KS. Lâm Hoài Bảo

GIÁO TRÌNH

Visual Basic

Ebook.moet.gov.vn, 2008

CHƯƠNG 1 TỔNG QUAN VỀ VISUAL BASIC 6.0

Mục tiêu:

Chương này giới thiệu về môi trường phát triển tích hợp (IDE) Microsoft Visual Basic 6.0; cũng như giúp sinh viên có cái nhìn tổng quan về Visual Basic.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Sử dụng môi trường phát triển tích hợp VB 6.0 để phát triển ứng dụng.
- Cách tạo dự án mới (New Project) trong VB 6.0.

Kiến thức có liên quan:

- Sử dụng hệ điều hành Windows.

Tài liệu tham khảo:

- **Visual Basic 6 Certification Exam Guide** - Chapter 1, Page 1 - **Dan Mezick & Scot Hillier** - **McGraw-Hill** - **1998.**

I. Giới thiệu về Visual Basic 6.0

Visual Basic 6.0 (VB6) là một phiên bản của bộ công cụ lập trình Visual Basic (VB), cho phép người dùng tiếp cận nhanh cách thức lập trình trên môi trường Windows. Những ai đã từng quen thuộc với VB thì tìm thấy ở VB6 những tính năng trợ giúp mới và các công cụ lập trình hiệu quả. Người dùng mới làm quen với VB cũng có thể làm chủ VB6 một cách dễ dàng.

Với VB6, chúng ta có thể :

- Khai thác thế mạnh của các điều khiển mở rộng.
- Làm việc với các điều khiển mới (ngày tháng với điều khiển MonthView và DataTimePicker, các thanh công cụ có thể di chuyển được CoolBar, sử dụng đồ họa với ImageCombo, thanh cuộn FlatScrollBar,…).
- Làm việc với các tính năng ngôn ngữ mới.
- Làm việc với DHTML.
- Làm việc với cơ sở dữ liệu.
- Các bổ sung về lập trình hướng đối tượng.

II. Cài đặt Visual Basic 6.0

Sử dụng chương trình Setup, người dùng có thể cài đặt VB6 lên máy tính của mình. Chương trình Setup này còn cài đặt các tập tin cần thiết để xem tài liệu trên đĩa CD MSDN (Microsoft Developer Network). Nếu cần, người dùng có thể cài đặt riêng phần tài liệu và ví dụ mẫu của Visual Basic lên máy tính.

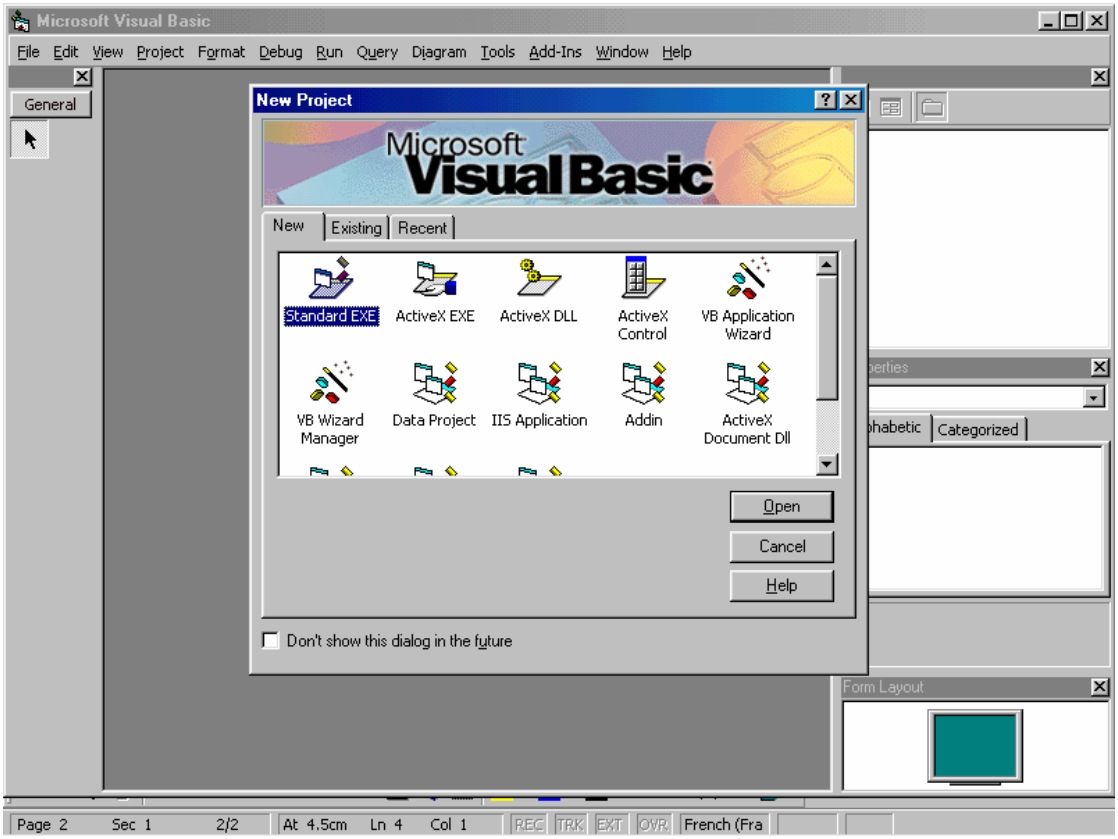
Để cài đặt VB6, người dùng nên kiểm tra máy tính của mình đảm bảo được cấu hình tối thiểu. Các yêu cầu hệ thống tối thiểu :

- Microsoft Windows 95 trở lên hoặc là Microsoft Windows NT Workstation 4.0 trở lên.
- Tốc độ CPU 66 MHz trở lên.
- Màn hình VGA hoặc màn hình có độ phân giải cao được hỗ trợ bởi Microsoft Windows.
- 16 MB RAM cho Microsoft Windows 95 hoặc 32MB RAM cho Microsoft Windows NT Workstation.

III. Làm quen với VB6

III.1 Bắt đầu một dự án mới với VB6

Từ menu Start chọn Programs, Microsoft Visual Basic 6.0. Khi đó bạn sẽ thấy màn hình đầu tiên như hình I.1 dưới đây.



Hình I.1 Cửa sổ khi kích hoạt VB6

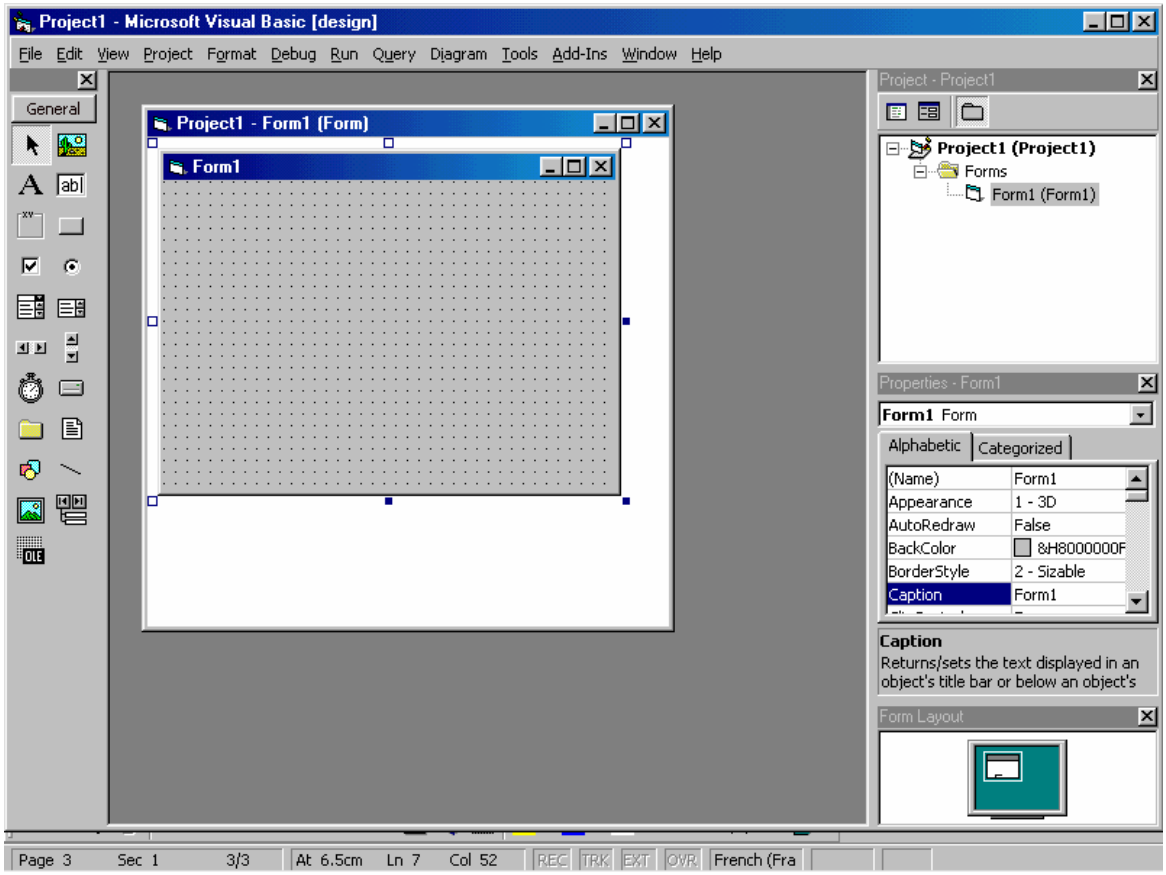
Ở đây, người dùng có thể chọn tạo mới một dự án thực thi được bằng cách chọn Standard EXE rồi nhấp Open (Hình I.2).

Tiếp theo là cửa sổ làm việc chính của VB6, gọi tắt là IDE (Integrated Development Environment) sẽ được giới thiệu chi tiết trong phần sau.

III.2 Tìm hiểu các thành phần của IDE

IDE là tên tắt của môi trường phát triển tích hợp (Integrated Development Environment), đây là nơi tạo ra các chương trình Visual Basic.

IDE của Visual Basic là nơi tập trung các menu, thanh công cụ và cửa sổ để tạo ra chương trình. Mỗi một thành phần của IDE có các tính năng ảnh hưởng đến các hoạt động lập trình khác nhau.



Hình I.3 Cửa sổ IDE của VB6

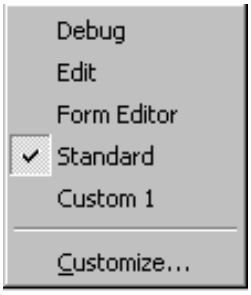
Thanh menu cho phép bạn tác động cũng như quản lý trực tiếp trên toàn bộ ứng dụng. Bên cạnh đó thanh công cụ cho phép truy cập các chức năng của thanh menu thông qua các nút trên thanh công cụ.

Các biểu mẫu (Form) - **khối xây dựng chương trình chính của VB** - xuất hiện trong cửa sổ Form. Hộp công cụ để thêm các điều khiển vào các biểu mẫu của đề án. Cửa sổ Project Explorer hiển thị các đề án khác nhau mà người dùng đang làm cũng như các phần của đề án. Người dùng duyệt và cài đặt các thuộc tính của điều khiển, biểu mẫu và module trong cửa sổ Properties. Sau cùng, người dùng sẽ xem xét và bố trí một hoặc nhiều biểu mẫu trên màn hình thông qua cửa sổ Form Layout.

III.3 Sử dụng thanh công cụ trong IDE của VB

Thanh công cụ là tập hợp các nút bấm mang biểu tượng thường đặt dưới thanh menu. Các nút này đảm nhận các chức năng thông dụng của thanh menu (New, Open, Save ...).

Hình I.4 Thanh công cụ ở dạng standard



Hình I.5 Popup menu thêm, xóa công cụ

Hơn nữa, người dùng có thể kéo rê thanh công cụ trên IDE đến vị trí bất kỳ nào đó thuận tiện cho việc sử dụng.

Người dùng có thể thêm hay xóa thanh công cụ trên IDE:

* Chọn Toolbars từ menu View hoặc ấn chuột phải vào điểm bất kỳ nào trên thanh menu, một popup menu bật ra.

* Chọn loại thanh công cụ mà ta muốn thêm vào hoặc xóa đi. Nếu có đánh dấu check ở bên trái thì loại công cụ đó đang được chọn.

Sử dụng thanh công cụ gỡ rối (debug)

Hình I.6 Thanh công cụ gỡ rối

Với thanh công cụ gỡ rối, người dùng có thể thực thi, tạm ngưng hoặc dừng một đề án. Với thanh công cụ Debug, người dùng có thể kiểm tra chương trình và giải quyết các lỗi có thể xảy ra. Khi gỡ rối chương trình, người dùng có thể chạy từng dòng lệnh, kiểm tra giá trị các biến, dừng chương trình tại một điểm nào đó hoặc với một điều kiện nào đó.

Sử dụng thanh công cụ Edit

Hình I.7 Thanh công cụ Edit

Thanh công cụ Edit được dùng để viết chương trình trong cửa sổ Code, thanh công cụ Edit có đầy đủ các tính năng của menu Edit. Ngoài ra người sử dụng có thể sử dụng chức năng viết chương trình tự động như là Quick Info.

Thanh công cụ Edit của VB6 có tính năng lý thú đó là tự hoàn tất các từ khóa. Tính năng này rất hữu dụng giúp cho người dùng tránh các lỗi mắc phải do gõ sai từ khóa.

Sử dụng thanh công cụ Form Editor

Hình I.8 Thanh công cụ thiết kế biểu mẫu

Thanh công cụ Form Editor có chức năng giống như menu Format dùng để di chuyển và sắp xếp các điều khiển trên biểu mẫu.

Trong quá trình thiết kế biểu mẫu, đôi khi chúng ta phải sử dụng thuộc tính ZOrder để cho phép một điều khiển có thể thay thế một điều khiển khác hay không hoặc là xuất hiện bên trên một điều khiển khác hay không.

Sử dụng hộp công cụ (Toolbox)

Hộp công cụ là nơi chứa các điều khiển được dùng trong quá trình thiết kế biểu mẫu. Các điều khiển được chia làm hai loại: Điều khiển có sẵn trong VB và các điều khiển được chứa trong tập tin với phần mở rộng là .OCX.

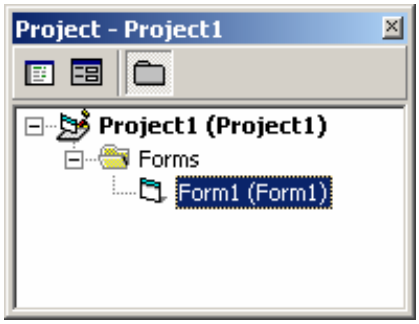
Đối với các điều khiển có sẵn trong VB thì ta không thể gỡ bỏ khỏi hộp công cụ, trong khi đó đối với điều khiển nằm ngoài ta có thêm hoặc xóa bỏ khỏi hộp công cụ.

Một điều khiển có thể được đưa vào biểu mẫu bằng cách chọn điều khiển đó và đưa vào biểu mẫu. Chúng ta sẽ trở lại phần này trong chương tiếp theo khi thiết kế các biểu mẫu.

Hình I.9 Hộp công cụ của Visual Basic

III.4 Quản lý ứng dụng với Project Explorer

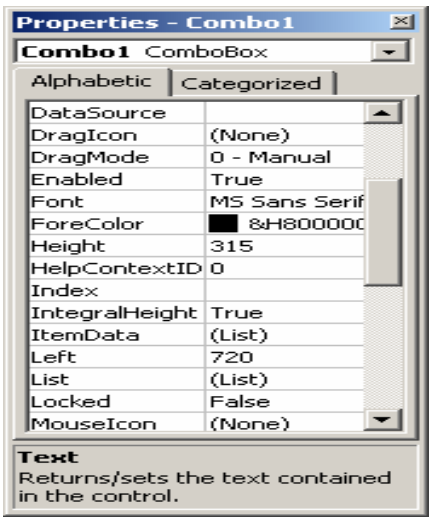
Project Explorer trong VB6 giúp quản lý và định hướng nhiều đề án.VB cho phép nhóm nhiều đề án trong cùng một nhóm. Người dùng có thể lưu tập hợp các đề án trong VB thành một tập tin nhóm đề án với phần mở rộng **.vbp**.



Hình I.10 Cửa sổ Project Explorer

Project Explorer có cấu trúc cây phân cấp như cây thư mục trong cửa sổ Explorer của hệ điều hành. Các đề án có thể được coi là gốc của cây, các thành phần của đề án như biểu mẫu, module ... là các nút của cây. Khi muốn làm việc với thành phần nào thì ta có thể nhấn đúp lên thành phần đó trên cửa sổ Project Explorer để vào cửa sổ viết code cho thành phần đó.

Khi làm việc với một dự án lớn, chúng ta sẽ thấy Project Explorer cực kỳ hữu ích cho việc tổ chức và quản lý một dự án lớn.



III.5 Cửa sổ Properties

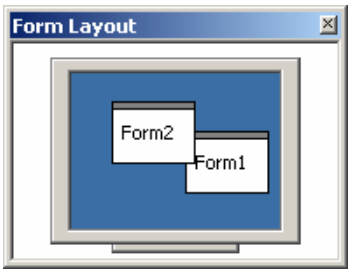
Mỗi một thành phần, điều khiển điều có nhiều thuộc tính. Mỗi một thuộc tính lại có một hoặc nhiều giá trị.

Cửa sổ Properties cho phép người dùng xem, sửa đổi giá trị các thuộc tính của điều khiển nhằm giúp điều khiển hoạt động theo đúng ý đồ của người sử dụng.

III.6 Cửa sổ Form Layout

Đây chính là cửa sổ trình bày biểu mẫu cho phép định vị trí của một hoặc nhiều biểu mẫu trên màn hình khi chương trình ứng dụng được thi hành.

Ta định vị một biểu mẫu trên màn hình bằng cách dùng chuột di chuyển biểu mẫu trong cửa sổ Form Layout.



Hình I.12 Cửa sổ Form

Sử dụng cửa sổ Form Layout không đơn giản như các cửa sổ khác vì nó không được kích hoạt sẵn, người dùng cần phải chạy ứng dụng sau đó mới có thể bố trí được các biểu mẫu thông qua Form Layout.

Nếu ta không định vị các biểu mẫu thì vị trí của biểu mẫu trên màn hình lúc thiết kế cũng là vị trí khởi động của biểu mẫu khi thực thi.

III.7 Biên dịch đề án thành tập tin thực thi

Sau khi đề án đã hoàn thành, người dùng có thể biên dịch thành tập tin thực thi được. Cách tiến hành như sau:

- Trước tiên ta cần chỉ cho VB6 biết phần chương trình nào sẽ được thực thi trước bằng cách chọn Project Properties từ menu Project. Chọn tab General, chú ý phần Startup Object, đây là nơi quy định điểm khởi đầu của chương trình sau khi biên dịch kết thúc.

- Từ menu File, chọn Make ... EXE... Một hộp thoại xuất hiện cho phép bạn nhập vào tên của tập tin thực thi. Bạn chỉ cần gõ tên tập tin, VB sẽ tự động thêm phần mở rộng .EXE.

- Nhấn vào nút Options để mở hộp thoại Project Properties và điền tên của ứng dụng vào ô Title, ta có thể ghi chú thông tin cho từng phiên bản trong phần Version Information. Ta có thể chọn Auto Increment để VB tự động tăng số Revision mỗi lần ta tạo lại tập tin EXE cho dự án.

- Cuối cùng, nhấn OK để trở về hộp thoại Make Project.

CHƯƠNG 2 BIỂU MẪU VÀ MỘT SỐ ĐIỀU KHIỂN THÔNG DỤNG

Mục tiêu:

Chương này giới thiệu về một số điều khiển cơ bản để tạo nên giao diện cho các ứng dụng cũng như một số khái niệm trong lập trình với VB; những yêu cầu tối thiểu cần có trong việc "lập trình sự kiện" với VB.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Khái niệm về điều khiển, thuộc tính, phương thức, sự kiện.
- Quy tắc đặt tên danh biểu trong VB.
- Sử dụng biểu mẫu trong thiết kế giao diện.
- Sử dụng điều khiển ô nhập liệu, nút nhấn, nhãn, khung.

Kiến thức có liên quan:

- Cách thức sử dụng môi trường phát triển VB.

Tài liệu tham khảo:

- **Microsoft Visual Basic 6.0 và Lập trình Cơ sở dữ liệu** - Chương 2, trang 26; Chương 3, trang 29 - **Nguyễn Thị Ngọc Mai** (chủ biên), **Nhà xuất bản Giáo dục** - **2000**.

I. Các khái niệm

✓ **Điều khiển**: Các thành phần có sẵn để người lập trình tạo giao diện tương tác với người dùng.

Mỗi điều khiển thực chất là một đối tượng, do vậy nó sẽ có một số điểm đặc trưng cho đối tượng, chẳng hạn như các thuộc tính, các phương thức & các sự kiện.

✓ **Thuộc tính**: Các đặc trưng của một điều khiển tạo nên dáng vẻ của điều khiển đó.

✓ **Phương thức**: Các điều khiển có thể thực thi một số tác vụ nào đó, các tác vụ này được định nghĩa sẵn bên trong các phương thức (còn gọi là chương trình con: hàm & thủ tục), người lập trình có thể gọi thực thi các phương thức này nếu cần.

✓ **Sự kiện**: là hành động của người dùng tác động lên ứng dụng đang thực thi.

Thí dụ: - Nhấn phím bất kỳ trên bàn phím.

- Nhấp chuột.

Các thành phần giao diện có khả năng đáp ứng lại sự kiện. Chẳng hạn khi chúng ta nhấp chuột vào button, lúc đó button nhận biết được sự kiện này; hay như textbox nhận biết được sự kiện bàn phím tác động lên nó.

Một ứng dụng trên Windows thường được thực hiện nhờ vào việc đáp ứng lại các sự kiện của người dùng.

✓ **Lập trình sự kiện**:

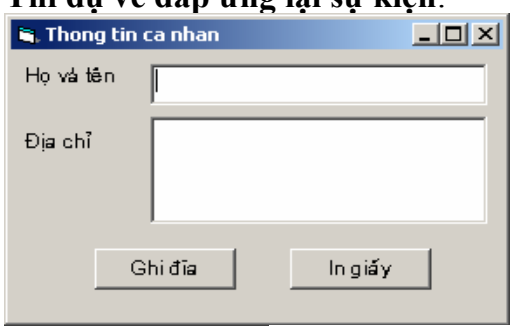
Các thành phần giao diện có khả năng nhận biết được các sự kiện từ phía người dùng. Tuy nhiên khả năng đáp ứng lại các sự kiện được thực hiện bởi người lập trình.

Khi một thành phần giao diện được sử dụng, người lập trình phải xác định chính xác hành động của thành phần giao diện đó để đáp ứng lại một sự kiện cụ thể. Lúc đó người lập trình phải viết đoạn mã lệnh mà đoạn mã lệnh này sẽ được thực thi khi sự kiện xảy ra.

Chẳng hạn, trong ứng dụng Paint của Windows; khi người sử dụng nhấp chuột vào nút vẽ hình elip sau đó dùng chuột vẽ nó trên cửa sổ vẽ, một hình elip được vẽ ra.

Trong lập trình sự kiện, một ứng dụng được xây dựng là một chuỗi các đáp ứng lại sự kiện. Tất cả các hành động của ứng dụng là đáp ứng lại các sự kiện. Do vậy người lập trình cần phải xác định các hành động cần thiết của ứng dụng; phân loại chúng; sau đó viết các đoạn mã lệnh tương ứng.

Thí dụ về đáp ứng lại sự kiện:



Mã lệnh

- Mã lệnh cho sự kiện Click của Ghi đĩa.

- Mã lệnh cho sự kiện Click của In giấy

Hình II.1: Thí dụ về đáp ứng sự kiện

- Khi người dùng không tác động vào ứng dụng, ứng dụng không làm gì cả.

- Khi người dùng nhập dữ liệu vào các ô nhập Họ và tên, Địa chỉ; sự kiện bàn phím xảy ra trên các ô nhập. Tuy nhiên, ứng dụng vẫn không làm gì cả vì không có đoạn mã lệnh nào đáp ứng các sự kiện này.

- Khi người dùng nhấp nút chọn Ghi đĩa, ứng dụng tìm kiếm trong mã lệnh của mình thấy có đoạn mã lệnh đáp ứng lại sự kiện này; lúc đó đoạn mã lệnh được thực thi.

- Tương tự như vậy đối với nút chọn In giấy.

✓ **Cách xác lập các thuộc tính & các phương thức trong chương trình**

<*Thuộc tính Name của điều khiển***>.<Tên thuộc tính>**

<*Thuộc tính Name của điều khiển***>.<***Tên phương thức***>[(<*Các tham số*>)]**

✓ **Tên điều khiển (thuộc tính Name)**

Đây là thuộc tính xác định tên của điều khiển trong ứng dụng. Tên này được đặt theo quy tắc:

- Tên có thể dài từ 1 - 40 ký tự.
- Tên phải bắt đầu với ký tự chữ, có thể chữ hoa hay thường.
- Sau ký tự đầu tiên, tên có thể chứa ký tự, số hay dấu gạch dưới.

Ví dụ: Num, StudentCode, Class12A2 là những tên hợp lệ. 345, 7yu là những tên không hợp lệ.

II. Biểu mẫu (Form)

II.1. Khái niệm:

Chương trình ứng dụng giao tiếp với người dùng thông qua các biểu mẫu (hay còn gọi là cửa sổ, xuất phát từ chữ Form hay Windows); các điều khiển (Control) được đặt lên bên trên giúp cho biểu mẫu thực hiện được công việc đó.

Biểu mẫu là các cửa số được lập trình nhằm hiển thị dữ liệu và nhận thông tin từ phía người dùng.

II.2. Thuộc tính

- **Name**: thuộc tính này như là một định danh nhằm xác định tên của biểu mẫu là gì? Ta sẽ sử dụng thuộc tính này để truy xuất đến các thuộc tính khác cùng với phương thức có thể thao tác được trên biểu mẫu.
- **Caption**: chuỗi hiển thị trên thanh tiêu đề của biểu mẫu.
- **Icon**: hình icon được dùng trong thanh tiêu đề của biểu mẫu, nhất là khi biểu mẫu thu nhỏ lại.
- **WindowState**: xác định biểu mẫu sẽ có kích thước bình thường (Normal=0), hay Minimized (=1), Maximized =(2).

- **Font**: xác lập Font cho biểu mẫu. Thuộc tính này sẽ được các điều khiển nằm trên nó thừa kế. Tức là khi ta đặt một điều khiển lên biểu mẫu, thuộc tính Font của điều khiển ấy sẽ tự động trở nên giống y của biểu mẫu.

- **BorderStyle**: xác định dạng của biểu mẫu.

II.3. Phương thức

- **Move**: di chuyển biểu mẫu đến tọa độ X,Y: **Move X, Y**.

II.4. Sự kiện

- **Form_Initialize**: Sự kiện này xảy ra trước nhất và chỉ một lần thôi khi ta tạo ra thể hiện đầu tiên của biểu mẫu. Ta dùng sự kiện Form_Initialize để thực hiện những gì cần phải làm chung cho tất cả các thể hiện của biểu mẫu này.

- **Form_Load**: Sự kiện này xảy ra mỗi lần ta gọi thể hiện một biểu mẫu. Nếu ta chỉ dùng một thể hiện duy nhất của một biểu mẫu trong chương trình thì Form_Load coi như tương đương với Form_Initialize.

  Ta dùng sự kiện Form_Load để *khởi tạo các biến, điều khiển* cho các thể hiện của biểu mẫu này.

- **Form_Activate**: Mỗi lần một biểu mẫu được kích hoạt (active) thì một sự kiện Activate phát sinh. Ta thường dùng sự kiện này để cập nhật lại giá trị các điều khiển trên biểu mẫu.

- **Form_QueryUnload**: Khi người sử dụng chương trình nhấp chuột vào nút **X** phía trên bên phải để đóng biểu mẫu thì một sự kiện QueryUnload được sinh ra. Đoạn chương trình con dưới đây mô tả thủ tục xử lý sự kiện QueryUnload.

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


End Sub
```

  Sự kiện này cho ta khả năng hủy bỏ hành động đóng biểu mẫu bằng cách đặt lại Cancel là 1.

- **Form_Resize**: Sự kiện này xảy ra mỗi khi biểu mẫu thay đổi kích thước.

III. Nhãn (Label)

III.1. Khái niệm:

Nhãn là điều khiển dạng đồ họa cho phép người sử dụng hiển thị chuỗi ký tự trên biểu mẫu nhưng họ không thể thay đổi chuỗi ký tự đó một cách trực tiếp.

Biểu tượng (shortcut) trên hộp công cụ: 

III.2. Thuộc tính:

- **Name**: Đây là một tên xác định một định danh, người lập trình có thể thay đổi tên này theo cách của mình để tiện sử dụng.

- **Caption**: Thuộc tính quy định chuỗi ký tự hiển thị khi ta tạo một điều khiển nhãn. Khi ta tạo mới một điều khiển thì thuộc tính Caption có giá trị mặc nhiên là "`Label…`".

Ví dụ: Ta muốn tạo một nhãn là "Chào mừng bạn đến với Visual Basic", ta thay đổi giá trị của thuộc tính Caption thành "Chào mừng bạn đến với Visual Basic".

Ta có thể thay đổi giá trị của thuộc tính Caption tại thời điểm ứng dụng đang chạy nhờ vào đoạn mã lệnh đơn giản như sau:

`L1.Caption = "Đã đổi giá trị Caption"` với L1 là tên của điều khiển nhãn mà ta muốn đổi.

- **Font, Fore Color**: Quy định kiểu chữ, kích thước, màu hiển thị.
- **BackStyle, BackColor**: BackStyle quy định là nhãn trong suốt hay không. BackColor quy định màu nền của nhãn trong trường hợp không trong suốt.

III.3. Phương thức:

- **Move**: di chuyển nhãn đến tọa độ X,Y: **Move X, Y**.

III.4. Sự kiện:

- **Change**: Xảy ra mỗi khi nhãn thay đổi giá trị.
- **Click**: Mỗi khi nhãn được chuột nhấp lên, sự kiện này xảy ra.
- **DblClick:** Xảy ra khi người sử dụng nhấp đúp chuột lên điều khiển nhãn.

IV. Khung (Frame)

IV.1. Khái niệm:

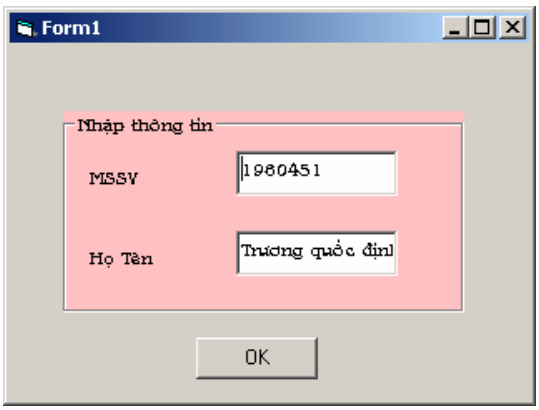
Khung là một điều khiển dùng trong việc bố trí giao diện của biểu mẫu một cách trong sáng và rõ nét. Thông thường các điều khiển cùng phục vụ cho một công việc nào đó sẽ được đặt trong một khung nhằm làm nổi bật vai trò của chúng.

Biểu tượng (shortcut) trên hộp công cụ: 

Khi chúng ta tạo mới một khung để chứa các điều khiển khác, ta có hai cách thực hiện:

- Tạo khung chứa trước, sau đó đưa các điều khiển vào trong khung chứa. Đây là cách đơn giản nhất.

- Tạo khung chứa sau khi đã tạo mới các điều khiển, khi đó khung chứa sẽ che mất các điều khiển, vì vậy ta cần phải đưa khung chứa ra sau các điều khiển bằng cách nhấp chuột phải và chọn **Send to Back**. Nhưng đối với cách này, các điều khiển khác không nằm trên khung chứa. Do vậy ta có thể giải quyết bằng cách *cắt (**Cut**) các điều khiển này đi, sau đó dán (**Paste**) vào trong khung chứa*.



Hình II.2 Ví dụ về khung chứa (Frame)

IV.2. Thuộc tính*:*

Khung cũng có các thuộc tính thông dụng như của điều khiển nhãn chẳng hạn như: Name, Caption,...

IV.3. Phương thức:

- **Move**: di chuyển khung đến tọa độ X,Y: **Move X, Y**.

IV.4. Sự kiện:

- **Click, DblClick**: xảy ra khi khung nhận được một thao tác nhấp (nhấp đúp) chuột.

V. Nút lệnh (Command Button)

V.1. Khái niệm:

Nút lệnh là một điều khiển dùng để bắt đầu, ngắt hoặc kết thúc một quá trình. Khi nút lệnh được chọn thì nó trông như được nhấn xuống, do đó nút lệnh còn được gọi là nút nhấn (Push Button). Người sử dụng luôn có thể chọn một nút lệnh nào đó bằng cách nhấn chuột trên nút lệnh đó.

Biểu tượng (shortcut) trên hộp công cụ: 

V.2. Thuộc tính:

- **Name**: sử dụng như một định danh nhằm xác định tên của nút lệnh.
- **Caption**: Dùng để hiển thị một chuỗi nào đó trên nút lệnh.
- **Default**: Nếu giá trị của thuộc tính này là True thì ta có thể chọn nút lệnh bằng cách nhấn phím Enter.
- **Cancel**: Nếu giá trị của thuộc tính này là True thì ta có thể chọn nút lệnh nào đó bằng cách nhấn phím ESC.
- **Enabled**: Trong một biểu mẫu, có thể có nhiều nút lệnh để thực hiện nhiều công việc khác nhau và tại một thời điểm nào đó ta chỉ được phép thực hiện một số công việc. Nếu giá trị thuộc tính Enabled là False thì nút lệnh đó không có tác dụng. Giá trị mặc định của thuộc tính này là True. Ta có thể thay đổi giá trị của thuộc tính tại thời điểm chạy ứng dụng.

- **ToolTipText**: cho phép hiển thị một đoạn văn bản chú thích công dụng của nút lệnh khi người sử dụng dùng chuột rê trên nút nhấn.
- **Font, Fore Color**: Quy định kiểu chữ, kích thước, màu hiển thị.

V.3. Phương thức

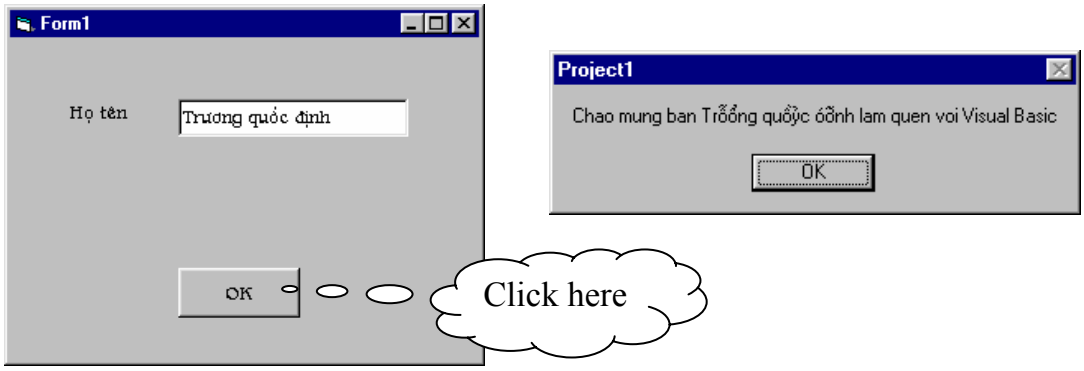
- **Move**: di chuyển nút lệnh đến tọa độ X,Y: **Move X, Y**.

V.4. Phương thức

- **Click**: đây là sự kiện thường xảy ra với nút lệnh. Mỗi khi một nút lệnh được chọn, sự kiện này được kích hoạt. Do đó, người sử dụng sẽ viết mã các lệnh để đáp ứng lại sự kiện này.

Ví dụ: Tạo một biểu mẫu có một ô nhập liệu với nhãn là họ tên và một nút lệnh cho phép đưa ra câu chào người dùng đó.

```
Private Sub Command1_Click()
      MsgBox "Chao mung ban " & Text1.Text & _
            " lam quen voi Visual Basic"
End Sub
```



Hình II.3 Sử dụng nút lệnh

VI. Ô nhập liệu (TextBox)

VI.1. Khái niệm:

Ô nhập liệu là một điều khiển cho phép nhận thông tin do người dùng nhập vào. Đối với ô nhập liệu ta cũng có thể dùng để hiển thị thông tin, thông tin này được đưa vào tại thời điểm thiết kế hay thậm chí ở thời điểm thực thi ứng dụng. Còn thao tác nhận thông tin do người dùng nhập vào dĩ nhiên là được thực hiện tại thời điểm chạy ứng dụng.

Biểu tượng (shortcut) trên hộp công cụ 

VI.2. Thuộc tính:

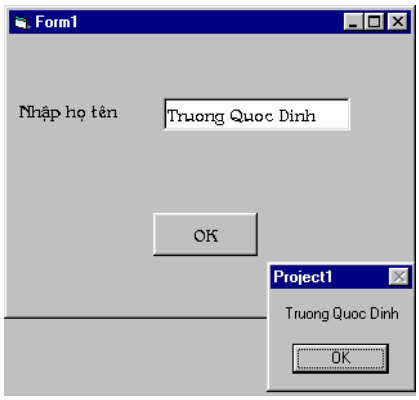
- **Name**: Đây là tên của ô nhập liệu, được sử dụng như một định danh.
- **MaxLength**: Thuộc tính quy định số ký tự tối đa có thể nhập vào ô nhập liệu. Nếu số ký tự nhập vào vượt quá số ký tự tối đa thì chỉ có đúng số ký tự tối đa được ghi nhận vào trong thuộc tính Text.

- **Text**: Dùng để nhập vào thông tin cần hiển thị trong Textbox tại thời điểm thiết kế hoặc nhận giá trị do người dùng nhập vào tại thời điểm chạy ứng dụng.

Ví dụ:

```
MsgBox Text1.Text
```

Đoạn mã này viết trong sự kiện Click của nút lệnh OK. Cho phép hộp thông báo hiển thị nội dung do người dùng nhập vào ô nhập liệu.



Hình II.3 Ví dụ về điều khiển ô nhập liệu

- **Locked**: Thuộc tính cho phép người dùng thay đổi nội dung của ô nhập liệu được hay không? Thuộc tính này có thể nhận 2 giá trị True hoặc False. Nếu False thì người dùng có thể thay đổi nội dung của ô nhập liệu & mặc định thì thuộc tính này có giá trị là False.

- **PasswordChar**: Thuộc tính này quy định cách hiển thị thông tin do người dùng nhập vào. Chẳng hạn, nếu ta nhập vào giá trị thuộc tính này là * thì các ký tự nhập vào điều hiển thị bởi dấu * . Thuộc tính này thường được dùng trong trường hợp thông tin nhập vào cần được che giấu (Ví dụ mật khẩu đăng nhập một chương trình ứng dụng nào đó mà trong đó các người dùng khác nhau thì có các quyền khác nhau).

- **Multiline**: Thuộc tính quy định ô nhập liệu có được hiển thị thông tin dưới dạng nhiều *hàng hay không*, nếu là TRUE thì ô nhập liệu cho phép nhiều hàng.

- **Font, Fore Color**: Quy định kiểu chữ, kích thước, màu hiển thị.

- **SelLength**:Cho phép trả về hoặc đặt trước số lượng ký tự được chọn trong ô nhập liệu.

- **SelStart**: Trả về hoặc xác định điểm bắt đầu của chuỗi được chọn. Đây là vị trí bắt đầu chèn một chuỗi mới trong trường hợp không có đánh dấu chọn chuỗi.

- **SelText**: Trả về hoặc xác định chuỗi ký tự được đánh dấu chọn, chỗi trả về sẽ là rỗng nếu như không đánh dấu chọn chuỗi nào.

Ba thuộc tính SelLength, SelStart, SelText chỉ có tác dụng tại thời điểm chạy ứng dụng.

VI.3. Phương thức

- **Move**: Di chuyển ô nhập liệu đến tọa độ X, Y: **Move X, Y**.

- **SetFocus**: Phương thức này nhằm mục đích thiết lập cho điều khiển ô nhập liệu nhận được Focus, nghĩa là nó sẵn sàng được tương tác bởi người sử dụng.

VI.4. Sự kiện:

- **KeyPress**: xảy ra khi người sử dụng chương trình nhấn một phím. Đối với điều khiển TextBox, ta thường dùng nó để lọc (filter out) các phím không chấp nhận. Sự kiện KeyPress cho ta một mã Ascii, một số có giá trị từ 0 đến 255, của phím vừa nhấn. Trong ví dụ dưới đây, TextBox Text1 sẽ chỉ nhận biết các phím là số (0 - 9), không nhận biết các phím khác:

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
 If KeyAscii < 48 Or KeyAscii > 57 Then ' Mã Ascii của  0 là 48, của 9 là 57
                KeyAscii = 0
 End If
End Sub
```

- **KeyDown, KeyUp**: mỗi sự kiện KeyPress lại cho ta một cặp sự kiện KeyDown/KeyUp. Sự kiện KeyDown/KeyUp có 2 tham số là KeyCode và Shift. Sự kiện này cho phép ta nhận biết được các phím đặc biệt trên bàn phím. Trong ví dụ dưới đây, ta hiển thị tên các phím chức năng mà người sử dụng chương trình nhấn vào:

```
Private Sub Text3_KeyDown(KeyCode As Integer, Shift As Integer)
        If (KeyCode >= 112) And (KeyCode <= 123) Then
                MsgBox "Ban vua nhan phim chuc nang: F" & _
                        Trim(Str(KeyCode - 111))
        End If
  End Sub
```

CHƯƠNG 3: LẬP TRÌNH CẤU TRÚC TRONG VISUAL BASIC

Mục tiêu:

Chương này giới thiệu về các cấu trúc lập trình trong VB; đây là các cấu trúc cốt lõi để xây dựng nên một chương trình VB.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Sử dụng môi trường lập trình VB để viết mã lệnh.
- Các kiểu dữ liệu trong VB.
- Cách khai báo hằng, biến trong VB.
- Biểu thức trong VB.
- Các câu lệnh đơn cũng như các câu lệnh có cấu trúc.
- Chương trình con trong VB.
- Bẫy lỗi trong VB.

