồm nhiều thành phần dữ liệu khác nhau.

Cú pháp :

```
Public | Private Type <Tên kiểu>
        <thành phần I> As <Kiểu dữ liệu>
        <thành phần II> As <Kiểu dữ liệu>
        ……..
End Type
```

Mô tả :

Các thành phần có thể thuộc các kiểu dữ liệu khác nhau : Long, String, Boolean, Date, và cả Record …

III.2, Ví dụ : Khai báo bản ghi quản lý thông tin sinh viên

```
Public Type SinhVien
   masv As String
   hotensv As String
   Ngaysinh As String
   DiemThi As Double
End Type
Public objSV As SinhVien

Sub NhapHoSo()
        With objSV
                .masv = "012199026"
                .hotensv = "donTRUONGBT"
                .Ngaysinh = "28/12"
        End With
End Sub
Sub InHoSo()
        With objSV
                lblThongTinSV = "Mã SV : " & .masv & " " & "Họ Tên : " & _
                        .hotensv & " " & "Ngày Sinh :" & .Ngaysinh
        End With
End Sub
```

Chương V : Menu Editor – MDI Form

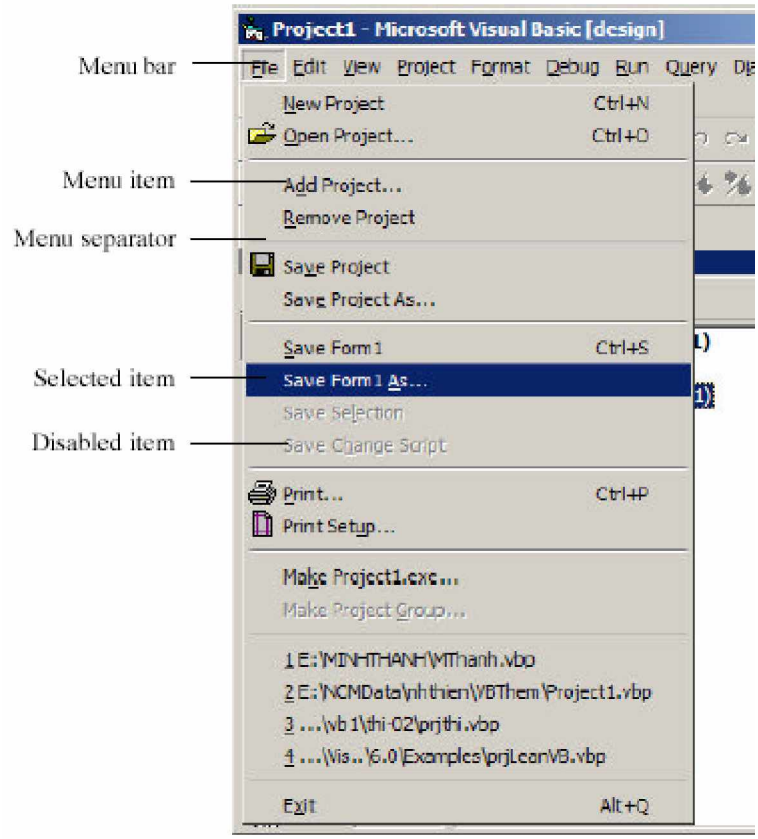
I, Menu Editor

I.1, Giới thiệu về hệ thống thực đơn của một ứng dụng

Khi xây dựng ứng dụng trên nền của hệ điều Windows, giao tiếp giữa ứng dụng và người sử dụng chủ yếu thông qua hệ thống hộp thoại của ứng dụng, hộp thoại đơn giản chỉ là những chức năng của chương trình được thể hiện bằng giao diện đồ hoạ.

Để quản lý các chức năng của ứng dụng ta cần phải xây dựng những điều khiển để gọi thực hiện từng chức năng một. Khi này, hãy tưởng tượng một chương trình có hơn 20 điều khiển nằm khắp nơi trên màn hình của ứng dụng thì sẽ không còn chỗ cho những thao tác xử lý của ứng dụng.

Do đó, hệ thống thực đơn là cách tiếp cận đơn giản nhất khi thiết kế hệ thống chức năng cho ứng dụng: thay vì hiển thị hết các chức năng một lúc ta sẽ tổ chức chúng thành các phần tử của thực đơn và ẩn đi cho đến khi cần sử dụng. Hệ thống thực đơn sẽ chứa những điều khiển của nó trong những cửa sổ riêng (drop down) và có thể sử dụng ngay khi cần

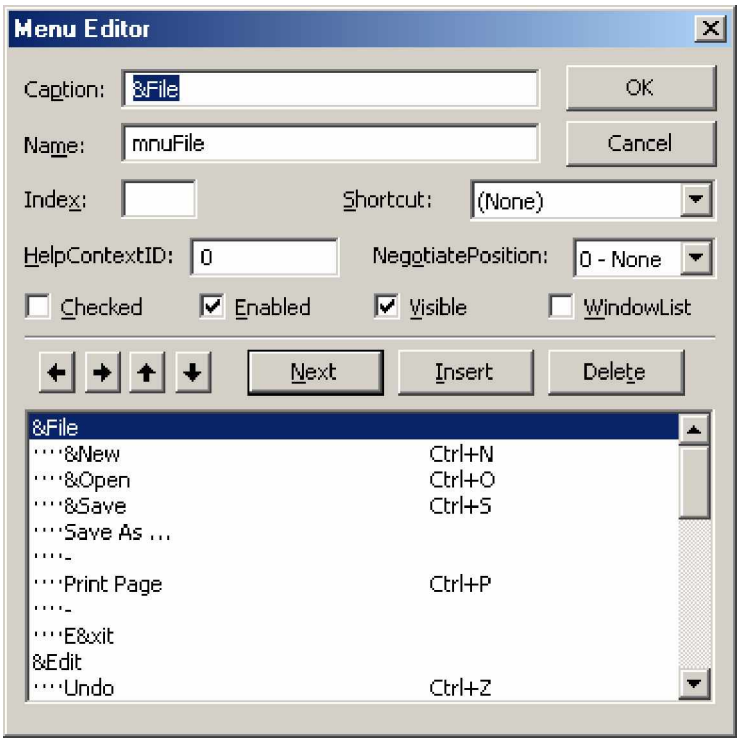


Menu bar giống như một vật chứa, lưu giữ tất cả các chức năng trong hệ thống. Trên menu bar sẽ liệt kê các chức năng hay nhóm chức năng của chương trình gọi là menu. Khi người dùng chọn một nhóm chức năng trên menu bar thì sẽ mở cửa sổ chứa các chức năng tương ứng, chẳng hạn trong hình trên ta chọn mục File trên menu bar

Mỗi cửa số của menu bar sẽ chức một số chức năng, và được tổ chức dưới dạng 1 danh sách theo chiều dọc. Những chức năng này gọi là **Menu item** , những menu item thường được sắp xếp theo từng nhóm có ý nghĩa giống nhau và ngăn cách bởi một một thành phần gọi là **Menu separator** . Người sử dụng có thể di chuyển các phím mũi tên lên xuống hay dùng chuột di chuyển qua các item để chọn một menu item. Khi một menu item được chọn thì sẽ có một vệt sáng xuất hiện tại vị trí của nó, gọi là **Selected item** . Sau đó, nhấn phím Enter hoặc click chuột trái để thực hiện chức năng của menu item đang chọn.
Menu item có thể ở tình trạng bị mờ là **Disabled item** , khi đó người sử dụng sẽ không thể thực hiện được chức năng của menu item đó.

I.2, Thiết kế hệ thống thực đơn cho ứng dụng

Trong môi trường phát triển của Visual Basic 6 thì hệ thống thực đơn sẽ không có sẵn trên màn hình thiết kế. Do đó ta phải tạo ra một Menu bar cho màn hình làm việc thông qua tiện ích Menu Editor do môi trường cung cấp, ta có thể mở Menu Editor bằng cách chọn menu item Menu Editor trong mục Tools của cửa sổ thiết kết, hoặc nhấn tổ hợp phím **Ctrl+E**



Tạo chức năng hay nhóm chức năng (menu) trên Menu bar

Để tạo một menu trên menu bar, ta chỉ cần cung cấp 2 thông tin chính là Caption và Name trong Menu Editor. Thuộc tính Caption dùng để hiển thị tên của chức năng trên menu bar và ta sẽ dùng thuộc tính Name trong khi viết lệnh xử lý cho chức năng tương ứng, giá trị của Name thường được bắt đầu với **mnu** và không được rỗng, không được trùng . Khi này bạn đã tạo được hệ thống thực đơn cho chương trình, công việc tiếp theo là tạo các Menu item cho các nhóm chức năng trên Menu bar.

Tạo Menu item
Khi có yêu cầu tạo một menu item **New** cho chức năng **File**, bạn phải làm thế nào?
Để thực hiện, trước tiên ta click vào nút Next trên Menu Editor, sau đó click vào nút có hình mũi tên qua phải trên Menu Editor để tạo một khoảng cách so với lề trái cho phần tử mới (*xem hình trên*). Bây giờ, bạn nhập giá trị cho Caption là New và Name là mnuFileNew. Khi này, Menu item mới tạo sẽ xuất hiện bên dưới menu File và thụt vào trong một đoạn 4 ký tự, giống như sau:

File
... New

Như vậy ta đã tạo được menu File với một Menu item là New. Nếu ta click Next (hay nhấn Enter) thì ta sẽ tiếp tục tạo Menu item, muốn tạo thêm menu trên Menu bar thì ta phải click vào nút có hình mũi tên qua trái cho đến khi không còn khoảng cách với lề trái.
Viết lệnh xử lý cho các Menu item cũng giống như cho các điều khiển khác, Menu item chỉ có 1 sự kiện duy nhất là Click.

Thêm/Xoá một thành phần trên hệ thống thực đơn
Để chèn thêm một thành phần vào hệ thống thực đã có, ta chọn một thành phần sau đó click vào nút Insert của Menu Editor, khi này ta sẽ có một thành phần mới trong hệ thống ngay trên thành phần đã chọn. Nhập giá trị cho các thuộc tính Caption và Name để tạo một thành phần mới trên thanh menu bar.
Muốn bỏ một thành phần đã có trong hệ thống thực đơn chỉ cần chọn thành phần đó và nhấn nút Delete trên Menu Editor

Sắp xếp thứ tự của các thành phần trên hệ thống thực đơn
Ta sử dụng các nút hình mũi tên lên, xuống, qua trái, qua phải trên Menu Editor để thực hiện một số thao tác sắp xếp và thay đổi tính chất của các thành phần trên hệ thống thực đơn.