Kiến thức có liên quan:

- Cách sử dụng môi trường phát triển của VB.

Tài liệu tham khảo:

- **Microsoft Visual Basic 6.0 và Lập trình Cơ sở dữ liệu** - Chương 4, trang 49 - **Nguyễn Thị Ngọc Mai** (chủ biên), **Nhà xuất bản Giáo dục** - **2000**.

I. Môi trường lập trình

I.1. Soạn thảo chương trình:

Trong Visual Basic IDE, cửa sổ mã lệnh (Code) cho phép soạn thảo chương trình. Cửa sổ này có một số chức năng nổi bật:

- Đánh dấu (Bookmarks): Chức năng này cho phép đánh dấu các dòng lệnh của chương trình trong cửa sổ mã lệnh để dễ dàng xem lại về sau này. Để bật tắt khả năng này, chọn Bookmarks từ menu Edit, hoặc chọn từ thanh công cụ Edit.
- Các phím tắt trong cửa sổ mã lệnh:

Chức năng	Phím tắt
Xem cửa sổ Code	F7
Xem cửa sổ Object Browser	F2
Tìm kiếm	CTRL+F
Thay thế	CTRL+H
Tìm tiếp	SHIFT+F4
Tìm ngược	SHIFT+F3
Chuyển đến thủ tục kế tiếp	CTRL+DOWN ARROW
Chuyển đến thủ tục trước đó	CTRL+UP ARROW
Xem định nghĩa	SHIFT+F2
Cuộn xuống một màn hình	CTRL+PAGE DOWN
Cuộn lên một màn hình	CTRL+PAGE UP
Nhảy về vị trí trước đó	CTRL+SHIFT+F2
Trở về đầu của mô-đun	CTRL+HOME
Đến cuối mô-đun	CTRL+END

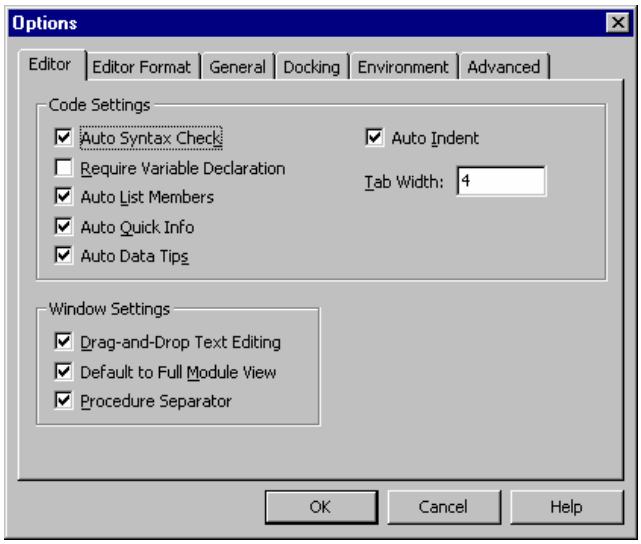
I.2. Các chức năng tự động:

- Tự động kiểm tra cú pháp (Auto Syntax Check)

Nếu chức năng này không được bật thì khi ta viết một dòng mã có chứa lỗi, VB chỉ hiển thị dòng chương trình sai với màu đỏ nhưng không kèm theo chú thích gì và tất nhiên ta có thể viết tiếp các dòng lệnh khác. Còn khi chức năng này được bật, VB sẽ cho ta biết một số thông tin về lỗi và hiển thị con trỏ ngay dòng chương trình lỗi để chờ ta sửa.

- Yêu cầu khai báo biến (Require Variable Declaration)

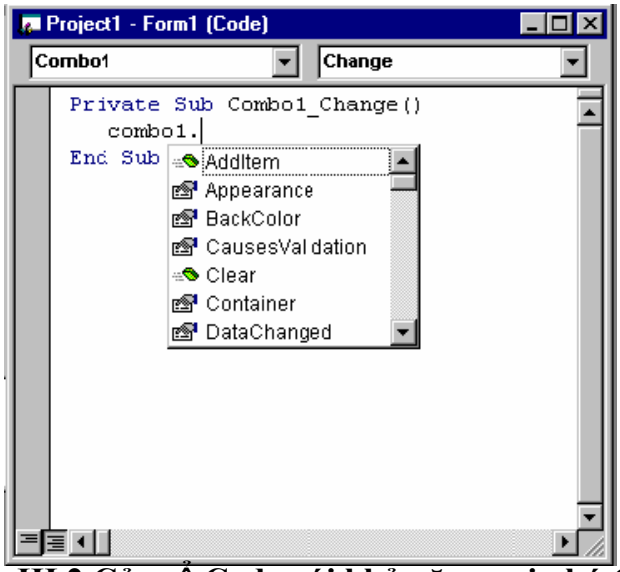
VB sẽ thông báo lỗi khi một biến được dùng mà không khai báo và sẽ chỉ ra vị trí của biến đó.



Hình III.1: Cửa sổ Options

- Gợi nhớ mã lệnh (Code):

Khả năng Auto List Members: Tự động hiển thị danh sách các thuộc tính và phương thức của 1 điều khiển hay một đối tượng khi ta gõ vào tên của chúng. Chọn thuộc tính hay phương thức cần thao tác và nhấn phím Tab hoặc Space để đưa nó vào chương trình.



Hình III.2 Cửa sổ Code với khả năng gợi nhớ Code

II. Kiểu dữ liệu

II.1. Khái niệm

Kiểu dữ liệu là một tập hợp các giá trị mà một biến của kiểu có thể nhận và một tập hợp các phép toán có thể áp dụng trên các giá trị đó.

II.2. Các kiểu dữ liệu cơ sở trong Visual Basic

Kiểu dữ liệu	Mô tả
Boolean	Gồm 2 giá trị: TRUE & FALSE.
Byte	Các giá trị số nguyên từ 0 – 255
Integer	Các giá trị số nguyên từ -32768 – 32767
Long	Các giá trị số nguyên từ -2147483648 – 2147483647. Kiểu dữ liệu này thường được gọi là số nguyên dài.
Single	Các giá trị số thực từ -3.402823E+38 – 3.402823E+38. Kiểu dữ liệu này còn được gọi là độ chính xác đơn.
Double	Các giá trị số thực từ -1.79769313486232E+308 - 1.79769313486232E+308. Kiểu dữ liệu này được gọi là độ chính xác kép.
Currency	Dữ liệu tiền tệ chứa các giá trị số từ -922.337.203.685.477,5808 - 922.337.203.685.477,5807.
String	Chuỗi dữ liệu từ 0 đến 65.500 ký tự hay ký số, thậm chí là các giá trị đặc biệt như ^%@. Giá trị kiểu chuỗi được đặt giữa 2 dấu ngoặc kép (“”).
Date	Dữ liệu kiểu ngày tháng, giá trị được đặt giữa cặp dấu ##. Việc định dạng hiển thị tùy thuộc vào việc thiết lập trong Control Panel.
Variant	Chứa mọi giá trị của các kiểu dữ liệu khác, kể cả mảng.

III. Hằng số

III.1. Khái niệm

Hằng số (Constant) là giá trị dữ liệu không thay đổi.

III.2. Khai báo hằng

[Public|Private] Const <tên hằng> [As <kiểu dữ liệu>] = <biểu thức>

Trong đó, ***tên hằng*** được đặt giống theo quy tắc đặt tên của điều khiển.

Ví dụ:

```
Const g = 9.8
Const Num As Integer = 4*5
```

Ta có thể dùng cửa sổ Object Browser để xem danh sách các hằng có sẵn của VB và VBA (Visual Basic for Application).

Trường hợp trùng tên hằng trong những thư viện khác nhau, ta có thể chỉ rõ tham chiếu hằng.

[<Libname>.][<tên mô-đun>.] <tên hằng>

III) Biến

III.1. Khái niệm

Biến (Variable) là vùng lưu trữ được đặt tên để chứa dữ liệu tạm thời trong quá trình tính toán, so sánh và các công việc khác.

Biến có 2 đăc điểm:

- Mỗi biến có một tên.
- Mỗi biến có thể chứa duy nhất một loại dữ liệu.

III.2. Khai báo

[Public|Private|Static|Dim] <tên biến> [As <kiểu dữ liệu>]

Trong đó, ***tên biến***: là một tên được đặt giống quy tắc đặt tên điều khiển. Nếu cần khai báo nhiều biến trên một dòng thì mỗi khai báo cách nhau dấu phẩy (,).

Nếu khai báo biến không xác định kiểu dữ liệu thì biến đó có kiểu Variant.

Khai báo ngầm: Đây là hình thức không cần phải khai báo một biến trước khi sử dụng. Cách dùng này có vẻ thuận tiện nhưng sẽ gây một số sai sót, chẳng hạn khi ta đánh nhầm tên biến, VB sẽ hiểu đó là một biến mới dẫn đến kết quả chương trình sai mà rất khó phát hiện.

Ví dụ:

```
Dim Num As Long, a As Single
Dim Age As Integer
```

Khai báo tường minh: Để tránh rắc rối như đã nêu ở trên, ta nên quy định rằng VB sẽ báo lỗi khi gặp biến chưa được khai báo bằng dòng lệnh:

Option Explicit trong phần Declaration (khai báo) của mô-đun.

Option Explicit chỉ có tác dụng trên từng mô-đun do đó ta phải đặt dòng lệnh này trong từng mô-đun của biểu mẫu, mô-đun lớp hay mô-đun chuẩn.

IV. Biểu thức

IV.1. Khái niệm

Toán tử hay phép toán (Operator): là từ hay ký hiệu nhằm thực hiện phép tính và xử lý dữ liệu.

Toán hạng: là giá trị dữ liệu (biến, hằng…).

Biểu thức: là tập hợp các toán hạng và các toán tử kết hợp lại với nhau theo quy tắc nhất định để tính toán ra một giá trị nào đó.

IV.2. Các loại phép toán

a. Các phép toán số học: Thao tác trên các giá trị có kiểu dữ liệu số.

Phép toán	Ý nghĩa	Kiểu của đối số	Kiểu của kết quả
-	Phép lấy số đối	Kiểu số (Integer, Single…)	Như kiểu đối số
+	Phép cộng hai số	Kiểu số (Integer, Single…)	Như kiểu đối số
-	Phép trừ hai số	Kiểu số (Integer, Single…)	Như kiểu đối số
*	Phép nhân hai số	Kiểu số (Integer, Single…)	Như kiểu đối số
/	Phép chia hai số	Kiểu số (Integer, Single…)	Single hay Double
\	Phép chia lấy phần nguyên	Integer, Long	Integer, Long
Mod	Phép chia lấy phần dư	Integer, Long	Integer, Long
^	Tính lũy thừa	Kiểu số (Integer, Single…)	Như kiểu đối số

b. Các phép toán quan hệ

Đây là các phép toán mà giá trị trả về của chúng là một giá trị kiểu Boolean (TRUE hay FALSE).

Phép toán	Ý nghĩa
=	So sánh bằng nhau
<>	So sánh khác nhau
>	So sánh lớn hơn
<	So sánh nhỏ hơn
>=	So sánh lớn hơn hoặc bằng
<=	So sánh nhỏ hơn hoặc bằng

c. Các phép toán Logic: là các phép toán tác động trên kiểu Boolean và cho kết quả là kiểu Boolean. Các phép toán này bao gồm AND (và), OR (hoặc), NOT (phủ định). Sau đây là bảng giá trị của các phép toán:

X	Y	X AND Y	X OR Y	NOT X
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE

V. Câu lệnh

Một câu lệnh (statement) xác định một công việc mà chương trình phải thực hiện để xử lý dữ liệu đã được mô tả và khai báo. Các câu lệnh được ngăn cách với nhau bởi ký tự xuống dòng. *Ký tự xuống dòng báo hiệu kết thúc một câu lệnh*.

V.1. Lệnh gán

Cú pháp:

<Tên biến> = <Biểu thức>

Ví dụ:

Giả sử ta có khai báo sau:

```
Dim TodayTemp As Single, MinAge As Integer
Dim Sales As Single, NewSales As Single, FullName As String
```

Các lệnh sau gán giá trị cho các biến trên:

```
TodayTemp = 30.5
MinAge = 18
Sales = 200000
NewSales = Sales * 1.2
```

Giả sử người dùng cần nhập họ và tên vào ô nhập liệu TextBox có thuộc tính Name là txtName, câu lệnh dưới đây sẽ lưu giá trị của ô nhập liệu vào trong biến FullName:

```
FullName = txtName.Text
```

Lưu ý: Kiểu dữ liệu của biểu thức (vế phải của lệnh gán) phải phù hợp với biến ta cần gán trị.

V.2. Lệnh rẽ nhánh If

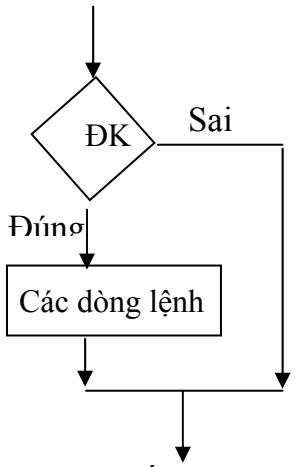
- Một dòng lệnh:

  If <điều kiện> **Then** <*dòng lệnh*>

- Nhiều dòng lệnh:

  If <điều kiện> **Then**
  Các dòng lệnh
  End If

Lưu đồ cú pháp:



Trong đó, <điều kiện>: biểu thức mà kết quả trả về kiểu Boolean.

Ý nghĩa câu lệnh: *Các dòng lệnh* hay *dòng lệnh* sẽ được thi hành nếu như điều kiện là đúng. Còn nếu như điều kiện là sai thì câu lệnh tiếp theo sau cấu trúc If ... Then được thi hành.

- Dạng đầy đủ: **If ... Then ... Else**

If <điều kiện 1> **Then**
[Khối lệnh 1]
ElseIf <điều kiện 2> **Then**
[Khối lệnh 2]**...**
[**Else**
[Khối lệnh n]]
End If

VB sẽ kiểm tra các điều kiện, nếu điều kiện nào đúng thì khối lệnh tương ứng sẽ được thi hành. Ngược lại nếu không có điều kiện nào đúng thì khối lệnh sau từ khóa Else sẽ được thi hành.

Ví dụ:

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

V.3. Lệnh lựa chọn Select Case

Trong trường hợp có quá nhiều các điều kiện cần phải kiểm tra, nếu ta dùng cấu trúc rẽ nhánh **If**…**Then** thì đoạn lệnh không được trong sáng, khó kiểm tra, sửa đổi khi có sai sót. Ngược lại với cấu trúc **Select**…**Case**, biểu thức điều kiện sẽ được tính toán một lần vào đầu cấu trúc, sau đó VB sẽ so sánh kết quả với từng trường hợp (**Case**). Nếu bằng nó thi hành khối lệnh trong trường hợp (**Case**) đó.

Select Case <biểu thức kiểm tra>
Case <Danh sách kết quả biểu thức 1>
[Khối lệnh 1]
Case <Danh sách kết quả biểu thức 2>
[Khối lệnh 2]
.
.
.
[**Case Else**
[Khối lệnh n]]
End Select

Mỗi danh sách kết quả biểu thức sẽ chứa một hoặc nhiều giá trị. Trong trường hợp có nhiều giá trị thì mỗi giá trị cách nhau bởi dấu phẩy (**,**). Nếu có nhiều Case cùng thỏa điều kiện thì khối lệnh của Case đầu tiên sẽ được thực hiện.

Ví dụ của lệnh rẽ nhánh **If…Then** ở trên có thể viết như sau:

```
Select Case TheColorYouLike
Case vbRed
   MsgBox "You are a lucky person"
Case vbGreen
   MsgBox "You are a hopeful person"
Case vbBlue
   MsgBox "You are a brave person"
Case vbMagenta
   MsgBox "You are a sad person"
Case Else
```

```
        MsgBox "You are an average person"
    End Select
```

Toán tử Is & To

Toán tử Is: Được dùng để so sánh <Biểu thức kiểm tra> với một biểu thức nào đó.

Toán tử To: Dùng để xác lập miền giá trị của <Biểu thức kiểm tra>.

Ví dụ:

```
Select Case Tuoi
    Case Is <18
        MsgBox "Vi thanh nien"
    Case 18 To 30
        MsgBox "Ban da truong thanh, lo lap than di"
    Case 31 To 60
        MsgBox "Ban dang o lua tuoi trung nien"
    Case Else
        MsgBox "Ban da lon tuoi, nghi huu duoc roi day!"
End Select
```

Lưu ý: Trong ví dụ trên không thể viết **Case** Tuoi < 18.

V.4. Cấu trúc lặp

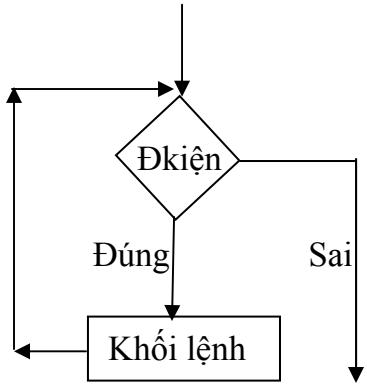
Các cấu trúc lặp cho phép thi hành một khối lệnh nào đó nhiều lần.

a. Lặp không biết trước số lần lặp

Do ... Loop: Đây là cấu trúc lặp không xác định trước số lần lặp, trong đó, số lần lặp sẽ được quyết định bởi một biểu thức điều kiện. Biểu thức điều kiện phải có kết quả là True hoặc False. Cấu trúc này có 4 kiểu:

Kiểu 1:

Do While <điều kiện>
    <khối lệnh>
Loop



Khối lệnh sẽ được thi hành đến khi nào điều kiện không còn đúng nữa. Do biểu thức điều kiện được kiểm tra trước khi thi hành khối lệnh, do đó có thể khối lệnh sẽ không được thực hiện một lần nào cả.

Kiểu 2:

Do
    <khối lệnh>
Loop While <điều kiện>

Khối lệnh sẽ được thực hiện, sau đó biểu thức điều kiện được kiểm tra, nếu điều kiện còn đúng thì, khối lệnh sẽ được thực hiện tiếp tục. Do biểu thức điều kiện được kiểm tra sau, do đó khối lệnh sẽ được thực hiện ít nhất một lần.

Kiểu 3:

Do Until <điều kiện>
<khối lệnh>
Loop

Cũng tương tự như cấu trúc Do While ... Loop nhưng khác biệt ở chỗ là khối lệnh sẽ được thi hành khi điều kiện còn sai.

Kiểu 4:

Do
<khối lệnh>
Loop Until <điều kiện>

Khối lệnh được thi hành trong khi điều kiện còn sai và có ít nhất là một lần lặp.

Ví dụ: Đoạn lệnh dưới đây cho phép kiểm tra một số nguyên N có phải là số nguyên tố hay không?

```
Dim i As Integer
i = 2
Do While (i <= Sqr(N)) And (N Mod i = 0)
     i = i + 1
Loop
If (i > Sqr(N)) And (N <> 1) Then
     MsgBox Str(N) & " la so nguyen to"
Else
     MsgBox Str(N) & " khong la so nguyen to"
End If
```

Trong đó, hàm Sqr: hàm tính căn bậc hai của một số

b. Lặp biết trước số lần lặp

✓ **For ... Next**

Đây là cấu trúc biết trước số lần lặp, ta dùng biến đếm tăng dần hoặc giảm dần để xác định số lần lặp.

For <biến đếm> = <điểm đầu> **To** <điểm cuối> [**Step** <bước nhảy>]
[khối lệnh]
Next

Biến đếm, điểm đầu, điểm cuối, bước nhảy là những giá trị số (Integer, Single,…). Bước nhảy có thể là âm hoặc dương. Nếu bước nhảy là số âm thì điểm đầu phải lớn hơn điểm cuối, nếu không khối lệnh sẽ không được thi hành.

Khi Step không được chỉ ra, VB sẽ dùng bước nhảy mặc định là một.

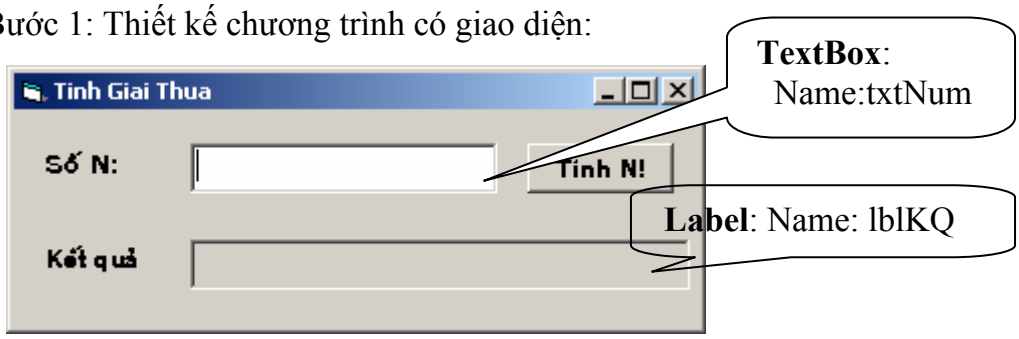
Ví dụ: Đoạn lệnh sau đây sẽ hiển thị các kiểu chữ hiện có của máy bạn.

```
Private Sub Form_Click( )
     Dim i As Integer
```

```
        For   i  = 0  To  Screen.FontCount
              MsgBox Screen.Fonts(I)
        Next
End  Sub
```

Ví dụ: Tính N!

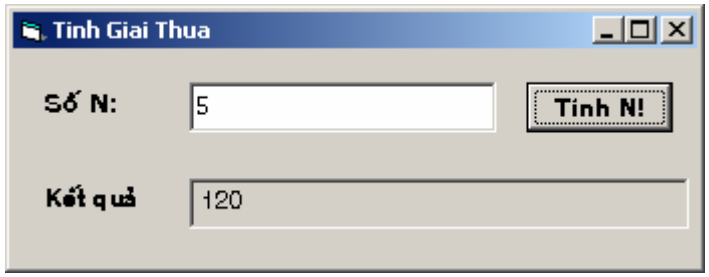
- Bước 1: Thiết kế chương trình có giao diện:



- Bước 2: Sự kiện Command1_Click được xử lý:

```
Private Sub Command1_Click()
    Dim i As Integer, n As Integer, Kq As Long
    n = Val(txtNum.Text)
    Kq = 1
    For i = 1 To n
        Kq = Kq * i
    Next

    lblKQ.Caption = Str(Kq)
End Sub
```

- Lưu dự án và chạy chương trình ta được kết quả như hình dưới:



✓ **For Each ... Next**

Tương tự vòng lặp For ... Next, nhưng nó lặp khối lệnh theo số phần tử của một tập các đối tượng hay một mảng thay vì theo số lần lặp xác định. Vòng lặp này tiện lợi khi ta không biết chính xác bao nhiêu phần tử trong tập hợp.

For Each <phần tử> **In** <nhóm>
<khối lệnh>
Next <phần tử>

Lưu ý:

- Phần tử trong tập hợp chỉ có thể là biến Variant, biến Object, hoặc một đối tượng trong Object Browser.
- Phần tử trong mảng chỉ có thể là biến Variant.

- Không dùng For Each ... Next với mảng chứa kiểu tự định nghĩa vì Variant không chứa kiểu tự định nghĩa.

VI. Chương trình con

VI.1. Khái niệm

Trong những chương trình lớn, có thể có những đoạn chương trình viết lặp đi lặp lại nhiều lần, để tránh rườm rà và mất thời gian khi viết chương trình người ta thường phân chia chương trình thành nhiều module, mỗi module giải quyết một công việc nào đó. Các module như vậy gọi là các chương trình con.

Một tiện lợi khác của việc sử dụng chương trình con là ta có thể dễ dàng kiểm tra xác định tính đúng đắn của nó trước khi ráp nối vào chương trình chính và do đó việc xác định sai sót để tiến hành hiệu đính trong chương trình chính sẽ thuận lợi hơn.

Trong Visual Basic, chương trình con có hai dạng là hàm (Function) và thủ tục (Sub).

Hàm khác thủ tục ở chỗ hàm trả về cho lệnh gọi một giá trị thông qua tên của nó còn thủ tục thì không. Do vậy ta chỉ dùng hàm khi và chỉ khi thoả mãn đồng thời các yêu cầu sau đây:

- Ta muốn nhận lại một kết quả (chỉ một mà thôi) khi gọi chương trình con.
- Ta cần dùng tên chương trình con (có chứa kết quả) để viết trong các biểu thức.

Nếu không thỏa mãn hai điều kiện ấy thì dùng thủ tục.

VI.2. Thủ tục

a. Khái niệm:

Thủ tục là một chương trình con thực hiện một hay một số tác vụ nào đó. Thủ tục có thể có hay không có tham số.

b. Khai báo thủ tục

[Private | Public] [Static] Sub <tên thủ tục> [(<tham số>[**As** <Kiểu tham số>])]
<Các dòng lệnh> hay <Các khai báo>
End Sub

Trong đó:

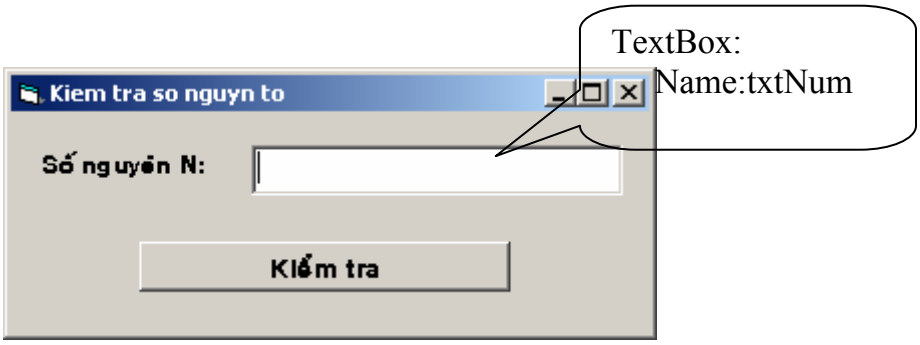
- <Tên thủ tục>: Đây là một tên được đặt giống quy tắc tên biến, hằng,…
- <tham số>[: <Kiểu tham số>]: có thể có hay không? Nếu có nhiều tham số thì mỗi tham số phân cách nhau dấu phẩy. Nếu không xác định kiểu tham số thì tham số có kiểu Variant.

Để gọi thủ tục để thực thi, ta có 2 cách:

- <Tên thủ tục> [<Các tham số thực tế>]
- **Call** <Tên thủ tục> ([<Các tham số thực tế>])

Ví dụ: Thiết kế chương trình kiểm tra xem số nguyên N có phải là số nguyên tố hay không?

- Bước 1: Thiết kế chương trình có giao diện



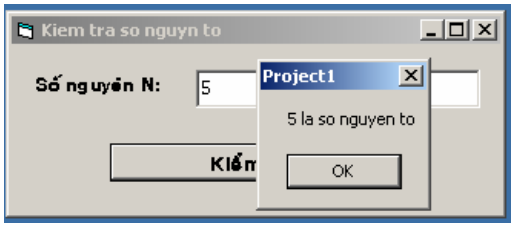
- Bước 2: Viết thủ tục KtraNgTo trong phần mã lệnh của Form

```
Sub KTraNgTo(N As Integer)
    Dim i As Integer
    i = 2
    Do While (i <= Sqr(N)) And (N Mod i <> 0)
            i = i + 1
    Loop
    If (i > Sqr(N)) And (N <> 1) Then
        MsgBox Str(N) & " la so nguyen to"
    Else
        MsgBox Str(N) & " khong la so nguyen to"
    End If
End Sub
```

- Bước 3: Xử lý sự kiện Command1_Click; trong thủ tục xử lý sự kiện này ta có gọi thủ tục KtraNgTo như sau:

```
Private Sub Command1_Click()
    KTraNgTo Val(txtNum.Text)
    ' Call KtraNgTo(Val(txtNum.Text))
End Sub
```

- Bước 4: Lưu dự án và chạy chương trình. Ta được kết quả sau:



Trong ví dụ trên thay vì gọi thủ tục bằng lời gọi:

```
KTraNgTo Val(txtNum.Text)
```

Ta có thể sử dụng cách khác:

```
Call KtraNgTo(Val(txtNum.Text))
```

VI.3. Hàm

a. Khái niệm

Hàm (Function) là một chương trình con có nhiệm vụ tính toán và cho ta một kết quả. Kết quả này được trả về trong tên hàm cho lời gọi nó.

b. Khai báo hàm

[Private | Public | Static] Function <Tên hàm> [(<tham số>[**As** <Kiểu tham số>])] _
[**As** <KIỂU DỮ LIỆU>]
<Các dòng lệnh> hay <Các khai báo>
End Function

Trong đó:

- <Tên hàm>: Đây là một tên được đặt giống quy tắc tên biến, hằng,…

- <tham số>[: <Kiểu tham số>]: có thể có hay không? Nếu có nhiều tham số thì mỗi tham số phân cách nhau dấu phẩy. Nếu không xác định kiểu tham số thì tham số có kiểu Variant.

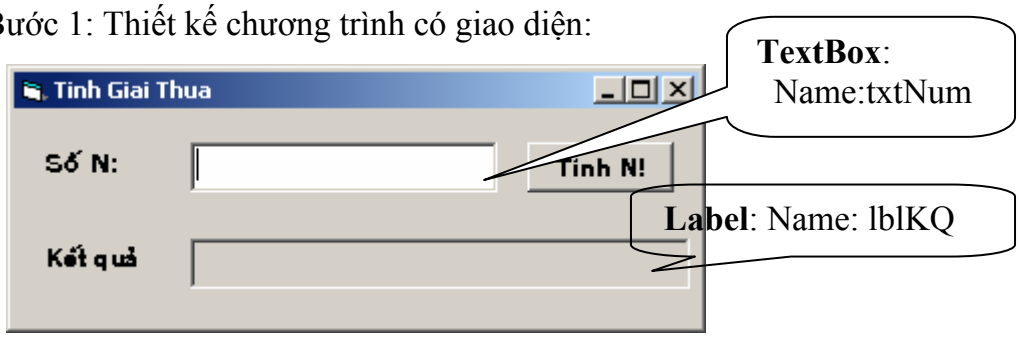
- <KIỂU DỮ LIỆU>: Kết quả trả về của hàm, trong trường hợp không khai báo **As** <kiểu dữ liệu>, mặc định, VB hiểu kiểu trả về kiểu Variant.

Khi gọi hàm để thực thi ta nhận được một kết quả. Cần chú ý khi gọi hàm thực thi ta nhận được một kết quả có kiểu chính là kiểu trả về của hàm (hay là kiểu Variant nếu ta không chỉ rõ kiểu trả về trong định nghĩa hàm). Do đó lời gọi hàm phải là thành phần của một biểu thức.

Cú pháp gọi hàm thực thi: *<Tên hàm>[(tham số)]*.

Ví dụ: Tính N!

- Bước 1: Thiết kế chương trình có giao diện:

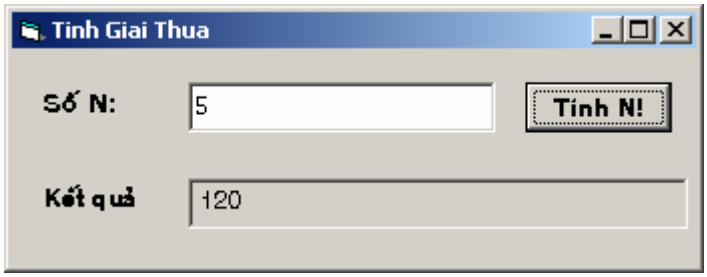


- Bước 2: Thêm một hàm vào cửa sổ mã lệnh của Form

```
Function Giaithua(N As Integer) As Long
    Dim i As Integer, Kq As Long
    Kq = 1
    For i = 1 To n
        Kq = Kq * i
    Next

    Giaithua = Kq
End Function

Private Sub Command1_Click()
```

```
    Dim n As Integer
    n = Val(txtNum.Text)

    lblKQ.Caption = Str(Giaithua(n))
End Sub
```

Lưu dự án và chạy chương trình ta được kết quả như hình dưới:



Lưu ý: Do khi gọi hàm ta nhận được một kết quả nên bên trong phần định nghĩa hàm, trước khi kết thúc ta phải gán kết quả trả về của hàm thông qua tên hàm (trong ví dụ trên là dòng lệnh `Giaithua = Kq`)

VII. Truy xuất dữ liệu trong Visual Basic

VII.1. Các khái niệm

- ***Module:***

- Một ứng dụng đơn giản có thể chỉ có một biểu mẫu, lúc đó tất cả mã lệnh của ứng dụng đó được đặt trong cửa sổ mã lệnh của biểu mẫu đó (gọi là Form Module). Khi ứng dụng được phát triển lớn lên, chúng ta có thể có thêm một số biểu mẫu nữa và lúc này khả năng lặp đi lặp lại nhiều lần của một đoạn mã lệnh trong nhiều biểu mẫu khác nhau là rất lớn.

- Để tránh việc lặp đi lặp lại trên, ta tạo ra một Module riêng rẽ chứa các chương trình con được dùng chung. Visual Basic cho phép 3 loại Module:

Module biểu mẫu (Form module): đi kèm với mỗi một biểu mẫu là một module của biểu mẫu đó để chứa mã lệnh của biểu mẫu này. Với mỗi điều khiển trên biểu mẫu, module biểu mẫu chứa các chương trình con và chúng sẵn sàng được thực thi để đáp ứng lại các sự kiện mà người sử dụng ứng dụng tác động trên điều khiển. Module biểu mẫu được lưu trong máy tính dưới dạng các tập tin có đuôi là ***.frm**.

Module chuẩn (Standard module): Mã lệnh không thuộc về bất cứ một biểu mẫu hay một điều khiển nào sẽ được đặt trong một module đặc biệt gọi là module chuẩn (được lưu với đuôi ***.bas**). Các chương trình con được lặp đi lặp lại để đáp ứng các *sự kiện khác nhau của các điều khiển khác nhau* thường được đặt trong module chuẩn.

Module lớp (Class module): được sử dụng để *tạo các điều khiển* được gọi thực thi trong một ứng dụng cụ thể. Một module chuẩn chỉ chứa mã lệnh nhưng module lớp chứa cả mã lệnh và dữ liệu, chúng có thể được coi là các điều khiển do người lập trình tạo ra (được lưu với đuôi ***.cls**).

- ***Phạm vi (scope)***: xác định số lượng chương trình có thể truy xuất một biến. Một biến sẽ thuộc một trong 3 loại phạm vi:

  Phạm vi biến cục bộ.

  Phạm vi biến module.

  Phạm vi biến toàn cục.

VII.2. Biến toàn cục

- ***Khái niệm***: Biến toàn cục là biến có phạm vi hoạt động trong toàn bộ ứng dụng.
- ***Khai báo***:

  Global <Tên biến> [**As** <Kiểu dữ liệu>]

VII.3. Biến cục bộ

- ***Khái niệm***: Biến cục bộ là biến chỉ có hiệu lực trong những chương trình mà chúng được định nghĩa.
- ***Khai báo***:

  Dim <Tên biến> [**As** <Kiểu dữ liệu>]
- ***Lưu ý***:

  Biến cục bộ được định nghĩa bằng từ khóa Dim sẽ kết thúc ngay khi việc thi hành thủ tục kết thúc.

VII.4. Biến Module

- ***Khái niệm***: Biến Module là biến được định nghĩa trong phần khai báo (General|Declaration) của Module và *mặc nhiên* phạm vi hoạt động của nó là toàn bộ Module ấy.
- ***Khai báo***:

  - Biến Module được khai báo bằng từ khóa *Dim hay Private* & đặt trong phần khai báo của Module.

  Ví dụ:

```
Private Num As Integer
```

  - Tuy nhiên, các biến Module này có thể được sử dụng bởi các chương trình con trong các Module khác. Muốn thế chúng phải được khai báo là Public trong phân Khai báo (General|Declaration) của Module.

  Ví dụ:

```
Public Num As Integer
```

Lưu ý: Không thể khai báo biến với từ khóa là Public trong chương trình con.

VII.5. Truyền tham số cho chương trình con

- ***Khái niệm***

Một chương trình con đôi lúc cần thêm một vài thông tin về trạng thái của đoạn mã lệnh mà nó định nghĩa để thực thi. Những thông tin này là các biến được truyền vào khi gọi chương trình con, các biến này gọi là tham số của chương trình con.

Có hai cách để truyền tham số cho chương trình con: Truyền bằng giá trị & truyền bằng địa chỉ.

- ***Truyền tham số bằng giá trị***

Với cách truyền tham số theo cách này, mỗi khi một tham số được truyền vào, một bản sao của biến đó được tạo ra. Nếu chương trình con có thay đổi giá trị, những thay đổi này chỉ tác động lên bản sao của biến. Trong VB, từ khóa *ByVal* được dùng để xác định tham số được truyền bằng giá trị.

Ví dụ:

```
Sub Twice (ByVal Num As Integer)
   Num = Num * 2
   Print Num
End Sub

Private Sub Form_Click()
   Dim A As Integer
   A = 4
   Print A
   Twice A
   Print A
End Sub
```

Kết quả thực hiện của đoạn chương trình trên:

4

8

4

- ***Truyền tham số bằng địa chỉ***

Truyền tham số theo địa chỉ cho phép chương trình con truy cập vào giá trị gốc của biến trong bộ nhớ. Vì thế, giá trị của biến có thể sẽ bị thay đổi bởi đoạn mã lệnh trong chương trình con. Mặc nhiên, trong VB6 các tham số được truyền theo địa chỉ; tuy nhiên ta có thể chỉ định một cách tường minh nhờ vào từ khóa ByRef.

Ví dụ:

```
Sub Twice (Num As Integer)
   Num = Num * 2
   Print Num
End Sub

Private Sub Form_Click()
   Dim A As Integer
   A = 4
   Print A
   Twice A
   Print A
End Sub
```

Kết quả thực hiện của đoạn chương trình trên:

4

8

8

VIII. Bẫy lỗi trong Visual Basic

Các thao tác bẫy các lỗi thực thi của chương trình là cần thiết đối với các ngôn ngữ lập trình. Người lập trình khó kiểm soát hết các tình huống có thể gây ra lỗi. Chẳng hạn người ta khó có thể kiểm tra chặt chẽ việc người dùng đang chép dữ liệu từ đĩa mềm (hay CD) khi chúng không có trong ổ đĩa. Nếu có các thao tác bẫy lỗi ở đây thì tiện cho người lập trình rất nhiều.

Visual Basic cũng cung cấp cho ta một số cấu trúc để bẫy các lỗi đang thực thi.

Cú pháp:

Dạng 1:

On Error GoTo <*Tên nhãn***>**
<Các câu lệnh có thể gây ra lỗi**>**

<*Tên nhãn***>***:*
<Các câu lệnh xử lý lỗi**>**

Ý nghĩa:

- <Tên nhãn>: là một tên được đặt theo quy tắc của một danh biểu.
- Nếu một lệnh trong <Các câu lệnh có thể gây ra lỗi> thì khi chương trình thực thi đến câu lệnh đó, chương trình sẽ tự động nhảy đến đoạn chương trình định nghĩa bên dưới <Tên nhãn> để thực thi.

Dạng 2:

On Error Resume Next
<Các câu lệnh có thể gây ra lỗi**>**

Ý nghĩa:

- Nếu một lệnh trong <Các câu lệnh có thể gây ra lỗi> thì khi chương trình thực thi đến câu lệnh đó, chương trình sẽ tự động bỏ qua câu lệnh bị lỗi và thực thi câu lệnh kế tiếp.

CHƯƠNG 4 CÁC KIỂU DỮ LIỆU CÓ CẤU TRÚC

Mục tiêu:

Chương này giới thiệu về các cấu trúc dữ liệu trong VB. Việc nắm bắt được các vấn đề này giúp cho việc tổ chức dữ liệu khi viết chương trình VB được hợp lý hơn.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Sử dụng kiểu dữ liệu chuỗi.
- Sử dụng kiểu ngày tháng.
- Kiểu động (Variant)
- Kiểu mảng

Kiến thức có liên quan:

- Các cấu trúc lập trình trong VB.

Tài liệu tham khảo:

http://www.vovisoft.com/VisualBasic/VB6Chapter5.htm

http://www.vovisoft.com/VisualBasic/VB6Chapter6.htm

I. Kiểu chuỗi ký tự (String)

I.1. Khai báo

Có hai đặc tả chuỗi ký tự theo cú pháp như sau:

- **String** * <Chiều dài> Chỉ ra một chuỗi ký tự có độ dài cố định là bao nhiêu ký tự. Trong trường hợp giá trị thực của chuỗi có độ dài ngắn hơn độ dài khai báo thì độ dài của chuỗi thì một số khoảng trắng được thêm vào cho đủ độ dài thực. Trong trường hợp giá trị thực của chuỗi có độ dài lớn hơn độ dài khai báo thì sẽ cắt bớt các ký tự dư thừa bên phải. Một chuỗi không có ký tự nào (độ dài bằng 0) gọi là chuỗi rỗng.

- **String**: Khi không chỉ ra chiều dài tối đa của chuỗi thì mặc nhiên chuỗi có chiều dài tối đa là 65.500 ký tự.