 Chuyển ptử được chọn qua phải, như là chuyển một Menu item thành một Menu

 Chuyển ptử được chọn qua trái , như là chuyển một Menu thành một Menu item

 Di chuyển phần tử được chọn lên trên

 Di chuyển một phần tử được chọn xuống dưới

Menu Separator
Menu separator không phải là một chức năng của chương trình, nó chỉ có ý nghĩa chia các menu item theo chức năng tương tự nhau. Menu separator có dạng là một đường kẻ nằm ngang.
Để thêm menu separator vào hệ thống cũng tương tự như thêm một menu item nhưng giá trị của thuộc tính Caption là dấu trừ (-).

Access character

Ta có thể kích hoạt chức năng của một menu, menu item thông qua bàn phím bằng cách nhất phím Atl sau đó chọn chọn phím ứng với chức năng cần thực hiện. Phím này gọi là access character và phải là một trong những ký tự có trong thuộc tính Caption của chức năng.

Để một ký tự trong Caption của menu hay menu item là Access character ta chỉ cần đặt vào trước ký tự đó một ký tự &. Điều quan trọng là các Access character trong cùng một cấp, nghĩa là các thành cùng là menu hay các menu item của cùng một menu, nên là duy nhất. Ví dụ:

&File
... &New
... &Open

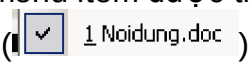
Shortcut key

Ngoài ra để thực hiện một chức năng của hệ thống ta còn có thể sử dụng Shortcut key, đó là một phím hay tổ hợp phím, mà không cần mở menu chứa nó. Shortcut key là một số phím, tổ hợp phím do Menu Editor cung cấp, ta không thể tự ý thêm vào.

Để tạo Shortcut key cho một menu hay menu item ta vào Menu Editor, chọn thành phần cần tạo shortcut key, sau đó chọn một phím hay tổ hợp phím đã được xây dựng sẵn trong ComboBox Shortcut. Ví dụ:

&File
... &New Ctrl+N
... &Open Ctrl+O

Một số thao tác xử lý

Trong hệ thống thực đơn, menu và menu item cũng có những thuộc tính giống với điều khiển khác là Enabled và Visible, ngoài ra menu item còn có thêm thuộc tính Checked. Nếu thuộc tính Check của một menu item được thiết lập là True thì bên phải menu item đó sẽ xuất hiện một dấu check (

). Giá trị của các thuộc tính này có thể được thiết lập lúc thiết kế hay lúc thực thi chương trình.

Thiết lập giá trị cho thuộc tính Checked, Enabled, Visible khi thiết kế

Ta có thể dễ dàng thiết lập giá trị cho 3 thuộc tính thông qua Menu Editor bằng cách click vào các check box tương ứng của một menu item đang chọn. Khi này, menu item sẽ thể hiện theo thuộc tính đã chọn.

Xác lập giá trị khi chương trình đang thực thi

Giá trị các thuộc tính trên đều có thể thiết lập khi chương trình đang chạy. Dựa vào đây ta có thể dễ dàng thay đổi trạng thái của phần tử theo sự chọn lựa của người sử dụng.

Xét ví dụ : nếu ta muốn khi người dùng click chọn menu item NoiDung thì menu item xuất hiện dấu Check để báo là chức năng NoiDung đang thực thi. Nếu click vào chức năng NoiDung một lần nữa thì dấu check sẽ mất đi. Để làm điều đó, ta viết lệnh trong sự kiện Click của menu item NoiDung như sau:

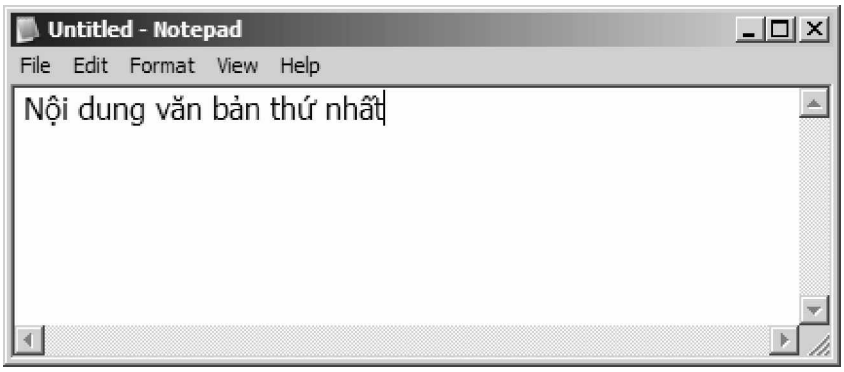
```
Private Sub mnuNoiDung_Click()
    mnuNoiDung.Checked = Not mnuNoiDung.Checked
End Sub
```

II, MDI Form :

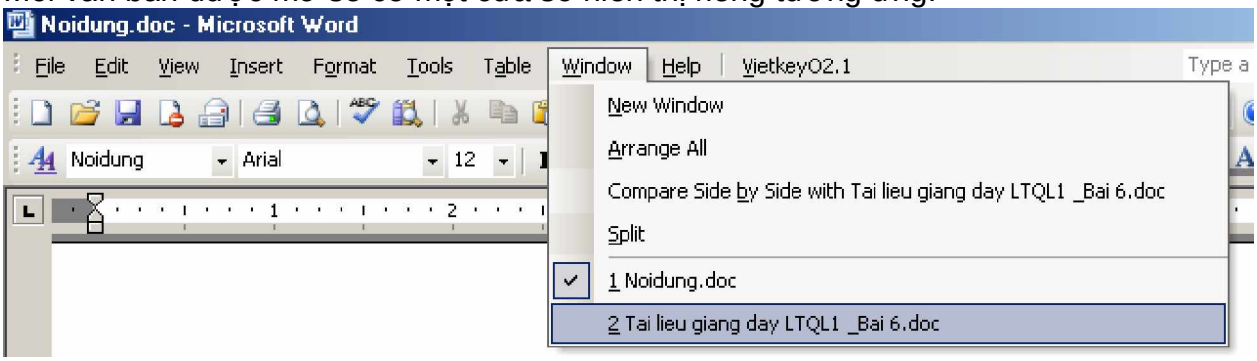
II.1, Màn hình SDI và màn hình MDI

Các chương trình chạy trên Windows có giao diện thuộc một trong hai loại chính : giao diện một cửa sổ **(SDI : Single Document Interface**) và giao diện đa cửa sổ **(MDI : Multi Document Interface)**

Một ví dụ dễ thấy nhất về ứng dụng có giao diện một cửa sổ là chương trình Notepad trong hệ điều hành Windows. Tại mỗi thời điểm chúng ta chỉ có thể mở một văn bản. Khi cần mở văn bản thứ hai, chúng ta phải đóng văn bản thứ nhất lại.



Khác với Notepad, Microsoft Excel và Microsoft Word là hai ứng dụng có giao diện đa cửa sổ (MDI) trên Windows. Các ứng dụng này cho phép mở nhiều văn bản cùng lúc. Mỗi văn bản được mở sẽ có một cửa sổ hiển thị riêng tương ứng.



Mỗi loại giao diện, SDI hay MDI, sẽ thích hợp với những loại ứng dụng khác nhau.

II.2, Màn hình MDI và màn hình MDI Child

Như đã trình bày ở trên, các ứng dụng giao diện MDI cho phép mở nhiều thể hiện khác nhau tại một thời điểm. Mỗi thể hiện sẽ được mở trong một cửa sổ khác nhau. Một ứng dụng đa cửa sổ sẽ có hai màn hình giao tiếp : một đóng vai trò cửa sổ chính MDI và một đóng vai trò dạng chung cho các thể hiện cửa sổ con MDI Child bên trong cửa sổ MDI.

Màn hình MDI

Màn hình MDI hoàn toàn giống với các màn hình bình thường mà chúng ta đã từng làm việc trong những chương trước. Điểm khác nhau là màn hình MDI không thể chứa các điều khiển trên nó ngoại trừ điều khiển Picture box và các điều khiển không hiển thị như

Timer,... Ngoài ra, trong một ứng dụng sẽ chỉ có duy nhất một màn hình MDI và với những lý do này màn hình MDI thường được dùng như màn hình chính của ứng dụng.

Màn hình MDI Child

Màn hình MDI Child là một màn hình giao tiếp giống như tất cả các màn hình giao tiếp Form mà chúng ta đã làm việc trước đây. Ở thời điểm thiết kế, màn hình MDI Child sẽ không bị giới hạn bên trong màn hình MDI. Chúng ta hoàn toàn có thể đặt thuộc tính, thêm các điều khiển cũng như các lệnh,... liên kết với cửa sổ con một cách bình thường.

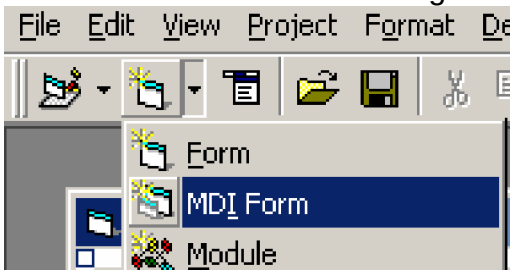
Điều cần phải nhớ là đặt thuộc tính MDIChild cho màn hình giao tiếp loại MDI Child là True.