Ví dụ:

```
Dim Name As String * 30, Class As String * 10
Dim A As String
```

I.2. Các hàm xử lý chuỗi

- **Ghép chuỗi**: cho phép ghép 2 hay nhiều chuỗi lại với nhau nhờ phép toán **&**.

  Ví dụ:

```
Dim FirstWord As String, SecondWord As String
Dim Greeting As String
FirstWord = "Hello"
SecondWord = "World"
Greeting = FirstWord & SecondWord
      ' Greeting bây giờ là "HelloWorld"
```

- **Len**: trả về chiều dài một chuỗi được chỉ định.

  Ví dụ:

```
Greeting = "Hi John!"
Dim iLen As Integer
iLen = Len(Greeting) ' iLen bây giờ bằng 8
```

- **Left**: Trích chuỗi con từ phần đầu chuỗi gốc `Left (String, [length]).`
- **Right**: Trích chuỗi con từ phần đuôi chuỗi gốc `Right (String, [length])`
- **Mid**: Trích chuỗi con từ giữa chuỗi gốc

```
Mid(String, Start As Long, [length])
```

  Ví dụ 1:

```
Dim Today As String, StrDay As String, StrMonth As String
Dim StrYear As String, StrMonthYear As String
Today = "24/05/2001"
' Lấy ra 2 ký tự từ bên trái của chuỗi Today
StrDay = Left(Today,2) ' StrDay bây giờ bằng "24"
' Lấy ra 4 ký tự từ bên phải của String Today
StrYear = Right(Today,4) ' StrYear bây giờ bằng "2001"
' Lấy ra 2 characters bắt đầu từ ký tự thứ tư của chuỗi
' Today, ký tự đầu tiên từ bên trái là thứ nhất
StrMonth = Mid(Today,4,2) ' StrMonth bây giờ bằng "05"
```

```
' Lấy ra phần còn lại bắt đầu từ ký tự 4 của chuỗi Today
StrMonthYear = Mid(Today,4) ' StrMonthYear bằng "05/2001"
```

Ví dụ 2:

```
Today = "24/05/2001"
' Thay thế character thứ 3 của Today bằng "-"
Mid(Today,3,1) = "-"
' Thay thế 2 ký tự bắt đầu từ ký tự 4 của Today bằng "10"
Mid(Today,4,2) = "10"
' Thay thế character thứ 6 của Today bằng "-"
Mid(Today,6,1) = "-" ' Today bây giờ bằng "24-10-2001"
```

- **InStr**: Tìm chuỗi con trong chuỗi gốc. Nếu hàm InStr trả về 0, nghĩa là không tìm thấy.

Cú pháp: `InStr([start,] string1, string2 [, compare])`

Trong đó:

- Start: Xác định vị trí trong chuỗi bắt đầu việc tìm kiếm. Nếu giá trị là Null thì sẽ bắt đầu từ đầu chuỗi. Nếu như tham số Compare có đặc tả thì bắt buộc phải khai báo tham số Start.

- String1: Biểu thức chuỗi để so sánh.

- String2: Chuỗi cần tìm.

- Compare: Xác định kiểu so sánh chuỗi.

Giá trị: `vbTextCompare, vbBinaryCompare.`

Ví dụ 1:

```
Dim myString As String, Position As Integer
myString = "The *rain in Spain mainly..."
Position = Instr(myString,"*") ' Position sẽ là 5
```

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

Ví dụ 2:

```
Dim KeyValuePair As String, Key As String
Dim Value As String
KeyValuePair = "BeatlesSong=Yesterday"
Pos = Instr(KeyValuePair, "=")
Key = Left(KeyValuePair, Pos-1)
Value = Mid(KeyValuePair, Pos+1)
```

- **Replace**: tìm và thay thế chuỗi.

Cú pháp:

```
Replace(Expression, find, replace[, start[, count[, compare]]])
```

Trong đó:

- Expression: Biểu thức chuỗi chứa chuỗi cần thay thế.

- find:Chuỗi cần tìm.

- replace: Chuỗi thay thế chuỗi tìm được.

- start: Tương tự như hàm InStr.

- count: Xác định số lần thay thế. Mặc định là 1.

- compare: Tương tự như hàm InStr.

- **LTrim (RTrim)**: cắt tất cả các khoảng trắng bên trái (bên phải của chuỗi)

Cú pháp: `LTrim(string)`
`RTrim(string)`

- **UCase**: đổi chuỗi sang chuỗi gồm các ký tự là chữ hoa.

  Cú pháp: `UCase(string)`

- **Asc**: cho mã Ascii của một ký tự.
- **Chr**: trả về ký tự ứng với mã Ascii được chỉ định.

```
Dim ASCIINumberA As Integer, CharB As String * 1
Dim StrFive As String * 1
ASCIINumberA = Asc("A") ' ASCIINumberA bây giờ bằng 65
CharB = Chr(66)
StrFive = Chr(Asc("0") + 5) ' ta có digit "5"
```

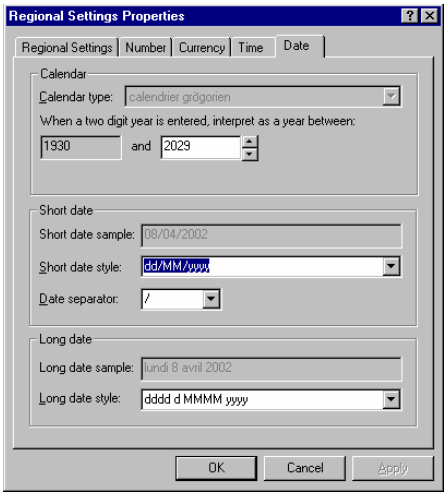
- **InstrRev**: tương tự như **InStr** nhưng việc tìm kiếm được tiến hành từ phải sang.
- **Val**: Hàm đổi chuỗi sang số.
- **Str**: Hàm đổi số sang chuỗi.

II. Kiểu ngày tháng (Date)

- Là kiểu mà các biến của nó chứa giá trị ngày tháng.
- Để cho VB biết dữ liệu là kiểu Date ta cần đặt giữa hai dấu # (hoặc cặp “”).

Ví dụ:

```
Dim D As Date
D = #01/02/98#  ‘ Hay “01/02/98”
```



Hình IV.1 Hộp thoại xác lập

Nếu hiểu theo kiểu người Mỹ, đây là ngày 2 tháng giêng năm 1998, còn nếu theo kiểu Anh thì đây là ngày 1 tháng hai năm 1998. Tuy nhiên, định dạng ngày tháng hiển thị phụ thuộc vào quy định của Windows.

- Hộp thoại hình IV.1 hiển thị khi ta chọn Regional Setting trong cửa sổ Control Panel của Windows, nó cho phép quy định kiểu ngày tháng tùy thuộc cách mà người dùng quy định. VB xử lý ngày tháng theo kiểu Mỹ, nhưng nếu máy hiển thị theo kiểu Anh thì nó vẫn hiển thị theo kiểu Anh.

- Hàm **Now**: trả về ngày giờ hiện tại.

Ví dụ: Dùng hàm Now & Format:

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

NOW IS Tue 05-Oct-2004 16:15:53

III. Các loại số

- Để chuyển đổi một chuỗi ra số ta có các hàm Val, CInt, CSng. Ngược lại để chuyển đổi từ số sang chuỗi ta dùng CStr, Str.

Ví dụ:

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

Ví dụ:

```
Dollars = "500.0"
ExchangeRatePerDollar = "7000.0"
'Dùng hàm CSng để đổi chuỗi ra Single
tempValue = CSng(Dollars) * CSng(ExchangeRatePerDollar)
'Dùng hàm Format để có các dấu phẩy ở ngàn và triệu
' và phải có 2 chữ số sau dấu chấm thập phân.
VNDong = Format (tempValue, "#,###,###.00")
MsgBox "Amount in VN Dong is " & VNDong
```

- **Round**: bỏ bớt một số chữ số sau dấu chấm thập phân

Ví dụ:

Round (12.3456789, 4)

chỉ giữ lại 4 con số sau dấu chấm thập phân và cho ta 12.3457

IV.Kiểu Object

Biến kiểu Object chứa một địa chỉ 4 Byte trỏ đến đối tượng trong ứng dụng hiện hành hoặc các ứng dụng khác. Dùng lệnh **Set** để chỉ ra đối tượng cụ thể.

```
Dim   ObjDb   As   Object
Set   ObjDb   =  OpenDatabase("d:\tqdinh\thu.mdb")
```

Khi khai báo biến đối tượng, ta nên chỉ ra tên lớp tường minh, chẳng hạn như TextBox thay vì Control, ứng dụng của ta sẽ chạy nhanh hơn.

Ta có thể xem danh sách các lớp có sẵn trong cửa sổ Object Browser.

V. Kiểu Variant

Biến kiểu Variant có thể chứa mọi kiểu dữ liệu kể cả kiểu mảng, kiểu do người dùng định nghĩa nhưng ngoại trừ kiểu chuỗi có độ dài cố định .

Biến kiểu Variant có thể nhận các giá trị đặc biệt như Empty, Nothing, Error, Null. Ta có thể xác định kiểu dữ liệu của biến Variant bằng các sử dụng hàm VarType hoặc hàm TypeName.

Hàm VarType dùng để kiểm tra kiểu dữ liệu

Hằng	Giá trị	Diễn giải
vbEmpty	**0**	Không chứa gì cả
vbNull	1	Dữ liệu không hợp lệ
vbInteger	2	Dữ liệu kiểu Integer chuẩn

vbLong	3	Dữ liệu kiểu Long Integer
vbSingle	4	Dữ liệu kiểu dấu chấm động Single
vbDouble	5	Dữ liệu kiểu dấu chấm động Double
vbCurrency	6	Kiểu Currency
vbDate	7	Kiểu Date
vbString	8	Kiểu String
vbObject	9	Kiểu Object
vbError	10	Có một đối tượng lỗi
vbBoolean	11	Kiểu giá trị Boolean chuẩn
vbVariant	12	Kiểu Variant
vbDataObject	13	Kiểu DAO chuẩn (data access object)
vbDecimal	14	Giá trị thuộc hệ thập phân
vbByte	17	Kiểu Byte
vbUserDefinedType	36	Kiểu do người dùng định nghĩa
vbArray	**8192**	Kiểu mảng

Một số chú ý khi dùng biến kiểu Variant:

- Nếu muốn thi hành các hàm toán học, Variant phải chứa giá trị kiểu số.
- Nếu muốn nối chuỗi, dùng toán tử & thay vì toán tử +.

Giá trị Empty:

- Đây là giá trị đặc biệt xuất hiện khi một biến chưa được gán trị. Ta dùng hàm IsEmpty để kiểm tra giá trị Empty.
- Giá trị Empty biến mất khi có một giá trị bất kỳ được gán cho biến Variant, để trở về giá trị Empty, ta gán từ khoá Empty cho biến Variant.

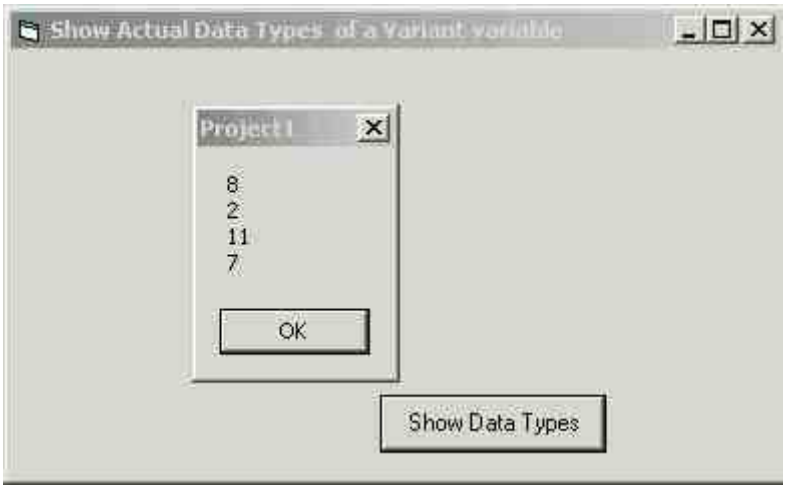
Giá trị Null: Biến Variant chứa giá trị Null trong trường hợp những ứng dụng cơ sở dữ liệu thể hiện không có dữ liệu hoặc dữ liệu không xác định.

Giá trị Error: Trong một biến kiểu Variant, Error là một giá trị đặc biệt cho biết đã có một lỗi đã xảy ra bên trong thủ tục.

Ví dụ:

```
Private Sub cmdShowDataTypes_Click()
   Dim sMess As String
   Dim vVariant As Variant
   vVariant = "Xin chao"  'String
   sMess = VarType(vVariant) & vbCrLf  ' xuống dòng & về đầu dòng
   vVariant = 25  ' Integer
   sMess = sMess & VarType(vVariant) & vbCrLf
   vVariant = True  ' Boolean
   sMess = sMess & VarType(vVariant) & vbCrLf
   'Date
   vVariant = #1/1/2001#  'trong cặp dấu #
   sMess = sMess & VarType(vVariant)
   MsgBox sMess
End Sub
```

Khi chạy chương trình kết quả là:



VI. Kiểu Mảng

VI.1. Khái niệm

- Mảng là tập hợp các phần tử có cùng một kiểu.

- Dùng mảng sẽ làm cho chương trình đơn giản và gọn hơn vì ta có thể sử dụng vòng lặp. Mảng sẽ có biên trên và biên dưới, trong đó các thành phần của mảng là liên tiếp trong khoảng giữa hai biên này.

- Có hai loại biến mảng: mảng có chiều dài cố định và mảng có chiều dài thay đổi lúc thi hành.

VI.2. Khai báo

- **Mảng có chiều dài cố định**:

  Dim <Tên biến mảng>(<Kích thước>) [**As** <Kiểu>]

  Lúc này phần tử đầu tiên có chỉ số là 0 & phần tử cuối cùng có chỉ số là <Kích thước>.

  Dim <Tên biến mảng>(<Chỉ số đầu> **To** <Chỉ số cuối>) [**As** <Kiểu>]

  Ví dụ:

```
' Khai báo một biến mảng 15 phần tử kiểu Integer
Dim  Counters(14)As Integer
' Khai báo một biến mảng 21 phần tử kiểu Double
Public Sums(20)As Double
' Khai báo một biến mảng 10 phần tử kiểu chuỗi ký tự
Dim List (1 To 10) As String * 12
```

- Hàm **UBound** trả về biên trên của một mảng.

- Hàm **LBound** trả về biên dưới của một mảng.

Ví dụ:

`UBound(List)` sẽ trả về giá trị là 10.

`LBound(List)` sẽ trả về giá trị là 1.

✓ ***Lưu ý:*** ta có thể khai báo một mảng nhiều chiều như sau

```
Dim  Multi3D (3, 1 To 10, 9) As Double
```

Khai báo này tạo ra một mảng 3 chiều với kích thước 4 x 10 x 10.

o **Mảng động**:

- Đây là mảng có kích thước thay đổi, đó là một trong những ưu điểm của mảng động vì nó giúp ta tiết kiệm tài nguyên hệ thống. Ta có thể sử dụng một mảng có kích thước lớn trong một thời gian nào đó rồi xoá bỏ để trả lại vùng nhớ cho hệ thống.

- Khai báo một mảng động bằng cách cho nó một danh sách không theo chiều nào cả. Cú pháp: ***Dim*** *<Tên mảng> () [**As** <Kiểu>]*

Ví dụ:

```
Dim DynArray() As Integer
```

Sau đó ta có thể cấp phát số phần tử thật sự bằng lệnh **ReDim**.

ReDim <*Tên mảng***>(N)** ' Trong đó N là một biểu thức kiểu Integer.

ReDim dùng để xác định hay thay đổi kích thước của một mảng động. Ta có thể dùng ReDim để thay đổi số phần tử, số chiều của một mảng nhiều lần nhưng không thể thay đổi kiểu dữ liệu của mảng ngoại trừ kiểu mảng là kiểu Variant.

Mỗi lần gọi ReDim tất cả các giá trị chứa trong mảng sẽ bị mất. VB khởi tạo lại giá trị cho chúng (Empty đối với mảng Variant, 0 cho mảng kiểu số, chuỗi rỗng cho mảng chuỗi hoặc Nothing cho mảng các đối tượng). Nhưng đôi khi ta muốn tăng kích cỡ của mảng nhưng không muốn làm mất dữ liệu, ta dùng **ReDim** đi kèm với từ khoá **Preserve**. Ta xem ví dụ dưới đây:

```
ReDim Preserve DynArray (UBound(DynArray) +10)
```

Tuy nhiên chỉ có biên trên của chiều cuối cùng trong mảng được thay đổi khi ta dùng **Preserve**. Nếu ta cố tình thay đổi chiều khác hoặc biên dưới thì VB sẽ báo lỗi.

VI.3. Một số thao tác trên mảng

o Truy xuất từng phần tử trong mảng: <Tên mảng>(<Vị trí>)

o Sao chép mảng: Đối với VB6, ta có thể gán một mảng cho một mảng khác, hoặc kết quả trả về của một hàm có thể là một mảng.

Ví dụ:

```
Sub ByteCopy (old () As Byte,  New () As Byte)
    New = old
End Sub
```

Tuy nhiên, cách này cũng chỉ áp dụng được cho mảng khai báo động mà thôi. Khi gán biến, có một số quy luật mà ta cần lưu ý: Đó là quy luật về kiểu dữ liệu và quy luật về kích thước và số chiều của mảng.

Lỗi khi gán mảng có thể xảy ra lúc biên dịch hoặc khi thi hành. Ta có thể thêm bẫy lỗi để đảm bảo rằng hai mảng là tương thích trước khi gán.

- Mảng là kết quả trả về của hàm. Chẳng hạn như:

```
Public Function ArrayFunction (b As Byte) As Byte()
    Dim   x(2)   As   Byte
    x(0)  = b
    x(1)  = b  + 2
    x(2)  = b  + b
    ArrayFunction  =  x
End Function
```

Khi gọi hàm trả về mảng, biến giữ giá trị trả về phải là một mảng và có kiểu như kiểu của hàm, nếu không nó sẽ báo lỗi "*không tương thích kiểu*".

VII. Kiểu do người dùng định nghĩa - Kiểu mẩu tin

Cú pháp:

Type <tên kiểu>
<Tên trường 1> : <Kiểu trường 1>
<Tên trường 2> : <Kiểu trường 2>
:
<Tên trường n> : <Kiểu trường n>
End Type

Ví dụ:

```
Type    TEmployee
    Fullname   As   String
    Salary     As   Single
    Age        As  Integer
End   Type
```

Chúng ta vừa định nghĩa một kiểu dữ liệu mới có tên là TEmployee. Kiểu này có nét tương tự như một lớp. Về mặt chức năng, cả hai là như nhau, nhưng một lớp có thể chứa trong DLL và sẵn sàng cho việc dùng chung với các ứng dụng khác, trong khi đó kiểu dữ liệu do người dùng định nghĩa phải được khai báo lại trong từng dự án. Do vậy, kiểu lớp có nhiều mặt tiện lợi hơn.

Cách truy xuất từng trường của kiểu mẩu tin:

<Tên biến mẩu tin>.<Tên trường>

Ví dụ: Giả sử ta có khai báo biến sau:

```
Dim e As TEmployee
```

Ta có thể gán:

```
e.Fullname = "Nguyen Van An"
e.Salary = 300000.00
e.Age = 26
```

Câu lệnh With:

- Được sử dụng để viết gọn hơn khi thao tác với dữ liệu kiểu mẩu tin.
- Cú pháp:

With <Tên biến mẩu tin>

[Truy xuất đến từng trường của mẩu tin theo dạng:

.<Tên trường>

]

End With

Ví dụ:
```
Dim e As TEmployee
```
Ta có thể gán:
```
With e
    .Fullname = "Nguyen Van An"
    .Salary = 300000.00
    .Age = 26
End With
```

CHƯƠNG 5 THIẾT KẾ BIỂU MẪU DÙNG CÁC ĐIỀU KHIỂN

Mục tiêu:

Chương này giới thiệu về các điều khiển dùng trong việc tạo giao diện cho các ứng dụng chạy trên Windows. Việc nắm bắt được các vấn đề này làm cho công việc tạo giao diện cho ứng dụng được nhanh chóng.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Sử dụng các điều khiển hộp danh sách, hộp lựa chọn để lưu các danh sách.
- Sử dụng các điều khiển hộp đánh dấu, nút lựa chọn để nhận/hiển thị dữ liệu dạng Yes/No.
- Sử dụng các điều khiển hộp hình ảnh, điều khiển ảnh để hiển thị ảnh.
- Sử dụng điều khiển thanh cuộn để nhận/hiển thị dữ liệu số.
- Sử dụng điều khiển thời gian để đáp ứng sự trôi đi của thời gian.
- Một số điều khiển khác.

Kiến thức có liên quan:

- Các cấu trúc lập trình trong VB.
- Cách thức xử lý sự kiện.

Tài liệu tham khảo:

- **Microsoft Visual Basic 6.0 và Lập trình Cơ sở dữ liệu** - Chương 3, trang 29 - **Nguyễn Thị Ngọc Mai** (chủ biên), **Nhà xuất bản Giáo dục** - **2000**.

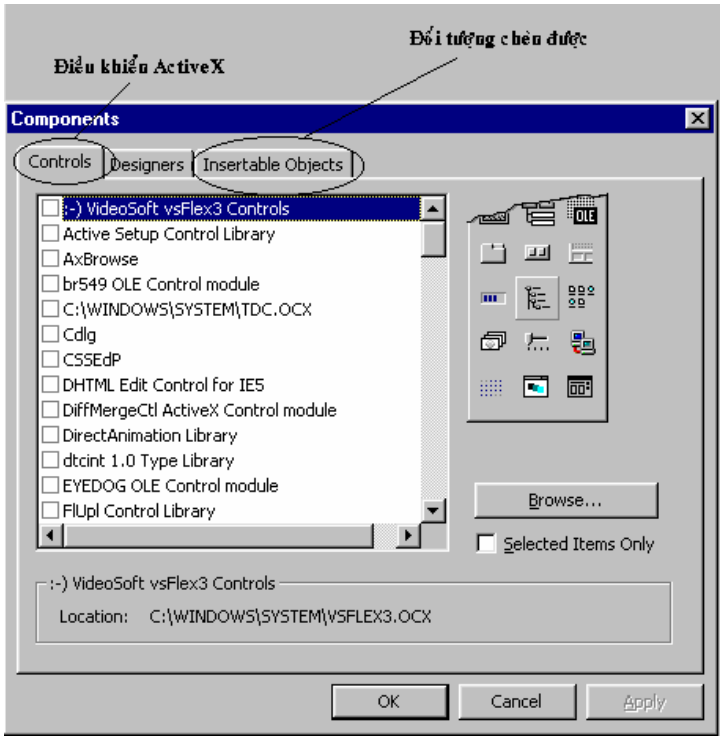
I. Phân loại điều khiển

Có 3 nhóm điều khiển trong Visual Basic:

Các điều khiển nội tại (Intrinsic control). Các điều khiển nội tại luôn chứa sẵn trong hộp công cụ (nhãn, khung, nút lệnh, khung ảnh...). Ta không thể gỡ bỏ các điều khiển nội tại ra khỏi hộp công cụ.

Các điều khiển ActiveX tồn tại trong các tập tin độc lập có phần mở rộng .OCX: Đó là các điều khiển có thể có trong mọi phiên bản của VB hoặc là các điều khiển chỉ hiện diện trong ấn bản Professional và Enterprise. Mặt khác còn có rất nhiều điều khiển ActiveX do nhà cung cấp thứ ba cung cấp.

Các đối tượng chèn được (Insertable Object): Các đối tượng này có thể là Microsoft Equation 3.0 hoặc bảng tính (Worksheet) của Microsoft Excel... Một vài đối tượng kiểu này cho phép ta lập trình với các đối tượng sinh ra từ các ứng dụng khác ngay trong ứng dụng VB.



II. Sử dụng các điều khiển

II.1 Điều khiển danh sách các lựa chọn (List Box)

II.1.1. Khái niệm:

Điều khiển này hiển thị một danh sách các đề mục mà ở đó người dùng có thể chọn lựa một hoặc nhiều đề mục

Biểu tượng (Shortcut) trên hộp công cụ 

Điều khiển này hiển thị một danh sách các đề mục mà ở đó người dùng có thể chọn lựa một hoặc nhiều đề mục

List Box giới thiệu với người dùng một danh sách các lựa chọn. Một cách mặc định, các lựa chọn hiển thị theo chiều dọc trên một cột và bạn có thể thiết lập là hiển thị theo nhiều cột. Nếu số lượng các lựa chọn nhiều và không thể hiển thị hết trong danh sách thì một thanh trượt sẽ tự động xuất hiện trên điều khiển. Dưới đây là một ví dụ về danh sách các lựa chọn đơn cột.

Hình V.1: Ví dụ về List Box

II.1.2. Thuộc tính:

- **Name**: Đây là tên của danh sách lựa chọn, được sử dụng như một định danh.
- **MultiSelect**: Thuộc tính này cho phép List Box có được phép có nhiều lựa chọn khi thực thi hay không?
- **Sort**: List Box có sắp xếp hay không?
- Ngoài ra còn có một số thuộc tính thông dụng khác như: Font, Width, Height…
- **ListIndex**: Vị trí của phần tử được lựa chọn trong List Box.
- **Select(<*Index*>)**: cho biết phần tử thứ <*Index*> trong List Box có được chọn hay không?

II.1.3. Phương thức:

- **AddItem**: Thêm một phần tử vào List Box. Cú pháp:

<Name>.AddIem(Item As String, [*Index*])

Tham số	Diễn giải
Name	Tên của List Box.
Item	Biểu thức chuỗi (đề mục) cần thêm vào.
Index	Xác định vị trí đề mục mới được chèn vào, giá trị 0 xác định cho vị trí đầu tiên. Khi không chỉ định rõ *Index* thì phần tử thêm vào là mục cuối cùng trong List Box mới.

Sau đây là đoạn mã ví dụ tạo một List Box có tên List1 với các đề mục "Germany," "India," "France," và "USA" vào lúc *biểu mẫu được nạp* (Load).

```
Private Sub Form_Load ()
```

```
        List1.AddItem "Germany"
        List1.AddItem "India"
        List1.AddItem "France"
        List1.AddItem "USA"
End Sub
```

Người dùng cũng có thể thêm vào một đề mục mới một cách tự động vào bất kỳ thời điểm nào nhằm đáp lại tác động từ phía người sử dụng ứng dụng. Dưới đây là hình ảnh minh họa cho List Box tương ứng với đoạn mã ở trên.

Hình V.2 List box hiển thị các quốc gia

Thêm một đề mục mới tại vị trí xác định: để thực hiện công việc này ta chỉ cần chỉ ra vị trí cần xen đề mục mới vào.

Ví dụ: `List1.AddItem "Japan", 0`

Thêm mới đề mục tại thời điểm thiết kế: Sử dụng thuộc tính List của điều khiển List Box, ta có thể thêm mới các đề mục và dùng tổ hợp phím CTRL+ENTER để bắt đầu thêm vào đề mục mới trên dòng khác. Khi đã thêm xong danh sách các đề mục, ta có thể sắp xếp lại các đề mục bằng cách sử dụng thuộc tính Sorted và đặt giá trị của thuộc tính này là TRUE.

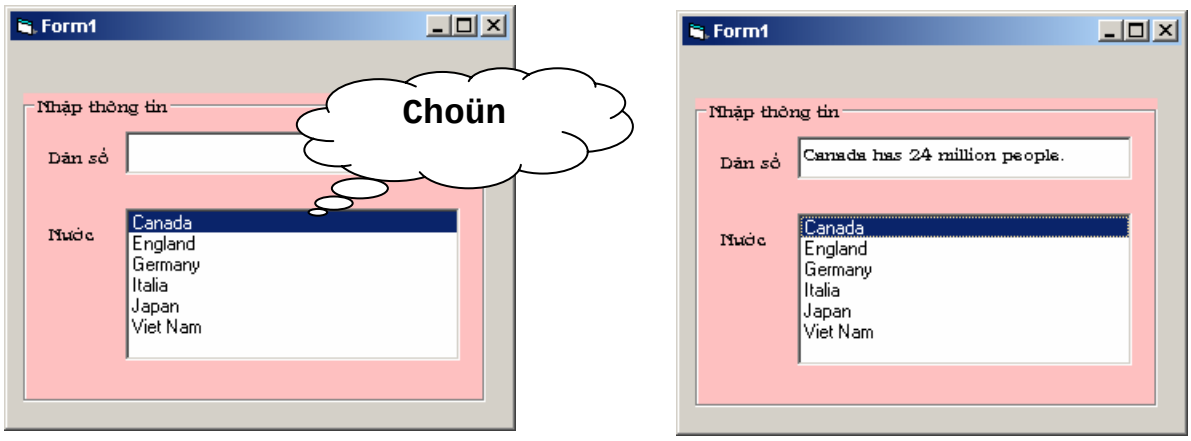
- **RemoveItem**: Xóa một phần tử ra khỏi List Box.

  Cú pháp: <**Name**>.**RemoveItem** *Index*
  Tham số **Name** và *Index* giống như ở trường hợp thêm vào một đề mục.

- **Clear**: Xóa tất cả các mục trong List Box. Cú pháp <**Name**>.*Clear*

- **Text**: Nhận giá trị từ List Box khi một đề mục được chọn. Chẳng hạn đoạn mã sau đây sẽ cho biết dân số của Canada khi người dùng chọn Canada từ List Box.

```
Private Sub List1_Click ()
     If List1.Text = "Canada" Then
          Text1.Text = "Canada has 24 million people."
     End If
End Sub
```



Hình V.3 Ví dụ về List Box

- **List**: truy xuất nội dung phần tử bất kỳ trong List Box.

Thuộc tính này cho phép truy xuất tất cả các đề mục của điều khiển List Box. Thuộc tính này chứa một mảng và mỗi đề mục là một phần tử của mảng. Mỗi đề mục được hiển thị dưới dạng chuỗi, để tham chiếu đến một đề mục trong danh sách, sử dụng cú pháp sau:

<**Name**>.**List**(*Index*)

Ví dụ: `Text1.Text = List1.List(2)`

II.1.4. Sự kiện:

- **Click & Double Click**: Xảy ra khi người sử dụng nhấp chuột (hay nhấp đúp) vào List Box.

Thông thường người sử dụng sẽ thiết kế một nút lệnh đi kèm để nhận về giá trị do người dùng chọn. Khi đó công việc thực hiện sau khi nút lệnh được chọn sẽ phụ thuộc vào giá trị người dùng chọn từ List Box.

Double Click lên một đề mục trong danh sách cũng có kết quả tương tự như việc chọn một đề mục trong danh sách rồi ấn lên nút lệnh. Để thực hiện công việc như trên trong sự kiện Double Click của List Box ta sẽ gọi đến sự kiện Click của nút lệnh.

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

Hoặc ta có thể thiết đặt giá trị **True** cho thuộc tính **Value** của nút lệnh.

```
Private Sub List1_DblClick ()
     Command1.Value = True
End Sub
```

II.2 Điều khiển hộp lựa chọn (Combo Box)

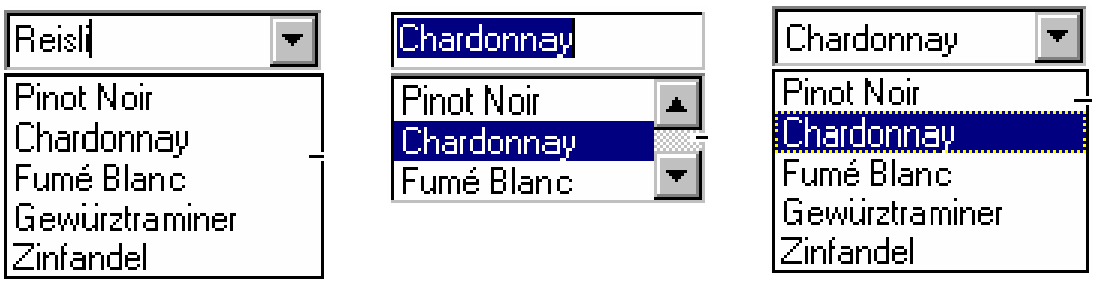
Điều khiển Combo Box có thể được xem là tích hợp giữa hai điều khiển Text Box và List Box. Người dùng có thể chọn một đề mục bằng cách đánh chuỗi văn bản vào Combo Box hoặc chọn một đề mục trong danh sách.

Điểm khác nhau cơ bản giữa Combo Box và List Box là điều khiển Combo chỉ gợi ý (hay đề nghị) các lựa chọn trong khi đó điều khiển List thì giới hạn các đề mục nhập vào tức là người dùng chỉ có thể chọn những đề mục có trong danh sách. Điều khiển Combo chứa cả ô nhập liệu nên người dùng có thể đưa vào một đề mục không có sẵn trong danh sách.

Biểu tượng short cut trên hộp công cụ: 

Các dạng của điều khiển Combo Box: Có tất cả 3 dạng của điều khiển Combo Box. Ta có thể chọn dạng của Combo tại thời điểm thiết kế bằng cách dùng giá trị hoặc hằng chuỗi của VB.

Kiểu	Giá trị	Hằng
Drop-down Combo Box	**0**	VbComboDropDown
Simple Combo Box	1	VbComboSimple
Drop-down List Box	2	vbComboDropDownList



Drop-down combo box | Simple combo box | Drop-down list box

Hình V.4: Các dạng combo box

- *Drop-down Combo Box*: Đây là dạng mặc nhiên của Combo. Người dùng có thể nhập vào trực tiếp hoặc chọn từ danh sách các đề mục.

- *Simple Combo Box*: Ta có thể hiển thị nhiều đề mục cùng một lúc. Để hiển thị tất cả các đề mục, bạn cần thiết kế Combo đủ lớn. Một thanh trượt sẽ xuất hiện khi còn đề mục chưa được hiển thị hết. Ở dạng này, người dùng vẫn có thể nhập một chuỗi vào trực tiếp hoặc chọn từ danh sách các đề mục.

- *Drop down List Box*: Dạng này rất giống như một List box. Một điểm khác biệt đó là các đề mục sẽ không hiển thị đến khi nào người dùng Click lên mũi tên phía phải của điều khiển. Điểm khác biệt với dạng thứ 2 đó là người dùng không thể nhập vào trực tiếp một chuỗi không có trong danh sách.

Các thuộc tính cũng như các phương thức áp dụng trên Combo Box giống như trên List Box.

II.3 Điều khiển hộp đánh dấu (Check Box)

II.3.1. Khái niệm:

Đây là điều khiển hiển thị dấu ✓ nếu như được chọn và dấu ✓ bị xoá nếu như không chọn. Dùng điều khiển Check Box để nhận thông tin từ người dùng theo dạng Yes/No hoặc True/False. Ta cũng có thể dùng nhiều điều khiển trong một nhóm để hiển thị nhiều khả năng lựa chọn trong khi chỉ có một được chọn. Khi Check Box được chọn, nó có giá trị 1 và ngược lại có giá trị 0.

Biểu tượng shortcut trên hộp công cụ 

II.3.2. Thuộc tính:

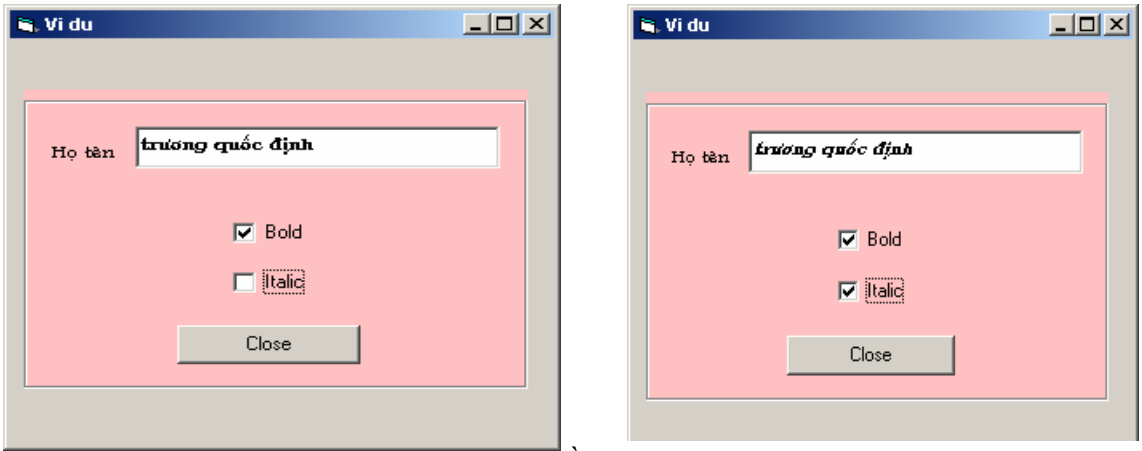
- **Name**: thuộc tính tên.
- **Value**: Giá trị hiện thời trên Check Box. Có thể nhận các giá trị: vbChecked, vbUnchecked, vbGrayed.

II.3.3. Sự kiện:

- **Click**: Xảy ra khi người sử dụng nhấp chuột trên Check Box.

II.3.4. Ví dụ:

Thiết kế chương trình có giao diện:



Hình V.5 Ví dụ về Check Box

Hình V.5 là hình minh họa cách dùng Check Box để hiển thị chuỗi dưới dạng tô đậm và nghiêng.

```
Private Sub chkBold_Click ()
     If ChkBold.Value = vbChecked Then    ' If checked.
          txtDisplay.Font.Bold = True
     Else                  ' If not checked.
               txtDisplay.Font.Bold = False
     End If
End Sub

Private Sub chkItalic_Click ()
     If   ChkItalic.Value = vbChecked Then    ' If checked.
          txtDisplay.Font.Italic = True
     Else If not checked.
```

```
            txtDisplay.Font.Italic = False
            End If
End Sub
```

II.4 Điều khiển nút lựa chọn (Option Button)

II.4.1. Khái niệm:

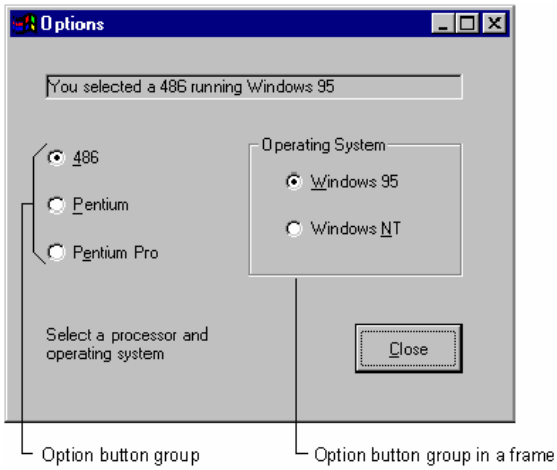
Công dụng của điều khiển Option button cũng tương tự như điều khiển Check Box. Điểm khác nhau chủ yếu giữa hai loại điều khiển này đó là: Các Option Button của cùng một nhóm tại mỗi thời điểm chỉ có một điều khiển nhất định được chọn.

Biểu tượng Shortcut trên hộp công cuï 

Cách sử dụng Option button cũng tương tự như của Check Box.

Tạo nhóm Option Button

Tất cả các Option button đặt trực tiếp trên biểu mẫu (có nghĩa là không thuộc vào Frame hoặc Picture Box) sẽ được xem như là một nhóm. Nếu người dùng muốn tạo một nhóm các Option button khác thì bắt buộc phải đặt chúng bên trong phạm vi của một Frame hoặc Picture box.



Hình V.6 Nhóm các option button

II.4.2. Thuộc tính:

- **Name**: thuộc tính tên của điều khiển Option Button.
- **Value**: Giá trị hiện thời trên Option Button. Có thể nhận các giá trị: True & False.

II.4.3. Sự kiện:

- **Click**: Xảy ra khi người sử dụng nhấp chuột trên Option Button.

II.5 Điều khiển thanh cuộn ngang (HScrollBar)

II.5.1. Khái niệm:

Là điều khiển có thanh trượt cho phép cuộn ngang và người dùng có thể sử dụng HScrollBar như một thiết bị nhập hoặc một thiết bị chỉ định cho số lượng hoặc vận tốc. Ví dụ ta thiết kế volume cho một trò chơi trên máy tính hoặc để diễn đạt có bao nhiêu thời gian trôi qua trong một khoảng định thời nhất định.

Biểu tượng Shortcut trên hộp công cuï 

Khi người dùng sử dụng Scroll Bar như một thiết bị chỉ định số lượng thì người dùng cần xác định giá trị cho hai thuộc tính Max và Min để đưa ra khoảng thay đổi thích hợp.

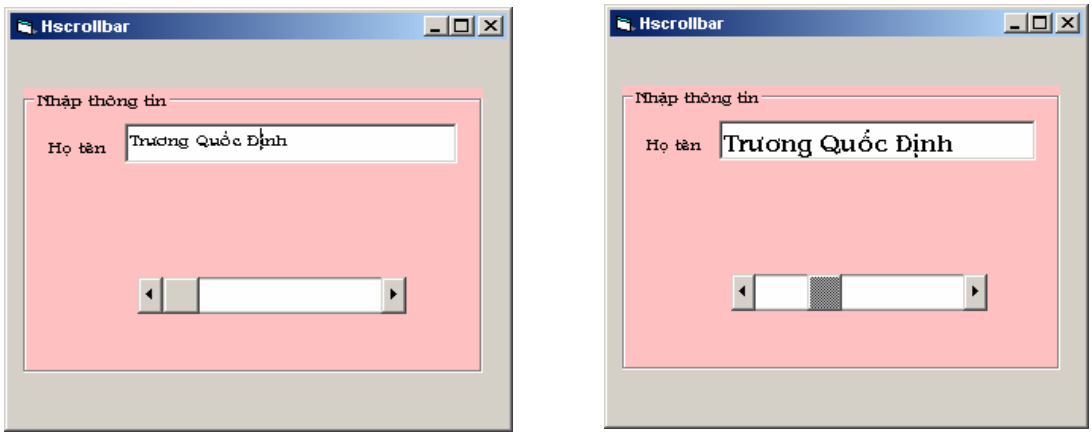
II.5.2. Thuộc tính:

- **Name**: Tên của thanh cuộn.
- **Min**: Là giá trị nhỏ nhất trên thanh cuộn.
- **Max**:: Giá trị lớn nhất của thanh cuộn.
- **Large change**: Thuộc tính này dùng để xác định khoảng thay đổi khi người dùng ấn chuột lên Hscrollbar.
- **Small change**: Thuộc tính này dùng để xác định khoảng thay đổi khi người dùng ấn lên mũi tên phía cuối thanh cuộn.
- **Value**: Thuộc tính này trả về giá trị tại một thời điểm của thanh cuộn nằm trong khoảng giá trị [Min, Max] mà người dùng đã xác định.