Đặc điểm của hai màn hình : Tại thời điểm chạy chương trình, màn hình MDI và các màn hình MDI child có những đặc điểm như sau:

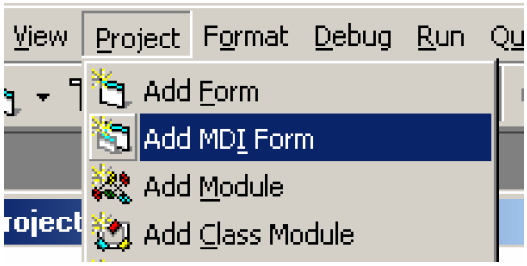
- Tất cả các màn hình MDI Child sẽ luôn hiển thị bên trong màn hình MDI.
- Người dùng chỉ có thể di chuyển, thay đổi kích thước, phóng to hay thu nhỏ các màn hình MDI Child nhưng chỉ trong giới hạn kích thước của màn hình MDI mà thôi.
- Khi một màn hình MDI Child được thu nhỏ, biểu tượng của nó sẽ xuất hiện bên trong màn hình chính MDI chứ không xuất hiện trên thanh ứng dụng (Taskbar) của Windows. Nếu màn hình MDI được thu nhỏ thì tất cả các màn hình MDI Child trong màn hình MDI này cũng được thu nhỏ theo
- Khi phóng to màn hình MDI Child, tiêu đề của màn hình con này sẽ tự động nối với tiêu đề của màn hình MDI và hiển thị trong thanh tiêu đề của màn hình MDI. Hình vẽ dưới đây minh họa cho điều này.
- Khi đóng màn hình MDI, các màn hình MDI Child sẽ tự động đóng theo. Nếu một màn hình con nào đó không đóng được, màn hình MDI sẽ không đóng lại.

II.3, Các bước đưa ứng dụng giao diện MDI vào chương trình

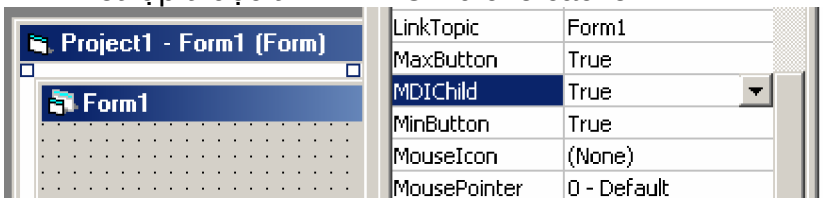
- Đưa MDI Form vào chương trình

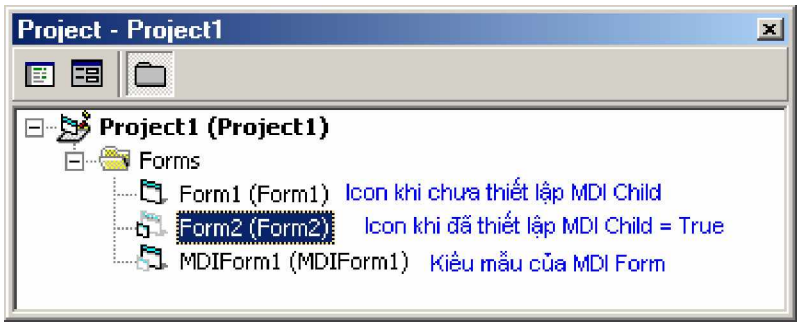


hoặc



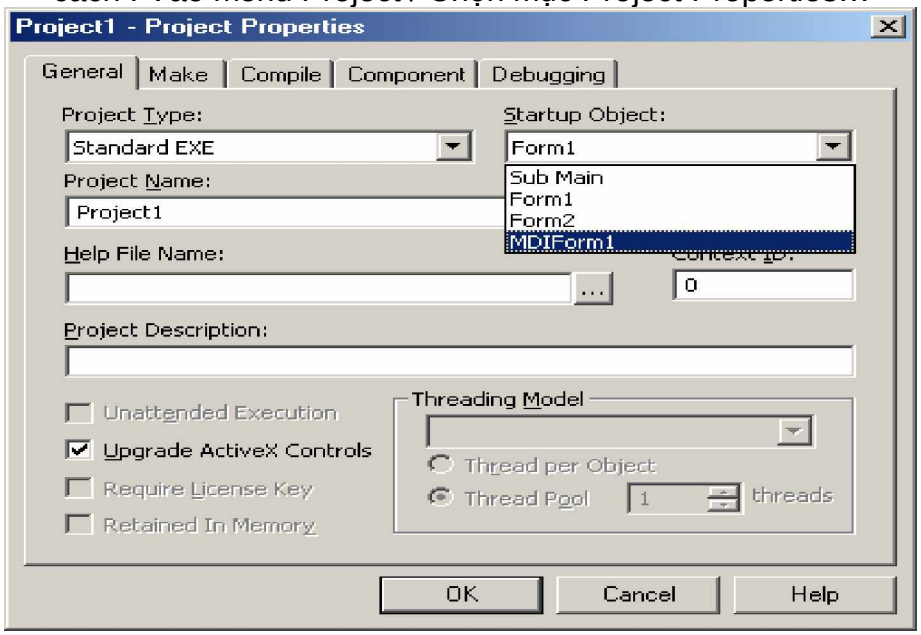
- Thiết lập thuộc tính MDI Child cho các form





- Thiết kế menu (hệ thống thực đơn) cho MDI Form

- Ta thiết lập cho MDI trở thành Form khởi động đầu tiên trong ứng dụng bằng cách : Vào menu Project / Chọn mục Project Properties…



- Viết lệnh cho hệ thống menu gọi thực hiện các chức năng :

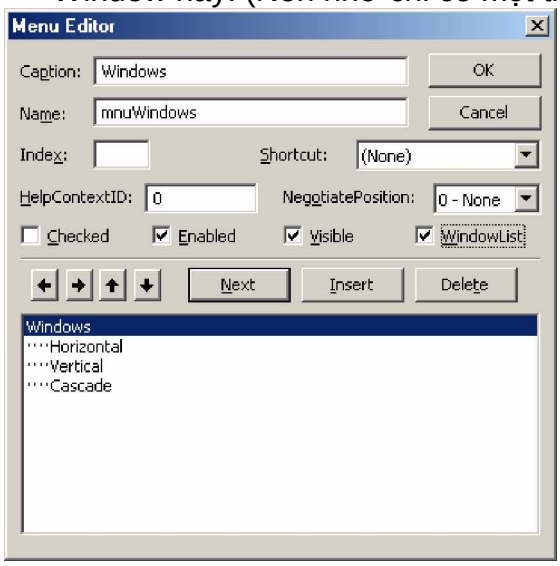
```
Private Sub mnuhanghoa_Click()
    Form1.Show
End Sub

Private Sub mnuinfo_Click()
    Dim info As String
    info = "Hoc Vien: Thu " & vbCrLf & "Stt: 006" & vbCrLf & "Thu Muc Bai Lam:" _
        & App.Path & vbCrLf & "Ten File Project: " & App.Title & ".vbp"
    MsgBox info, vbInformation, "Thong Tin Bai Lam"
End Sub

Private Sub mnuthoat_Click()
    If MsgBox("Co Thoat Khong?", vbYesNo + vbInformation, "Thoat") = vbYes Then
        Unload Me
    End If
End Sub
```

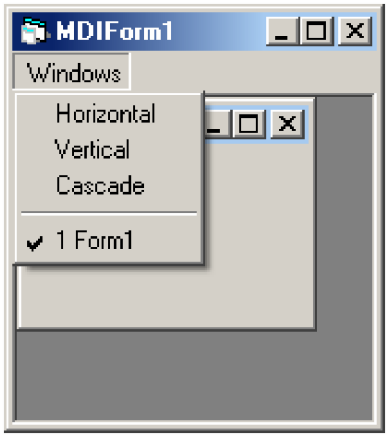
- Thiết lập các thuộc tính của MDI Form : WindowState = 2 – Maximized
  - 0 – Normal : Hiển thị khung cửa sổ bình thường theo thiết kế
  - 1 – Minimnized : Thu nhỏ cửa sổ xuống thanh TaskBar
  - 2 – Maximized : Phóng to cửa sổ ra toàn bộ màn hình

- Nếu các MDI Child bị thay đổi kích cỡ so với khi thiết kế lúc bạn chạy chương trình thì bạn cần thiết lập lại thuộc tính BorderStyle của Form đó :
  - BorderStyle = 1- Fixed Single

II.4, Tạo thực đơn Window trong các ứng dụng MDI

- Tạo thực đơn chính Window trong cửa sổ chính MDI hay cửa sổ con MDI Child
- Sử dụng công cụ Menu Editor, chọn thuộc tính WindowList cho thực đơn Window này. (Nên nhớ chỉ có một thực đơn được chọn thuộc tính WindowList)






```
Private Sub mnuHorizontal_Click()
        Me.Arrange vbTileHorizontal
End Sub

Private Sub mnuVertical_Click()
        Me.Arrange vbTileVertical
End Sub

Private Sub mnuCascade_Click()
        Me.Arrange vbCascade
End Sub
```

Chương VI : Kết Nối Cơ Sở Dữ Liệu Với ADO

I, Tổng quan về lập trình cơ sở dữ liệu

I.1, Hệ quản trị CSDL

Khái niệm về hệ quản trị CSDL

Hệ quản trị cơ sở dữ liệu là một hệ thống phần mềm, cung cấp các chức năng cho phép người dùng quản lý các đối tượng trong một CSDL.
Ví dụ: Hệ quản trị CSDL Microsoft Access, Microsoft SQL Server,…
Khác với CSDL, hệ quản trị CSDL chỉ đóng vai trò quản lý, cho phép người dùng tạo ra các đối tượng như bảng, query, quan hệ,… CSDL chỉ là một hay nhiều file, chứa các đối tượng mà người dùng tạo ra. Người dùng chỉ có thể làm việc với CSDL thông qua hệ quản trị CSDL.

Các chức năng quản lý CSDL của hệ quản trị

- Thông qua hệ quản trị CSDL, người dùng tạo ra các cấu trúc bảng, mối liên hệ giữa các bảng, nhập dữ liệu, tạo các query,…
- Với một số hệ quản trị CSDL chuyên nghiệp (khác với Microsoft Access là một hệ quản trị CSDL nhỏ – Desktop Relational Database Management), người dùng còn có thể tạo ra các đối tượng khác trong CSDL như View, Trigger, UDDT, phân quyền cho người dùng khác,…

I.2, Các kỹ thuật lập trình với CSDL

Các kỹ thuật lập trình CSDL cấp thấp

Windows API : Hệ điều hành Windows cung cấp một bộ thư viện rất nhiều hàm riêng lẻ để làm việc với CSDL gọi là MS DB-LIB. Sử dụng những hàm trong DB-LIB rất phức tạp với nhiều kiểu dữ liệu, nhiều tham số, tên hàm khó nhớ.