II.5.3. Sự kiện:

- **Change**: Xảy ra mỗi khi HScrollBar thay đổi giá trị.
- **Scroll**: Xảy ra mỗi khi ta di chuyển con trỏ thanh cuộn.

II.5.4. Ví dụ:



Hình V.7 Ví dụ về Hscrollbar

Hình V.7 minh họa việc sử dụng thanh cuộn để thay đổi kích cỡ của ô Text họ tên với đoạn mã đơn giản như sau:

```
Private Sub HScroll1_Change()
        Text1.FontSize = HScroll1.Value
End Sub
```

II.6 Điều khiển thanh cuộn đứng (VScrollBar)

Biểu tượng shortcut trên hộp công cuï 

Các thuộc tính và công dụng của VScrollBar cũng tương tự như HScrollBar.

II.7 Điều khiển hộp hình ảnh (Picture Box)

II.7.1. Khái niệm:

Điều khiển Picture Box cho phép người dùng hiển thị hình ảnh lên một biểu mẫu.

Biểu tượng Shortcut trên hộp công cụ 

II.7.2. Thuộc tính:

- **Name**: tên của điều khiển Picture Box.
- **Picture**: Đây là thuộc tính cho phép xác định hình ảnh nào sẽ được hiển thị bên trong Picture box. Bao gồm tên tập tin hình ảnh và cả đường dẫn nếu có.

Để hiển thị hoặc thay thế một hình ảnh tại thời điểm chạy chương trình thì người dùng có thể dùng phương thức LoadPicture để đặt lại giá trị của thuộc tính Picture với cú pháp như trong ví dụ dưới đây:

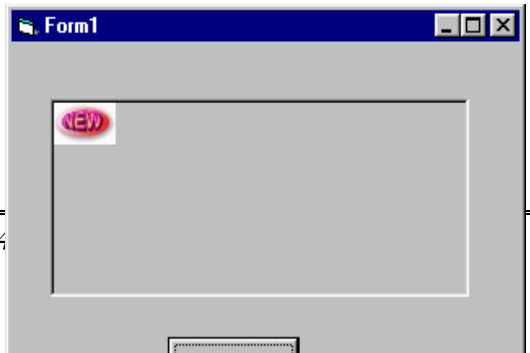
```
picMain.Picture = LoadPicture("NEW.JPG")
```

- **Autosize**: Khi giá trị của thuộc tính này là TRUE thì điều khiển Picture box sẽ tự động thay đổi kích thước cho phù hợp với hình ảnh được hiển thị. Ta nên cẩn thận khi sử dụng thuộc tính này vì khi điều khiển Picture Box thay đổi kích thước, nó không quan tâm đến vị trí của các điều khiển khác.

II.7.3. Sự kiện:

- **Mouse Down**: Xảy ra khi người sử dụng chương trình nhấn giữ phím chuột.
- **Mouse Move**: Xảy ra khi người sử dụng chương trình di chuyển chuột.
- **Mouse Up**: Xảy ra khi người sử dụng chương trình thả phím chuột.

II.7.4. Ví dụ:



```
Private Sub Picture1_MouseMove(Button As Integer, Shift As _
                Integer,    X  As Single, Y As Single)
        Picture1.Picture = LoadPicture("d:\quocdinh\new.jpg")
End Sub
```

II.7.5. Lưu ý:

- Điều khiển Picture Box có thể được dùng như một vật chứa các điều khiển khác (tương tự như một Frame).
- Ngoài ra người dùng cũng có thể sử dụng Picture Box như một khung vẽ hoặc như một khung soạn thảo và có thể in được nội dung trên đó.

II.8 Điều khiển hình ảnh (Image)

II.8.1. Khái niệm:

Điều khiển Image dùng để hiển thị một hình ảnh. Các dạng có thể là Bitmap, Icon, Metafile, Jpeg, Gif. Tuy nhiên khác với điều khiển Picture Box điều khiển Image sử dụng tài nguyên hệ thống ít và cũng nạp ảnh nhanh hơn; hơn nữa số lượng thuộc tính và phương thức áp dụng ít hơn điều khiển Picture box.

Biểu tượng Shortcut trên hộp công cụ 

II.8.2. Thuộc tính:

- **Name**: tên của điều khiển Image.
- **Picture**: Đây là thuộc tính cho phép xác định hình ảnh nào sẽ được hiển thị bên trong điều khiển Image. Bao gồm tên tập tin hình ảnh và cả đường dẫn nếu có.

Để hiển thị hoặc thay thế một hình ảnh tại thời điểm chạy chương trình thì người dùng có thể dùng phương thức LoadPicture để đặt lại giá trị của thuộc tính Picture với cú pháp như trong ví dụ dưới đây:

```
imgMain.Picture = LoadPicture("NEW.JPG")
```

- **Stretch**: Khi giá trị của thuộc tính này là TRUE thì điều khiển Image sẽ tự động thay đổi kích thước cho phù hợp với hình ảnh được hiển thị.

II.8.3. Sự kiện:

- **Mouse Down**: Xảy ra khi người sử dụng chương trình nhấn giữ phím chuột.
- **Mouse Move**: Xảy ra khi người sử dụng chương trình di chuyển chuột.
- **Mouse Up**: Xảy ra khi người sử dụng chương trình thả phím chuột.

II.9 Điều khiển hình dạng (Shape)

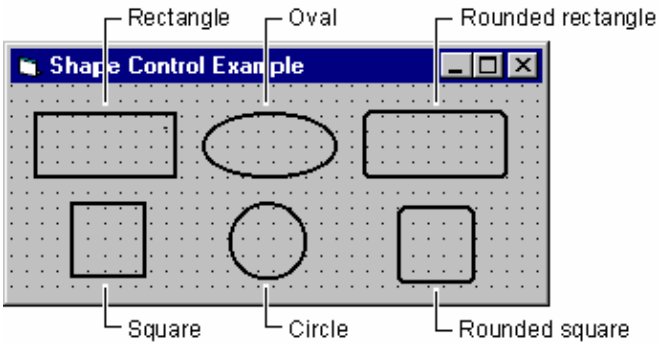
Biểu tượng Shortcut trên hộp công cụ 

Điều khiển Shape dùng để vẽ các hình dạng như: hình chữ nhật, hình vuông, oval, hình tròn, hình chữ nhật góc tròn hoặc hình vuông góc tròn.

Thuộc tính Shape cho phép người dùng chọn 1 trong 6 dạng như đã nêu ở trên. Sau đây là bảng giá trị của thuộc tính này

Hình dạng	Giá trị	Hằng
Rectangle	0	vbShapeRectangle
Square	1	vbShapeSquare
Oval	2	vbShapeOval
Circle	3	vbShapeCircle
Rounded Rectangle	4	vbShapeRoundedRectangle
Rounded Square	5	vbShapeRoundedSquare

Hình V.9 dưới đây minh họa cho các dạng của điều khiển Shape



Hình V.9: 6 dạng của điều khiển Shape

Người dùng có thể sử dụng các thuộc tính như BorderColor, BorderStyle, BorderWidth để xác định màu, dạng cũng như độ dày của đường biên. Tương tự với các thuộc tính FillColor, FillStyle để xác định màu và kiểu tô nền.

II.10 Điều khiển thời gian (Timer)

II.10.1. Khái niệm:

Điều khiển Timer đáp ứng lại sự trôi đi của thời gian. Nó độc lập với người sử dụng và ta có thể lập trình để thực hiện một công việc nào đó cứ sau một khoảng thời gian đều nhau.

Biểu tượng Shortcut trên hộp công cụ 

Việc đưa một điều khiển Timer và trong một biểu mẫu cũng tương tự như những điều khiển khác. Ở đây, ta chỉ có thể quan sát được vị trí của điều khiển Timer

tại giai đoạn thiết kế, khi chạy ứng dụng điều khiển Timer coi như không có thể hiện trên biểu mẫu.

II.10.2. Thuộc tính:

- **Name**: tên của điều khiển Timer.
- **Interval**: Đây là thuộc tính chỉ rõ số *ms* giữa hai sự kiện kế tiếp nhau. Trừ khi nó bị vô hiệu hóa, mỗi điều khiển Timer sẽ luôn nhận được một sự kiện sau một khoảng thời gian đều nhau.

  Thuộc tính Interval nhận giá trị trong khoảng 0...64.767 *ms* có nghĩa là khoảng thời gian dài nhất giữa hai sự kiện chỉ có thể là khoảng một phút (64.8 giây).
- **Enabled**: nếu giá trị là True nghĩa là điều khiển Timer được kích hoạt và ngược lại.

II.10.3. Sự kiện:

- **Timer**: xảy ra mỗi khi đến thời gian một sự kiện được thực hiện (xác định trong thuộc tính **Interval**).

II.10.4. Sử dụng điều khiển Timer:

- **Khởi tạo một điều khiển Timer**: Nếu lập trình viên muốn điều khiển Timer hoạt động ngay tại thời điểm biểu mẫu chứa nó được nạp thì đặt thuộc tính Enable là TRUE hoặc có thể dùng một sự kiện nào đó từ bên ngoài để kích hoạt điều khiển Timer.
- **Lập trình đáp ứng sự kiện trả về từ điều khiển Timer**: Ta sẽ đưa mã lệnh của công viêc cần thực hiện vào trong sự kiện Timer của điều khiển Timer. Sau đây là ví dụ khởi tạo một đồng hồ số nhờ vào điều khiển Timer.

```
Private Sub Timer1_Timer()
      If Label1.Caption <> CStr(Time) Then
                Label1.Caption = Time
      End If
End Sub
```

Thuộc tính Interval được thiết lập là 500 (tức 0.5 giây).

Điều khiển Timer còn hữu ích trong việc tính toán thời gian cho một công việc nào đó, đến một thời điểm nào đó thì ta sẽ khởi tạo một công việc mới hoặc ngưng một công việc không còn cần nữa.

II.11 Điều khiển danh sách ổ đĩa (DriveListbox), danh sách thư mục (DirListbox), danh sách tập tin (FileListbox)

II.11.1. Khái niệm:

- Điều khiển DriveListbox trình bày những ổ đĩa của hệ thống và cho phép người dùng chọn một trong những ổ đĩa đó.

  Biểu tượng shortcut (DriveListbox) trên hộp công cụ 
- Điều khiển DirListbox cho phép trình bày những thư mục và đường dẫn tại thời điểm chạy ứng dụng. Sử dụng điều khiển này để trình bày một danh sách có thứ bậc của các thư mục, có nghĩa là người dùng có thể mở một tài liệu từ danh sách các tài liệu đã có sẵn.

Biểu tượng shortcut (DirListbox) trên hộp công cụ 

- Điều khiển FileListbox cho phép trình bày những tài liệu có sẵn trong thư mục hiện hành. Bạn có thể dùng thuộc tính Partten để xác định kiểu tập tin nào được hiển thị trong danh sách.

Biểu tượng shortcut (FileListbox) trên hộp công cụ 

Thông thường các ứng dụng sẽ sử dụng kết hợp cả ba loại điều khiển trên là DriveListbox, DirListbox và FileListbox.

II.11.2. Ví dụ:

Các thuộc tính được thiết lập tại thời điểm thiết kế

FileListbox: Partten = "*.doc"

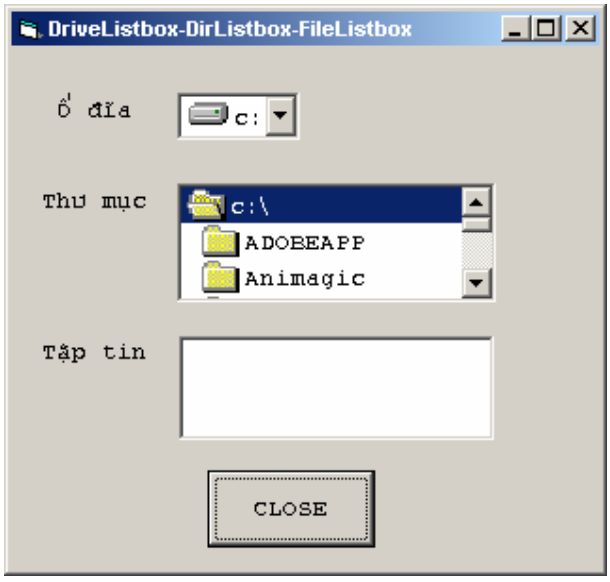
DriveListbox, DirListbox

FileListbox: FontName = "Vncourier New".

Tức là điều khiển FileListbox chỉ hiển thị các tập tin của Microsoft Word.

Đoạn mã dưới đây được viết trong sự kiện Load của biểu mẫu

```
Private Sub Form_Load()
    Dir1.Path = "c:\"
    File1.Path = "c:\"
End Sub
```



Hình V.10 : Ví dụ về các điều khiển ổ đĩa, thư mục và tập tin

Đoạn mã trên cho phép xác định đường dẫn hiện hành của điều khiển DirListbox, FileListbox tại thời điểm nạp ứng dụng là "C:\". Khi chạy ứng dụng ta được như hình V.10.

Khi ta thay đổi ổ đĩa thì các thư mục sẽ thay đổi theo đó là những thư mục ở ổ đĩa hiện hành, sự thay đổi cũng diễn ra tương tự đối với danh sách các tập tin. Cần chú ý rằng đối với ổ đĩa mềm và CD thì cần thiết phải có đĩa mềm hoặc CD tại thời điểm chọn ở đĩa nếu không sẽ xảy ra lỗi tại thời điểm chạy ứng dụng.

```
Private Sub Dir1_Change()
     File1.Path = Dir1.Path
End Sub

Private Sub Drive1_Change()
     Dir1.Path = Drive1.Drive
End Sub
```

Bên trên là những đoạn mã đáp ứng lại việc người dùng chọn việc thay đổi ổ đĩa và thư mục. Trong danh sách các tập tin ta chỉ hiển thị những file *.doc.

CHƯƠNG 6: LẬP TRÌNH XỬ LÝ GIAO DIỆN & ĐỒ HỌA

Mục tiêu:

Chương này giới thiệu về cách tạo menu cũng như một số hàm xử lý đồ họa, những thành phần quan trọng trong các ứng dụng chạy trên Windows.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Sử dụng menu trong thiết kế giao diện.
- Sử dụng các hộp thoại trong thiết kế ứng dụng.
- Sử dụng các phương thức đồ họa cùng với cách xử lý một số sự kiện để tạo các ứng dụng đồ họa.

Kiến thức có liên quan:

- Các cấu trúc lập trình trong VB.
- Cách thức xử lý sự kiện.

Tài liệu tham khảo:

- **Microsoft Visual Basic 6.0 và Lập trình Cơ sở dữ liệu** - Chương 7, trang 99 - **Nguyễn Thị Ngọc Mai** (chủ biên), **Nhà xuất bản Giáo dục** - **2000**.

I. Menu

I.1. Khái niệm

Menu là một loại điều khiển trong đó người sử dụng có thể lựa chọn các mục từ một danh sách cho trước.

Có 2 loại menu:

- Menu thả xuống (Drop-Down Menu): là dạng menu thông dụng nhất.
- Menu bật ra (Pop-Up Menu): thường hiển thị khi ta ấn nút phải chuột.

I.2. Các thuộc tính của Menu

Các thuộc tính của Menu không chứa trong cửa sổ Properties mà chứa trong Menu Editor.

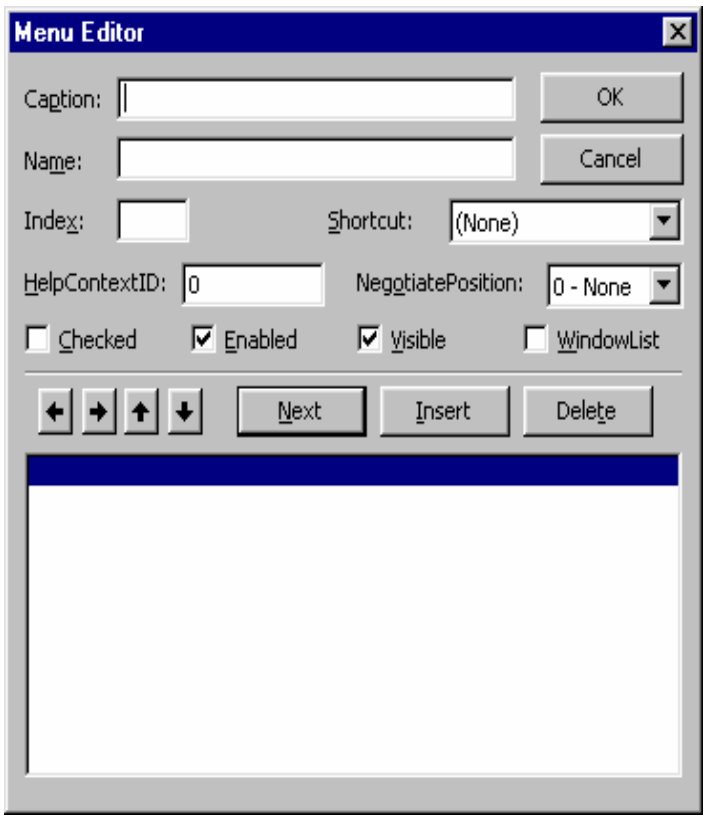
- **Caption**: Là chuỗi hiển thị trên menu.
- **Name**: Phải duy nhất và dễ nhớ. Đây là tên để nhận biết thành phần nào của Menu được chọn.
- **Shortcut**: dùng để thiết lập các phím tắt (Shortcut key).
- **WindowList**: Dùng trong các ứng dụng MDI (Multipe Document Interface). Đây là ứng dụng có một biểu mẫu chính và nhiều biểu mẫu con. Thuộc tính này ra lệnh cho VB hiển thị tiêu đề của các biểu mẫu con trên menu.
- **Checked**: Nếu chọn thuộc tính này thì sẽ có một dấu hiển thị bên cạnh trái, nhưng thuộc tính này không được áp dụng cho những mục menu có chứa menu con.
- **Enabled**: Nếu thuộc tính này không được chọn thì mục này sẽ bị xám đi và người dùng không thể chọn.
- **Visible**: Nếu thuộc tính này không được chọn thì mục này sẽ không được hiển thị.
- **NegotiatePosition**: quản lý vị trí gắn menu trong trường hợp sử dụng các đối tượng ActiveX.

I.3. Các sự kiện

- **Click**: Xảy ra khi người sử dụng chương trình nhấp chuột vào một mục nào đó của Menu.

I.4. Cách tạo Menu

Menu cũng là một loại điều khiển, nhưng Windows sẽ kiểm soát việc vẽ menu. Lập trình viên chỉ quản lý việc điều hành các sự kiện mà thôi.

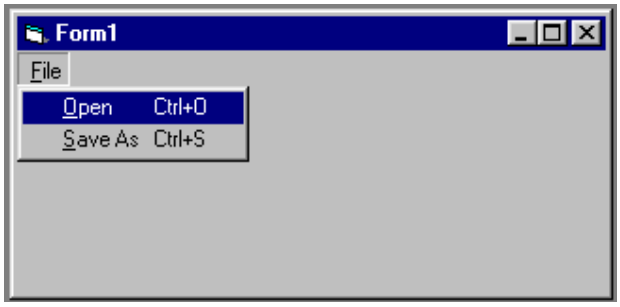


Hình VI.1 Trình soan thảo Menu

Menu không chứa trong hộp công cụ mà được thiết kế từ trình soạn thảo menu. Trong Visual Basic 6.0 IDE, chọn *Tools*, rồi *Menu Editor* để mở chương trình này.

Ví dụ: *Tạo một Drop-Down Menu*.

- Tạo một đề án mới.
- Ấn Ctrl-E để mở Menu Editor
- Ta sẽ tạo một Menu File với Menu con là Open và Save As.
- Trước tiên ta nhập vào &File trong ô Caption và nhập một tên bất kỳ vào ô Name (chẳng hạn là mfile). Ký tự & trước chữ F cho biết chữ F sẽ là phím tắt (ấn Ctrl-F) coi như chọn menu File.
- Tiếp theo ta nhập &Open và &Save As. Để Open và Save As là Sub menu của File, ta chọn Open rồi ấn mũi tên sang trái. Tương tự đối với Save As.

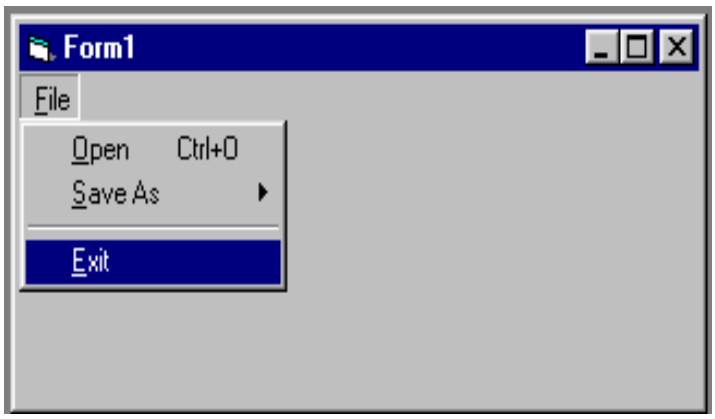


Hình VI.2 Ví dụ tạo Popup Menu

Tách nhóm menu:

Trong trường hợp Menu có nhiều mục, ta sẽ tách nhóm Menu để tiện theo dõi. Chẳng hạn ta thêm vào Menu File mục Exit và tách riêng ra với Open và Save As.

Ta sẽ xen vào giữa hai mục Save As và Exit một mục mới có Caption là "-". Ta có thể theo dõi qua hình VI.3.



Hình VI.3 Ví dụ tách nhóm các

Ví dụ: *Tạo Pop-up menu*

- Sử dụng lại menu đã dùng ở ví dụ trước, nhưng ta sẽ tắt thuộc tính Visible của menu File.
- Sau đó, mở cửa sổ Code của ứng dụng, dùng sự kiện MousUp, nhập vào đoạn lệnh sau:

```
Private Sub Form_MouseUp (Button As Integer, Shift As _
      Integer, X As Single, Y As Single)
            If Button = vbRightButton Then
                  PopupMenu mfile, vbPopupMenuLeftAlign
            End If
End Sub
```

- Chạy thử ứng dụng, khi ta ấn chuột phải, một menu sẽ bật ra.
- Lệnh PopupMenu cho biết tên menu cần bật ra, đó là *tên mà ta đã đặt trong trình soạn thảo MenuEditor*, ở đây là mfile.
- Kế đến, đó là tham số xác định cách hiển thị menu: vbPopupMenuLeftAlign, vbPopupMenuRightAlign, vbPopupMenuCenterAlign.

Sau khi đã thiết kế xong menu, ta sẽ viết các đoạn mã để VB sẽ thi hành một công việc nào đó tương ứng với mục được chọn. *Công việc thi hành sẽ được viết bên trong sự kiện Click của mục đó*.

II. Hộp thoại

II.1. Khái niệm

Hộp thoại (Dialog Box) là một trong những cách VB dùng để giao tiếp với người dùng. Có 4 loại hộp thoại:

- Hộp thông điệp (Message Box).
- Hộp nhập (Input Box).
- Các hộp thoại thông dụng (Common Dialog)
- Hộp thoại hiệu chỉnh (Custom Dialog).

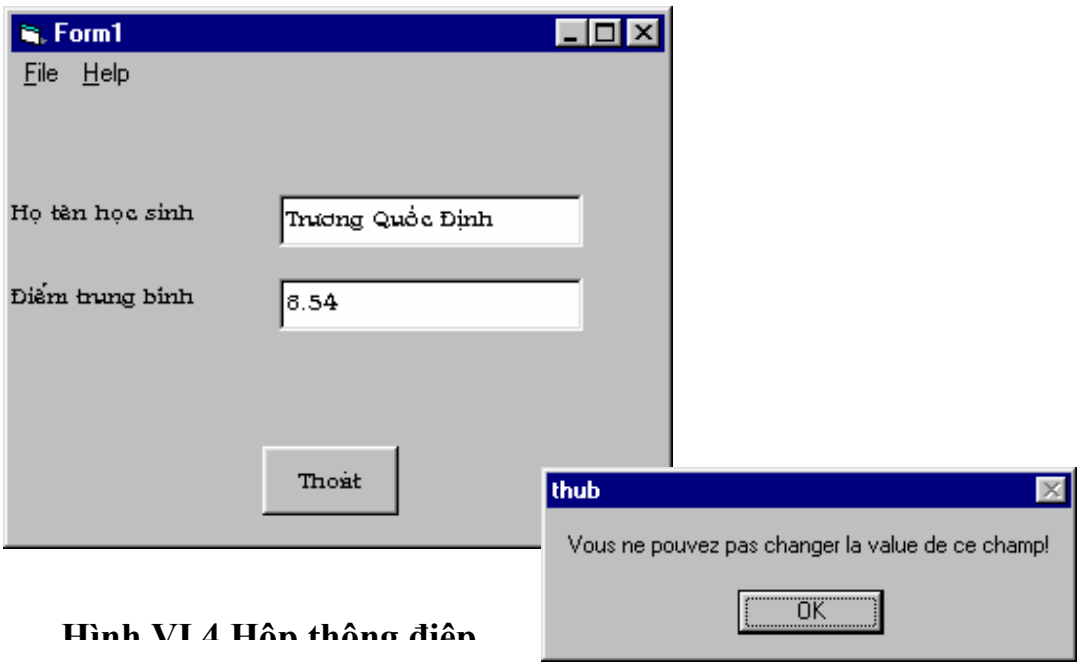
II.2. Hộp thông điệp

Hộp thông điệp cũng có 2 loại: Loại chỉ xuất thông báo, loại có tương tác với người dùng.

II.2.1 Loại chỉ xuất thông báo

- Lúc này ta dùng MsgBox như là một thủ tục.
- Cú pháp: `MsgBox Prompt, Button, Title`. Trong đó:
  Prompt: Chuỗi thông báo sẽ hiển thị.
  Button: Các nút nhấn sẽ được hiển thị trên hộp thông báo.
  Title: Chuỗi hiển thị trên thanh tiêu đề của hộp thông báo.

- Ví dụ:



Hình VI.4 Hộp thông điệp

Chẳng hạn ta xây dựng một biểu mẫu dùng để hiển thị tên và điểm trung bình cuối năm của một học sinh khối lớp 12. Do đó giá trị điểm trung bình cũng như họ tên học sinh là không thể thay đổi. Do đó khi người dùng Click vào một ô Text nào đó, ta sẽ xuất thông báo rằng giá trị này không thể thay đổi.

```
Private Sub Text2_Click()
     MsgBox "Vous ne pouvez pas changer la value de ce champ!"
End Sub
```

Sau khi xuất thông báo, VB sẽ đợi ta ấn vào nút OK hoặc Enter. Sau đó VB sẽ thi hành dòng lệnh ngay sau dòng lệnh MsgBox.

Đôi khi dòng thông báo quá dài, VB sẽ tự động cắt để đưa xuống dòng khác, tuy nhiên có khi sẽ không như mong muốn của lập trình viên. Ta có thể thực hiện công việc này như sau:

```
MsgBox "This is a multi-line " & chr$(10) & " message"
```

Tùy theo thông số truyền vào MsgBox mà có nhiều loại hộp thoại thông điệp khác nhau.

Hằng số	Giá trị	Diễn giải
vbOKOnly	**0**	Chỉ hiển thị nút OK .
vbOKCancel	1	Hiển thị 2 nút OK và Cancel.
vbAbortRetryIgnore	2	Hiển thị các nút Abort, Retry, và Ignore.
vbYesNoCancel	3	Hiển thị các nút Yes, No, và Cancel.
vbYesNo	4	Hiển thị 2 nút Yes và No.
vbRetryCancel	5	Hiển thị 2 nút Retry và Cancel.

Các loại biểu tượng trên hộp công cụ

Hằng số	Diễn giải
vbCritical	Dùng cho những thông báo lỗi thất bại khi thi hành công việc nào đó.
vbQuestion	Dùng cho những câu hỏi yêu cầu người dùng chọn lựa.
vbExclamation	Dùng cho các thông báo của chương trình.
vbInformation	Dùng cho các thông báo cung cấp thêm thông tin.

II.2.2 Loại tương tác với người dùng

Lúc này MsgBox được dùng như một hàm, khi một nút nào đó trên hộp thông báo được ấn, VB sẽ trả về giá trị của nút ấn đó.

Cú pháp:

```
MsgBox (Prompt, Button, Title) As Integer
```

Hằng số	Giá trị	Nút
vbOK	**1**	OK
vbCancel	2	Cancel
vbAbort	3	Abort
vbRetry	4	Retry
vbIgnore	5	Ignore
vbYes	6	Yes

vbNo	7	No

Với những thông điệp quan trọng, ta mong muốn người dùng phải chọn lựa một trong các đề xuất mà ta đưa ra trước khi chuyển qua ứng dụng khác, ta sẽ dùng thông số vbSystemModal.

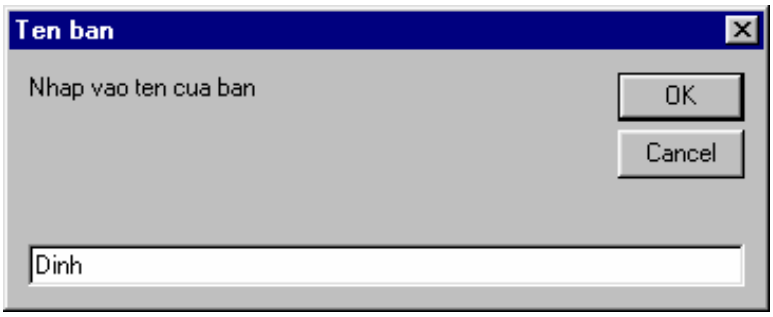
II.3. Hộp nhập

Đây là loại hộp thông điệp cho phép nhận thông tin từ phía người sử dụng. Tuy nhiên trong các ứng dụng, hộp nhập rất ít khi được dùng do:

- Ta không có cách nào để kiểm tra thông tin do người dùng nhập vào khi mà Enter chưa được ấn.
- Thông tin được nhập là rất ít.

Sau đây là một ví dụ về hộp nhập:

```
Public Sub Main ()
     Dim   ReturnString    As    String
     ReturnString = InputBox("Nhap vao ten cua ban")
End  Sub
```



Giá trị trả về của hộp nhập là một chuỗi.

Hình: Một dạng nhập tên

II.4. Các hộp thoại thông dụng

Có 6 loại hộp thoại thông dụng:

- Mở tập tin
- Lưu tập tin
- Chọn màu
- Chọn Font
- In ấn
- Trợ giúp

Tuy có 6 loại, nhưng khi thiết kế biểu mẫu, ta chỉ thấy một công cụ duy nhất đó là CommonDialog. Muốn đưa Common Dialog vào dự án, ta chọn: *Project/Components…/Controls/Microsoft Common Dialog Control 6.0*. Sau đó, Common Dialog sẽ xuất hiện trong hộp công cụ ToolBox.

II.4.1 Hộp thoại mở và lưu tập tin

Hai hộp thoại này có chức năng và thể hiện như nhau. Cả hai hộp thoại đều hiển thị danh sách các tập tin, người dùng có thể duyệt qua các ổ đĩa để tìm các tập tin. Chúng chỉ khác nhau phần tiêu đề và nút nhấn.

Các thuộc tính quan trọng:

- Name: tên của Common Dialog.
- Filter: đây là một chuỗi xác định phần mở rộng của tên các tập tin mà hộp thoại có thể mở hay lưu.

- FilterIndex: nếu có nhiều phần mở rộng của tên tập tin được mô tả trong thuộc tính Filter thì thuộc tính này xác định mặc định loại tập tin nào được chọn (là một số nguyên).
- FileName: trả về tên tập tin sau khi người sử dụng hộp thoại chọn một tập tin nào đó.
- CancelError: nếu TRUE thì trả về giá trị lỗi khi người dùng chọn nút Cancel, mặc nhiên giá trị này là False.

Phương thức:

- ShowOpen: mở ra hộp thoại mở tập tin.
- ShowSave: mở ra hộp thoại lưu tập tin.

Ví dụ:

```
Private Sub Form_Load()
      On Error GoTo ErrHandler
      dlgFile.Filter = "All Files (*.*)|*.*|Text Files " & _
            "(*.txt)|*.txt|Batch Files (*.bat)|*.bat"
      dlgFile.FilterIndex = 2
      dlgFile.ShowOpen
      Exit Sub

   ErrHandler:
      MsgBox Err.Description
End Sub
```

Ở ví dụ trên, ta thiết kế một hộp thoại mở tập tin, trong đó các tập tin được hiển thị theo 3 nhóm tập tin đó là:

- All Files: (*.*)
- Text Files: (*.txt)
- Batch Files: (*.bat)

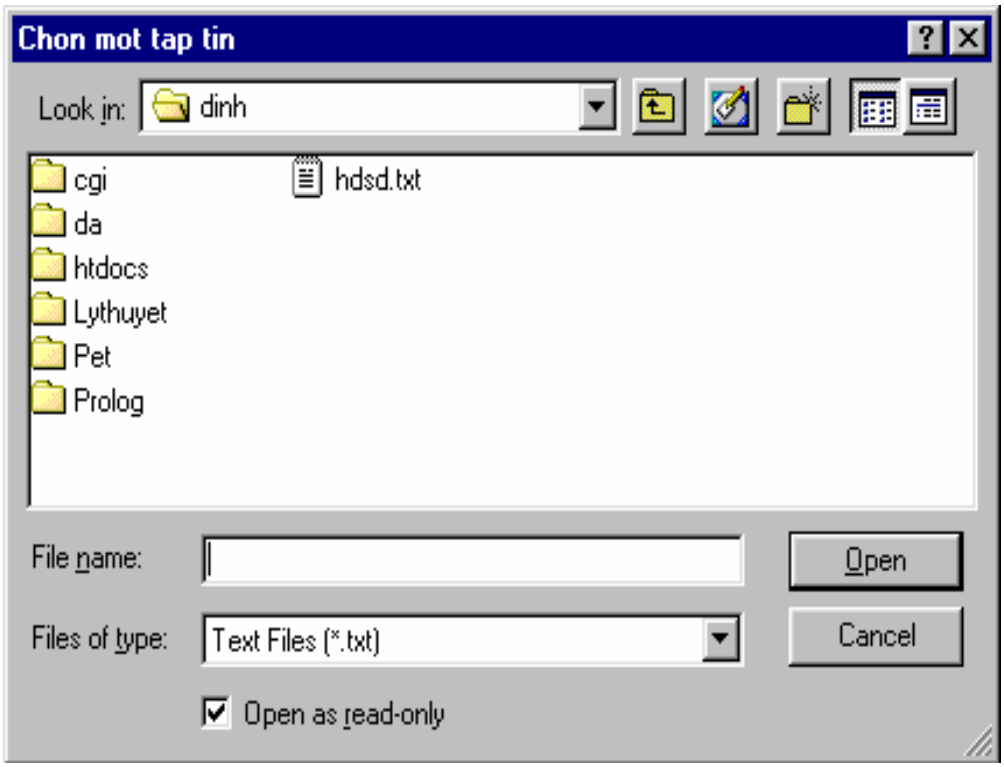
Các nhóm tập tin được thể hiện trong thuộc tính Filter. Mỗi nhóm tập tin cách nhau bởi dấu phân cách |.

Thuộc tính FilterIndex = 2 tức là khi hộp thoại Open được mở lên, thì loại tập tin hiển thị mặc định là Text Files.

Sau khi đã chọn một tập tin và nhấn nút Open, ta sử dụng thuộc tính FileName để nhận về tên tập tin đã chọn.

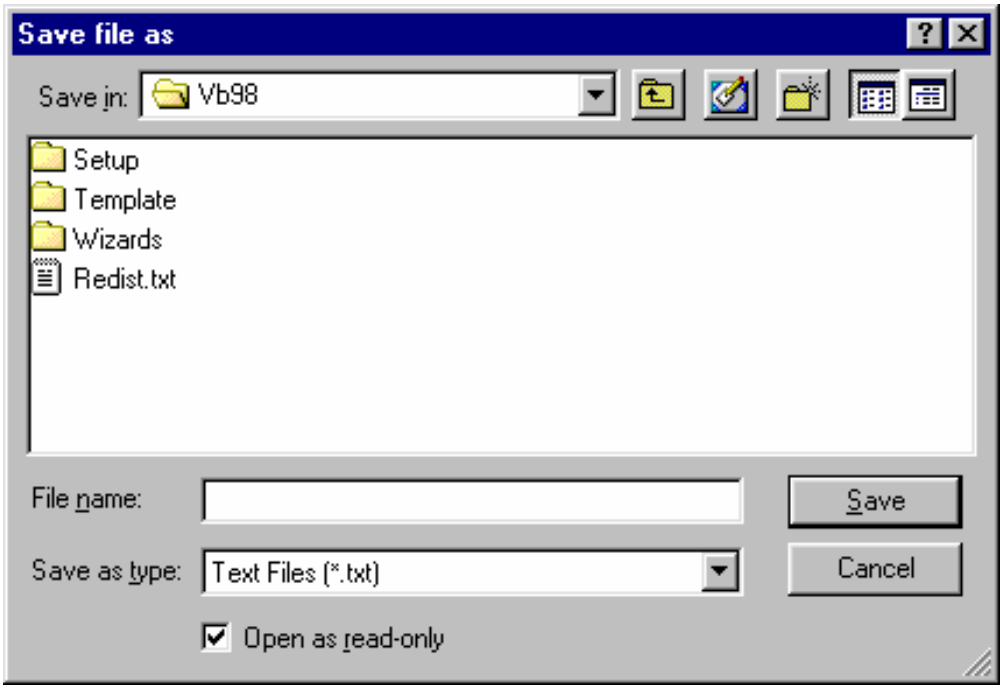
Khi ta chọn thuộc tính CancelError là TRUE, thì khi người dùng ấn nút Cancel trên hộp thoại, ta sẽ nhận được một lỗi và sẽ có cách xử lý lỗi này.

Ta chọn phương thức ShowOpen để hiển thị hộp thoại mở tập tin.



Hình VI.6 Hộp thoại mở tập tin

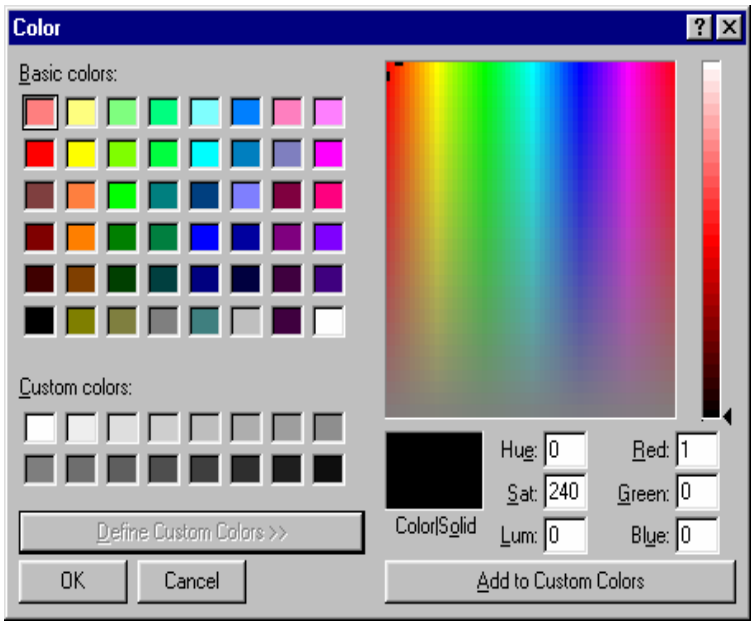
Các thuộc tính cũng tương tự đối với hộp thoại lưu tập tin, ta chỉ cần thay đổi tiêu đề của Dialog và dùng phương thức ShowSave. Dưới đây là minh họa cho hộp thoại lưu tập tin.