ODBC : Để đơn giản bớt việc sử dụng DB-LIB, bộ thư viện phần mềm ODBC được Microsoft xây dựng sau đó, cũng dựa trên MS DB-LIB nhưng đơn giản hơn. Trong ODBC, Microsoft hỗ trợ người lập trình với các Database driver để làm việc với nhiều loại CSDL khác nhau. Tận dụng Database driver, người lập trình có thể làm việc với các loại CSDL khác nhau mà ODBC hỗ trợ theo cùng một cách.
Tới nay, ODBC vẫn còn được áp dụng trong các phần mềm chuyên nghiệp để tận dụng tốc độ xử lý dù viết chương trình rất khó. ODBC hỗ trợ tới gần 20 loại CSDL khác nhau.

OLE DB : Là một kỹ thuật kết nối CSDL tương tự như ODBC nhưng mạnh hơn, OLE DB không chỉ có khả năng làm việc với các loại CSDL quan hệ mà cả những loại dữ liệu khác như cấu trúc thư mục/tập tin, mail,…
Thay vì sử dụng Database driver, OLE DB cung cấp cách lập trình thống nhất cho người lập trình trên các loại CSDL khác nhau thông qua Data provider.

Đặc điểm chung của các kỹ thuật lập trình CSDL cấp thấp là nhiều hàm, kiểu dữ liệu và tham số. Thứ tự các bước lập trình cũng phức tạp và dài dòng, tuy nhiên ứng dụng sẽ chạy nhanh hơn, người lập trình nếu nắm vững kỹ thuật sẽ chủ động trong quá trình xây dựng các chức năng.

Các kỹ thuật lập trình CSDL cấp cao

DAO : Tiếp sau ODBC là hai kỹ thuật kết nối CSDL JET và DAO. JET chỉ kết nối với các CSDL Access để nhắm tới mục tiêu tốc độ xử lý nhanh và lập trình đơn giản. DAO cũng được xây dựng dựa trên ODBC để đơn giản hoá việc lập trình.

Cả JET và DAO đều cung cấp cho người lập trình các đối tượng để làm việc với CSDL. Điểm này giúp người lập trình có được một cách làm việc đơn giản khi xử lý dữ liệu. Ít nhất, các hàm cũng được gom thành từng nhóm có ý nghĩa riêng trong các đối tượng và khi gọi, không còn cần thiết phải truyền nhiều tham số vì bản thân các đối tượng xử lý dữ liệu chỉ cần gán các giá trị tham số vào thuộc tính một lần duy nhất.

ADO : Tương tự như DAO nhưng là thế hệ đi sau, ADO được xây dựng dựa trên OLE DB. ADO cũng sử dụng các đối tượng làm công cụ để thao tác với CSDL. Điểm quan trọng là các đối tượng của ADO là những đối tượng độc lập, có thể được khai báo và sử dụng riêng rẽ, khác với các đối tượng trong DAO là một hệ thống đối tượng phân cấp, muốn tạo ra đối tượng này có thể phải cần đến một đối tượng khác.

Đặc điểm của các kỹ thuật lập trình CSDL cấp cao là cung cấp phương pháp lập trình CSDL đơn giản, che đi những thủ tục phức tạp mà người lập trình thực sự không cần quan tâm trong khi lập trình. Đổi lại, chương trình sẽ chạy chậm hơn

II, Giới thiệu về Microsoft Active Data Object (ADO)

Các phiên bản

- ADO ra đời phiên bản chính thức đầu tiên là 1.5. Bản thân ADO chỉ là một phần trong bộ công cụ lập trình CSDL của Microsoft có tên là MSDAC (Microsoft Data Access Component) gồm 3 thành phần: MS DB-LIB, OLE DB và ADO.
- Ban đầu, ADO là bộ công cụ lập trình CSDL rời rạc nhưng sau đó được Microsoft tích hợp vào hệ điều hành Windows và một số phần mềm của mình như Microsoft Visual Studio, Microsoft Office,…
- Phiên bản ADO 2.0 đi kèm với Windows 98. Windows 2000 có phiên bản 2.5 và Windows XP có phiên bản ADO 2.7

Tương quan giữa các đối tượng của ADODB và các thành phần trong CSDL

- Hệ thống các đối tượng của ADODB không nhiều, chỉ có khoảng 10 đối tượng được tổ chức phân cấp nhưng có thể được tạo ra một cách độc lập để sử dụng. 3 đối tượng chính của ADODB là:
  - Connection: dùng để tạo kết nối tới CSDL, một cách đơn giản, có thể coi Connection là đại diện cho CSDL đang làm việc
  - RecordSet: dùng để chứa dữ liệu trong các bảng hay truy vấn, view,… Một cách đơn giản, có thể coi RecordSet đại diện cho một bảng hay một query, view trong CSDL
  - Command: dùng để thực hiện các câu lệnh SQL hay các query được tạo sẵn trong CSDL.
- Ngoài những đối tượng chính, ADODB có một số đối tượng ở cấp thấp hơn, cung cấp những thông tin chi tiết. Ví dụ:
  - Đối tượng Fields là một tập hợp các Field có trong RecordSet mô tả các cột trong các bảng hay câu query mà RecordSet đó đại diện
  - Đối tượng Parameters là tập hợp các Parameter dùng trong trường hợp cần thực hiện các câu query có chứa tham số.

III, Đối tượng ADODC

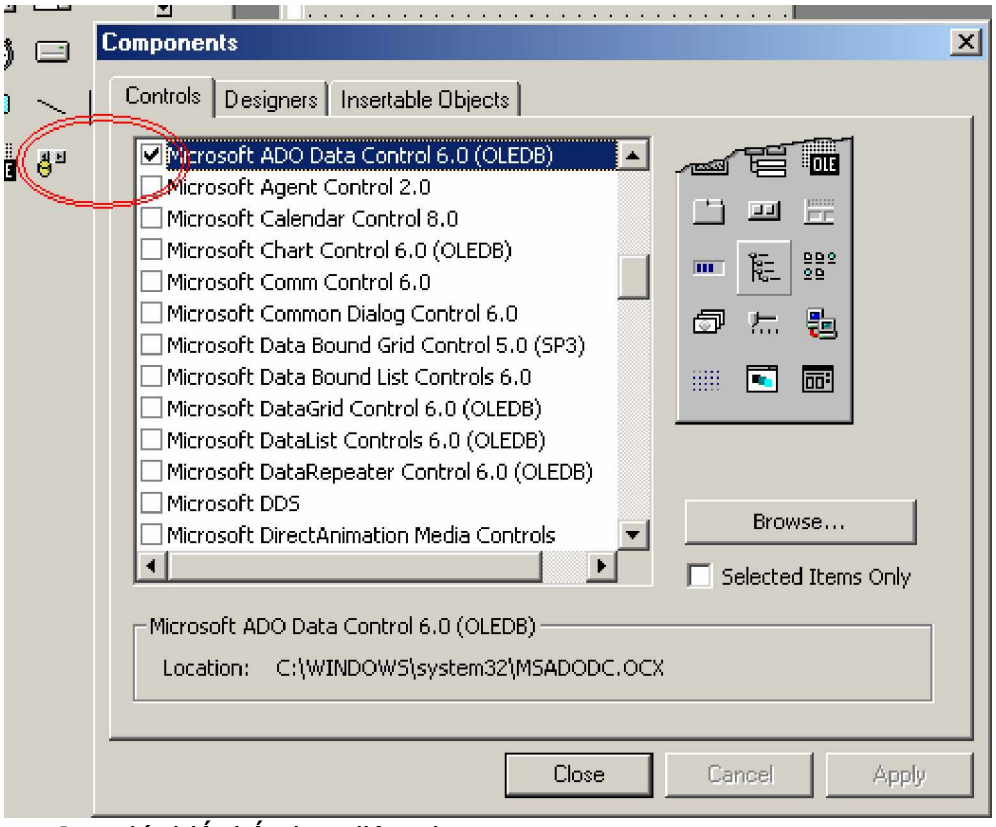
Bài toán : cho CSDL quanlysinhvien.mdb chứa bảng SinhVien có cấu trúc như sau

MaSV	Text
HoSV	Text
TenSV	Text
Makhoa	Text
Ngaysinh	Date/Time

Yêu cầu : sử dụng Adodc để truy cập và làm việc trên CSDL đó.

III.1, Đưa Adodc vào chương trình và thiết kế giao diện

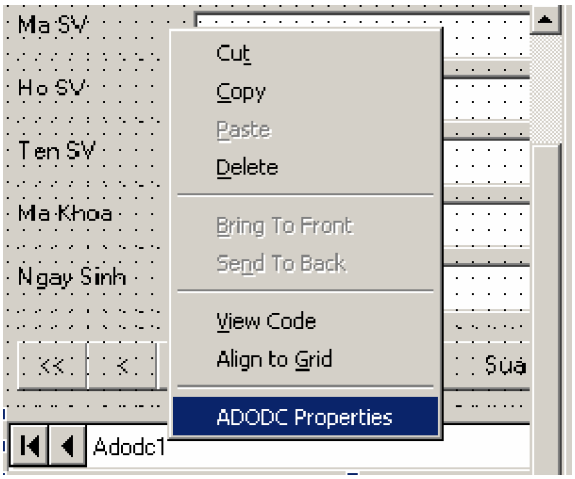
- Vào menu Project/Components.. (Ctrl + T)
- Chọn mục : **Microsoft ADO Data Control 6.0 (OLEDB)**