Hình VI.7 Hộp thoại lưu tập tin

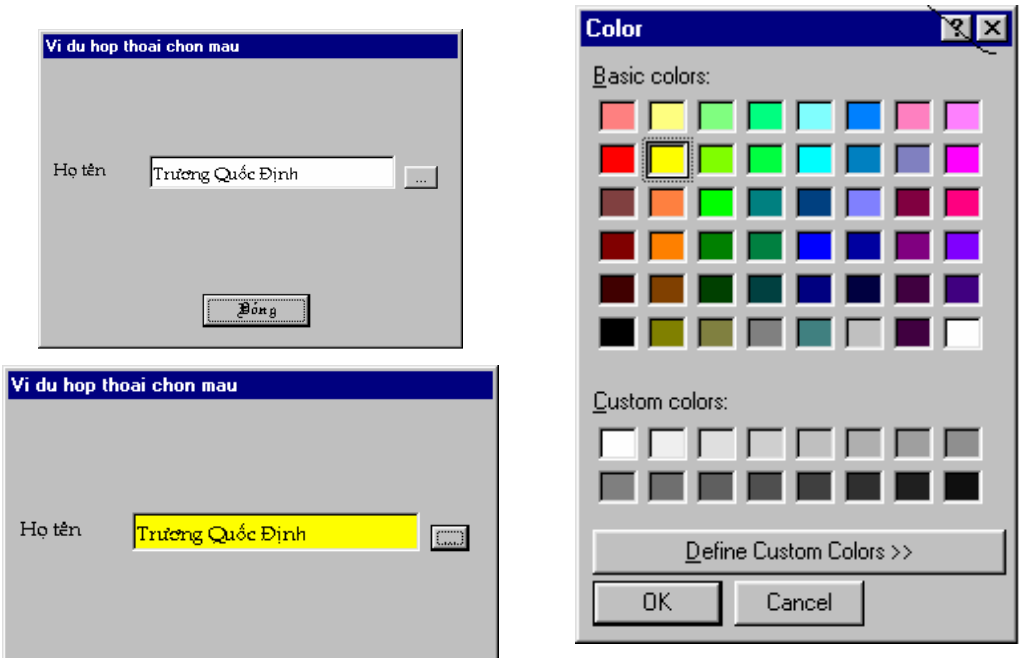
II.4.2 Hộp thoại chọn màu

Đây là hộp thoại cho phép người dùng chọn và hiển thị các màu có sẵn trong bảng màu của Windows cũng như thiết lập thêm nhiều màu mới. Một thuộc tính quan trọng đối với hộp thoại chọn màu đó là thuộc tính Color, thuộc tính này trả về giá trị của màu đã được chọn. Ta sẽ dùng phương thức ShowColor để hiển thị hộp thoại chọn màu.



Hình VI.8 Hộp thoại chọn màu

Trong một số ứng dụng, ta sẽ dùng hộp thoại chọn màu để thay đổi giá trị màu của các điều khiển trong một số trường hợp nào đó. Ví dụ thay đổi màu nền của điều khiển TextBox trong ví dụ dưới đây:



Hình VI.9 Ví dụ sử dụng hộp thoại chọn mầu

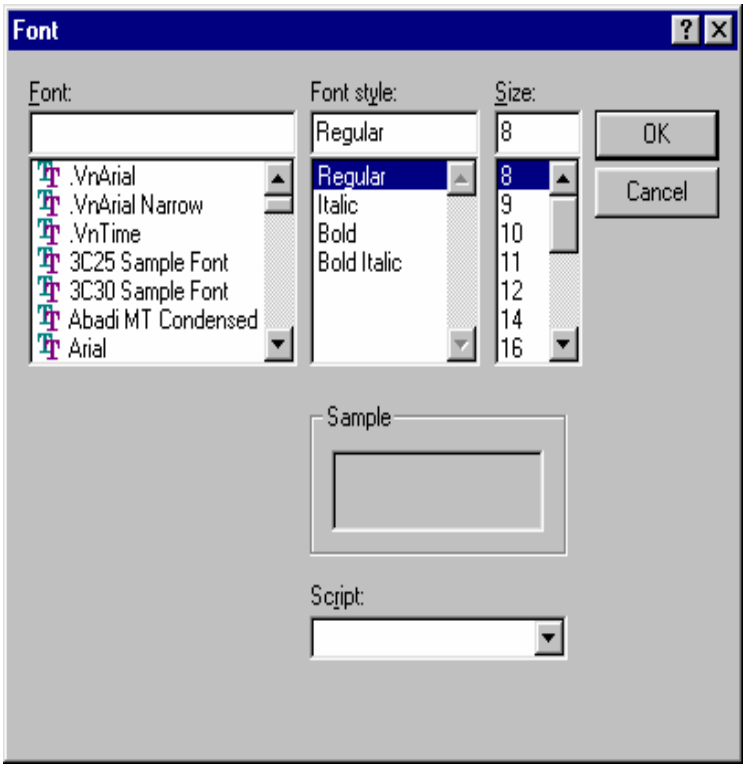
Ta sẽ thiết kế một nút nhấn nhỏ bên cạnh điều khiển TextBox, nút nhấn này cho phép người sử dụng chọn màu nền của TextBox. Ta có đoạn mã lệnh sau:

```
Private Sub Command2_Click()
      On Error GoTo ErrHandler
      CommonDialog1.ShowColor
      Text1.BackColor = CommonDialog1.Color

ErrHandler:
      CommonDialog1.ShowColor
End Sub
```

Trước khi chạy chương trình cần xác định thuộc tính CancelError = TRUE.

II.4.3 Hộp thoại chọn Font chữ

Cho phép người dùng chọn Font màn hình, máy in hay cả hai. Khi dùng hộp thoại chọn Font ta phải dùng thuộc tính Flags quy định loại Font nào sẽ được hiển thị.



Hình VI.10 Hộp thoại Font

Thuộc tính	Giải thích
Color	Lưu giữ giá trị của màu được chọn
FontBold	TRUE nếu người dùng chọn chế độ đậm (Bold) và FALSE nếu

	ngược lại.
FontItalic	TRUE nếu người dùng chọn chế độ nghiêng (Italic) và FALSE nếu ngược lại.
FontStrikeThru	TRUE nếu chọn chế độ gạch ngang các ký tự.
FontUnderLine	TRUE nếu chọn chế độ gạch dưới
FontName	Tùy ý
Max	Kích cỡ lớn nhất của Font được hiển thị
Min	Kích cỡ nhỏ nhất của font được hiển thị
FontSize	Kích cỡ của Font được chọn

Các giá trị của thuộc tính Flags:

Hằng	Giá trị	Hiệu quả
cdlCFPrinterFonts	&H2	Chỉ hiển thị font máy in
cdlCFScreenFonts	&H1	Chỉ hiển thị font màn hình
cdlCFBoth	&H3	Chỉ hiển thị font màn hình và font máy in
cdlCFScalableOnly	&H20000	Hiển thị font tỷ lệ như là fonts TrueType

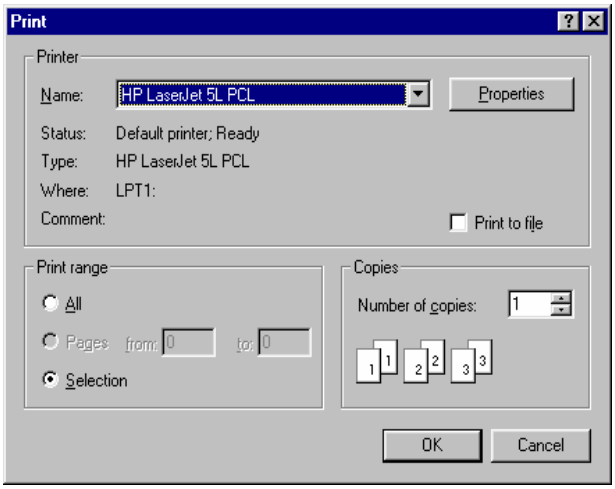
Nếu muốn chọn màu cho Font, ta thêm 256 vào giá trị của thuộc tính Flags. Nếu không có điều này, ta chỉ thấy tên Font, kiểu Font và kích cỡ Font mà thôi.

Để mở hộp thoại chọn Font, ta sử dụng phương thức ShowFont.

II.4.4 Hộp thoại in ấn

Đây là hộp thoại cho phép xác lập các thông tin về máy in chẳng hạn như bao nhiêu dữ liệu được in, máy in sẽ hoạt động như thế nào…

Hộp thoại in ấn, nó trả về 3 thuộc tính thông dụng: Copies, FromPage và ToPage.



Hình VI.11 Hộp thoại in ấn

Thuộc tính	Giải thích
Copies	Số bản in
FromPage	Số thứ tự của trang bắt đầu
Max	Số bản in tối đa cho phép

Min	Số bản in tối thiểu cho phép
PrinterDefault	Nếu gán thành TRUE, mọi thay đổi mà người dùng thực hiện sẽ được ghi lại thành các thay đổi trên hệ thống và có ảnh hưởng đến các ứng dụng khác nếu có sử dụng máy in.
ToPage	Số thứ tự của trang in cuối cùng

Để mở hộp thoại in ấn, ta sử dụng phương thức ShowPrinter.

III. Xử lý các sự kiện chuột và bàn phím

III.1 Sự kiện chuột

Biểu mẫu hoặc điều khiển có thể nhận biết sự kiện chuột khi có con trỏ chuột đi ngang qua.

Có 3 sự kiện chuột chủ yếu, đó là

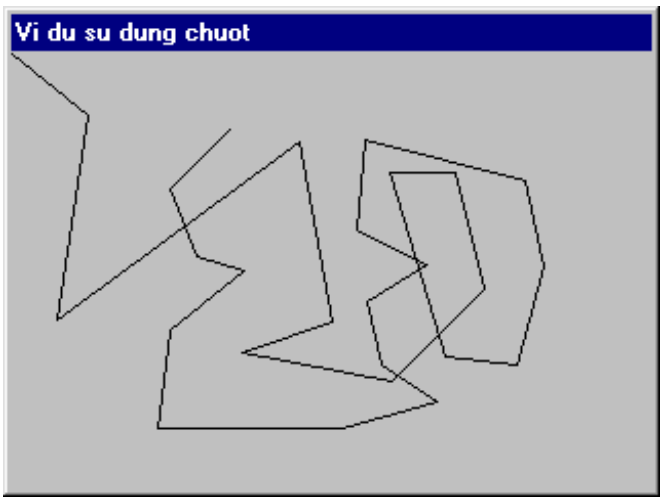
Sự kiện	Giải thích
MouseDown	Xảy ra khi người sử dụng ấn chuột (chuột trái hoặc phải)
MouseUp	Xảy ra khi người sử dụng thả một nút chuột bất kỳ
MouseMove	Xảy ra khi con trỏ chuột di chuyển đến một điểm mới trên màn hình.

Các tham số

Tham số	Giải thích
Button	Cho biết phím chuột nào được ấn
Shift	Cho biết SHIFT hay CTRL hay ALT được ấn
X, Y	Xác định vị trí của con trỏ chuột đối với hệ tọa độ của điều khiển

Ví dụ 1: Sử dụng sự kiện MouseDown để vẽ các đoạn thẳng nối tiếp nhau mỗi khi ta dùng chuột chấm một điểm trên biểu mẫu. Ta có thể thực hiện điều đó với đoạn mã lệnh xử lý sự kiện Form_MouseDown như sau:

```
Private Sub Form_MouseDown(Button As Integer, & _
     Shift As Integer, X As Single, Y As Single)
          Line -(X, Y)
End Sub
```

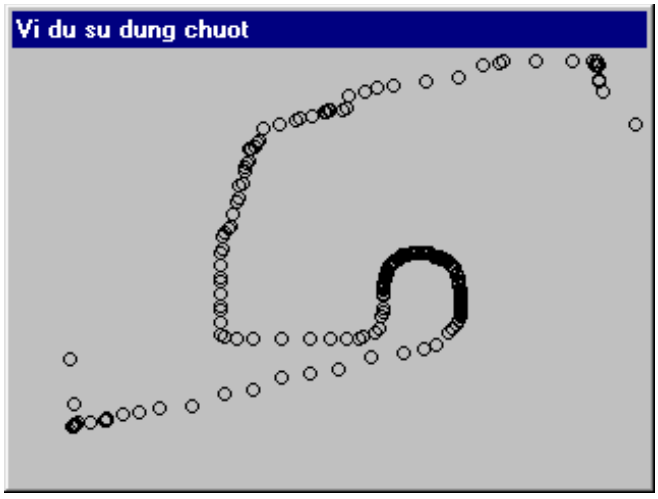


Ví dụ 2: Sử dụng sự kiện MouseUp để hiển thị một thông điệp cho biết nút chuột nào vừa được thả. Sự kiện Form_MouseUp được xử lý:

```
Private Sub Form_MouseUp (Button As Integer,  & _
     Shift As Integer, X As Single, Y As Single)
          If  Button  =  1  Then
               Print  "Ban vua tha phim chuot trai"
          End If
          If  Button  =  2  Then
               Print  "Ban vua tha phim chuot phai"
          End If
          If  Button  =  4  Then
               Print  "Ban vua tha phim chuot giua"
          End If
End Sub
```

Ví dụ 3: Sử dụng sự kiện MouseMove để vẽ các đường tròn liên tục trên biểu mẫu. Sự kiện Form_MouseMove được xử lý:

```
Private Sub Form_MouseMove(Button As Integer, & _
     Shift As Integer, X As Single, Y As Single)
          Circle (X, Y), 50
End Sub
```



Hình VI.13 Ví dụ về MouseMove

Với ví dụ 3 ta nhận thấy rằng: sự kiện MouseMove không nhất thiết phải xảy ra ứng với mỗi Pixel khi con trỏ chuột đi qua. Thực ra mỗi đơn vị thời gian nào đó, hệ điều hành phát ra một số thông điệp. Ở đây, ta vẽ đường tròn ứng với sự kiện MouseMove, nếu người dùng di chuyển chuột chậm, thì các đường tròn sẽ được vẽ sát nhau và ngược lại nếu chuột được di chuyển nhanh.

Hiệu chỉnh con trỏ chuột

Ta có thể dùng thuộc tính MousePointer để hiển thị một biểu tượng, con trỏ màn hình hay con trỏ chuột đã được hiệu chỉnh. Dưới đây là các giá trị của thuộc tính MousePointer:

Hằng	Giá trị	Diễn giải
ccDefault	0	(Default) Shape determined by the object.
ccArrow	1	Arrow.
ccCross	2	Cross (cross-hair pointer).
ccIbeam	3	I Beam.
ccIcon	4	Icon (small square within a square).
ccSize	5	Size (four-pointed arrow pointing north, south, east, and west).
ccSizeNESW	6	Size NE SW (double arrow pointing northeast and southwest).
ccSizeNS	7	Size N S (double arrow pointing north and south).
ccSizeNWSE	8	Size NW, SE.
ccSizeEW	9	Size E W (double arrow pointing east and west).
ccUpArrow	10	Up Arrow.
ccHourglass	11	Hourglass (wait).
ccNoDrop	12	No Drop.
ccArrowHourglass	13	Arrow and hourglass.
ccArrowQuestion	14	Arrow and question mark.
ccSizeAll	15	Size all.

ccCustom	99	Custom icon specified by the MouseIcon property.

III.2 Sự kiện bàn phím

Bàn phím cũng có 3 sự kiện, đó là sự kiện KeyPress (khi một phím có mã ASCII bất kỳ được ấn), KeyDown (khi một phím bất kỳ được ấn), KeyUp (khi một phím bất kỳ được thả)

Chỉ có điều khiển đang có Focus mới bắt sự kiện bàn phím. Còn đối với *biểu mẫu, nó chỉ bắt được sự kiện bàn phím mỗi khi nó đã được kích hoạt và không có bất kỳ điều khiển nào trên nó có Focus. Tuy nhiên ta có khể khắc phục điều này nếu như gán giá trị thuộc tính KeyPreview của biểu mẫu là True*, biểu mẫu sẽ nhận mọi sự kiện bàn phím của mọi điều khiển đặt trên nó, điều này hữu ích khi ta muốn thực hiện cùng một công việc nào đó cho một phím được ấn mà không quan tâm rằng Focus đang thuộc điều khiển nào.

Các sự kiện KeyDown, KeyUp có thể phát hiện một số tình huống mà sự kiện KeyPress không phát hiện:

- Khi người dùng bấm một tổ hợp phím SHIFT, CTRL và ALT.
- Phím định hướng.
- PAGEUP và PAGEDOWN.
- Phân biệt được phím số ở bên phải bàn phím và phím số ở bên trái bàn phím.
- Đáp ứng khi thả phím.
- Phím chức năng không trùng với menu.

Các sự kiện bàn phím là không loại trừ nhau. Tức là một phím được ấn thì có thể là cả hai sự kiện KeyPress và KeyDown cùng được phát ra. Nhưng nếu là một phím mà KeyPress không phát hiện được thì chỉ có KeyDown và KeyUp xảy ra.

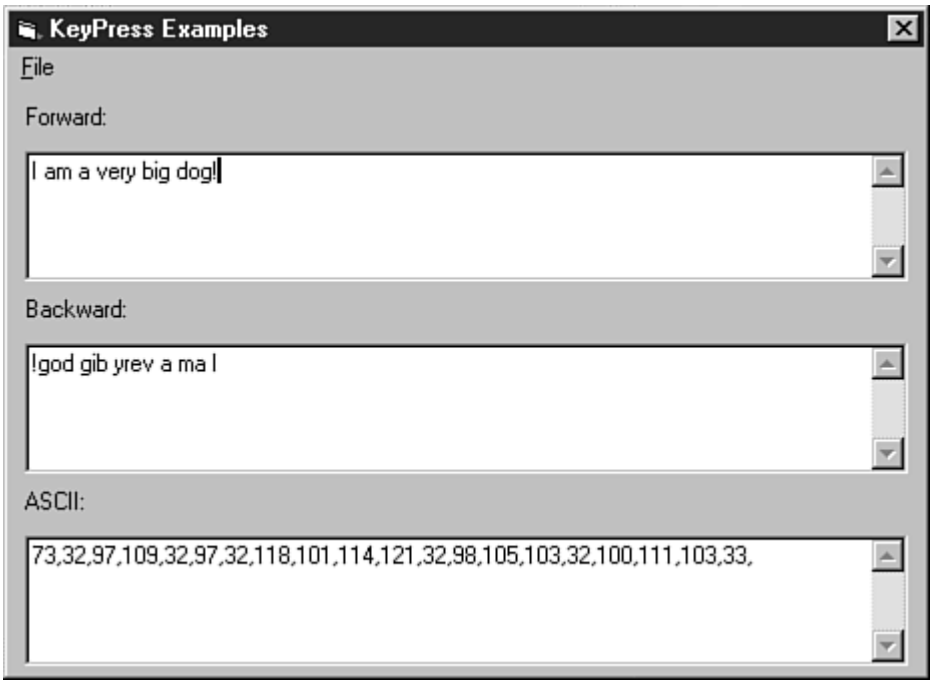
Thuộc tính KeyPreview

Đôi khi ta muốn tất cả các điều khiển trên Form nhận được sự kiện KeyPress chứ không phải chỉ có điều khiển đang nhận con trỏ (Focus), ta sẽ phải sử dụng thuộc tính KeyPreview.

Khi chúng ta thiết kế một Form, giá trị mặc định của thuộc tính này sẽ là False, khi đó bất kỳ một sự kiện bàn phím nào cũng đều được gởi đến điều khiển đang giữ quyền điều khiển. Tuy nhiên nếu giá trị của thuộc tính là True thì Form sẽ là nơi nhận mọi sự kiện bàn phím.

Sau đây là ví dụ về điều này:

```
Private   Sub   Form_KeyPress (KeyAscii   As   Integer)
       ' Gởi điều khiển đến textbox đầu tiên
       txtForward.SetFocus
       txtBackward.Text = Chr(KeyAscii) & txtBackward.Text
       txtAscii.Text = txtAscii.Text & CStr(KeyAscii) & ","
End Sub
```



Hình VI.15 Các sự kiện bàn phím

Trong ví dụ trên, nếu như giá trị của thuộc tính KeyPreview là False thì các TextBox Backward và Ascii không thể nhận được giá trị.

IV. Xử lý đồ họa và giao diện

IV.1 Hiển thị hình ảnh

Bởi vì hệ điều hành Windows là hệ điều hành sử dụng giao diện đồ họa, do đó một phần quan trọng trong ứng dụng của ta là cách thức để hiển thị hình ảnh trong ứng dụng của mình. VB cung cấp cho chúng ta 4 loại điều khiển để hiển thị và quản lý hình ảnh: Picture Box, Image, Shape và Line.

IV.1.1 Sử dụng Picture Box

Cách dùng chính của điều khiển Picture Box là hiển thị hình ảnh. Hình ảnh mặc định mà Picture Box hiển thị có tên được xác định bởi thuộc tính Picture (có thể bao gồm cả đường dẫn).

Ta cũng cần chú ý một điều đó là đối tượng Form cũng có thể hiển thị một hình ảnh xem như là ảnh nền thông qua thuộc tính Picture.

Thuộc tính AutoSize của điều khiển Picture Box quy định kích thước của điều khiển có thể thay đổi một cách tự động hay không? Nếu giá trị của thuộc tính này là True, thì kích thước của điều khiển sẽ thay đổi theo kích thước của hình ảnh mà nó chứa. Tuy nhiên sự thay đổi này có thể làm ứng dụng của chúng ta trở nên xấu đi do sự thay đổi kích thước của điều khiển Picture Box sẽ không quan tâm đến các vị trí của các điều khiển khác cùng có trên biểu mẫu. Tốt hơn hết là chúng ta nên thử qua tất cả các hình ảnh có thể hiển thị tại thời điểm thiết kế để quy định kích thước của điều khiển cho hợp lý.

Hơn thế nữa, có thể thay đổi hình ảnh hiển thị bên trong Picture Box bằng cách sử dụng phương thức LoadPicture để thay đổi giá trị của thuộc tính Picture.

Ngoài ra ta có thể dùng Picture Box như một vật chứa các điều khiển khác. Cũng như điều khiển Frame, ta có thể đặt các điều khiển khác bên trong Picture Box. Ta thường sử dụng Picture box chứa các điều khiển Label để hiển thị các thông tin và trạng thái của ứng dụng.

Một cách dùng khác của Picture box đó là xem như một khung vẽ trắng và ta dùng các phương thức Circle, Line, PSet hay Point để vẽ lên trên điều khiển này.

IV.1.2 Sử dụng Image Control

Image control cũng như điều khiển Picture Box nhưng chỉ dùng để hiển thị hình ảnh. Nó không thể dùng làm vật chứa và cũng không có một số thuộc tính như điều khiển Picture Box.

Các phương thức dùng để hiển thị, thay đổi hình ảnh cũng như điều khiển Picture Box, tuy nhiên thuộc tính quy định việc kích thước thay đổi một cách tự động là thuộc tính Stretch.

Một trong những ứng dụng chủ yếu của điều khiển Image Control đó là sử dụng như một nút lệnh, đây là một cách thức tiện lợi để thiết kế nút lệnh chứa hình ảnh thay vì là các câu văn bản.

Khi sử dụng Image Control như một nút lệnh, ta nên nhớ rằng điều khiển này sẽ không thể có trạng thái *ấn xuống* khi được Click, vì thế ta nên thay đổi hình ảnh hiển thị bởi Image Control để cho biết rằng nút lệnh đã được ấn.

IV.2 Xử lý đồ họa

IV.2.1 Tọa độ màn hình

Góc trái trên của màn hình có tọa độ là (0,0) có nghĩa là X = 0 và Y = 0. Như vậy tức là khi di chuyển sang phải màn hình thì X tăng lên cũng như di chuyển xuống dưới thì Y tăng lên.

Tuy nhiên VB chỉ cho phép ta vẽ trên biểu mẫu hay hộp hình (picture box). Khi đó hệ tọa độ sẽ được gắn với từng điều khiển.

Ta thường sử dụng 2 hệ tọa độ chủ yếu sau: Twips và Pixel.

Twips: Đây là hệ tọa độ mặc định dùng cho biểu mẫu. Mỗi điểm sẽ bằng 1/567 cm. Đây là hệ tọa độ không bị ảnh hưởng bởi thiết bị, kết quả vẽ sẽ như nhau trên màn hình VGA chuẩn, trên máy in hay trên màn hình có độ phân giải cao khác.

Pixel: Đây là hệ tọa độ phổ biến nhất, mỗi một điểm trên màn hình sẽ bằng chính xác với một Pixel, như vậy khi sử dụng hệ tọa độ này sẽ giúp cho các ứng dụng đồ họa thực hiện được nhanh hơn vì không phải thông qua quá trình đổi hệ tọa độ.

IV.2.2 Các phương thức đồ họa

Các điều khiển được vẽ lên biểu mẫu lúc thiết kế nhưng các phương thức đồ họa cho phép vẽ trực tiếp khi ứng dụng thi hành.

Phương thức PaintPicture

Phương thức PaintPicture cho phép sao chép nhanh các hình ảnh từ biểu mẫu, hộp hình và máy in.

Cú pháp:

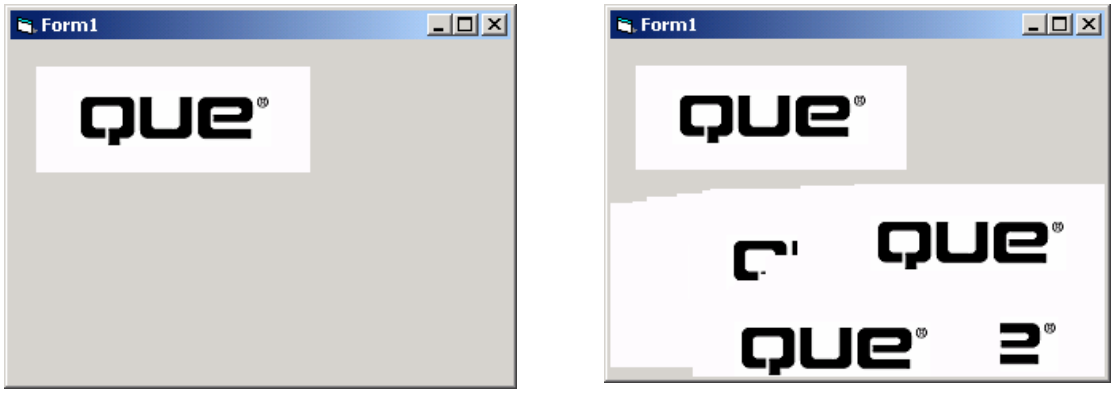
object.PaintPicture picture, x1, y1, width1, height1, x2, y2, width2, height2, opcode

Object	Object là đối tượng mà phương thức sẽ làm việc, nó có thể là biểu mẫu, hộp hình hay đối tượng máy in.
Picture	Hình ảnh nguồn sẽ được vẽ lên đối tượng phải được chỉ rõ bởi thuộc tính Picture của biểu mẫu hoặc hộp hình.
x1, y1	Giá trị chỉ định vị trí của hình ảnh trên đối tượng. Thuộc tính ScaleMode xác định hệ tọa độ nào được sử dụng.
Width1	Giá trị xác định độ rộng của hình ảnh, nếu bỏ qua thì mặc định là độ rộng của ảnh nguồn.
Height1	Giá trị xác định độ cao của hình ảnh, nếu bỏ qua thì mặc định là độ cao của ảnh nguồn.
x2, y2	Các giá trị xác định hình ảnh sẽ được vẽ lại từ vị trí nào. Nếu bỏ qua thì giá trị mặc định là 0, tức toàn bộ hình ảnh được vẽ lại.
Width2	Tương tự như Width1, nhưng ở đây là tác động đến ảnh nguồn.
Height2	Tương tự như Height1, nhưng ở đây là tác động đến ảnh nguồn.
Opcode	Đây là tùy chọn và chỉ có tác dụng với ảnh Bitmap.

Ví dụ: Thiết kế chương trình sao cho khi người sử dụng vừa di chuyển vừa nắm giữ phím chuột thì một hình ảnh sẽ được vẽ lại ở tọa độ mới.

```
Dim re

Private Sub Form_Load()
      re = False
End Sub

Private Sub Form_MouseDown(Button As Integer, & _
      Shift As Integer, X As Single, Y As Single)
      re = True
End Sub

Private Sub Form_MouseMove(Button As Integer, & _
      Shift As Integer, X As Single, Y As Single)


      If re Then
            Form1.PaintPicture Image1.Picture, X, Y, & _
                  Image1.Width, Image1.Height
      End If
End Sub

Private Sub Form_MouseUp(Button As Integer, & _
      Shift As Integer, X As Single, Y As Single)
      re = False
End Sub
```

Qua ví dụ trên ta thấy phương thức PaintPicture cho phép sao chép nhanh một ảnh nguồn trên các đối tượng khác .



Hình VI.16 Phương thức PaintPicture

Phương thức PSet

Phương thức này thao tác trên từng điểm và có công dụng gán một màu nào đó cho một điểm trên đối tượng.

Cú pháp :

Object.PSet [Step] (x, y), [color]

Object	Đối tượng mà phương thức làm việc.
Step	Tùy chọn. Xác định mối quan hệ với tọa độ X và Y hiện tại.
(x, y)	Tọa độ của điểm.
Color	Màu của điểm đó.

Điều khiển hình dáng

Đây là điều khiển cho phép vẽ các hình đơn giản lên một biểu mẫu trong khi thiết kế. Đây là một điều khiển rất đơn giản, ta chỉ quan tâm đến các thuộc tính sau:

- *Shape*: Quy định hình vẽ là hình oval, chữ nhật …
- *BorderStyle*: Quy định kiểu đường vẽ.
- *BackStyle*: Cho biết dạng tô màu đặc hay không.
- *BorderWidth*: Đây là độ rộng của đường vẽ.

Vẽ đường tròn, cung tròn và Ellipse

VB cung cấp phương thức Circle cho phép ta vẽ đường tròn, đường cong, cung tròn, ellipse …

Để vẽ một đường tròn ta dùng phương thức Circle do VB cung cấp.

Cú pháp:

Object.Circle (X, Y), Radius, [color]

Object	Đối tượng mà phương thức làm việc.
(x, y)	Tọa độ tâm đường tròn.
Radius	Bán kính của đường tròn
Color	Màu đặt cho đường tròn. .

Để vẽ một cung tròn, ta cũng sử dụng phương thức Circle, tuy nhiên ta cần cung cấp thêm 2 thông số đó là điểm bắt đầu và điểm kết thúc của cung tròn. Thông thường chúng ta quen sử dụng đơn vị đo góc là độ, tuy nhiên đối với VB ta cần phải đưa vào đơn vị là Radians.

Ví dụ vẽ một cung tròn tâm (1000, 1500), bán kính 500 bắt đầu từ góc 60^o đến góc 90^o ta dùng đoạn lệnh như sau:

```
Const pi = 3.1415
Circle (1000, 15000), 500,  , pi/3, pi/2
```

Để vẽ một Ellipse, ta cung cấp thêm thông số cuối cùng (thông số Aspect) đó là sự co giãn của đường tròn theo chiều ngang.

Cú pháp tổng quát của phương thức Circle:

object.**Circle** [**Step**] **(***x, y***),** *radius,* [*color, start, end, aspect*]

Chương 7 : TẬP TIN

Mục tiêu:

Chương này giới thiệu về cách thức truy cập hệ thống tập tin của Windows trong VB, thao tác thường gặp trong các ứng dụng chạy trên Windows.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Sử dụng mô hình đối tượng hệ thống tập tin để thao tác với ổ đĩa, thư mục, tập tin trong Windows.

- Sử dụng các hàm xuất/nhập tập tin để truy xuất tập tin văn bản, tập tin nhị phân, tập tin truy xuất ngẫu nhiên.

Kiến thức có liên quan:

- Các cấu trúc lập trình trong VB.

- Cách thức tổ chức hệ thống tập tin của hệ điều hành Windows (9X, 2000, XP…)

Tài liệu tham khảo:

- **Microsoft Visual Basic 6.0 và Lập trình Cơ sở dữ liệu** - Chương 6, trang 88 - **Nguyễn Thị Ngọc Mai** (chủ biên), **Nhà xuất bản Giáo dục** - **2000**.

I. Mô hình File System Object (FSO)

Cung cấp cho ứng dụng của ta các khả năng như tạo mới, thay đổi, xóa, di chuyển các thư mục, hoặc kiểm tra xem một thư mục nào đó có tồn tại hay không… Ngoài ra chúng ta cũng có thể lấy được các thông tin liên quan đến thư mục như tên, ngày tạo, ngày sửa đổi gần nhất…

Mô hình FSO chỉ hỗ trợ truy xuất trực tiếp tập tin dạng văn bản thông qua đối tượng TextStream, nó chưa hỗ trợ cho tập tin nhị phân, do đó với tập tin nhị phân ta phải dùng lệnh Open với cờ Binary.

Đối tượng	Giải thích
Drive	Cho phép thu thập thông tin về ổ đĩa, bao gồm cả các ổ đĩa chia sẻ qua mạng LAN, CD-ROM…
Folder	Cho phép tạo, xóa, di chuyển và thu nhận các thông tin hệ thống trên thư mục.
File	Đối tượng cho phép thao tác trên tập tin.
FileSystemObject	Các thuộc tính và phương thức cho phép thao tác trên tập tin, thư mục và ổ đĩa.
TextStream	Cho phép đọc và ghi tập tin dạng văn bản (dạng Text).

Nếu chưa có tham chiếu đến đối tượng FSO, ta cần chọn "Microsoft Scripting Runtime" từ menu *Project/References*... Các phương thức của FSO ta có thể xem trong cửa sổ Object Browser.

I.1 Tạo đối tượng FileSystemObject

Có hai cách, khai báo một biến kiểu FileSystemObject hoặc dùng phương thức CreateObject của lập trình hướng đối tượng.

Cách 1:

```
Dim fso As New FileSystemObject
```

Cách 2:

```
Set fso = CreateObject("Scripting. FileSystemObject")
```

Trong đó Scripting là tên thư viện và FileSystemObject là tên đối tượng.

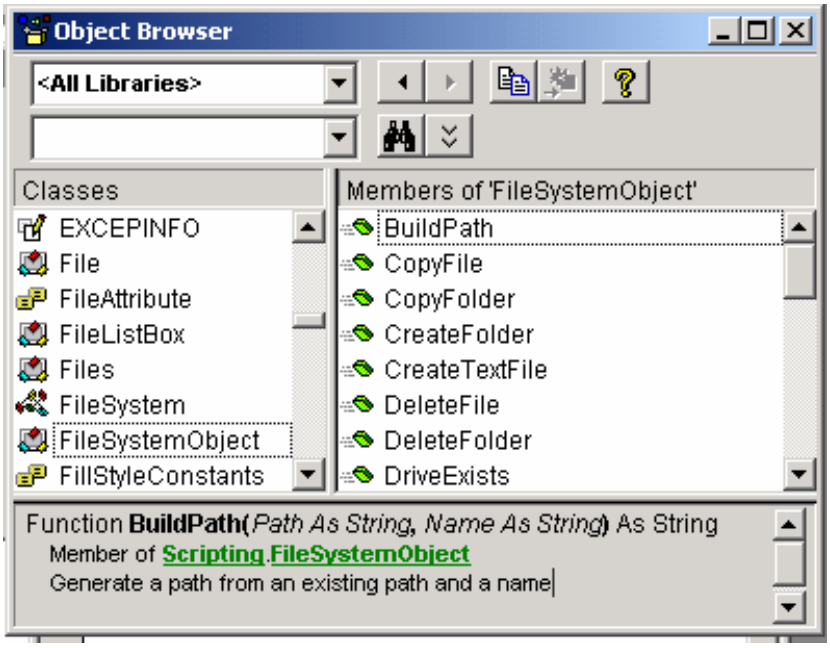
I.2 Truy cập ổ đĩa, thư mục, tập tin

Dùng các phương thức GetDrive, GetFolder, GetFile. Ví dụ để tạo một handle trỏ đến tập tin "d:\tqdinh\text.txt" ta dùng các dòng lệnh sau:

```
Dim fso As New FileSystemObject, f As File
Set f = fso.GetFile("d:\tqdinh\text.txt")
```

Hoặc ta có thể tạo mới thư mục, tập tin thông qua các phương thức CreateFolder, CreateTextFile. Ngoài ra ta có thể xóa một thư mục hoặc một tập tin thông qua DeleteFolder, DeleteFile.

Đối tượng FileSystemObject còn có rất nhiều phương thức, ta có thể xem qua cửa sổ ObjectBrowser.



Hình VII.1 Cửa sổ ObjectBrowser với đối tượng FileSystemObject

I.3 Thông tin về ổ đĩa

Các thông tin này được truy xuất thông qua các thuộc tính của đối tượng File.

- TotalSize: tổng dung lượng của ổ đĩa tính bằng Byte.
- AvailableSpace, FreeSpace: dung lượng còn trống của đĩa.
- DriveLetter: ký tự ổ đĩa.
- DriveType: loại ổ đĩa (ổ tháo lắp hay cố định, ổ mạng, CD-ROM, ổ RAM).
- FileSystem: ổ đĩa được quản lý bởi bản FAT nào: FAT16, FAT32, NTFS...

I.4 Làm việc với thư mục

Đây là các phương thức có cách sử dụng rất đơn giản, vì thế ta chỉ xét qua phương thức nào ứng với tác vụ gì (công việc gì) chứ không đi sâu phân tích từng phương thức.

Tác vụ	Phương thức
Tạo thư mục	FileSystemObject.CreateFolder
Xóa thư mục	FileSystemObject.DeleteFolder hay Folder.Delete
Di chuyển thư mục	FileSystemObject.MoveFolder hay Folder.Move
Sao chép thư mục	FileSystemObject.CopyFolder hay Folder.Copy
Lấy tên thư mục	Folder.Name
Kiểm tra thư mục có tồn tại trên ổ	FileSystemObject.FolderExists

đĩa hay không	
Trả về đối tượng Folder	FileSystemObject.GetFolder
Lấy tên của thư mục cha	FileSystemObject.GetParentFolderName
Lấy tên của thư mục hệ thống	FileSystemObject.GetSpecialFolder

I.5 Làm việc với tập tin

Mở tập tin để ghi dữ liệu

- Tạo tập tin mới: sử dụng phương thức CreateTextFile.

```
Dim fso As New FileSystemObject
fso.CreateTextFile("d:\home\lhbao\test.txt")
```

- Mở tập tin để ghi với cờ ForWriting, lúc này ta sử dụng phương thức OpenAsTextStream của đối tượng File cùng với đối tượng TextStream để thao tác.

Ví dụ:

```
Dim fso As New FileSystemObject, f As File
Dim ts  As  TextStream
fso.CreateTextfile("d:\home\lhbao\test.txt")
Set f = fso.GetFile("d:\home\lhbao\test.txt")
Set ts = f.OpenAsTextStream(ForWriting)
```

- Ghi dữ liệu lên tập tin: ta có thể ghi dữ liệu vào tập tin đang mở bằng phương thức Write hay WriteLine của đối tượng TextStream. Sự khác biệt giữa hai phương thức này là sẽ có thêm ký tự xuống dòng nếu như sử dụng WriteLine. Nếu muốn ghi một dòng trắng vào tập tin đang mở, ta sử dụng phương thức WriteBlankLines.

Cú pháp: ts là đối tượng TextStream

```
ts.Write(s) ' Ghi chuỗi s lên tập tin
ts.WriteLine(s) ' Ghi chuỗi s lên tập tin
ts.WriteBlankLines(N) ' Ghi N dòng trắng lên tập tin
```

Mở tập tin để đọc dữ liệu

- Mở tập tin để đọc với cờ ForReading, lúc này ta sử dụng phương thức OpenAsTextStream của đối tượng File cùng với đối tượng TextStream để thao tác.

Ví dụ:

```
Dim fso As New FileSystemObject, f As File
Dim ts As TextStream
Set f = fso.GetFile("D:\Home\lhbao\Test.txt")
Set ts = f.OpenAsTextStream(ForReading)
```

- Đọc dữ liệu từ tập tin: Ta có ba phương thức để đọc dữ liệu từ một tập tin văn bản, đó là Read, ReadLine và ReadAll. Ba phương thức này cho phép đọc một số ký tự, một dòng của văn bản và toàn bộ văn bản.

Trong khi đọc nội dung của tập tin, ta có thể sử dụng phương thức Skip, SkipLine để nhảy đến phần tử dữ liệu mới.

Cú pháp: ts là đối tượng TextStream

```
ts.Read(N) As String: Đọc N ký tự từ tập tin
ts.ReadLine As String
ts.ReadAll As String
```

Đóng tập tin: Sử dụng phương thức Close của đối tượng TextStream.

Di chuyển, sao chép và xóa tập tin

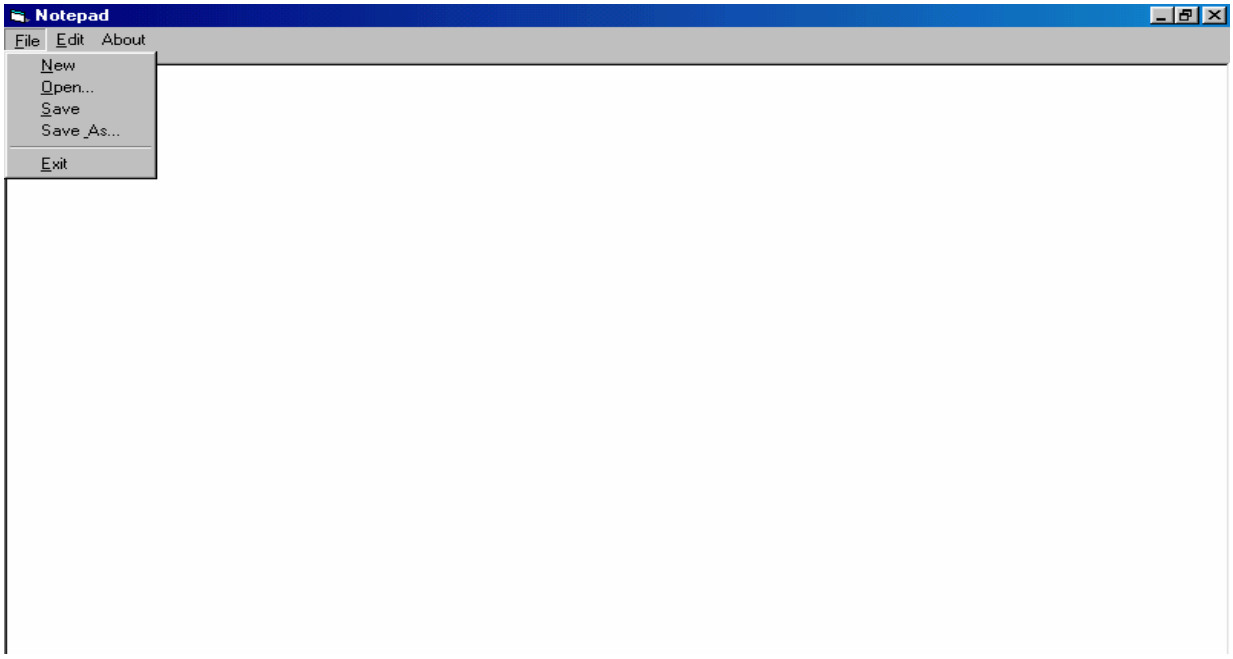
Sự khác biệt giữa di chuyển và sao chép một tập tin đó là tập tin nguồn có còn tồn tại ở thư mục nguồn hay không. Ứng với một thao tác, ta cũng có hai phương thức để thực hiện, đó là các phương thức thuộc đối tượng FileSystemObject và đối tượng File.