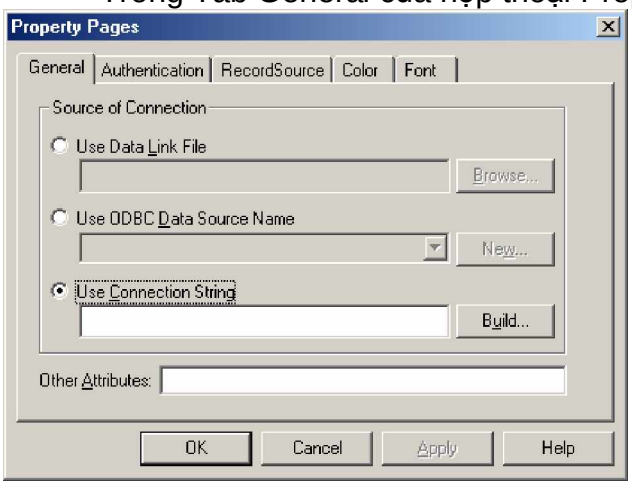
- Sau đó thiết kế giao diện như sau :

III.2, Kết nối cơ sở dữ liệu Access thông qua Adodc

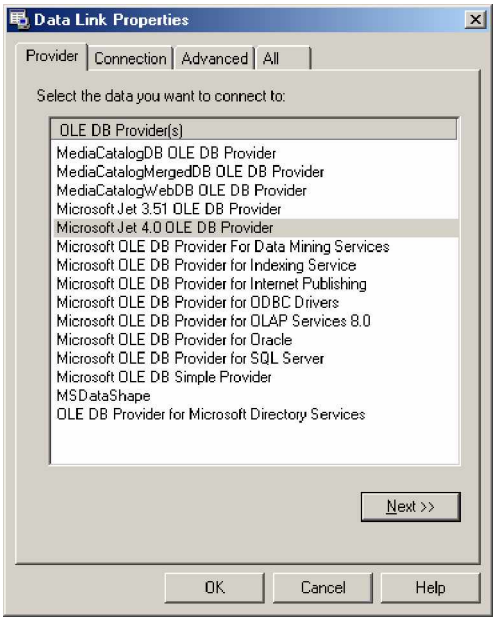
- Click phải chuột vào đối tượng, chọn mục ADODC Properties



- Trong Tab General của hộp thoại Property pages

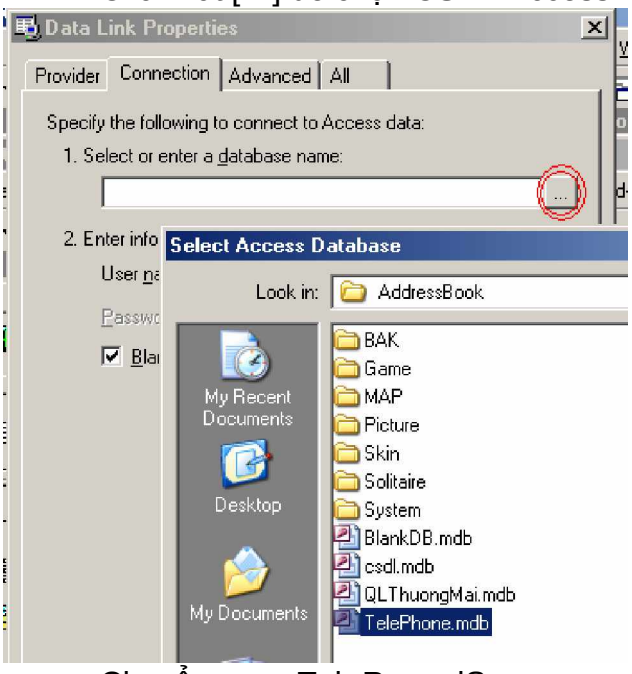


- Click nút Build… (ở Option chọn thứ 3), Trong cửa sổ Data Link Properties, Tab Provider

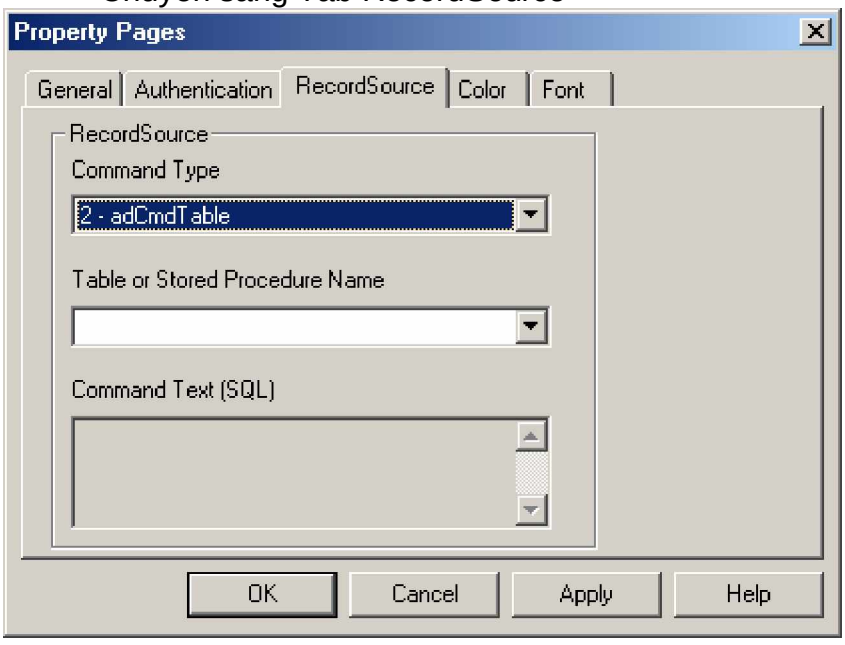


- chọn mục **: Microsoft.Jet 4.0 OLE DB Provider**
- ấn Next, hoặc ấn chuyển sang Tab Connection

- Click nút […] để chọn CSDL Access mong muốn



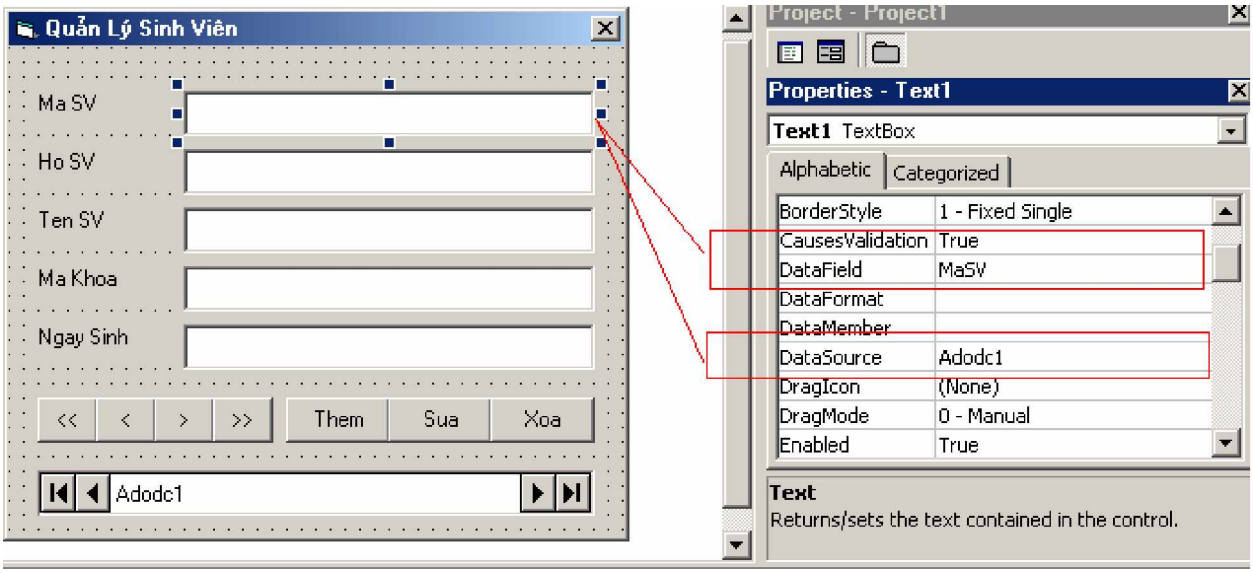
- Chuyển sang Tab RecordSource



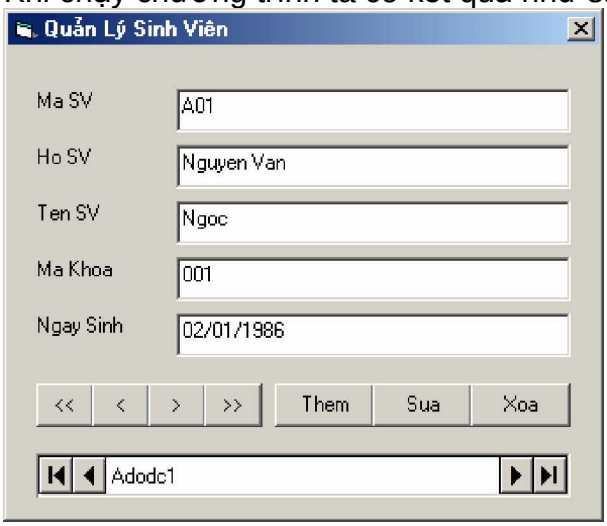
- Command Type : cho phép bạn lựa chọn phương thức sẽ lấy dữ liệu là từ bảng (2-adCmdTable) hay từ câu lệnh truy vấn SQL (1-adCmdText)
- Table or Stored Procedure Name : lựa chọn tên bảng (Table) hoặc tên truy vấn (Query)
- Command Text (SQL) : Bạn tự nhập 1 câu truy vấn SQL bất kỳ để lấy dữ liệu.