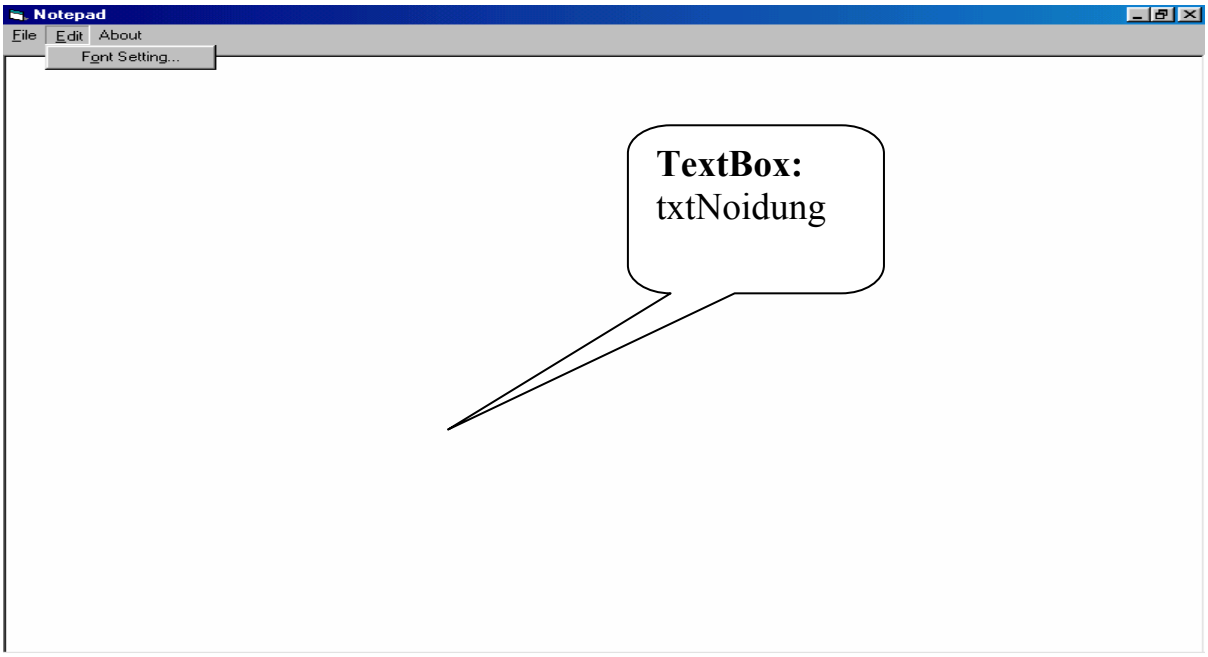
Tác vụ	Phương thức
Di chuyển một tập tin	FileSystemObject.MoveFile hoặc File.Move
Sao chép một tập tin	FileSystemObject.CopyFile hoặc File.Copy
Xóa một tập tin	FileSystemObject.DeleteFile hoặc File.Delete

Các phương thức ứng với thao tác di chuyển và sao chép tập tin cần có đối số là đường dẫn đến nơi chứa tập tin đích.

I.6 Ví dụ

Thiết kế một ứng dụng như Notepad của Windows, sử dụng FSO để truy xuất tập tin.





- Thêm vào điều khiển CommonDialog vào ứng dụng với Name: dlgFile.
- Sự kiện mnuNew_Click được xử lý:

```
Private Sub mnuNew_Click()
    txtNoiDung.Text = ""
End Sub
```

- Đoạn mã cho thủ tục xử lý sự kiện mnuOpen_Click:

```
Private Sub mnuOpen_Click()
    On Error GoTo Xuly
    dlgFile.Filter = "All Files (*.*)|*.*|" & _
                     "Text Files (*.txt)|*.txt"
    dlgFile.FilterIndex = 2
    dlgFile.ShowOpen
    Dim fso As New FileSystemObject, f As File
    Dim ts As TextStream
    Set f = fso.GetFile(dlgFile.FileName)
    Set ts = f.OpenAsTextStream(ForReading)

    txtNoiDung.Text = ts.ReadAll()
    ts.Close
Xuly:
End Sub
```

- Đối với mnuSave_Click:

```
Private Sub mnuSave_Click()
    On Error GoTo Xuly
    dlgFile.Filter = "All Files (*.*)|*.*|" & _
                     "Text Files (*.txt)|*.txt"
    dlgFile.FilterIndex = 2
    dlgFile.ShowSave

    Dim fso As New FileSystemObject, f As File
    Dim ts As TextStream
    fso.CreateTextFile (dlgFile.FileName)
```

```
            Set f = fso.GetFile(dlgFile.FileName)
            Set ts = f.OpenAsTextStream(ForWriting)
            ts.Write (txtNoiDung.Text)
            ts.Close
        Xuly:
        End Sub
```

- Sự kiện mnuSaveAs_Click cũng được xử lý tương tự.
- Sự kiện mnuFont:

```
Private Sub mnuFont_Click()
    On Error GoTo Xuly
    With dlgFile
        .Flags = cdlCFBoth + 256
        .ShowFont
        txtNoiDung.Font.Bold = .FontBold
        txtNoiDung.Font.Italic = .FontItalic
        txtNoiDung.Font.Name = .FontName
        txtNoiDung.Font.Size = .FontSize
    End With

Xuly:
End Sub
```

- Sự kiện mnuExit_Click:

```
Private Sub mnuExit_Click()
     End
End Sub
```

- Lưu dự án và chạy chương trình.

II. Hàm I/O và lệnh xử lý tập tin

Có ba kiểu tập tin: Tuần tự, ngẫu nhiên và nhị phân

- Tuần tự (Sequential): Đây là cách thức truy cập đến tập tin cho kiểu đọc và ghi thành theo các khối liên tục nhau.

- Ngẫu nhiên (Random): đọc và ghi các tập tin văn bản hoặc nhị phân có cấu trúc theo các mẩu tin có độ dài cố định.

- Nhị phân (Binary): đọc và ghi các tập tin có cấu trúc thay đổi.

Các hàm và dòng lệnh thao tác trên các kiểu truy cập tập tin, công dụng của từng hàm sẽ được xét đến trong phần sau:

Statements & Functions	**Sequential**	**Random**	**Binary**
Close	X	X	X
Get		X	X
Input( )	X		X
Input #	X		
Line Input #	X		

Open	X	X	X
Print #	X		
Put		X	X
Write #	X		

II.1. Tập tin tuần tự

II.1.1 Mở tập tin

Cú pháp:

```
Open pathname For [Input|Output|Append] _
     As  filenumber [Len = buffersize]
```

Nếu ta dùng tham số Input thì tập tin (có đường dẫn là pathname) đó phải tồn tại rồi, nếu không sẽ gây ra lỗi và tham số này được dùng trong trường hợp mở tập tin để đọc. Còn các tham số Output hoặc Append sẽ tạo mới tập tin và sau đó mở nó.

Tham số **Len** sẽ chỉ ra số ký tự trong vùng đệm khi sao chép dữ liệu giữa tập tin và chương trình xử lý.

`Filenumber` là số hiệu của tập tin được mở, nó mang giá trị kiểu nguyên và nằm trong khoảng từ 1 đến 511. Để lấy số hiệu tập tin mới hợp lệ, ta sử dùng hàm `FreeFile`.

II.1.2 Đọc nội dung tập tin

- **Hàm Input**

Cú pháp: `Input (number, filenumber) As String`: hàm này trả về `number` ký tự của tập tin có số hiệu được chỉ định bởi `filenumber`.

- **Lệnh Input #**

Cú pháp: `Input # filenumber, varlist`: lệnh này sẽ đọc nội dung của tập tin vào các biến được chỉ bởi `varlist`. Lưu ý rằng lệnh này chỉ sử dụng với các tập tin được ghi bởi lệnh `Write #`.

- **Lệnh Line Input #**

Cú pháp: `Line Input # filenumber, varname`: lệnh này sẽ đọc nội dung của một dòng trong tập tin tuần tự vào biến chuỗi.

II.1.3 Ghi dữ liệu lên tập tin

Ta có 2 câu lệnh để ghi dữ liệu lên tập tin là Write # và Print #.

- **Câu lệnh Write #**:

Cú pháp: `Write # filenumber, [outputlist]`: lệnh này ghi dữ liệu vào tập tin tuần tự, nội dung ghi mới chứa trong danh sách biến `outputlist`, các phần tử của danh sách cách nhau bởi dấu phẩy. Nếu ta không đặc tả `outputlist` thì một dòng trắng sẽ được ghi vào tập tin.

- **Câu lệnh Print#**:

Cú pháp: `Print # filenumber, [outputlist]`: tương tự như Write# nhưng dữ liệu có thể định dạng khi ghi lên tập tin.

Đối số outputlist có thể là:

[{**Spc(***n***)** | **Tab**[**(***n***)**]}] [*expression*] [*charpos*]

Trong đó:

- **Spc(n)**: dùng để xen khoảng trắng vào tập tin với n là số khoảng trắng.

- **Tab(n)**: dùng để xác định cột bắt đầu ghi dữ liệu trong vùng ghi với n là số thứ tự cột. Dùng Tab không đối số để chỉ vị trí bắt đầu của vùng ghi tiếp theo.

- **Expression**: biểu thức chuỗi hoặc biểu thức số.

- **Charpos**: chỉ định vị trí của ký tự kế tiếp. Trong đó: dấu **;** xác định dữ liệu mới sẽ ghi kế tiếp ký tự cuối cùng trước đó. **Tab(n)** xác định chính xác cột để ghi dữ liệu hay **Tab** chỉ ra vị trí của dữ liệu cần ghi vào là vùng ghi kế tiếp. Nếu tham số này bị bỏ qua, dữ liệu sẽ được ghi bắt đầu từ dòng kế tiếp.

II.1.4 Đóng tập tin

Cú pháp: `Close filenumberlist`: đóng lại các tập tin được mở với các số hiệu được mô tả trong `filenumberlist`, `filenumberlist` có dạng sau:

[[**#**]*filenumber*] [**,** [**#**]*filenumber*] **. . .**

II.1.5 Ví dụ

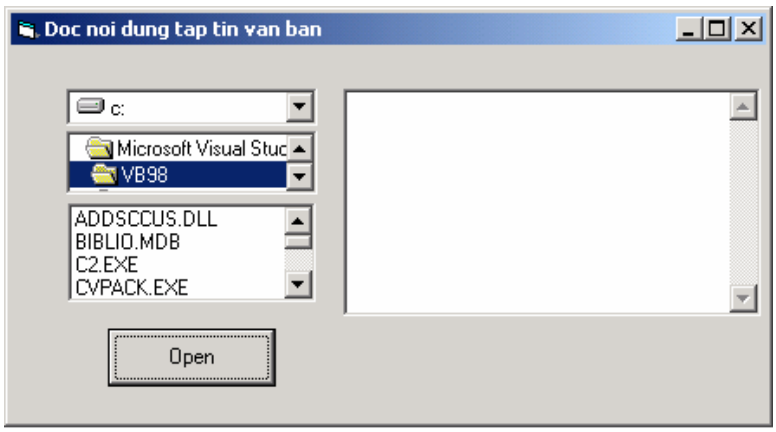
```
Open "TESTFILE" For Output As #1   ' Mở tập tin để ghi.
Print #1, "This is a test"   ' Ghi chuỗi vào tập tin.
Print #1,   ' Ghi một dòng trắng vào tập tin.
Print #1, "Zone 1"; Tab ; "Zone 2"   ' Tạo hai vùng ghi.

' Phân cách hai chuỗi bởi khoảng trống.
Print #1, "Hello" ; " " ; "World"
Print #1, Spc(5) ; "5 leading spaces "  ' Tạo 5 khoảng trống
Print #1, Tab(10) ; "Hello"   ' ghi chuỗi tại cột thứ 10.

' Gán giá trị  thuộc kiểu Boolean, Date, Null và Error.
Dim MyBool, MyDate, MyNull, MyError
MyBool = False : MyDate = #February 12, 1969# : MyNull = Null
MyError = CVErr(32767)
Print #1, MyBool ; " is a Boolean value"
Print #1, MyDate ; " is a date"
Print #1, MyNull ; " is a null value"
Print #1, MyError ; " is an error value"
Close #1   ' Đóng tập tin
```

Và sau đây là nội dung của tập tin TESTFILE sau khi thực thi chương trình:

```
This is a test

Zone 1       Zone 2
Hello World
    5 leading spaces
       Hello
False is a Boolean value
2/12/1969  is a date
Null is a null value
Error 32767 is an error value
```

Ứng dụng sau đây cho phép đọc nội dung 1 tập tin được lựa chọn.

- Thiết kế chương trình có giao diện:

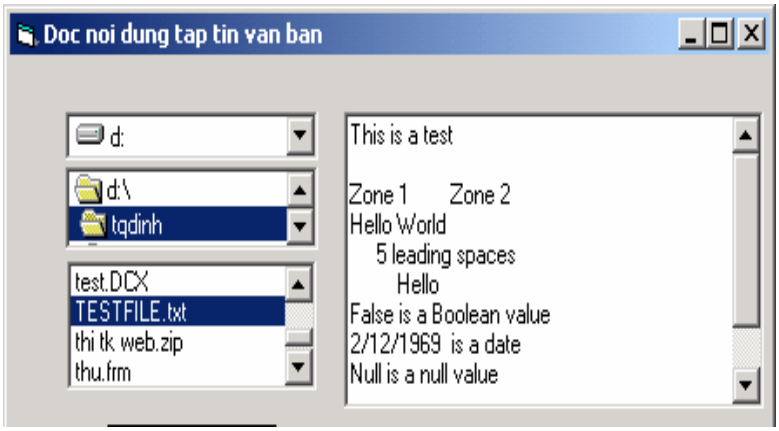


Ở đây ta có sử dụng điều khiển danh sách ổ đĩa, thư mục, tập tin để lựa chọn tập tin cùng với 1 TextBox để hiển thị nội dung.

- Các sự kiện được xử lý như sau:

```
Private Sub Drive1_Change()
      Dir1.Path = Drive1.Drive
End Sub

Private Sub Dir1_Change()
      File1.Path = Dir1.Path
End Sub

Private Sub Command1_Click()
      Dim fname As String, str As String
      fname = Dir1.Path + "\" + File1.filename
      Open fname For Input As #1
      Do While Not EOF(1)
            Line Input #1, str
            Text1.Text = Text1.Text + str + vbCrLf
      Loop
      Close #1
End Sub
```

- Lưu dự án và chạy chương trình, ta được kết quả:



II.2. Tập tin truy xuất ngẫu nhiên

II.2.1 Mở tập tin

Cú pháp

```
Open pathname [For Random] As filenumber _
     [Len = RecLength]
```

Bởi vì tham số Random là mặc định, do đó từ khóa `For Random` là không cần thiết.

Tham số Len cần một kích thước đúng của mẩu tin, nếu nhỏ hơn thì sẽ gây ra lỗi, lớn hơn sẽ gây lãng phí không gian đĩa.

`Filenumber`: số hiệu tập tin khi mở (có đường dẫn là `pathname`)

II.2.2 Đọc nội dung tập tin

- **Câu lệnh Get**:

Cú pháp: **`Get`** `[#] filenumber, [recnumber], varname`: lệnh này sẽ đọc nội dung của mẩu tin thứ `recnumber` trong tập tin vào biến mẩu tin `varname`, trong đó mẩu tin đầu tiên có số thứ tự là 1.

II.2.3 Ghi dữ liệu lên tập tin

- **Câu lệnh Put**:

Cú pháp: **`Put`** `[#] filenumber, [recnumber], varname`: Lệnh này sẽ thay thế hoặc thêm mới mẩu tin vào tập tin. Nếu như số thứ tự mẩu tin (`recnumber`) chỉ đến mẩu tin đã tồn tại, thì nội dung của mẩu tin đó được thay thế. Để thêm mới mẩu tin, số thứ tự mẩu tin cần có giá trị là số mẩu tin hiện có của tập tin cộng thêm 1.

II.2.4 Đóng tập tin

Cú pháp: `Close filenumberlist`: đóng lại các tập tin được mở với các số hiệu được mô tả trong `filenumberlist`, `filenumberlist` có dạng sau:

```
[[#]filenumber] [, [#]filenumber]...
```

II.2.5 Ví dụ

Giá sử ta có khai báo 1 mẩu tin như sau:

```
Type Hanghoa
```

```
    MaHang As String * 10
    TenHang As String * 40
    DVTinh As String * 15
    Gia As Double
End Type
```

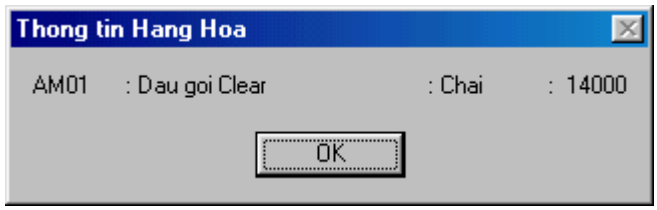
- Ghi dữ liệu lên tập tin có cấu trúc là các mẩu tin như trên

```
Dim h As Hanghoa
h.MaHang = "AM01"
h.TenHang = "Dau goi Clear"
h.DVTinh = "Chai"
h.Gia = 14000#

Open "E:\Test.dat" For Random As #1 Len = LenB(h)
Put #1, 1, h
Close #1
```

- Đọc dữ liệu từ tập tin có cấu trúc như trên:

```
Dim h As Hanghoa
Open "E:\Test.dat" For Random As #1 Len = LenB(h)
Get #1, 1, h
Close #1
MsgBox h.MaHang & ": " & h.TenHang & ": " & _
       h.DVTinh & ": " & Str(h.Gia), , _
       "Thong tin Hang Hoa"
```

Kết quả khi đọc dữ liệu như sau:



Chương 8: CÁC KHÁI NIỆM CƠ BẢN VỀ CƠ SỞ DỮ LIỆU

Mục tiêu:

Chương này giới thiệu về một số khái niệm trong lập trình cơ sở dữ liệu với VB, những vấn đề cần thiết khi thiết kế các ứng dụng truy cập cơ sở dữ liệu.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Một số khái niệm khi lập trình cơ sở dữ liệu trong VB.
- Phân biệt DAO, RDO, ADO.
- Sử dụng môi trường phát triển VB để tương tác với cơ sở dữ liệu.

Kiến thức có liên quan:

- Câu lệnh SQL để truy vấn dữ liệu.

Tài liệu tham khảo:

- **Microsoft Visual Basic 6.0 và Lập trình Cơ sở dữ liệu** - Chương 18, trang 447 - **Nguyễn Thị Ngọc Mai** (chủ biên), **Nhà xuất bản Giáo dục – 2000**.

I. Cơ sở dữ liệu

I.1. Khái niệm

Cơ sở dữ liệu là một kho chứa thông tin. Có nhiều loại cơ sở dữ liệu, nhưng trong khuôn khổ bài giảng này ta chỉ quan tâm đến các ứng dụng lập trình liên quan đến cơ sở dữ liệu quan hệ.

Một cơ sở dữ liệu quan hệ:

- Chứa dữ liệu trong các bảng, được cấu tạo bởi các dòng còn gọi là các mẩu tin, và cột còn gọi là các trường.
- Cho phép lấy về (hay truy vấn) các tập hợp dữ liệu con từ các bảng.
- Cho phép nối các bảng với nhau cho mục đích truy cập các mẩu tin liên quan với nhau chứa trong các bảng khác nhau.

I.2. Bộ máy (Engine) cơ sở dữ liệu

Chức năng cơ bản của một cơ sở dữ liệu được cung cấp bởi một bộ máy cơ sở dữ liệu, là hệ thống chương trình quản lý cách thức chứa và trả về dữ liệu.

Chẳng hạn **Microsoft Jet** là bộ máy cơ sở dữ liệu được sử dụng khi truy cập dữ liệu Access.

I.3. Bảng (Table) và trường (Field)

Các cơ sở dữ liệu được cấu thành từ các bảng dùng thể hiện các phân nhóm dữ liệu. Chẳng hạn, nếu ta tạo một cơ sở dữ liệu để quản lý các tài khoản trong công việc kinh doanh, ta phải tạo một bảng cho Khách hàng, một bảng cho Hóa đơn và một bảng cho Nhân viên. Bảng có cấu trúc định nghĩa sẵn và chứa dữ liệu phù hợp với cấu trúc này.

- Bảng: chứa các mẩu tin là các mẩu dữ liệu riêng rẽ bên trong phân nhóm dữ liệu.
- Mẩu tin: chứa các trường. Mỗi trường thể hiện một bộ phận dữ liệu trong một mẩu tin. Ví dụ như mỗi mẩu tin thể hiện một mục trong danh bạ địa chỉ chứa các trường tên và họ, địa chỉ, thành phố, số điện thoại…

Ta có thể dùng chương trình Visual Basic để tham chiếu và thao tác với cơ sở dữ liệu, bảng, mẩu tin và các trường.

I.4. Tập mẩu tin (Recordset)

Recordset là một cấu trúc dữ liệu thể hiện một tập hợp con các mẩu tin lấy về từ cơ sở dữ liệu. Về khái niệm, nó tương tự như một bảng nhưng có thêm một vài thuộc tính riêng biệt quan trọng.

Các Recordset được thể hiện như các đối tượng. Cũng như các đối tượng khác trong Visual Basic, các đối tượng recordset có các thuộc tính và phương thức riêng.

II. Truy xuất cơ sở dữ liệu trong Visual Basic 6.0

Visual Basic cung cấp kèm theo nó một bộ máy cơ sở dữ liệu có thể hiểu được dữ liệu của Microsoft Access gọi là *Joint Engine Technology (JET)*. JET là một bộ

máy truy cập cơ sở dữ liệu hướng đối tượng và nó là một phần không thể thiếu được của Visual Basic. Phiên bản của JET đi kèm với VB 6.0 là miễn phí nghĩa là VB có thể truy xuất trực tiếp cơ sở dữ liệu của Microsoft Access. Giao diện để VB truy xuất JET có tên là *Data Access Objects (DAO)*.

JET là một bộ máy cơ sở dữ liệu tuyệt vời cho các ứng dụng văn phòng chạy trên máy đơn, nhưng hiệu suất của nó giảm đáng kể khi số lượng người dùng tăng lên và cơ sở dữ liệu được mở rộng. Vì điều này JET không phải là một giải pháp tối ưu cho các ứng dụng cơ sở dữ liệu nhiều người dùng. Cho đến nay người ta chưa có một thống kê chính xác được kích thước dữ liệu tối đa hay số lượng người dùng tối đa của JET nhưng nhìn chung JET bị giới hạn nhiều hơn so với các giải pháp khác trong môi trường đa người dùng. Tuy vậy, JET là điểm khởi đầu tốt nhất cho người lập trình VB bởi vì sự đơn giản của nó.

Khi kích thước dữ liệu tăng lên, người lập trình bao giờ cũng muốn xây dựng một ứng dụng Khách/Chủ (Client/Server) có khả năng bảo mật cao và linh hoạt. Vì lẽ đó, Microsoft hỗ trợ trong VB để truy cập các cơ sở dữ liệu quan hệ được thông dịch bởi chuẩn *Open Database Connectivity (ODBC)*. ODBC là một kỹ thuật cho phép truy cập các cơ sở dữ liệu quan hệ cao cấp như SQL SERVER hay ORACLE. Tuy nhiên, ODBC cũng có thể được sử dụng để truy cập các cơ sở dữ liệu nhỏ tổ chức bằng Microsoft Access hay Foxpro, thậm chí các cơ sở dữ liệu máy chủ như IBM DB2. Visual Basic sử dụng giao diện đối tượng *Remote Data Objects (RDO)* để truy cập ODBC.

DAO và RDO là những kỹ thuật hỗ trợ việc truy xuất đến các cơ sở dữ liệu quan hệ. Tuy nhiên, Microsoft lại cung cấp một công cụ hữu ích hơn để truy cập dữ liệu goi là *OLEDB*. OLEDB là kỹ thuật cho phép dữ liệu được truy xuất từ cả 2 nguồn cơ sở dữ liệu: quan hệ và không quan hệ. Điều đó có nghĩa là gồm các cơ sở dữ liệu của Microsoft Access, Oracle, SQL SERVER và cả các nguồn dữ liệu không quan hệ như Excel, Microsoft Index Server, Microsoft Exchange, Active Directory… Visual Basic sử dụng giao diện đối tượng *ActiveX Data Objects (ADO)* để truy cập OLEDB.

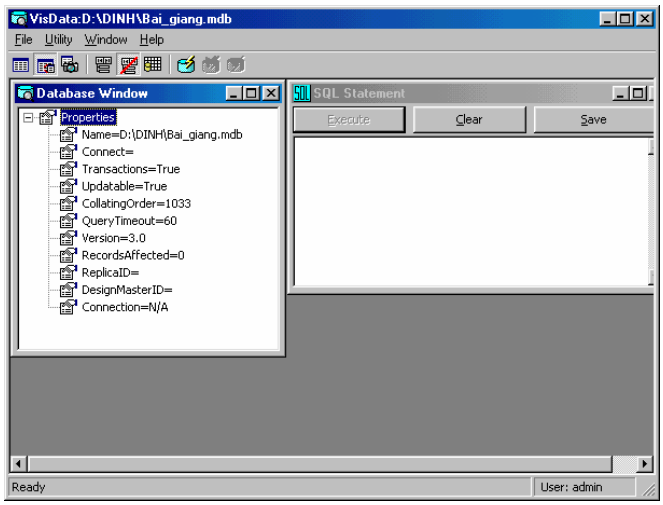
Visual Basic cung cấp cho ta nhiều công cụ để truy cập dữ liệu như DAO, RDO, ADO. Câu hỏi thường đặt ra là: Kỹ thuật nào được sử dụng lúc nào ở đâu? Nhiều người cho rằng DAO & RDO đã lỗi thời và người ta hiếm sử dụng chúng. Thật ra DAO & RDO là các điển hình cho một vài khả năng tiêu biểu của ADO. Hiện nay, vẫn còn khá nhiều ứng dụng sử dụng DAO & RDO và thật sự chúng bị giới hạn trong chừng mực nào đó. OLEDB thực sự cung cấp một khả năng rộng lớn để truy cập các cơ sở dữ liệu từ nhiều nguồn khác nhau. Tuy vậy, trong một số trường hợp một giải pháp dùng RDO lại hữu dụng hơn ADO.

III. Dùng Visual Basic để tạo một cơ sở dữ liệu

Thông thường chúng ta sẽ sử dụng các hệ quản trị cơ sở dữ liệu để tạo nên một cơ sở dữ liệu, nhưng trong phần này ta sẽ xét qua tính năng tạo cơ sở dữ liệu bằng Visual Basic 6.0. Ta có thể áp dụng phương pháp này cho những cơ sở dữ liệu nhỏ và tương thích với Microsoft Access.

III.1 Sử dụng cửa sổ cơ sở dữ liệu

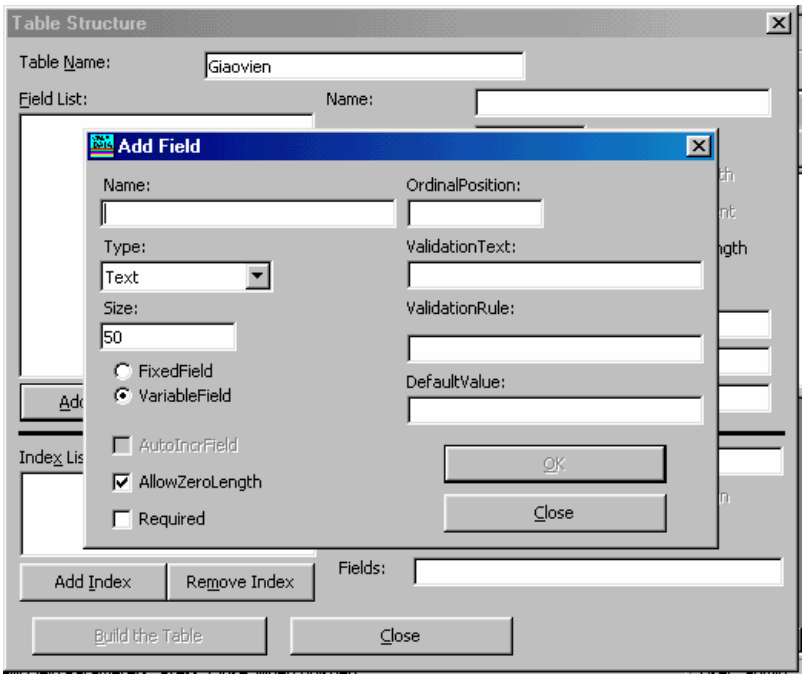
- Từ Menu của VB6, chọn mục *Add-Ins, Visual Data Manager*. Cửa sổ Visual Data Manager sẽ xuất hiện.
- Chọn mục *File -> New -> MicroSoft Access -> Version 7.0 MDB*.
- Chọn thư mục ta muốn lưu cơ sở dữ liệu và tên của cơ sở dữ liệu.



Hình VIII.1 Cửa sổ Visual Data Manager

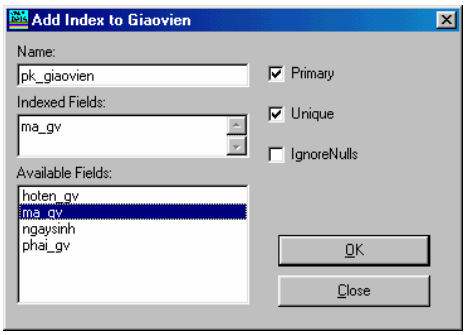
Tạo bảng

- Để tạo mới một bảng, ta chọn *Properties* trong cửa sổ *Databases*, nhấp chuột phải, chọn *New Table*, đặt tên cho Table tại ô Table Name, ấn *Add Field* để tạo mới các trường cho bảng.



Hình VIII.2 Cửa sổ tạo Table

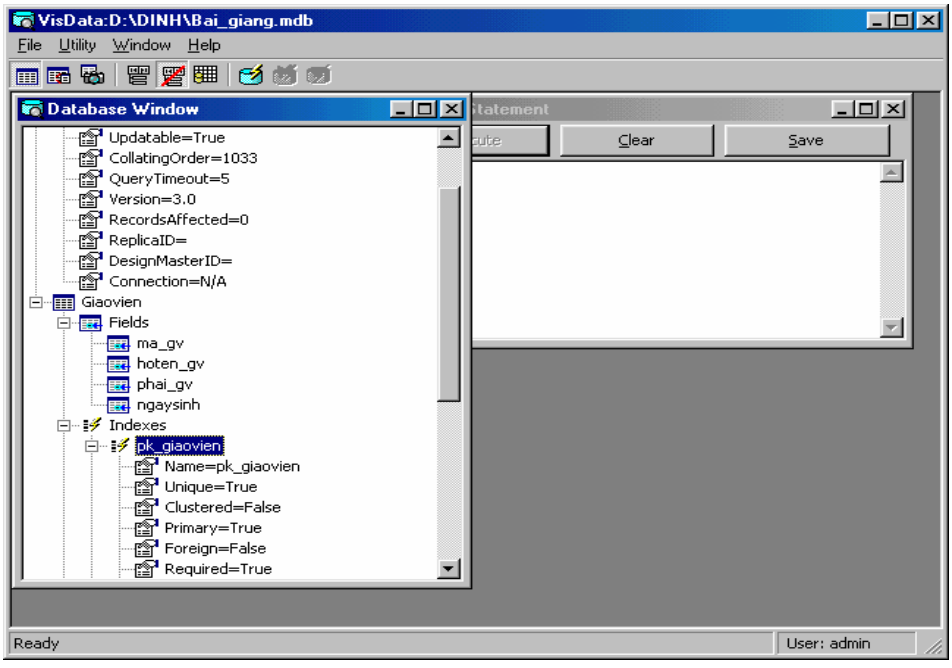
- Ta sẽ nhập tên trường tại ô *Name*, chọn kiểu của trường tại *Combo Type*, tùy chọn *FixedField* và *VariableField* xác định độ dài của trường là cố định hay thay đổi.
- Sau khi xác định đầy đủ các thuộc tính của trường, ấn OK và tiếp tục thêm vào các trường khác cho bảng. Nếu đã thêm mới đầy đủ các trường của bảng, ấn Close để quay về cửa sổ Table Structure.
- Sau khi quay về cửa sổ Table Structure, ta sẽ xác lập các chỉ mục cũng như khóa chính của bảng.



Hình VIII.3 Cửa sổ tạo khóa chính và chỉ mục

- Tại ô *Name*, ta sẽ nhập vào tên của chỉ mục, rồi chọn các trường tham gia vào chỉ mục đó. Nếu ta chọn *Primary* thì đó chính là các trường cấu thành khóa chính của bảng. Chọn *Unique* tức là giá trị của chỉ mục đó sẽ không có sự trùng lặp.
- Ấn Close xác nhận rằng ta đã xây dựng xong tập các chỉ mục của bảng.
- Sau khi đã hoàn thành tất cả các thao tác trên, để tạo bảng ta ấn Build the Table.

Tuy rằng đây là một tính năng mới của VB6, tuy nhiên chúng ta cũng sẽ gặp phải rất nhiều bất tiện khi phải thiết kế một cơ sở dữ liệu hoàn chỉnh cũng như trong quá trình bảo trì và sử dụng (khó khăn trong việc thay đổi các thuộc tính đã xác lập, không tạo liên kết giữa các bảng được …). Một phương cách tốt nhất đó là nên dùng các hệ quản trị cơ sở dữ liệu chuyên dùng để thực hiện công việc nêu trên.

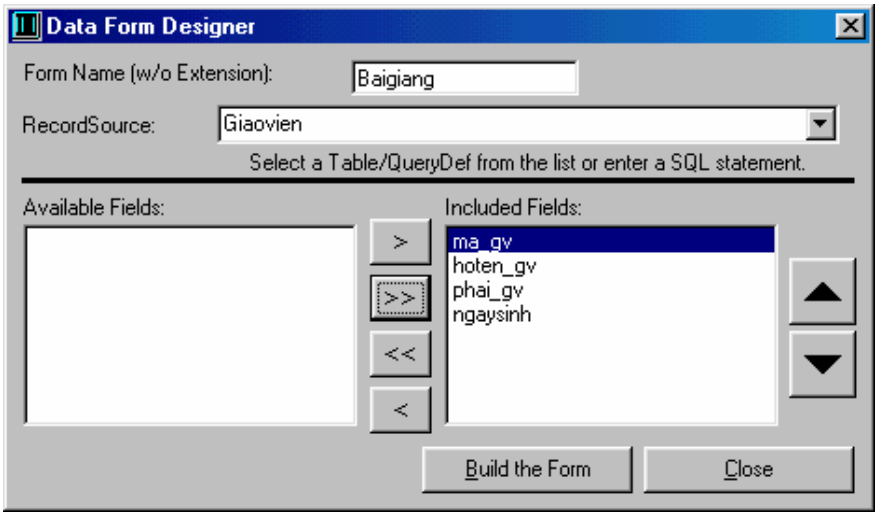


Hình VIII.4 Tạo bảng cho cơ sở dữ liệu

III.2 Dùng Visual Data Manager để tạo giao diện

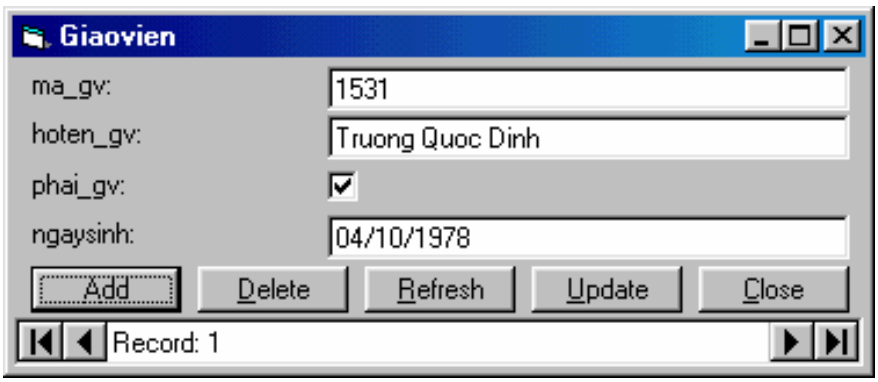
Ta có thể thiết kế một Form nhập liệu đơn giản cho một Table từ Visual Data Manager. Các bước tiến hành như sau:

- Từ Visual Data Manager chọn Cơ sở dữ liệu cần thao tác.
- Chọn *Data Form Design* từ mục *Utility*.
- Chọn Table cần cho việc tạo Form và các trường hiển thị trên Form (thông thường chúng ta sẽ cho hiển thị tất cả các trường).
- Chọn *Build the Form*, biểu mẫu mới đã được tạo trong đề án của chúng ta.



Hình VIII.5 Thiết lập các thuộc tính cho Form

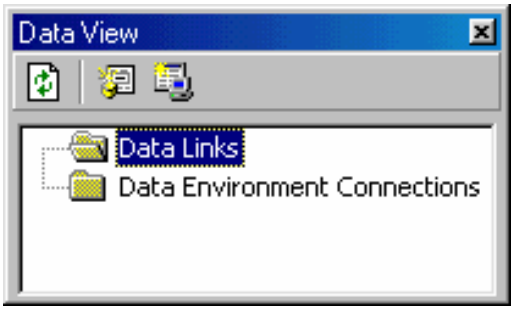
Kết quả sau khi chúng ta xây dựng Form bằng Visual Data Manager:



IV. Sử dụng cửa sổ xem dữ liệu (Data View)

VB6 còn hỗ trợ cho người lập trình khả năng làm việc với cơ sở dữ liệu mà không phải thông qua công cụ quản trị cơ sở dữ liệu hoặc các công cụ trong Add-In. Công cụ này chính là cửa sổ Data View.

- Từ Menu của VB chọn *Data View* hoặc nhấn nút *Data View* trên thanh công cụ.
- Cửa sổ *Data View* xuất hiện cho ta hai lựa chọn: *Data Links* và *Data Environment Connections*.

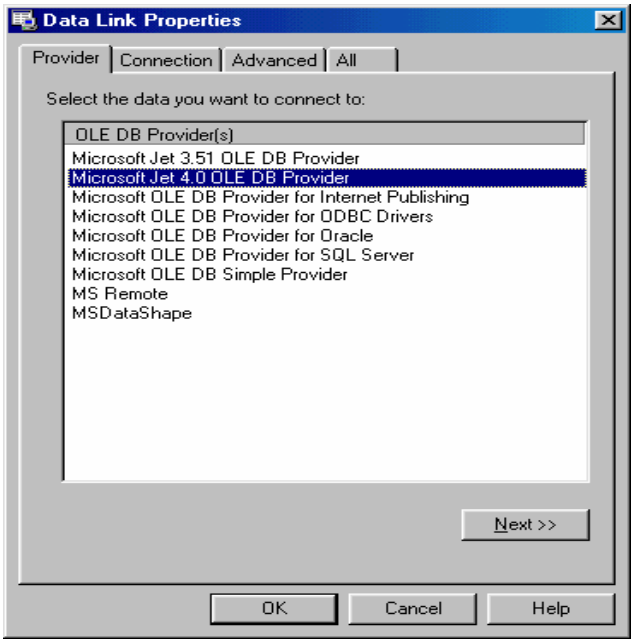


Hình VIII.6 Cửa sổ Data View

Liên kết dữ liệu (Data Link) là một cách kết nối môi trường phát triển của VB6 với một cơ sở dữ liệu nào đó. Một kết nối môi trường dữ liệu (Data Environment Connection) là cách thức sử dụng cơ sở dữ liệu trong một đề án cụ thể. Điểm khác biệt giữa hai thành phần này là sự liên quan giữa cơ sở dữ liệu với đề án cụ thể.

Để sử dụng cơ sở dữ liệu theo kiểu Data Link, ta tiến hành như sau:

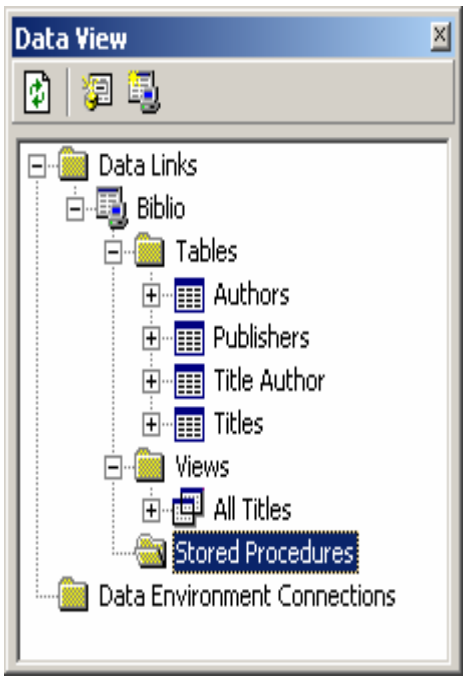
- Chọn *Data Links*, ấn chuột phải -> *Add a Data Link* hoặc ấn nút *Data Link* trên thanh công cụ của cửa sổ.
- Cửa sổ *Data Link Properties* xuất hiện.