III.3, Xuất thông tin thông qua Adodc

- Bạn chọn ô TextBox muốn đưa thông tin ra. Sau đó thiết lập 2 thuộc tính : DataSource (Adodc) và DataFiled (Tên trường)
- Tiến hành việc này với tất cả các Textbox còn lại



Khi chạy chương trình ta có kết quả như sau :



III.4, Cú pháp các câu lệnh di chuyển trên các bản ghi

- Di chuyển về bản ghi đầu tiên

```
Private Sub cmdDau_Click()
    Adodc1.Recordset.MoveFirst
End Sub
```

- Di chuyển về bản ghi phía trước bản ghi hiện hành

```
Private Sub cmdTruoc_Click()
    If Adodc1.Recordset.AbsolutePosition > 1 Then
```

```
        Adodc1.Recordset.MovePrevious
    End If
End Sub
```

- Di chuyển về bản ghi phía sau bản ghi hiện hành

```
Private Sub cmdSau_Click()
    If Adodc1.Recordset.AbsolutePosition < Adodc1.Recordset.RecordCount Then
        Adodc1.Recordset.MoveNext
    End If
End Sub
```

- Di chuyển về bản ghi cuối cùng

```
Private Sub cmdDau_Click()
     Adodc1.Recordset.MoveLast
End Sub
```

- Viết lệnh xem bản ghi hiện hành và tổng số bản ghi (ví dụ : 2 / 28)

```
Private Sub cmdXem_Click()
    MsgBox Adodc1.Recordset.AbsolutePosition & _
          " / " & Adodc1.Recordset.RecordCount
End Sub
```

III.5, Cú pháp các câu lệnh cập nhật dữ liệu

```
Private Sub cmdThem_Click()
    Adodc1.Recordset.AddNew
End Sub

Private Sub cmdSua_Click()
    Adodc1.Recordset.Update
End Sub

Private Sub cmdXoa_Click()
    Adodc1.Recordset.Delete
End Sub
```

IV, ADODB

Thư viện ADODB cung cấp cho người dùng các đối tượng độc lập với nhau để kết nối vào xử lý dữ liệu. Các đối tượng chính của ADODB bao gồm: Connection, Command và RecordSet. Bài này muốn giới thiệu cho sinh viên những chức năng của mỗi đối tượng trong thư viện ADODB, qua đó sinh viên có thể nắm bắt được một cách tổng quát thứ tự các bước lập trình với CSDL sử dụng ADODB

IV.1, Đối tượng Connection

Kết nối với CSDL

- Để truy cập và xử lý dữ liệu trong CSDL, ứng dụng cần phải kết nối với CSDL đó. ADODB cung cấp đối tượng *Connection* để kết nối với CSDL. Với Connection, người dùng có thể kết nối với nhiều loại CSDL khác nhau như Access, SQL Server, Oracle hay Excel, Mail, Thư mục,…

- Các thông tin về CSDL muốn kết nối tới cần phải cung cấp cho đối tượng Connection thông qua thuộc tính *ConnectionString*. *ConnectionString* bao gồm 2 thông tin chính là Provider và Nguồn dữ liệu, các thông tin khác có thể cần cung cấp thêm là Username và Password để có thể truy cập CSDL có bảo mật.

- Sau khi đã gán giá trị cho *ConnectionString*, sử dụng hành động *Open* để mở kết nối.

- Thuộc tính *State* giúp kiểm tra trạng thái của kết nối.
  - adStateOpen [1] : kết nối đang mở
  - adStateClosed [0] : kết nối đang đóng

- Thuộc tính *CursorLocation* giúp chỉ định cách Data Provider cung cấp các chức năng thao tác với CSDL với hai giá trị *adUseServer* và *adUseClient*.

Ví dụ:

```
Public cnn As New ADODB.Connection
Sub ketnoi()
        If cnn.State = 1 Then cnn.Close
cnn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source = " _
        & App.Path & "\CSDL.mdb"
        cnn.CursorLocation = adUseClient
        cnn.Open
        If cnn.State = 1 Then
                MsgBox “Ban da ket noi thanh cong”
        End If
End Sub
```

Thực hiện các câu truy vấn SQL

Sau khi đã kết nối với CSDL, người dùng có thể làm việc ngay với dữ liệu thông qua các câu truy vấn. Hành động *Execcute* của đối tượng Connection giúp thực hiện một câu lệnh SQL

Ví dụ :

```
Sub AddNewUser()
        strSQL = "INSERT INTO users(keyname,pass,fullname) " & _
                " VALUES( " & _
                " '" & Trim(txtUserName.Text) & " '," & _
                " ' " & Trim(txtPassW1.Text) & " ',' " & _
                " ' " & Trim(txtFullName.Text) & " ' )"
        cnn.Execute (strSQL)
End Sub
```

```
Sub UpdateUser()
        strSQL = "UPDATE Users SET " & _
                " keyname=' " & Trim(txtUserName) & " ',' " & _
                " pass=' " & Trim(txtPassW1) & " ',' " & _
                " fullname=' " & Trim(txtFullName) & " ' " & _
                " WHERE keyname = ' " & strKN_Old & " ' "
        cnn.Execute strSQL
End Sub

Sub DeleteUser()
        strSQL = "DELETE FROM Users WHERE keyname ='" & strUN & "'"
        cnn.Execute strSQL
End Sub
```

IV.2, Đối tượng RecordSet

Tạo nguồn dữ liệu cho ứng dụng

- Làm việc trực tiếp với dữ liệu thông qua các câu SQL có thể coi là một hình thức thao tác dữ liệu ở cấp thấy. ADODB cung cấp cho người dùng đối tượng *RecordSet* để thao tác với dữ liệu được dễ dàng.

- RecordSet đáp ứng các yêu cầu của người lập trình như hiển thị dữ liệu, thêm, xoá, sửa dữ liệu, làm việc trên từng dòng dữ liệu thay vì một tập hợp nhiều mẩu tin,… Có thể coi RecordSet như đại diện của một bảng hay một view trong CSDL.

- Người dùng có thể mở RecordSet để lấy dữ liệu từ một bảng hay nhiều bảng trong CSDL bằng một câu truy vấn SQL hay đơn giản bằng cách chỉ ra tên bảng.

- Việc mở RecordSet được thực hiện qua hành động *Open*. RecordSet cũng cung cấp các thuộc tính *CursorType*, *LockType* để người dùng có thể truy cập dữ liệu theo cách phù hợp với ngữ cảnh lập trình. Giá trị của các thuộc tính này có thể được cung cấp trực tiếp qua hành động Open.

Cú Pháp :

<Recordset>.Open [*Source*], [*ActiveConnection*], [*CursorType*], [*LockType*]

Mô Tả :

- **Source** : Nội dung cần truy xuất

- **ActiveConnection** : Nội dung khai báo ConnectionString hoặc tên Connection đang được mở

- **CursorType** : Phân loại recordset. Có các giá trị sau :

- adOpenStatic [3] : mẩu tin tạo tại máy con. Không tự động cập nhật
- adOpenDynamic [2] : mẩu tin tạo trên máy chủ. Tự động cập nhật.
- adOpenKeySet [1] : không tự động cập nhật các mẩu tin
- adOpenForewardOnle [0] : Chỉ di chuyển bằng MoveNext

- **CursorType** : Xác định cách khoá dữ liệu khi cập nhật. Có các giá trị sau :
  - adLockReadOnly : [1] chỉ cho phép đọc (luôn luôn khoá)
  - adLockOptimistic : [2] chỉ khoá khi cập nhật mẩu tin
  - adLockBatchOptimistic : [3] giống [2] nhưng cho phép cập nhật đồng thời nhiều mẩu tin.
  - adLockPessimistic : [4] mẩu tin sẽ khoá ngay khi thực hiện Update hay AddNew. chỉ dùng khi CursorLocation là adUseServer

Ví Dụ :

```
Dim rsSinhVien As New ADODB.Recordset

Sub Lay_Nguon_SinhVien()
    sql = "SINHVIEN"
    If rsSinhVien.State = 1 Then rsSinhVien.Close
    rsSinhVien.Open sql, cnn, 3, 3
End Sub
```

Hiển thị dữ liệu

- RecordSet được dùng làm nguồn dữ liệu (data source) cung cấp dữ liệu cho các control khác để hiển thị thông tin trong chương trình. Hầu hết các control cơ sở như TextBox, Label, CheckBox, Image,… đều có thể liên kết (binding) với RecordSet để tự động hiển thị thông tin có trong mẩu tin hiện hành

- Các control liên kết dữ liệu với RecordSet thông qua hai thuộc tính DataSource và DataField.

Ví dụ :

```
Sub Hien_Thi_DuLieu()
    Call Lay_Nguon_SinhVien
    Set txtMaSVn.DataSource = rsSinhVien
        txtMaSV.DataField = "MaSV"
    Set txtHoTen.DataSource = rsSinhVien
        txtHoTen.DataField = " HoTen"
    Set txtNgaySinh.DataSource = rsSinhVien
        txtNgaySinh.DataField = " NgaySinh"
End Sub
```

Làm việc với từng mẩu tin

- Với câu truy vấn SQL, người dùng chỉ không thể thao tác với từng dòng dữ liệu trong các dòng dữ liệu kết quả. Ngược lại, với RecordSet người dùng luôn làm việc với một mẩu tin duy nhất trong bộ mẩu tin là mẩu tin hiện hành.

Ví dụ :

vị trí bản ghi hiện hành = *rsSinhVien.AbsolutePosition*
tổng số bản ghi trong CSDL = *rsSinhVien.RecordCount*

- Để chọn làm việc với mẩu tin khác trong RecordSet, người dùng có thể sử dụng các hành động : MoveFirst, MoveNext, MoveLast, MovePrevious, Move

Ví dụ :

```
Dim rsSinhVien As New ADODB.Recordset

Private Sub cmdDau_Click()
   rsSinhVien.MoveFirst
End Sub

Private Sub cmdTruoc_Click()
   If rsSinhVien.AbsolutePosition > 1 Then
      rsSinhVien.MovePrevious
   End If
End Sub

Private Sub cmdSau_Click()
   If rsSinhVien.AbsolutePosition < rsSinhVien.RecordCount Then
      rsSinhVien.MoveNext
   End If
End Sub

Private Sub cmdCuoi_Click()
   rsSinhVien.MoveLast
End Sub
```

Cập nhật dữ liệu

- Ngoài việc dùng RecordSet như nguồn dữ liệu để hiển thị, người dùng có thể cập nhật dữ liệu gián tiếp vào CSDL thông qua RecordSet bằng cách sử dụng các hành động AddNew, Delete, Update và CancelUpdate.
  - AddNew : dùng để thêm mới 1 bản ghi
  - Update / UpdateBatch : chấp nhận cập nhật sự thay đổi
  - CancelUpdate / CancelBatch : bỏ qua sự thay đổi
  - Delete : Xoá bỏ bản ghi

- Việc cập nhật dữ liệu vào RecordSet và từ RecordSet xuống CSDL diễn ra không đồng thời, dữ liệu chỉ cập nhật xuống CSDL khi người dùng thực hiện hành động "Update". Cơ chế này cho phép người dùng có cơ hội kiểm tra lại dữ liệu có hợp lệ hay không trước khi lưu thực sự xuống CSDL. RecordSet cung cấp hai cơ chế cập nhật là Update và UpdateBatch được chỉ định vào lúc mở RecordSet.