Hình VIII.7 Hộp thoại Data Link Properties

- Chọn trình cung cấp *Microsoft Jet*, chọn *Next*.
- Chọn cơ sở dữ liệu muốn nối kết đến, nhấn *OK*.

Liên kết dữ liệu cung cấp một cách nhìn tóm lược về nguồn dữ liệu. Mỗi lần ta tạo một liên kết dữ liệu, ta có thể duyệt bằng cách mở rộng phần tử trong danh sách tóm lược. Thực hiện điều này bằng cách nhấn vào dấu cộng bên trái mỗi phần tử (hình VIII.8).



Hình VIII.8 Cửa sổ Data

V. Sử dụng điều khiển dữ liệu để tạo giao diện người sử dụng

Điều khiển dữ liệu giúp cho người sử dụng liên kết biểu mẫu của mình đến nguồn cơ sở dữ liệu. Điều khiển dữ liệu cung cấp cho người sử dụng những tính năng xử lý dữ liệu cơ bản như duyệt qua các mẩu tin, thêm mới, cập nhật.

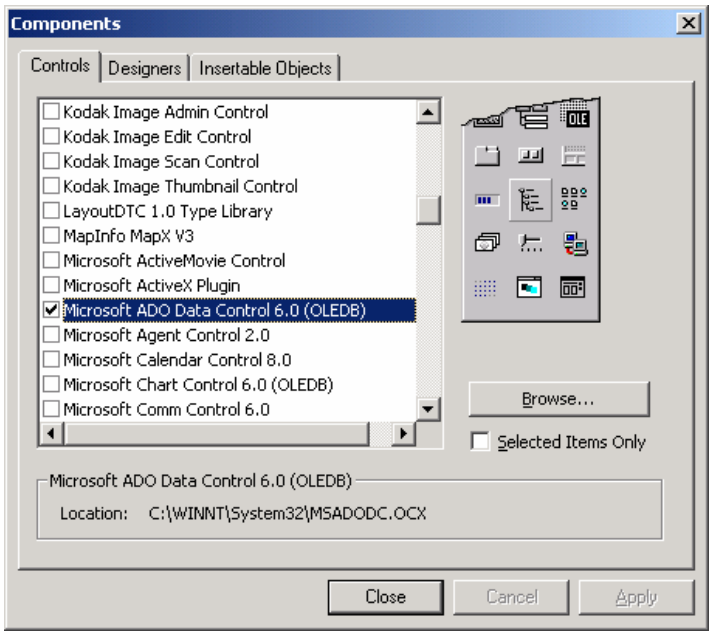
Đối với phiên bản VB6 cung cấp cho chúng ta 3 trình điều khiển dữ liệu: DAO (Data Access Object), RDO (Remote Data Object) và ADO (ActiveX Data Object).

V.1 Kết nối với cơ sở dữ liệu và làm việc với các mẩu tin thông qua điều khiển ADO Data

V.1.1 Hiển thị dữ liệu

Nếu như chúng ta xây dựng một biểu mẫu chỉ để hiển thị các mẩu tin của một bảng, điều này rất đơn giản và ta không cần phải lập trình gì cả.

Để sử dụng điều khiển ADO Data, ta cần đánh dấu Microsoft ADO Data Control 6.0 (OLEDB) trong hộp thoại Components.



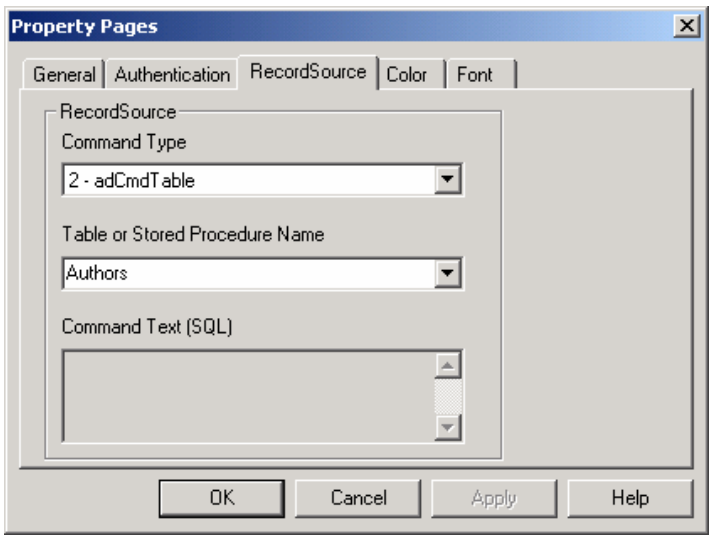
Hình VIII.9 Hộp thoại Components

Chọn điều khiển ADO Data từ hộp công cụ đưa vào biểu mẫu, liên kết đến nguồn dữ liệu thông qua hai thuộc tính ConnectionString và RecordSource.

- *ConnectionString*: Xác định nguồn dữ liệu cần nối kết, đó chính là chuỗi nối kết chỉ đến cơ sở dữ liệu mà ta thao tác.
- *RecordSource*: Xác định xem nối kết của ta đang thao tác trên bảng nào.

Ví dụ: Tạo một nối kết đến cơ sở dữ liệu "C:\Program Files\Microsoft Visual Studio\VB98 \Biblio.mdb".

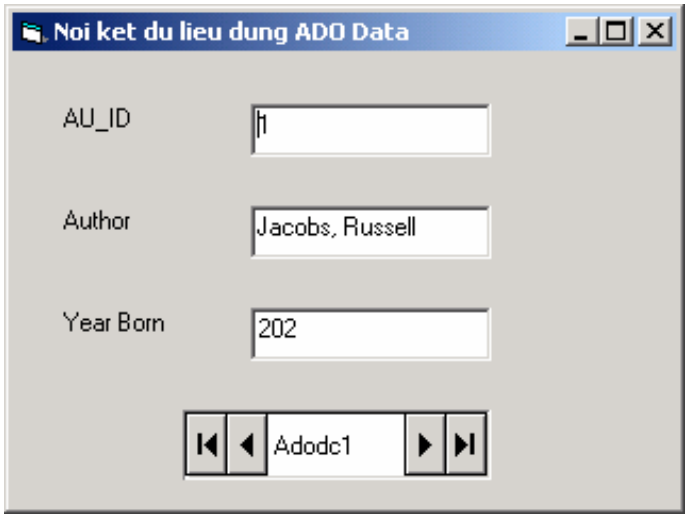
- Chọn Use Connection String, ấn Build.
- Chọn Microsoft Jet 4.0 OLE DB Provider.
- Chọn cơ sở dữ liệu như ví dụ.
- Ấn OK.
- Quay về cửa sổ Property Pages, chọn Tab RecordSource, xác định các tùy chọn như hình vẽ.
- Ấn Close.



Sau khi đã xác định được nối kết, ta vẫn không thấy được sự hoạt động của điều khiển dữ liệu, nguyên nhân do chúng ta không có điều khiển để hiển thị nội dung, cách giải quyết vấn đề là dùng điều khiển TextBox hiển thị dữ liệu.

Để dùng điều khiển Textbox hiển thị dữ liệu, ta xác định hai thuộc tính sau đây của điều khiển: *DataSource, DataField*. Các thuộc tính này xác định nguồn dữ liệu và tên trường, đối với ví dụ này đó là Adodc1 (tên của ADO Data) và Au_Id.

- Thực thi đề án, ta được kết quả sau:



Hình VIII.11 Ví dụ dùng ADO Data

V.1.2 Cập nhật dữ liệu

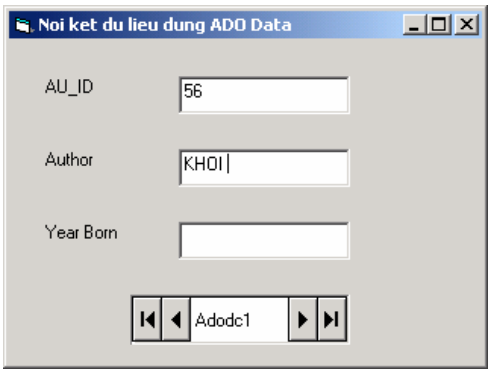
Thao tác cập nhật dữ liệu cũng khá đơn giản, điều khiển ADO Data sẽ tự động cập nhật lại giá trị của mẩu tin hiện hành mỗi khi ta duyệt qua mẩu tin khác, vì vậy ta cũng không phải làm gì cả.

V.1.3 Thêm mới mẩu tin

Để có thể thêm mới mẩu tin, ta có hai phương cách như sau:

- Thiết lập thuộc tính *EOFAction* của điều khiển ADO Data là *2-AddNew*. Cách này không cần phải lập trình gì cả.

Để thêm mới vào một mẩu tin, ta sẽ đi đến cuối mẩu tin, sau đó ấn nút tiếp, ta nhận thấy giá trị của các trường sẽ rỗng để chờ chúng ta nhập mới thông tin vào.

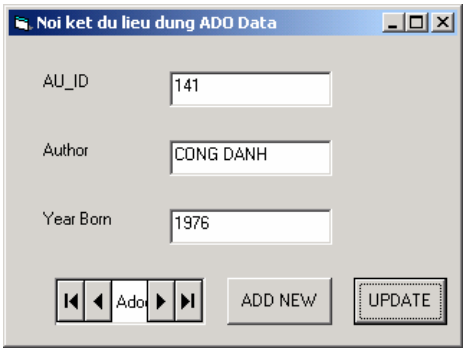


Hình VIII.12 Thêm mới mẩu tin dùng ADO Data

- Thêm mới mẩu tin bằng 2 phương thức *AddNew* và *Update*, các bước tiến hành sẽ như sau:

- Thiết kế hai nút lệnh là ADD NEW và UPDATE.

- Trong sự kiện Click của hai nút trên lần lượt nhập vào câu lệnh sau: `Adodc1.Recordset.AddNew`, `Adodc1.Recordset.Update` với `Adodc1` là thuộc tính Name của điều khiển dữ liệu.



Hình VIII.13 Sử dụng phương thức AddNew và Update

V.1.4 Dùng sự kiện MoveComplete để cập nhật giao diện người sử dụng

Ta có thể dùng sự kiện *MoveComplete* của điều khiển ADO Data để khởi động sửa đổi trong ứng dụng khi người sử dụng di chuyển từ mẩu tin này sang mẩu tin khác.

Sự kiện *MoveComplete* được kích hoạt khi một mẩu tin mới thành mẩu tin hiện hành. Đây là một trong vài sự kiện được kích hoạt khi điều khiển di chuyển từ mẩu tin này sang mẩu tin khác. Các sự kiện khác bao gồm *WillChange*, được kích hoạt khi điều khiển di chuyển từ mẩu tin này sang mẩu tin khác hay thay đổi một mẩu tin và sự kiện *RecordChangeComplete*, xảy ra khi một mẩu tin được sửa đổi thành công trong cơ sở dữ liệu như một kết quả của hoạt động trong điều khiển dữ liệu.

V.1.5 Xóa mẩu tin

Để xóa mẩu tin trong một ứng dụng sử dụng điều khiển dữ liệu, ta dùng phương thức *Delete* của đối tượng *Recordset* của điều khiển dữ liệu.

Ví dụ: `Adodc1.Recordset.Delete`

V.1.6 Dùng sự kiện WillChangeRecord để đảm bảo dữ liệu hợp lệ

Trong lập trình cơ sở dữ liệu, việc đảm bảo rằng dữ liệu nhập vào phù hợp với các quy tắc của một cơ sở dữ liệu người dùng cụ thể là yếu tố quan trọng bậc nhất và mang tính bắt buộc.

Đối với điều khiển ADO Data, việc xác định xem dữ liệu có hợp lệ hay không sẽ được viết trong sự kiện *WillChangeRecord* của điều khiển. Sự kiện này sẽ được kích hoạt khi người dùng thay đổi thông tin của một mẩu tin và di chuyển sang mẩu tin khác hoặc thêm mới mẩu tin.

V.2 Kết nối với cơ sở dữ liệu và làm việc với các mẩu tin thông qua điều khiển Data (DAO Data Control)

Điều khiển Data hạn chế hơn điều khiển ADO Data vì nó chỉ cho phép chúng ta nối kết đến một số nguồn dữ liệu cụ thể, chẳng hạn như Access, Excel, Foxpro,… nhưng Version của các hệ quản trị này cũng là những Version đã lâu đời.

Để sử dụng điều khiển này, ta cần xác định các giá trị của các thuộc tính sau: *Connect, DatabaseName, RecordSource*.

- *Connect* xác định cơ sở dữ liệu của ta là loại gì.
- *DatabaseName* chỉ đến một cơ sở dữ liệu cụ thể.
- *RecordSource* là một bảng dữ liệu trong cơ sở dữ liệu (đối với cơ sở dữ liệu Access).

Sau khi đã xác định giá trị của các thuộc tính trên thì việc duyệt qua các mẩu tin cũng tương tự như của điều khiển ADO Data.

Thuộc tính *EOFAction* của điều khiển Data quy định việc chúng ta có thể thêm mới một mẩu tin hay là không (tương tự điều khiển ADO Data).

Chương 9: CÁC ĐỐI TƯỢNG TRUY CẬP DỮ LIỆU (DATA ACCESS OBJECTS)

Mục tiêu:

Chương này giới thiệu về thư viện đối tượng Data Access Objects, cách thức được sử dụng để truy cập cơ sở dữ liệu của các hệ quản trị cơ sở dữ liệu nhỏ như Microsoft Access, Foxpro...

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Mô hình cây phân cấp của đối tượng DAO.
- Sử dụng thư viện đối tượng DAO để tương tác với cơ sở dữ liệu trong VB.

Kiến thức có liên quan:

- Các cấu trúc lập trình trong VB.
- Câu lệnh truy vấn dữ liệu trong cơ sở dữ liệu.

Tài liệu tham khảo:

- **Microsoft Visual Basic 6.0 & Lập trình cơ sở dữ liệu** - Chương 20, trang 571 - **Nguyễn Thị Ngọc Mai** (chủ biên), **Nhà xuất bản Giáo dục** - **2000**.
- **Tự học Lập trình cơ sở dữ liệu với Visual Basic 6 trong 21 ngày** (T1) – Chương 8, trang 305 - **Nguyễn Đình Tê** (chủ biên), **Nhà xuất bản Giáo dục** - **2001**.

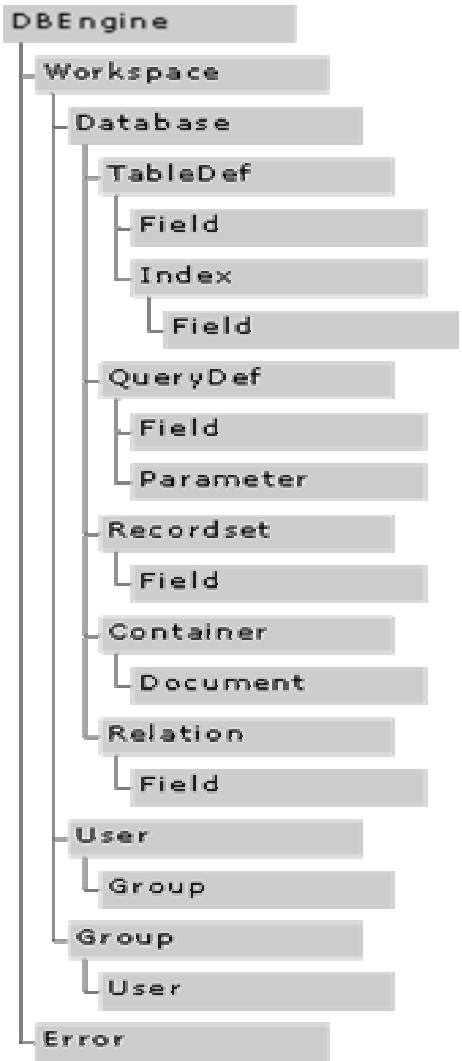
Các ứng dụng Visual Basic có thể thao tác trên cơ sở dữ liệu thông qua DAO (Data Access Objects). Dùng DAO ta có thể thi hành các câu truy vấn để xem, cập nhật, cũng như tạo mới các giá trị cho các mẩu tin của bảng …

DAO có thể được sử dụng cho các ứng dụng truy cập dữ liệu từ máy cá nhân hoặc các ứng dụng theo kiểu Khách/Chủ (Client/Server). Tuy nhiên, vào thời điểm hiện tại, chúng ta chỉ dùng DAO để thao tác với cơ sở dữ liệu Jet vì dạng ứng dụng Client/Server là thế mạnh của ADO (ActiveX Data Objects).

I. Mô hình đối tượng Data Access Objects (DAO)

Mô hình đối tượng DAO khá phức tạp với hàng trăm yếu tố với rất nhiều tập hợp chứa khá nhiều đối tượng, mỗi đối tượng lại có các thuộc tính, phương thức và các đối tượng con của riêng nó.

Sau đây là mô hình cây phân cấp của đối tượng DAO:



Trong lập trình DAO, có một tập hợp cốt lõi các kỹ thuật thông dụng được sử dụng gần như mọi chương trình. Chúng bao gồm:

- Thi hành câu truy vấn SELECT để lấy về dữ liệu từ cơ sở dữ liệu.
- Duyệt qua từng mẩu tin trong một Recordset.
- Thi hành câu truy vấn hành động (Update, Delete, Insert).
- Sửa đổi cấu trúc cơ sở dữ liệu.

- Xử lý lỗi phát sinh bởi truy cập cơ sở dữ liệu.

II. Sử dụng DAO để làm việc với cơ sở dữ liệu

Để sử dụng đối tượng DAO, ta cần tham chiếu đến đối tượng này bằng cách chọn Project -> References, sau đó đánh dấu chọn **Microsoft DAO 3.51 Object Library**. Nhấn OK và ta đã có thể sử dụng các đối tượng do DAO cung cấp.

II.1 Đối tượng Database

Đối tượng Database là nơi bắt đầu việc truy cập đến cơ sở dữ liệu, hay nói cách khác để có thể kết nối và thao tác với cơ sở dữ liệu thông qua DAO thì cần thông qua đối tượng đầu tiên đó là đối tượng Database.

Trước tiên ta khai báo một đối tượng Database như sau:

```
Dim db As Database
```

Đối tượng Database có rất nhiều phương thức, ta sẽ chỉ xét qua những phương thức cơ bản và quan trọng nhất cho phép ta thao tác với cơ sở dữ liệu.

II.1.1 Sử dụng phương thức OpenDatabase để tạo một đối tượng Database

Ta dùng phương thức OpenDatabase để cho phép một đối tượng Database tham khảo đến một cơ sở dữ liệu cụ thể, đối số bắt buộc của phương thức là một chuỗi, kết quả trả về là một đối tượng Database, vì vậy trước khi sử dụng phương thức này, ta cần khai báo một đối tượng Database. Chẳng hạn như:

```
Dim db As Database
Set db = OpenDatabase("..\..\baigiang.mdb")
```

Cú pháp đầy đủ của phương thức OpenDatabase:

***Set** database = **OpenDatabase** (dbname, options, read-only, connect)*

Ý nghĩa các tham số của phương thức OpenDatabase như sau:

Thành phần	Ý nghĩa
database	Biến kiểu đối tượng Database mà ta muốn sử dụng.
dbname	Chuỗi xác định sự tồn tại của cơ sở dữ liệu Jet hoặc là tên nguồn dữ liệu (DSN) dạng ODBC data source.
options	Biến mang giá trị xác định tùy chọn loại cơ sở dữ liệu.
read-only	Mang giá trị kiểu Boolean, TRUE nếu như mở cơ sở dữ liệu dạng chỉ đọc, ngược lại cho kiểu truy xuất đọc và ghi.
connect	Kiểu chuỗi, xác định kiểu nối kết bao gồm cả mật khẩu.

Các giá trị của tùy chọn Options

Giá trị	Ý nghĩa
dbDriverNoPrompt	Trình quản lý ODBC dùng chuỗi nối kết cung cấp tên cơ sở dữ liệu và nối kết. Nếu

	ta không cung cấp thông tin cụ thể, sẽ xảy ra lỗi tại thời điểm thực thi.
dbDriverPrompt	Trình quản lý ODBC hiển thị hộp thoại các nguồn dữ liệu ODBC. Chuỗi kết nối sẽ được tạo từ nguồn dữ liệu do người dùng chọn thông qua hộp thoại, nếu không nguồn dữ liệu mặc định sẽ được sử dụng.
dbDriverComplete	Nếu nối kết và tên nguồn dữ liệu xác định đầy đủ các thông tin cần thiết cho một nối kết thì trình quản lý ODBC sẽ dùng chuỗi trong nối kết nếu không sẽ như trường hợp sử dụng dbDriverPrompt.

II.1.2 Sử dụng phương thức Execute để thi hành câu truy vấn hành động

Ta sử dụng phương thức **Excute** của đối tượng **Database** để thi hành một câu lệnh SQL trên cơ sở dữ liệu. tuy vậy phương thức này không nên dùng cho mọi trường hợp. ta chỉ nên dùng phương thức **Excute** để thi hành các lệnh SQL cho các mục đích sau:

- Cập nhật, xóa hay sao chép mẩu tin (trong Access/Jet, ta gọi là các truy vấn hành động).
- Sửa cấu trúc cơ sở dữ liệu (được biết như là các lệnh **DDL** – Ngôn ngữ định nghĩa dữ liệu, Data Definition Language).

Các câu truy vấn SELECT theo quy ước (lấy về các mẩu tin) được thi hành nhờ phương thức OpenRecordset của đối tượng Database.

Cú pháp:

Object.Execute Source

- Object: đối tượng Database.
- Source: câu SQL kiểu biến chuỗi.

Ví dụ: Tăng giá bán mỗi sản phẩm của bảng Products của CSDL Northwind.MDB lên 10%.

```
Dim db As Database
Private Sub Form_Load()
   Set db = OpenDatabase("Northwind.mdb")
End Sub
Private Sub cmdExcute_Click()
   db.Excute "UPDATE Products " & _
             "SET [Unit Price] = [Unit Price] * 1.1"
End Sub
```

II.2 Đối tượng Recordset

Đối tượng Recordset xác định một tập hợp các mẩu tin từ một bảng cơ sở hoặc kết quả của một câu lệnh truy vấn nào đó. Tại một thời điểm bất kỳ, đối tượng

Recordset chỉ tham khảo đến một mẩu tin đơn, đó là mẩu tin hiện hành. Để sử dụng đối tượng Recordset, ta dùng phương thức Open Recordset.

Cú pháp thông dụng của OpenRecordset như sau:

Set recordset = object.OpenRecordset source [, Type][, Options][, LockEdits]

Trong đó phương thức OpenRecordset trả về đối tượng Recordset và object là biến đối tượng kiểu Database, tham số Source ở đây sẽ là một câu lệnh truy vấn (SELECT).

Đối tượng Recordset có nhiều loại (*Type*) khác nhau, sau đây là bảng phân tích công dụng cũng như ưu, nhược điểm của từng kiểu Recordset.

Kiểu	Ưu điểm	Nhược điểm
Dynamic (dbOpenDynamic)	Cập nhật được, kết quả trả về có thể thuộc nhiều bảng khác nhau thông qua nối kết bảng. Một ưu điểm nổi trội đó là đối với cơ sở dữ liệu nhiều người sử dụng, nó có được khả năng tự cập nhật khi các người dùng khác cập nhật mẩu tin chứa trong đó.	Kém hiệu quả hơn so với kiểu Dynaset
Dynaset (dbOpenDynaset)	Các chức năng tương tự như Dynamic, nhưng đạt được hiệu quả hơn do không thao tác trên dữ liệu thực sự mà là sự tham chiếu đến dữ liệu.	Tìm kiếm không thật sự hiệu quả do không có các chỉ mục.
Forward-Only (dbOpenForwardOnly)	Có thể lấy về mẩu tin từ nhiều bảng thông qua nối kết bảng. Đặc biệt hiệu quả đối với các Recordset nhỏ.	Ta chỉ có thể di chuyển đến phía trước.
Snapshot (dbOpenSnapshot)	Tương tự như Forward-Only	Không cập nhật được với dữ liệu Jet. Trả về một bảng sao đến dữ liệu, nên thao tác chậm hơn rất nhiều so với Dynaset.
Table (dbOpenTable)	Có thể định vị và lấy về các mẩu tin một cách nhanh chóng vì các bảng được lập chỉ mục.	Không thể thực hiện một câu truy vấn liên quan đến nhiều bảng.

Lưu ý:

✓ Nếu ta mở một Recordset với bộ máy CSDL Microsoft Jet và ta không xác định tham số *type* của OpenRecordset thì *dbOpenTable* là mặc định (nếu có thể).

✓ Với một Recordset xác định một câu truy vấn, *dbOpenDynaset* là mặc định.

✓ Với cách truy cập CSDL theo ODBC, *dbOpenForwardOnly* là mặc định.

Một số giá trị của tham số *Option*, một hằng số có thể được kết hợp bởi nhiều giá trị khác nhau, xác định đặc tính của Recordset.

Hằng số	Ý nghĩa
dbAppendOnly	Cho phép người dùng thêm mẩu tin mới vào Recordset, nhưng không được sửa đổi hay xóa các mẩu tin có sẵn (chỉ với dynaset-Recordset của JET).
dbSQLPassThrough	Cho phép tham khảo đến các câu SQL của bộ máy CSDL JET khi khi nó được nối với một nguồn dữ liệu ODBC (chỉ với snapshot-Recordset của JET).
dbSeeChanges	Một lỗi thực thi sẽ xuất hiện khi một người dùng thay đổi dữ liệu mà người khác đang thao tác (chỉ với dynaset-Recordset của JET). Điều này thật sự có ích cho ứng dụng đa người dùng cần đồng bộ hóa dữ liệu.
dbDenyWrite	Ngăn cản người dùng khác sửa đổi hay thêm mẩu tin.
dbDenyRead	Ngăn cản người dùng khác đọc dữ liệu từ một bảng (chi với Table-Recordset của JET).
dbForwardOnly	Một Recordset chỉ cho phép di chuyển tới (snapshot-Recordset của JET).
dbReadOnly	Không cho người dùng thay đổi dữ liệu.
dbRunAsync	Thực thi một câu truy vấn không đồng bộ (truy cập dữ liệu theo ODBC).
dbExecDirect	Thực thi câu truy vấn bỏ qua phương thức SQLPrepare và trực tiếp gọi phương thức SQLExecDirect (truy cập dữ liệu ODBC trong môi trường đa người dùng).

Lưu ý: Ta không thể sử dụng tham số *lockedits* khi *options* là dbReadOnly.

Một số các giá trị của tham số *lockedits*:

Hằng số	Ý nghĩa
dbReadOnly	Ngăn cản người dùng sửa đổi dữ liệu (mặc nhiên đối với cách truy cập dữ liệu theo ODBC). Ta có thể sử dụng hằng số này ở tham số *Options* hay *LockEdits* đều được, nhưng không thể cùng một lúc (lỗi thực thi xảy ra).
dbPessimistic	Khóa trang bi quan trong môi trường đa người dùng. Trang chứa mẩu tin đang sửa đổi sẽ bị khóa lại khi phương thức Edit được thực thi (mặc định đối với JET).
dbOptimistic	Khóa trang lạc quan trong môi trường đa người dùng. Trang chứa mẩu tin đang sửa đổi sẽ không bị khóa cho tới khi phương thức Update được gọi thực thi.
dbOptimisticValue	Khóa trang lạc quan đồng thời dựa vào giá trị của một dòng cụ thể (chỉ đối với cách truy cập dữ liệu theo ODBC).
dbOptimisticBatch	Cho phép cập nhật theo lô (chỉ đối với cách truy cập dữ liệu theo ODBC).

Lưu ý: Xét ví dụ sau:

```
Dim db As Database
Dim rs As Recordset
Set db = OpenDataBase ("..\..\baigiang.mdb")
Set rs = db.OpenRecordset ("Select * From Canbo " & _
        "Order   by  hotencb  =  "Truong"")
```

- Như vậy câu lệnh cuối cùng trong ví dụ trên sẽ sai ở chỗ là VB không xác định được đâu là dấu trích dẫn của chuỗi và đâu là dấu trích dẫn hết câu lệnh truy vấn. Cách khắc phục là đổi dấu trích dẫn chuỗi thành dấu nháy đơn.

- Một điểm cần chú ý khác là khi viết một câu truy vấn trên nhiều dòng thì cần có ký tự nối dòng _ cuối mỗi dòng.

- Nếu giá trị của tham số trong câu truy vấn không phải là cứng nhắc, tức ta lấy giá trị từ một biến thì ta theo nguyên tắc sau:

```
Set rs = db.OpenRecordset ("Select * From Canbo " & _
       "Order by hotencb = '"& name & "'")
```

II.3 Đối tượng Field

Đối tượng Field giúp chúng ta truy xuất giá trị của một trường trong Recordset. Giá trị của trường sẽ truy xuất qua thuộc tính Value của đối tượng Field, tuy nhiên thuộc tính Value là thuộc tính mặc định của Field, nên ta không cần tham khảo tường minh đến thuộc tính này.

Như vậy để truy xuất giá trị của một trường trong 1 Recordset cụ thể, ta có thể dùng một trong các cách sau:

- Fields(Num): Num là số thứ tự của trường trong Recordset (bắt đầu tính từ 0)
- Fields("name"): Với name là tên trường
- Fields![name]: Với name là tên trường.

II.4 Các phương thức duyệt qua đối tượng Recordset

Sau khi nhận về một đối tượng Recordset, ta cần có những cách thức để duyệt qua các mẩu tin phục vụ cho một công việc cụ thể nào đó. Ta có một số phương thức duyệt Recordset như sau:

Phương thức	Ý nghĩa
MoveFirst	Di chuyển đến mẩu tin đầu tiên trong Recordset
MoveNext	Di chuyển đến mẩu tin kế tiếp trong Recordset
MovePrevious	Di chuyển đến mẩu tin liền trước trong Recordset
MoveLast	Di chuyển đến mẩu tin cuối trong Recordset
Move **N**	Di chuyển đi **N** mẩu tin được chỉ định trong Recordset

Cũng như đã nêu ở trên, có nhiều loại kiểu Recordset, tùy vào từng kiểu mà chúng ta chỉ có thể duyệt tới mà không thể đi lui, khi đó các phương thức như MoveFirst, MovePrevious sẽ gây ra lỗi.

Để biết được rằng chúng ta đang di chuyển trong phạm vi các mẩu tin của Recordset, ta sử dụng hai thuộc tính sau đây để xác định điều đó:

- **BOF**: Trả về TRUE nếu ta di chuyển đến trước mẩu tin đầu tiên của Recordset.

- **EOF**: Trả về TRUE nếu ta di chuyển đến sau mẩu tin cuối cùng của Recordset.

Hơn thế nữa, ta có thể dùng hai thuộc tính này để kiểm tra một Recordset có rỗng hay không, một Recordset rỗng khi tại một thời điểm bất kỳ **cả hai thuộc tính EOF và BOF đều có giá trị là TRUE**.

Để xác định số mẩu tin có trong một Recordset, ta dùng **thuộc tính RecordCount**. Nhưng chú ý rằng ta *cần di chuyển đến mẩu tin cuối cùng* trước khi sử dụng thuộc tính RecordCount thì kết quả trả về mới chính xác. Tại sao lại như vậy? Bởi vì câu lệnh truy vấn được xử lý thông qua hai giai đoạn, trả về số lượng đủ mẩu tin cho xử lý và xử lý bên dưới câu lệnh truy vấn trên một số lượng đúng dữ liệu kết quả, và ta không thể điều khiển được hai quá trình này.

Để cập nhật giá trị của 1 mẩu tin ta làm theo các bước như sau:

- Dùng các phương thức duyệt mẩu tin để đi đến mẩu tin cần thay đổi giá trị.
- Thi hành **phương thức Edit**.
- Dùng thuộc tính **Fields để gán trị cho trường trong mẩu tin**, chẳng hạn:

```
rs.Fields("hotencb") = "Truong Quoc Dinh"
```

- Lưu lại sự thay đổi bằng cách **thi hành phương thức Update**.

Để thêm mới một mẩu tin ta làm theo các bước:

- Thi hành **phương thức AddNew**, VB sẽ thêm mới một mẩu tin trắng.
- Sử dụng các **cách thức gán trị** để cập nhật giá trị cho mẩu tin mới thêm vào.
- Thi hành **phương thức Update**.

Sau khi đã hoàn thành công việc chúng ta cần thi hành **phương thức Close để đóng một đối tượng Recordset**. Điều này thật sự có ý nghĩa khi Recordset hiện hành đang khóa dữ liệu, phương thức Close sẽ mở khóa và các người dùng khác có thể thao tác trên dữ liệu.

II.5 Tìm kiếm dữ liệu trong Recordset và Table (bảng)

Đôi khi đối với một số công việc nào đó, ta cần tìm kiếm một mẩu tin cụ thể trong một tập các mẩu tin của Recordset, có nhiều phương thức tìm kiếm mẩu tin, tùy vào nội dung công việc mà ta áp dụng phương thức nào cho hiệu quả.

Ta có các phương thức tìm kiếm trên Recordset như sau:

```
FindFirst|FindLast|FindNext|FindPrevious
```

Cú pháp của phương thức Find:

recordset.{FindFirst | FindLast | FindNext | FindPrevious} criteria

Thành phần	Ý nghĩa
recordset	Một biến đối tượng Recordset kiểu dynaset hoặc snapshot.
criteria	Chuỗi dùng để xác định mẩu tin, giống như mệnh đề WHERE trong câu lệnh SQL nhưng không có từ khóa WHERE.

Phương thức	Bắt đầu từ	Hướng tìm kiếm
FindFirst	Mẩu tin đầu tiên	Đến cuối Recordset
FindLast	Mẩu tin cuối cùng	Đến đầu Recordset
FindNext	Mẩu tin hiện hành	Đến cuối Recordset
FindPrevious	Mẩu tin hiện hành	Đến đầu Recordset

Các phương thức tìm kiếm này sẽ không làm nảy sinh một Recordset, nó chỉ di chuyển đến mẩu tin hợp điều kiện và mẩu tin đó trở thành mẩu tin hiện hành, nếu không tìm thấy, mẩu tin hiện hành không thay đổi, khi này **thuộc tính NoMacth có giá trị là TRUE**.

Ngoài ra đối tượng Recordset còn cung cấp **phương thức Seek** giúp ta tìm kiếm trên một *Recordset kiểu bảng có chỉ mục*, cú pháp như sau:

recordset.Seek comparison, key1, key2...key13

Thành phần	Ý nghĩa
recordset	Một biến đối tượng Recordset kiểu bảng đã định nghĩa chỉ mục thông qua thuộc tính Index.
comparison	Một trong các biểu thức so sánh sau <,<=, =, >=, or >.
key1, key2...key13	Một hoặc nhiều giá trị tương ứng với trường chỉ mục hiện hành, ta có thể dùng tối đa đến 13 giá trị.

III. Sử dụng điều khiển DAO Data

Hiện tại mặc dù việc liên kết với cơ sở dữ liệu đều có thể thực hiện thông qua điều khiển ADO Data với nhiều tính năng mạnh hơn, tuy nhiên ta cũng có thể dùng điều khiển DAO Data để tham khảo đến cơ sở dữ liệu Jet cũng như một số loại cơ sở dữ liệu khác như DBASE, văn bản, bảng tính Excel mà chúng ta không cần dùng ODBC.

Điều khiển này chính là điều khiển Data mà ta đã xét ở chương 8. Tuy nhiên khi sử dụng điều khiển này thì ta cần chú ý đến thuộc tính Connect, đây là thuộc tính quy định loại dữ liệu sẽ kết nối. Một số kiểu cơ sở dữ liệu được hỗ trợ bởi điều khiển DAO Data:

- Microsoft Access.
- DBASE III, IV và 5.0.
- Phiên bản Excel 3.0, 4.0, 5.0 và 8.0.
- Phiên bản FoxPro 2.0,2.5 2.6 và 3.0.
- Lotus spreadsheet với định dạng WK1, WK3 và WK4.
- Phiên bản Paradox 3.x, 4.x và 5.x.
- Tập tin văn bản ASCII có phân cách.

Chương 10 : ODBC VÀ CÁC ĐỐI TƯỢNG DỮ LIỆU TỪ XA (REMOTE DATA OBJECTS)

Mục tiêu:

Chương này giới thiệu về thư viện đối tượng Remote Data Objects, cách thức được sử dụng để truy cập các đối tượng dữ liệu từ xa.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Khái niệm Open Database Connectivity (ODBC).
- Sử dụng điều khiển dữ liệu từ xa (Remote Data Control) để truy cập dữ liệu.
- Cây phân cấp của mô hình đối tượng RDO.
- Sử dụng thư viện đối tượng RDO để tương tác với cơ sở dữ liệu trong VB.

Kiến thức có liên quan:

- Các cấu trúc lập trình trong VB.
- Câu lệnh truy vấn dữ liệu trong cơ sở dữ liệu.
- Nắm bắt được các mô hình DAO là một lợi thế vì lúc đó việc tiếp thu mô hình ADO được nhanh hơn.

Tài liệu tham khảo:

- **Microsoft Visual Basic 6.0 & Lập trình cơ sở dữ liệu** – Chương 23, trang 735 - **Nguyễn Thị Ngọc Mai** (chủ biên), **Nhà xuất bản Giáo dục** - **2000**.
- **Tự học Lập trình cơ sở dữ liệu với Visual Basic 6 trong 21 ngày** (T2) – Chương 17, trang 227 - **Nguyễn Đình Tê** (chủ biên), **Nhà xuất bản Giáo dục** - **2001**.

I. Open Database Connectivity (ODBC)

1. Khái niệm

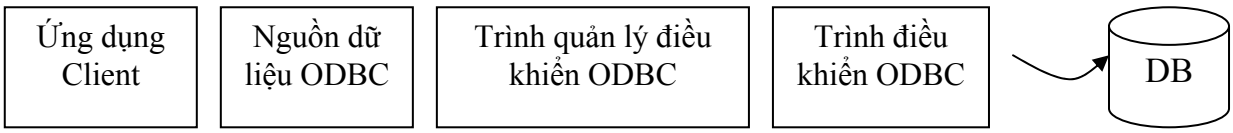
ODBC là công nghệ Windows cho phép ứng dụng Client nối với cơ sở dữ liệu từ xa. Nằm trên máy Client, ODBC làm cho nguồn dữ liệu quan hệ trở nên trong suốt đối với ứng dụng Client. Vì thế ứng dụng Client không cần quan tâm đến kiểu cơ sở dữ liệu là gì.

ODBC gồm 3 phần:

- Trình quản lý điều khiển (*driver manager*).
- Một hay nhiều trình điều khiển (driver).
- Một hay nhiều nguồn dữ liệu (data source).

2. Kiến trúc

Kiến trúc ODBC chứa kết nối giữa ứng dụng Client và cơ sở dữ liệu Server thông qua trình quản lý điều khiển ODBC.



Hình 10.1: Kiến trúc ODBC trình bày kết nối giữa ứng dụng Client và CSDL Server thông qua trình quản lý điều khiển ODBC

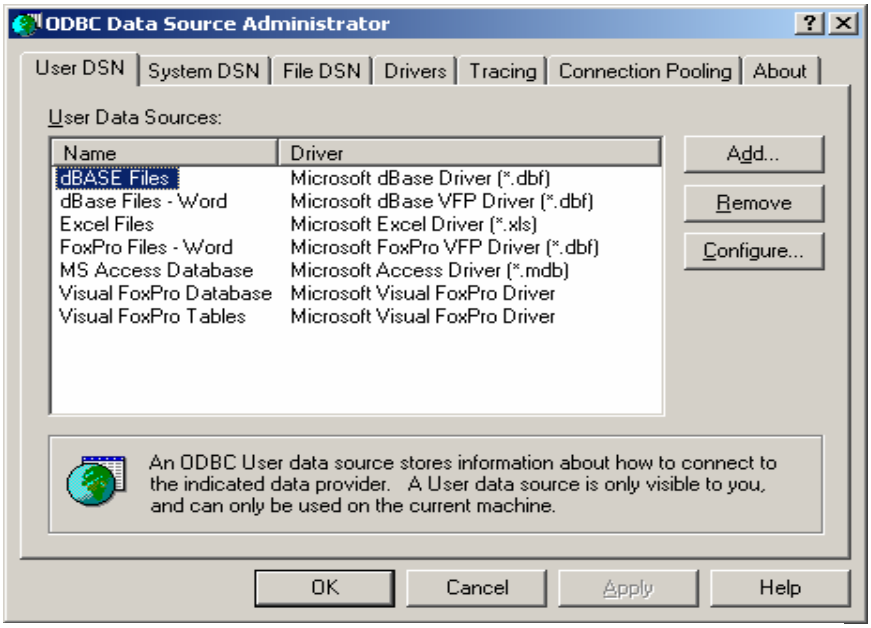
3. Tạo nguồn dữ liệu ODBC

Để một ứng dụng Client nối với cơ sở dữ liệu Client/Server dùng ODBC; trước hết, ta phải cung cấp thông tin về nguồn dữ liệu ODBC trên Client. Mỗi Server yêu cầu những gói thông tin khác nhau để nối với Client. ODBC cung cấp cho thông tin này một tên đơn giản để ta có thể tham chiếu đến nó, thay vì phải thiết lập gói thông tin từ đầu mỗi lần ta cần đến nó.

Ứng dụng Client có thể tham chiếu một cách dễ dàng đến tổ hợp của một điều khiển, một cơ sở dữ liệu và có thể thêm một người sử dụng và mật khẩu. Tên này chính là tên nguồn dữ liệu hay Data Source Name (DSN).