Ví dụ :

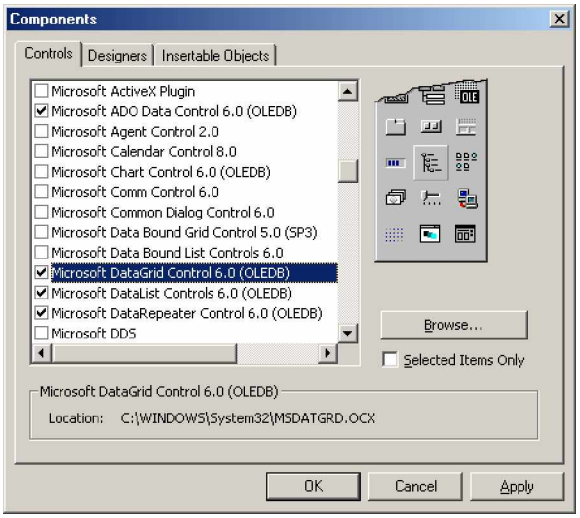
```
Dim rsSinhVien As New ADODB.Recordset

Private Sub cmdThemMoi_Click()
   rsSinhVien.AddNew
End Sub

Private Sub cmdSuaDoi_Click()
   rsSinhVien.Update
End Sub

Private Sub cmdXoaBo_Click()
   rsSinhVien.Delete
End Sub
```

Chương VII : DataGrid – DataList – DataCombo

I, Sử dụng công cụ DataGrid

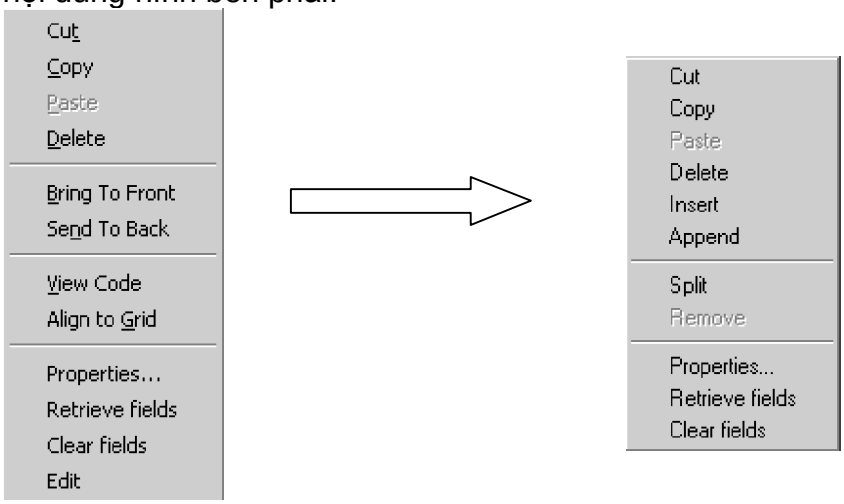
I.1, Đưa DataGrid vào chương trình :

DataGrid là một điều khiển ActiveX, nên muốn sử dụng chúng ta phải đưa vào ứng dụng thông qua chức năng Project – Components – Microsoft DataGrid Control 6.0... Điều khiển này chứa trong tập tin MSDATGRD.OCX



I.2, Thiết kế DataGrid :

DataGrid khi đưa vào form mặc định chỉ có hai cột, muốn tăng giảm cột ta có thể làm như sau : nhấn chuột phải trên DataGrid và chọn Edit trên shortcut menu (hình dưới bên trái). Sau đó, nhấn chuột phải lần thứ hai trên DataGrid, lúc này shortcut menu có nội dung hình bên phải.



Sử dụng các mục :

Delete : để xoá cột đang được chọn trên DataGrid

Insert : để chèn thêm một cột vào trước cột đang được chọn trên DataGrid

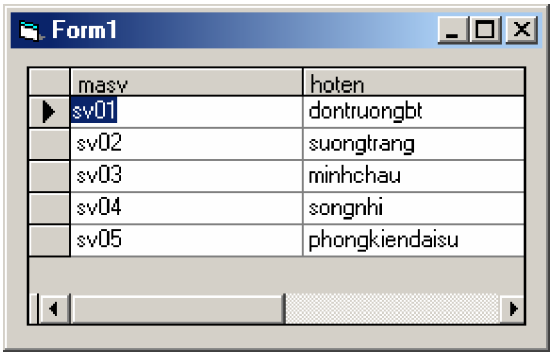
Append : thêm một cột vào vị trí sau cùng trên DataGrid

I.3, Liên kết Recordset với DataGrid

Ta liên kết DataGrid với Recordset qua thuộc tính Datsource

Set <tên DataGrid>.DataSource = <Recordset>

Ví dụ :




```
Dim rsSV As New ADODB.Recordset

Sub LayNguonDG()
   sql = "select * from sinhvien"
   If rsSV.State = 1 Then rsSV.Close
   rsSV.Open sql, cnn, 3, 3
   Set DataGrid1.DataSource = rsSV
End Sub
```

I.4, Truy xuất trị của một ô trên DataGrid

Cách 1 : Trước khi lấy nội dung của một ô bất kỳ trên DataGrid, ta dùng thuộc tính Col và Row để chuyển ô hiện hành trên DataGrid đến ô muốn lấy nội dung và dùng thuộc tính Text để lấy nội dung.

Ví dụ : Muốn lấy trị của ô ở cột thứ 3, dòng thứ 4 (lưu ý cột, dòng đánh số từ 0)

DataGrid1.Col = 2
DataGrid1.Row = 3
MsgBox DataGrid1.Text

Cách 2 : Sử dụng thuộc tính Text,Value của đối tượng Columns(<số>) trên DataGrid để lấy nội dung trên dòng hiện hành

Ví dụ : Muốn lấy trị cột thứ 3 của dòng hiện hành :

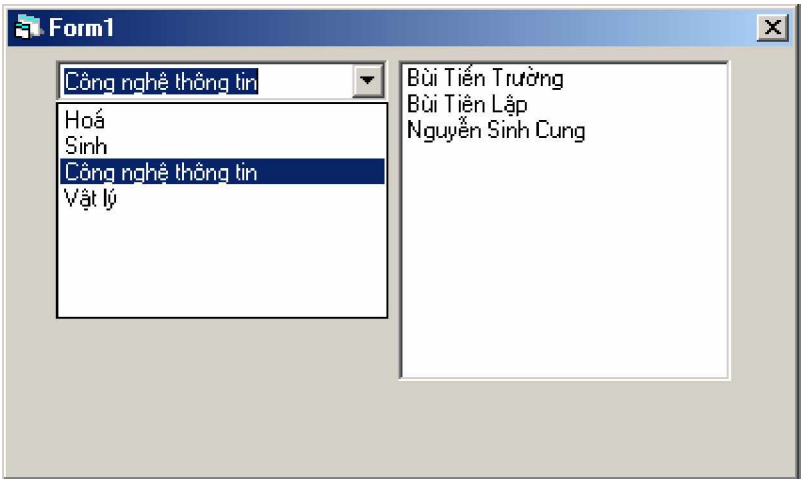
MsgBox DataGrid1.Columns(2).Text *(giá trị hiển thị)*
MsgBox DataGrid1.Columns(2).Value *(giá trị lưu trữ)*

II, Data List – Data Combo

DataList và DataCombo là hai control đặc biệt có khả năng kế t nối với nguồn dữ liệu. Hai control này có thể sử dụng 2 nguồn dữ liệu, một để hiển thị, một để cập nhật dữ liệu.

- Các thuộc tính dùng khi muốn cập nhật dữ liệu :
  DataSource : chứa recordset nguồn
  DataField : chứa Field (trường dữ liệu)
- Các thuộc tính hiển thị dữ liệu
  RowSource : chứa nội dung các mẩu tin
  ListField : chọn trường hiển thị nội dung
  BoundColumn : cột nội dung lưu trữ (không hiển thị)
  BoundText : nội dung lưu trữ (không hiển thị)

Ví dụ : Thiết kế giao diện như dưới, bao gồm 1 DataCombo và 1 DataGrid.
Yêu cầu khi chọn 1 khoa bất kỳ từ DataCombo thì sẽ liệt kê toàn bộ danh sách sinh viên của khoa đó lên DataList bên cạnh.



Mã nguồn :

Dim rsKhoa As New ADODB.Recordset

```
Public cn As New ADODB.Connection
Dim rsSV As New ADODB.Recordset

Public Sub ketnoicsdl()
    Dim strProvider As String, strSource As String
    strProvider = "provider=MICROSOFT.JET.OLEDB.4.0"
    strSource = "Data Source =" & App.Path & "\qlsinhvien.mdb"
    If cn.State = adStateOpen Then cn.Close
    cn.ConnectionString = strProvider & "; " & strSource
    cn.CursorLocation = adUseClient
    cn.Open
End Sub

Private Sub Form_Load()
    Call ketnoicsdl
    rsKhoa.Open "Khoa", cn, adOpenStatic, adLockReadOnly, adCmdTable
    Set cboKhoa.RowSource = rsKhoa
        cboKhoa.ListField = "TenKhoa"
        cboKhoa.BoundColumn = "MaKhoa"
End Sub
Private Sub cboKhoa_Change()
    sql = "select * from sinhvien where makhoa='" & cboKhoa.BoundText & "'"
    If rsSV.State <> adStateClosed Then rsSV.Close
    rsSV.Open sql, cn, adOpenStatic, adLockOptimistic, adCmdText
    Set lstSV.RowSource = rsSV
        lstSV.ListField = "HoTen"
End Sub
```

III, Hướng Dẫn Bài Tập :

III.1, Bài tập 1 : quản lý sinh viên (sử dụng kết hợp DataCombo với TextBox)

Bước 1 : Viết chương trình kết nối CSDL (trong Module)

```
Public cnn As New ADODB.Connection
Public duong_dan As String, strProvider As String

Sub Mo_CSDL()
   duong_dan = App.Path & "\CSDL.MDB"
   strProvider = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source =" & duong_dan
   If cnn.State = 1 Then cnn.Close
   cnn.CursorLocation = adUseClient
   cnn.Open strProvider
End Sub
```

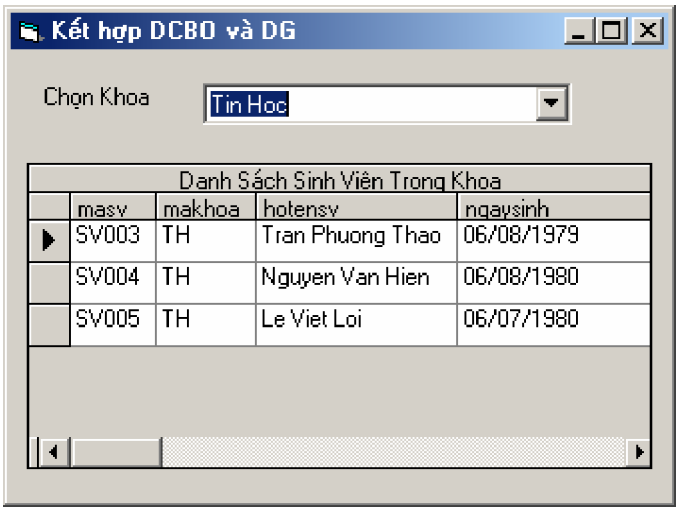
Bước 2 : mã nguồn trong Form

```
Dim rsSinhVien As New ADODB.Recordset
Dim rsKhoa As New ADODB.Recordset

Sub LayNguonKhoa()
   If rsKhoa.State = 1 Then rsKhoa.Close
   sql = "KHOA"
   rsKhoa.Open sql, cnn, 3, 3

   Set dcboKhoa.RowSource = rsKhoa
      dcboKhoa.ListField = "TenKhoa"
      dcboKhoa.BoundColumn = "MaKhoa"
End Sub

Sub LayNguonSinhVien()
   If rsSinhVien.State = 1 Then rsSinhVien.Close
   sql = "SINHVIEN"
   rsSinhVien.Open sql, cnn, 3, 3

   Set txtMasv.DataSource = rsSinhVien
      txtMasv.DataField = "MaSV"
   Set txtHoten.DataSource = rsSinhVien
      txtHoten.DataField = "HoTenSV"
   Set txtNgaysinh.DataSource = rsSinhVien
      txtNgaysinh.DataField = "Ngaysinh"
   Set txtDiachi.DataSource = rsSinhVien
      txtDiachi.DataField = "Diachi"

   Set dcboKhoa.DataSource = rsSinhVien
      dcboKhoa.DataField = "MaKhoa"
End Sub

Private Sub Form_Load()
   Call Mo_CSDL
   Call LayNguonKhoa
```

```
    Call LayNguonSinhVien
End Sub

Private Sub cmdDau_Click()
    rsSinhVien.MoveFirst
End Sub

Private Sub cmdTruoc_Click()
    If rsSinhVien.AbsolutePosition > 1 Then
        rsSinhVien.MovePrevious
    End If
End Sub

Private Sub cmdSau_Click()
    If rsSinhVien.AbsolutePosition < rsSinhVien.RecordCount Then
        rsSinhVien.MoveNext
    End If
End Sub

Private Sub cmdCuoi_Click()
    rsSinhVien.MoveLast
End Sub

Private Sub txtMasv_Change()
On Error Resume Next
    lblVT = rsSinhVien.AbsolutePosition & "/" & rsSinhVien.RecordCount
End Sub
```

III.2, Bài tập 2 : Sử dụng kết hợp DataCombo với DataGrid

DataCombo sẽ liệt kê toàn bộ danh mục các khoa. Khi ta chọn 1 khoa nào đó thì toàn bộ thông tin của các sinh viên khoa đó sẽ được liệt kê hiển thị ra bên DataGrid ở bên dưới.



Bước 1 : Viết chương trình kết nối CSDL (trong Module)

```
Public cnn As New ADODB.Connection
Public duong_dan As String, strProvider As String

Sub Mo_CSDL()
   duong_dan = App.Path & "\CSDL.MDB"
   strProvider = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source =" & duong_dan
   If cnn.State = 1 Then cnn.Close
   cnn.CursorLocation = adUseClient
   cnn.Open strProvider
End Sub
```

Bước 2 : mã nguồn trong Form

```
Dim rsSinhVien As New ADODB.Recordset
Dim rsKhoa As New ADODB.Recordset

Sub LayNguonKhoa()
   If rsKhoa.State = 1 Then rsKhoa.Close
   sql = "KHOA"
   rsKhoa.Open sql, cnn, 3, 3

   Set dcboKhoa.RowSource = rsKhoa
      dcboKhoa.ListField = "TenKhoa"
      dcboKhoa.BoundColumn = "MaKhoa"
End Sub

Private Sub Form_Load()
   Call Mo_CSDL
   Call LayNguonKhoa
End Sub

Sub LayNguonSinhVien()
   Dim strMaKhoa As String
   strMaKhoa = dcboKhoa.BoundText
   If rsSINHVIEN.State = 1 Then rsSINHVIEN.Close
   sql = "SELECT * FROM SinhVien Where Makhoa ='" & strMaKhoa & "'"
   rsSINHVIEN.Open sql, cnn, 3, 3
   Set dgSinhVien.DataSource = rsSINHVIEN
End Sub

Private Sub dcboKhoa_Click(Area As Integer)
   If Area = 2 Then
      Call LayNguonSinhVien
   End If
End Sub
```

MỤC LỤC

Chương I : Tổng quan về Visual Basic 02
I, Khởi động chương trình VB 02
II, Giới thiệu giao diện cửa sổ làm việc VB 03
II.1, Thanh công cụ 03
II.2, Cửa sổ Project Explorer 04
II.3, Thanh thuộc tính 04
III, Tạo/Lưu Project làm việc 04
III.1, Tạo Project 04
III.2, Lưu Project 05
IV, Các bước cơ bản xây dựng 1 chương trình 06
IV.1, Tạo giao diện 06
IV.2, Đặt thuộc tính cho điều khiển 07
IV.3,Viết mã lệnh 07
IV.4, Chạy chương trình 08
Chương II : Form và Control 08
I, Thuộc tính, sự kiện, phương thức 08
II, Các điều khiển cơ sở 09
II.1, Các thuộc tính chung của điều khiển 09
II.2, Công cụ điều khiển TextBox 10
II.3, Command Button, CheckBox và Option Group 13
II.4, List Box và Combo Box 16
II.5, Picture Box và Image 18
II.6, Timer 19
II.7, Frame, Label, Shape và Line 20
Chương III : Ngôn Ngữ Lập Trình Visual Basic 22
I, Sử dụng Code Editor : 22
II, Qui ước viết lệnh trong Visual Basic 23
II.1, Chia 1 lệnh thành nhiều dòng : Bằng cách sử dụng ký tự: [_] 23
II.2, Nối nhiều lệnh vào 1 dòng : Bằng cách sử dụng kí tự: [:] 23
II.3, Thêm chú giải vào mã lệnh : Bằng cách sử dụng kí tự: [‘] 24
II.4, Hệ thống số 24
II.5, Qui tắc đặt tên 24
II.6, Một số khái niệm 25
III, Biến - Hằng - Kiểu Dữ Liệu 26
III.1, Biến số 26
III.2, Hằng số 27
III.3, Kiểu dữ liệu 27
IV, Các cấu trúc điều khiển 28
IV.1, Các lệnh rẽ nhánh 28
IV.2, Cú pháp vòng lặp 31
V, Hàm - Thủ Tục 34
V.1, Thủ Tục 34
V.2, Hàm 35
Chương IV : Mảng – Record 37

I, Tổng quan về mảng 37
I.1, Khái niệm, cấu trúc và ý nghĩa sử dụng mảng 37
I.2, Khai báo mảng và cách truy xuất đến phần tử mảng 37
I.3, Một số giải thuật trên mảng 38
II, Các hàm thao tác trên mảng 42
II.1, Hàm Array() 42
II.2, Hàm Choose() 43
II.3, Hàm Split() 43
II.4, Hàm Join() 43
III, Kiểu Dữ Liệu Record 44
III.1, Khái niệm 44
III.2, Ví dụ 44
Chương V : Menu Editor – MDI Form 45
I, Menu Editor 45
I.1, Giới thiệu về hệ thống thực đơn của một ứng dụng 45
I.2, Thiết kế hệ thống thực đơn cho ứng dụng 46
II, MDI Form : 49
II.1, Màn hình SDI và màn hình MDI 49
II.2, Màn hình MDI và màn hình MDI Child 49
II.3, Các bước đưa ứng dụng giao diện MDI vào chương trình 50
II.4, Tạo thực đơn Window trong các ứng dụng MDI 52
Chương VI : Kết Nối Cơ Sở Dữ Liệu Với ADO 53
I, Tổng quan về lập trình cơ sở dữ liệu 53
I.1, Hệ quản trị CSDL 53
I.2, Các kỹ thuật lập trình với CSDL 53
II, Giới thiệu về Microsoft Active Data Object (ADO) 54
III, Đối tượng ADODC 55
III.1, Đưa Adodc vào chương trình và thiết kế giao diện 55
III.2, Kết nối cơ sở dữ liệu Access thông qua Adodc 56
III.3, Xuất thông tin thông qua Adodc 59
III.4, Cú pháp các câu lệnh di chuyển trên các bản ghi 59
III.5, Cú pháp các câu lệnh cập nhật dữ liệu 60
IV, ADODB 60
IV.1, Đối tượng Connection 60
IV.2, Đối tượng RecordSet 62
Chương VII : DataGrid – DataList – DataCombo 65
I, Sử dụng công cụ DataGrid 65
I.1, Đưa DataGrid vào chương trình 65
I.2, Thiết kế DataGrid 66
I.3, Liên kết Recordset với DataGrid 66
I.4, Truy xuất trị của một ô trên DataGrid 66
II, Data List – Data Combo 67
III, Hướng Dẫn Bài Tập 68
III.1, Bài tập 1 68
III.2, Bài tập 2 70