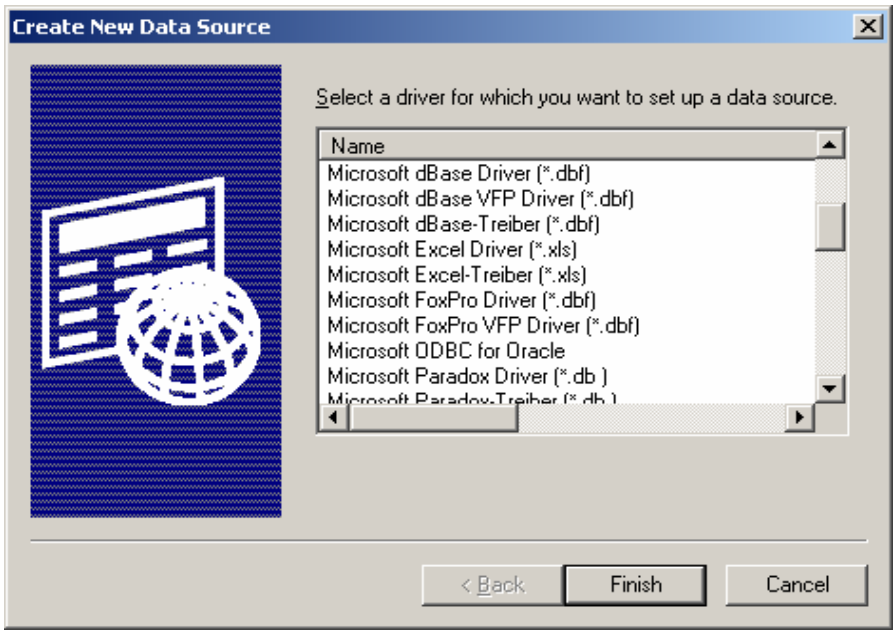
Để tạo một tên nguồn dữ liệu ODBC trên máy Client, ta làm như sau:

- Mở Control Panel.
- Chọn Administrative Tools\Data Source (ODBC), hộp thoại quản trị nguồn dữ liệu xuất hiện:



Hình 10.2: Hộp thoại quản trị nguồn dữ liệu ODBC

- Ta có thể tạo một trong ba kiểu nguồn dữ liệu ODBC:
  - ✓ **User DSN**: chỉ có người dùng tạo ra nó mới có thể sử dụng (trên máy đang dùng).
  - ✓ **System DSN**: bất kỳ ai sử dụng máy này đều có thể dùng được. Đây cũng là kiểu nguồn dữ liệu mà ta cần tạo khi cài đặt ứng dụng cơ sở dữ liệu Web.
  - ✓ **File DSN**: có thể được copy và sử dụng bởi máy khác.
- Khi hộp thoại ODBC đã mở ra, chọn lớp *UserDSN* (hay *System DSN*), Tạo một kết nối mới, nhấn nút Add, màn hình sẽ hiện ra như sau:



Hình 10.3: Lựa chọn loại cơ sở dữ liệu cần thiết để tạo kết nối

o Chọn loại CSDL mà ta muốn thao tác (Access, Foxpro, SQL Server,…), nhấn Finish. Sau khi nhấn **Finish**, một màn hình sẽ hiện ra cho phép ta nhập vào **Data Source Name**, đây là tên của kết nối CSDL. Tên của kết nối không cần phải giống với tên của cơ sở dữ liệu. Phần **Description** dùng để gõ các thông tin mô tả về kết nối. Ngoài ra ta còn phải chọn đường dẫn đến tập tin CSDL tương ứng.

4. Truy cập nguồn dữ liệu với điều khiển DAO Data và ODBC Direct

DAO tự động nạp bộ máy cơ sở dữ liệu Jet mỗi khi nó truy cập dữ liệu Client/Server, thậm chí khi ta không thực sự sử dụng cơ sở dữ liêu Jet/Access.

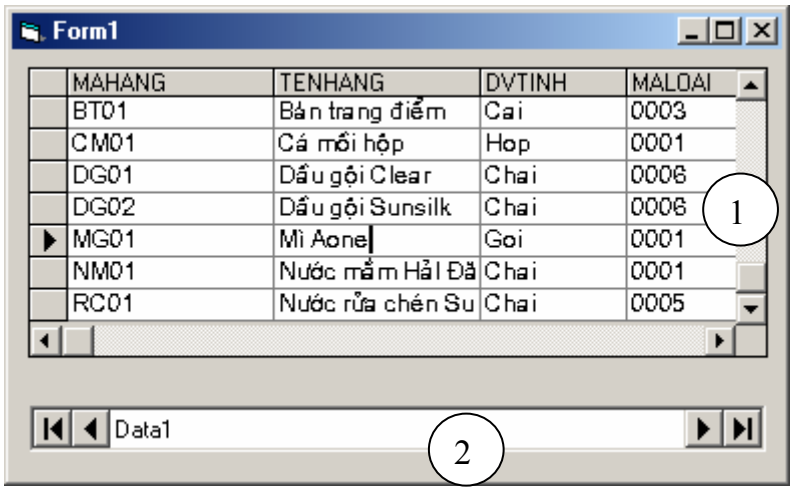
Ngoài ra ta còn có thêm tùy chọn sử dụng ODBCDIRECT để truy cập dữ liệu Client/Server. Đây là một chuyển đổi để truy cập cơ sở dữ liệu server trực tiếp thông qua DAO mà không cần nạp bộ máy cơ sở dữ liệu Jet. Tùy chọn này thích hợp khi ta phải dùng DAO để truy cập dữ liệu Client/Server mà không cần bận tâm về tính linh hoạt, khả năng sử dụng lại và tính dễ bảo trì của chương trình.

Ví dụ: Tạo một Form có sử dụng ODBCDirect và DAO Data Control.

o Tạo dự án mới.

o Tham chiếu đến điều khiển lưới Microsoft Data Bound Grid Control trong mục Project\Components.

o Tạo Form có dạng sau:



Hình 10.4: Ví dụ về ODBC Direct

1: DBGrid. Name: dbGrid1.
2: Data Control. Name: Data1.

o Đổi thuộc tính DefaultType của DataControl là 1 – Use ODBC. Điều này làm cho chương trình thực hiên nhanh hơn.

o Thuộc tính Connect của Data1 là: ODBC;DSN=DBHH

o Đặt thuộc tính Record Source của Data Control:

Select * From THANGHOA

Đặt thuộc tính Data Source của DBGrid1 là: Data1.

Lưu dự án và thực thi chương trình.

II. Truy cập dữ liệu dùng điều khiển dữ liệu từ xa

Điều khiển dữ liệu từ xa (Remote Data Control - RDC) là một cách khác truy cập dữ liệu từ xa trong các ứng dụng Visual Basic. Điều khiển này dùng một giao diện lập trình tương tự như điều khiển ADO Data hay DAO Data.

Ví dụ: Sử dụng RDC để hiển thị dữ liệu trên lưới:

- Ở đây ta có sử dụng Remote Data Control và lưới hiển thị dữ liệu, do đó ta tham chiếu đến các thành phần này bằng cách chọn Project\Components…, thiết lập tham chiếu đến Microsoft Remote Data Control và Microsoft Data Bound Grid Control. Nhấp OK.
- Thiết kế Form có dạng sau:

Hình 10.5: Ví dụ về Remote Data Control

1: RemoteDataControl. Name: rdcHangHoa
2: DBGrid. Name: dbgHangHoa.

- Đặt thuộc tính DataSourceName của điều khiển rdcHangHoa là DBHH (DSN đã tạo trước đây).
- Định thuộc tính SQL của điều khiển rdcHangHoa là:

  Select * From THANGHOA
- Chỉ định thuộc tính DataSource của điều khiển dbgHangHoa là rdcHangHoa.
- Thực thi chương trình.

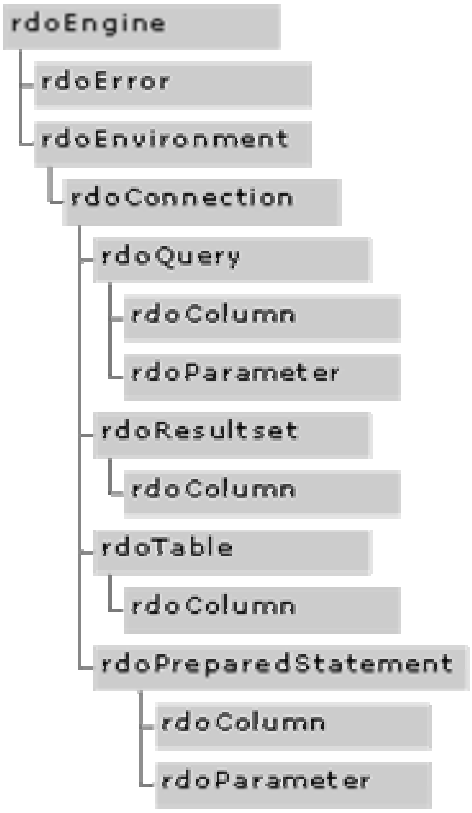
III. Remote Data Object (RDO)

Đối tượng dữ liệu từ xa (Remote Data Objects - RDO) được mô tả dạng hình cây tương tự như DAO. Tuy nhiên mô hình đối tượng RDO đơn giản hơn DAO.

Dù vậy ta không thể dùng RDO để tạo đối tượng cơ sở dữ liệu như bảng, view, thủ tục chứa sẵn…

Để tham chiếu đến RDO ta vào Project\References…\Microsoft Remote Data Object 2.0.

Mô hình đối tượng RDO:



Hình 10.6: Kiến trúc của RDO

III.1 Đối tượng RDOEngine

Đây là đối tượng ở cấp cao nhất trong mô hình RDO, giới thiệu về các thành phần của mô hình. Ta không cần phải khởi tạo đối tượng này cách tường minh bởi vì nó được khởi tạo tự động.

Đối tượng rdoEngine cho phép ta truy cập toàn bộ các đối tượng khác trong mô hình RDO. Ngoài ra, nó còn được sử dụng để tạo mới một Data Source Name (DSN) nhờ vào phương thức rdoRegisterDataSource. Đoạn mã lệnh dưới đây cho phép đăng ký một nguồn dữ liệu mới cho cơ sở dữ liệu Pubs (của SQL Server):

```
Dim strTemp As String
strTemp = "Description=Test DSN" & Chr(13) & _
     "SERVER=(local)" & Chr(13) & "Database=Pubs"
rdoEngine.rdoRegisterDataSource "MyTestPubs", "SQL Server", _
                                True, strTemp
```

III.2 Đối tượng RDOError

Đối tượng lỗi xác định thao tác nào trên một nguồn dữ liệu gây ra lỗi. Đây không phải là các lỗi của Visual Basic vì nó xảy ra đối với cơ sở dữ liệu. Vì thế, ta

không cần phải bẫy lỗi cho các lỗi này. Thay vào đó, tập hợp rdoErrors sẽ xác định thao tác của ta là thành công hay thất bại. Chẳng hạn như đoạn mã lệnh sau:

```
Dim objError As RDO.rdoError
Dim strError As String
For Each objError In rdoEngine.rdoErrors
    strError = strError & objError.Description & vbCrLf
Next

        ' Display Errors
MsgBox "The following errors occurred: " & strError
```

III.3 Đối tượng RDOEnvironment

Đối tượng này chỉ ra cách thức bảo mật của cơ sở dữ liệu. Ta sử dụng đối tượng này để xác định định danh người dùng cùng mật khẩu hay thi hành một phiên giao dịch (Transation) trên cơ sở dữ liệu.

Đối tượng này không được khởi tạo trực tiếp, chúng được tạo ra tự động.

III.4 Đối tượng RDOConnection

Phần lớn các chức năng của RDO bắt đầu với đối tượng rdoConnection; đối tượng này cho phép thiết lập một nối kết đến một nguồn dữ liệu ODBC. Khi nguồn dữ liệu đã được định nghĩa, các thuộc tính cùng các phương thức của đối tượng này được sử dụng để giao tiếp với nguồn dữ liệu ấy.

Đoạn mã lệnh đơn giản dưới đây cho phép tạo một nối kết đến nguồn dữ liệu có tên là Biblio.

```
Set m_Connection = New RDO.rdoConnection
m_Connection.Connect = "DSN=Biblio"
m_Connection.EstablishConnection
```

III.4.1 Đối tượng RDOQuery

Đối tượng này được sử dụng để thực thi các câu truy vấn trên nguồn dữ liệu ODBC. Các câu truy vấn này có thể là các câu SQL hay là lời gọi thực thi các thủ tục lưu trữ sẵn trong cơ sở dữ liệu. Nếu là lời gọi các thủ tục lưu trữ sẵn thì tham số của các thủ tục này được xác định nhờ đối tượng rdoParameter.

Hai yếu tố then chốt để xác định một đối tượng rdoQuery là *nối kết nào cần truy vấn và câu lệnh SQL để truy vấn dữ liệu*.

Ví dụ: Giả sử ta có đoạn mã lệnh bên trên để tạo nối kết đến nguồn dữ liệu Biblio. Đoạn mã lệnh dưới đây thực thi câu lệnh SQL lấy về thông tin về tất cả các nhà xuất bản:

```
' Tạo câu SQL
Dim strSQL As String
strSQL = "SELECT * FROM Publishers"
' Tạo đối tượng Query
Dim m_Query As RDO.rdoQuery
Set m_Query = New RDO.rdoQuery
Set m_Query.ActiveConnection = m_Connection
```

```
m_Query.SQL = strSQL
m_Query.Excute
```

III.4.2 Đối tượng RDOResultset

Đối tượng này quản lý các mẩu tin được trả về từ một nối kết qua ODBC. Tập các mẩu tin có thể được trả về thông qua đối tượng rdoQuery hay nhờ phương thức OpenResultset của một đối tượng rdoConnection. Trong một số trường hợp, tập các mẩu tin được truy xuất nhờ vào thuộc tính LastQueryResults của đối tượng rdoConnection:

```
Set m_Resultset = m_Connection.LastQueryResults
```

Khi đối tượng rdoResultset được tạo ra, ta có thể di chuyển đến mẩu tin xác định bằng các phương thức MoveFirst, MoveLast, MoveNext, MovePrevious.

III.4.3 Đối tượng RDOTable

Đối tượng này xác định một bảng trong một nguồn dữ liệu ODBC. Đối tượng này không được sử dụng cho các thao tác truy xuất dữ liệu mà nó chỉ được sử dụng khi ta cần xác định cấu trúc các bảng của cơ sở dữ liệu của ta.

III.4.4 Đối tượng RDOParameter

Đối tượng này xác định các tham số đầu vào hay các kết quả nhận được từ các thủ tục lưu trữ sẵn trong cơ sở dữ liệu.

III.5 Sử dụng RDO trong chương trình

III.5.1 Thiết lập kết nối đến nguồn dữ liệu

- Trước tiên ta phải tham chiếu đến mô hình đối tượng RDO.
- Nối kết đến nguồn dữ liệu nhờ đối tượng rdoConnection.

  - Khai báo một biến đối tượng rdoConnection.

  - Khởi tao đối tượng, sau đó thiết lập các thuộc tính và các phương thức thích hợp để hoàn tất kết nối; trong đó có hai thuộc tính cần quan tâm là chuỗi kết nối (ConnectionString) và loại con trỏ (Cursor Driver).

- Thiết lập nối kết nhờ phương thức EstablishConnection của đối tượng rdoConnection.

III.5.2 Chuỗi kết nối (ODBC Connect String)

Chuỗi kết nối được sử dụng khi thiết lập nối kết đến nguồn dữ liệu. Đây là một chuỗi (String) xác định các thông tin quan trọng gởi đến trình điều khiển ODBC để truy cập dữ liệu. Các thông tin cần quan tâm: tên nguồn dữ liệu (DSN Name), User ID, Password.

Các tham số của chuỗi kết nối ODBC:

Tham số	Ý nghĩa
DSN	Tên nguồn dữ liệu ODBC
UID	Tên người dùng cơ sở dữ liệu
PWD	Mật khẩu truy cập
DRIVER	Trình điều khiển DBC

DATABASE	Tên của cơ sở dữ liệu được nối kết
SERVER	Tên máy chứa cơ sở dữ liệu phục vụ (database server)
WSID	Tên máy chứa cơ sở dữ liệu khách (database client)
APP	Tên của tập tin chương trình (*.exe)

Chuỗi kết nối được xác lập nhờ thuộc tính Connect của đối tượng rdoConnection.

Ví dụ: Chuỗi nối kết đến 1 DSN của cơ sở dữ liệu Access Biblio:

```
Set m_Connection = New RDO.rdoConnection
m_Connection.Connect = "DSN=Biblo"
m_Connection.EstablishConnection
```

Chuỗi nối kết đến DSN Publications của cơ sở dữ liệu SQL Server:

```
Set m_Connection = New RDO.rdoConnection
m_Connection.Connect = "DSN=Biblo;UID=sa;PWD=;"
m_Connection.EstablishConnection
```

Bên cạnh nối kết chuẩn thông qua ODBC, RDO cũng hỗ trợ loại nối kết DSN cấp thấp (DSN-less connection). Đối với loại này, ta không cần định nghĩa tên nguồn dữ liệu (DSN) trong bộ quản trị ODBC của Windows. Lúc này tất cả các thông tin về cơ sở dữ liệu được cung cấp đầy đủ trong chuỗi nối kết.

Ví dụ: Chuỗi nối kết cấp thấp đến cơ sở dữ liệu SQL Server Pubs:

```
Set m_Connection = New RDO.rdoConnection
m_Connection.Connect ="DRIVER={SQL Server};" & _
   "SERVER=(local);DATABASE=Pubs;UID=admin;PWD=;DSN=;"
m_Connection.EstablishConnection
```

III.5.3 Trình điều khiển con trỏ (Cursor Drivers)

Trình điều khiển con trỏ xác định cách thức tập các mẩu tin trả về từ cơ sở dữ liệu được lưu trữ như thế nào, có thể chúng được lưu trữ tại máy chủ (server) hay máy trạm (client).

Trình điều khiển con trỏ được thiết lập nhờ thuộc tính CursorDriver của đối tượng Connection (trước khi gọi thực thi phương thức EstablishConnection) và các giá trị sau có thể đề cử cho nó:

Giá trị	Ý nghĩa
rdUseIfNeeded	Trình điều khiển ODBC tự động xác định loại con trỏ (được lưu phía server hay client). Nếu có thể, con trỏ loại được lưu phía server được đề cử.
rdUseODBC	Xác định loại con trỏ của ODBC chuẩn, nghĩa là tập tất cả các mẩu tin được lưu ở máy client.
rdUseServer	Tập các mẩu tin được lưu phia server. Tuy nhiên loại con trỏ này không thích hợp lắm trong môi trường đa người dùng.

rdUseClientBatch	Giống như rdUseODBC nhưng có thể được cập nhật đồng thời.
rdUseNone	Không sử dụng con trỏ, tập các mẩu tin được trả về một lần duy nhất lúc chúng được yêu cầu. Ta chỉ có thể di chuyển tới trong tập các mẩu tin kết quả.

Chương 11: ĐỐI TƯỢNG DỮ LIỆU ACTIVEX (ACTIVEX DATA OBJECTS)

Mục tiêu:

Chương này giới thiệu về thư viện ActiveX Data Object (ADO), thư viện đối tượng được sử dụng nhiều nhất trong các ứng dụng truy cập cơ sở dữ liệu dạng khách/chủ (Client/Server) hiện nay.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Kiến trúc OLEDB/ADO.
- Cây phân cấp của mô hình đối tượng ADO.
- Sử dụng thư viện đối tượng ADO để tương tác với cơ sở dữ liệu trong VB.

Kiến thức có liên quan:

- Các cấu trúc lập trình trong VB.
- Câu lệnh truy vấn dữ liệu trong cơ sở dữ liệu.
- Nắm bắt được các mô hình DAO, RDO là một lợi thế vì lúc đó việc tiếp thu mô hình ADO được nhanh hơn.

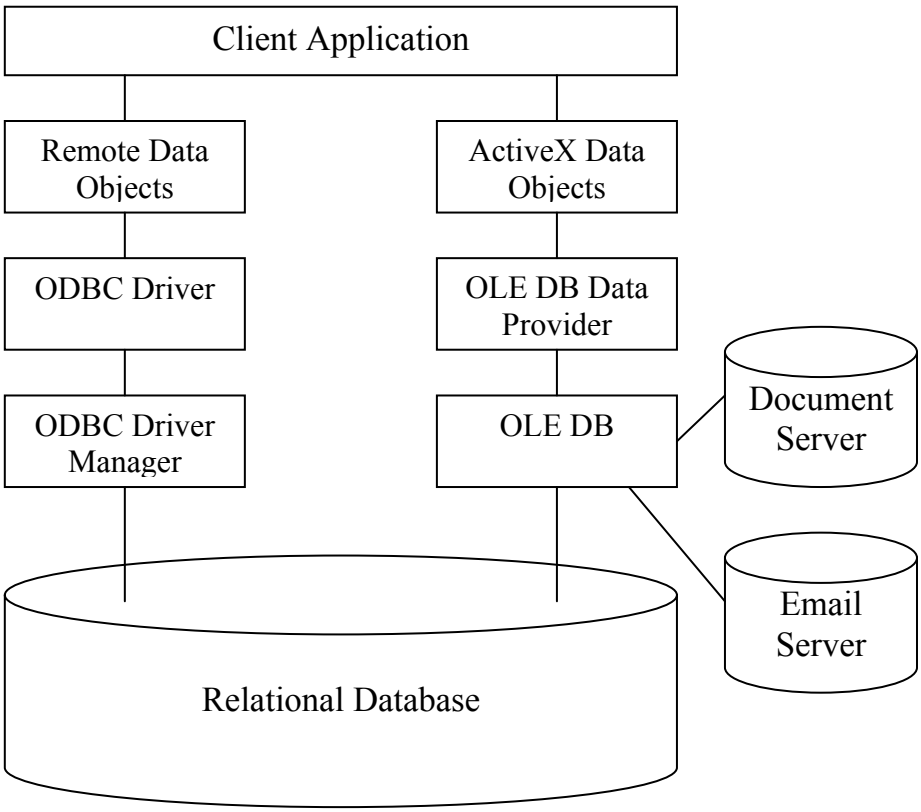
Tài liệu tham khảo:

- **Microsoft Visual Basic 6.0 & Lập trình cơ sở dữ liệu** - Chương 27, trang 877 - **Nguyễn Thị Ngọc Mai** (chủ biên) – **Nhà xuất bản Giáo dục** - **2000**.
- **Tự học Lập trình cơ sở dữ liệu với Visual Basic 6 trong 21 ngày** (T2) – Chương 18, trang 277 - **Nguyễn Đình Tê** (chủ biên) - **Nhà xuất bản Giáo dục** - **2001**.

ADO (ActiveX Data Objects) là công nghệ truy cập cơ sở dữ liệu hướng đối tượng tương tự như DAO. Hiện nay, ADO được Microsoft xem kỹ thuật chính để truy cập dữ liệu từ Web Server.

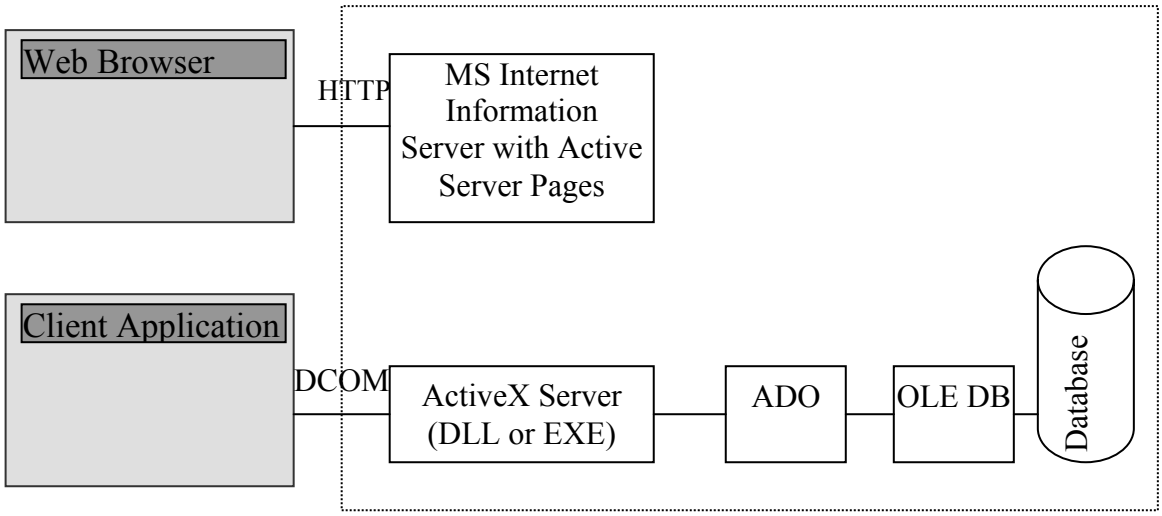
I. Kiến trúc OLE DB/ADO

ADO sử dụng OLEDB như là trình cung cấp dữ liệu cơ sở. Trình cung cấp OLE DB cho phép người lập trình có thể truy xuất dữ liệu từ cả hai nguồn: quan hệ và phi quan hệ. VB6.0 đã hỗ trợ các trình cung cấp cục bộ cho SQL Server, Oracle và Microsoft Jet/Access.



Hình 11.1: Mô hình lập trình CSDL Client - Server dùng RDO và ADO

Ta chỉ cần lập trình với phân giao diện người sử dụng ở phía Client. Việc truy cập cơ sở dữ liệu trên trình duyệt Web hay ứng dụng VB được thực hiện nhờ ADO. Cấu trúc này cho phép ta lập trình một cách nhất quán trên Web cũng như trên ứng dụng.

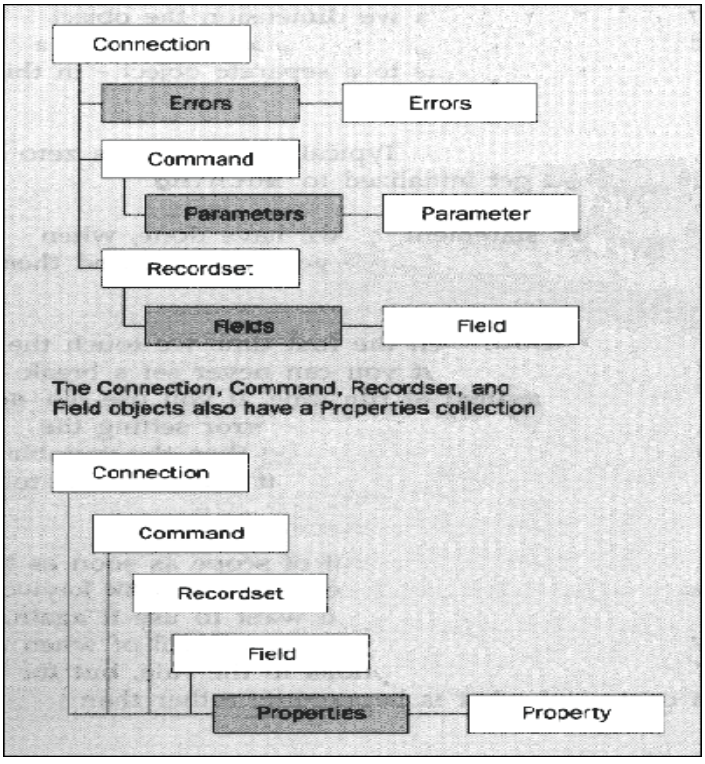


Hình 11.2: Truy cập CSDL từ trình ứng dụng & trình duyệt WEB theo ADO

II. Mô hình ADO

Mô hình ADO được trình bày theo dạng phân cấp (tương tự DAO và RDO).

Để có thể lập trình với thư viện ADO, ta phải tham chiếu đến thư viện này bằng cách chọn Project\References…\Microsoft ActiveX Data Object 2.0.



Hình 11.3: Mô hình ADO

Mô hình ADO có 3 đối tượng cốt lõi:

- Connection: kết nối CSDL thật sự.
- Command: thực thi các câu truy vấn dựa vào kết nối dữ liệu.
- RecordSet: là tập các mẩu tin được chọn từ câu truy vấn thông qua đối tượng Command.

III Các đối tượng trong mô hình ADO

III.1 Đối tượng Connection

Dùng phương thức Open của đối tượng Connection để thiết lập kết nối với nguồn dữ liệu. Để thực hiện điều này ta cần phải thông báo với ADO thông tin kết nối với dạng chuỗi theo kiểu chuỗi kết nối của ODBC. Thuộc tính ConnectionString thực hiện điều này. Ngoài ra ta còn có thể chọn trình cung cấp bằng cách quy định giá trị của thuộc tính Provider của đối tượng.

Để nối kết với dữ liệu, ta cần xác định trình cung cấp OLE DB và chuỗi kết nối. Nếu không xác định được hai yếu tố này, ta sẽ sử dụng trình cung cấp mặc định là ODBC: MSDASQL.

Một số trình cung cấp có sẵn:

- ✓ Microsoft OLEDB cho các trình điều khiển ODBC.
- ✓ Microsoft OLEDB cho Oracle.
- ✓ Microsoft Jet 3.51 OLEDB (Access).
- ✓ Microsoft Jet 4.0 OLEDB (Access)
- ✓ Microsoft OLEDB cho SQL Server.
- ✓ Microsoft OLEDB cho các dịch vụ thư mục.

Ví dụ:

Đối với trình cung cấp ODBC, thuộc tính ConnectionString có thể là một DSN hay là một kết nối không có DSN (DSN cấp thấp).

```
Dim cn As ADODB.Connection
Set cn = New ADODB.Connection
cn.Provider  =  "MSDASQL"
cn.ConnectionString  =  "DSN=Baigiang"
cn.Open
```

Kết nối DSN cấp thấp:

```
Dim cn As ADODB.Connection
Set cn = New ADODB.Connection
cn.Provider = "MSDASQL"
cn.ConnectionString = "DRIVER={SQL Server};" & _
   "DATABASE=Baigiang;UID=myuser;PWD=mypassword;"
cn.Open
```

Trong trường hợp này việc kết nối với cơ sở dữ liệu Server được thực hiện nhanh hơn vì chương trình không cần đọc thông tin về các DSN trên máy Client, tuy nhiên thông tin về nguồn cơ sở dữ liệu lại kết chặt với chương trình đã biên dịch.

Để kết nối với cơ sở dữ liệu Access, ta dùng trình cung cấp Jet với chuỗi kết nối là đường dẫn đến tập tin .mdb

```
Dim cn As ADODB.Connection
Set cn = New ADODB.Connection
cn.Provider = "MicroSoft.Jet.OLEDB.4.0"
cn.ConnectionString = "d:\data\baigiang.mdb"
```

```
cn.Open
```

Đối với cơ sở dữ liệu SQL Server, ta có thể dùng trình cung cấp SQLOLEDB.1, trong trường hợp này, chuỗi kết nối tương tự như trường hợp kết nối dùng trình cung cấp ODBC không có DSN, tuy nhiên ta không cần xác định giá trị của DRIVER:

```
Dim cn as ADODB.Connection
Set cn = New ADODB.Connection
cn.Provider = "SQLOLEDB.1"
cn.ConnectionString = "DATABASE=DBHH;" & _
      "SERVER=www;UID=user;PWD=user"
cn.Open
```

Mở và đóng nối kết nguồn dữ liệu

Để phát các yêu cầu đến nguồn dữ liệu sử dụng ADO, ta cần mở kết nối đến nguồn dữ liệu đó bằng phương thức Open của đối tượng Connection. Cú pháp đầy đủ như sau:

connection.Open [connect], [userid], [password]

Tất cả các tham số của phương thức Open đều là tùy chọn, nếu như các thông số này đã được xác định thông qua các thuộc tính khác của đối tượng Connection thì ta không cần mô tả chúng ở đây.

Khi đã hoàn thành tất cả các thao tác liên quan đến nối kết này, ta cần phải đóng nối kết một cách tường minh thông qua phương thức Close của đối tượng Connection.

connection.Close

Đóng nối kết một cách tường minh sẽ đảm bảo rằng tất cả các tài nguyên liên quan đến nối kết này trên Server cũng như Client đều được giải phóng một cách hợp lý.

Xác định vị trí con trỏ

Con trỏ (Cursor): một tập các mẩu tin được trả về cho chương trình. Vị trí con trỏ được xác định nhờ thuộc tính CursorLocation (có ở cả đối tượng Recordset). Có 2 giá trị có thể chỉ định:

- ✓ adUseClient: con trỏ phía Client.
- ✓ adUseServer: con trỏ phía Server (mặc định).

Thực thi các câu truy vấn hành động

Các câu truy vấn hành động (Insert, Update, Delete) được thực hiện nhờ phương thức Execute của đối tượng Connection; ngoài ra phương thức này cũng có thể được sử dụng để thực thi các thủ tục lưu trữ sẵn trong cơ sở dữ liệu hay các câu SELECT. Cú pháp phương thức này như sau:

Nếu không có kết quả trả về:

connection.Execute CommandText, RecordsAffected, Options

Có kết quả trả về:

***Set** recordset = connection.Execute (CommandText, RecordsAffected, Options)*

Trong đó:

- connection: Đối tượng Connection.

- recordset: Đối tượng Recordset là kết quả trả về của phương thức Execute, tuy nhiên, người ta thường ít khi sử dụng cách này. Thay vào đó, người ta thường sử dụng phương thức Open của đối tượng Recordset.

- CommandText: là một chuỗi xác định câu truy vấn hành động, SELECT, thủ tục lưu trữ sẵn hay tên một bảng trong cơ sở dữ liệu.

- RecordEffected: Tùy chọn, là một số nguyên dài (Long) xác định trình cung cấp trả về bao nhiêu mẩu tin thỏa điều kiện.

- Options: Tùy chọn, là một số nguyên dài (Long) xác định trình cung cấp sẽ đánh giá các đối số của CommandText như thế nào.

Thuộc tính Mode: Xác định trình cung cấp có thể hạn chế truy cập đến cơ sở dữ liệu khi có một recordset đang mở. Các giá trị có thể là:

Hằng số	Giá trị	Ý nghĩa
adModeUnknown	0	Mặc định, chỉ định quyền hạn chưa thiết lập hay không thể xác định
adModeRead	1	Mở Recordset với quyền chỉ đọc
adModeWrite	2	Mở Recordset với quyền chỉ ghi
adModeReadWrite	3	Mở Recordset với quyền đọc/ghi
adModeShareDenyRead	4	Ngăn người khác mở kết nối với quyền chỉ đọc
adModeShareDenyWrite	8	Ngăn người khác mở kết nối với quyền chỉ ghi
adModeShareExclusive	12	Ngăn người khác mở kết nối
adModeShareDenyNone	16	Ngăn người khác mở kết nối với bất cứ quyền nào

III.2 Đối tượng Recordset

Để có thể khởi tạo một đối tượng Recordset ta có thể thực hiện một trong hai cách:

- Phương thức Execute của đối tượng Connection. Tuy nhiên cách này ta chỉ tạo được các Recordset chỉ đọc và chỉ có thể di chuyển tới.

- Xác lập các thông số thích hợp cho đối tượng Recordset rồi thực thi phương thức Open của đối tượng Recordset. Điều này được thực hiện nhờ các bước:

  ✓ *Sau khi khởi tạo đối tượng Connection, chỉ định Recordset là của đối tượng Connection trên.*
  ✓ *Thiết lập các thuộc tính thích hợp của Recordset (Source, LockType…).*
  ✓ *Thực thi câu truy vấn nối kết nhờ phương thức Open.*

III.2.1 Thuộc tính CursorType (loại con trỏ)

Xác định loại con trỏ được trả về từ cơ sở dữ liệu. Các giá trị có thể nhận:

Hằng	Giá trị	Mô tả
adOpenForwardOnly	0	Chỉ có thể di chuyển phía trước
adOpenKeyset	1	Không thể thấy các mẩu tin do người dùng khác thêm vào nhưng khi họ xóa hay sửa đổi mẩu tin sẽ làm ảnh hưởng đến các mẩu tin ta đang làm việc.
adOpenDynamic	2	Có thể thấy toàn bộ sự thay đổi do người

		dùng khác tác động.
adOpenStatic	3	Bản sao tĩnh của tập mẩu tin. Mọi sự thay đổi của người dùng khác ta không thấy được

III.2.2 Thuộc tính LockType (khóa mẩu tin)

Xác định cách thức khóa mẩu tin trong Recordset. Dùng tính năng này khi muốn kiểm soát cách thức cập nhật mẩu tin với nhiều người dùng trong cơ sở dữ liệu.

Hằng	**Giá trị**	Mô tả
adLockReadOnly	1	Mặc định - Chỉ đọc.
adLockPessimistic	2	Khóa trang bi quan. Mẩu tin trong RecordSet bị khóa khi bắt đầu sửa đổi & tiếp tục khóa cho đến khi thi hành phương thức *Update* hay di chuyển sang mẩu tin khác.
adLockOptimistic	3	Khóa trang lạc quan. Mẩu tin chỉ bị khóa ngay lúc thi hành phương thức *Update* hay di chuyển sang mẩu tin khác.
adLockBatchOptimistic	4	Khóa trang lạc quan hàng loạt. Hỗ trợ cập nhật nhiều mẩu tin cùng một lúc.

III.2.3 Thuộc tính Source

Đây là một chuỗi xác định câu truy vấn để lấy dữ liệu, có thể là tên của bảng hay tên của thủ tục lưu trữ sẵn.

III.2.4 Thuộc tính ActiveConnection

Đây là một thuộc tính đối tượng xác định Recordset là của nối kết nào trong chương trình.

III.2.5 Ví dụ sử dụng đối tượng Recordset trong chương trình

Đối tượng Recordset có thể được sử dụng là đối tượng nguồn dữ liệu (DataSource) của điều khiển lưới: *Microsoft DataGrid Control 6.0 (OLEDB)*. Nhờ điều khiển lưới này ta có thể hiển thị dữ liệu từ một Recordset theo dạng hàng và cột.

Chẳng hạn ta có thể hiển thị trên lưới thông tin về các mặt hàng cùng với mã loại hàng của nó:

```
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset

Private Sub Form_Load()
    Set cn = New ADODB.Connection
    cn.Provider = "Microsoft.Jet.OLEDB.3.51"
    cn.ConnectionString = "F:\Data\DBHH.mdb"
    cn.Open
        Set rs = New ADODB.Recordset
    rs.Source = "SELECT MaHang, TenHang, DVTinh" & _
     "TenLoai FROM THangHoa, TLoaiHang WHERE " & _
     "THangHoa.MaLoai = TLoaiHang.MaLoai"
    Set rs.ActiveConnection = cn
    rs.CursorLocation = adUseClient
```

```
        rs.Open
        Set grdHH.DataSource = rs
End Sub
```

Kết quả thực thi của chương trình này như sau:

*: M

Hình 11.4: Sử dụng Datagrid để hiển thị dữ liêu từ Recordset

III.2.6 Cập nhật và thêm mới mẩu tin

Thêm mới mẩu tin

- Mở Recordset
- Thi hành phương thức AddNew
- Gán giá trị cho các trường trong mẩu tin của Recordset
- Lưu lại mẩu tin bằng cách thi hành phương thức Update (hay UpdateBatch).

Cập nhật mẩu tin

- Mở Recordset
- Thực hiện câu lệnh truy vấn để nhận về các mẩu tin thích hợp.
- Di chuyển đến mẩu tin cần cập nhật lại giá trị.
- Gán lại giá trị cho các trường.
- Thi hành phương thức Update (hay UpdateBatch tùy thuộc vào LockType).

Lưu ý:Chế độ khóa mẩu tin mặc định trong ADO là chỉ đọc, vì vậy ta phải đổi thuộc tính LockType của đối tượng Recordset sang chế độ soạn thảo trước khi thi hành cập nhật hay thêm mới mẩu tin.

III.2.7 Thuộc tính CursorLocation

Xác định tập mẩu tin trả về từ cơ sở dữ liệu được lưu ở đâu (Server hay Client, Server là mặc định). Thuộc tính cũng giống thuộc tính CursorLocation của đối tượng Connection.

III.2.8 Recordset ngắt kết nối

Khi chúng ta dùng con trỏ phía Client, ta có khả năng ngắt kết nối với Server cơ sở dữ liệu mà vẫn tiếp tục làm việc với dữ liệu. Cách này cho phép ứng dụng trở nên

linh hoạt hơn bởi vì nhiều người dùng có thể làm việc với cùng một dữ liệu tại một thời điểm nếu như họ không có nối kết với server.

Để ngắt nối kết với Server, ta quy định thuộc tính ActiveConnection của đối tượng Recordset là Nothing.

Ví dụ:

```
Dim cn As ADODB.Connection

Private Sub Form_Load()
    Set cn = New ADODB.Connection
    cn.Provider = "Microsoft.Jet.OLEDB.3.51"
    cn.ConnectionString = "F:\Data\GiangDay.mdb"
    cn.Open
End Sub
Public Function GetList (strState As String) _
    As ADODB.Recordset
     Dim rs  As ADODB.Recordset
     Set rs = New ADODB.Recordset
     Set rs.ActiveConnection = cn

     rs.CursorLocation = adUseClient
     rs.LockType = adLockBatchOptimistic
     rs.CursorType = adOpenKeyset
     rs.Open strState

     Set rs.ActiveConnection = Nothing
     Set GetList  = rs
     Set rs = Nothing
End Function
```

Để thi hành cùng một hành động trên một mẩu tin, ta sửa đổi lại các thuộc tính của đối tượng Recordset.

```
rs.LockType = adLockBatchOptimistic
rs.CursorType = adOpenKeyset
```

Chúng ta thiết lập giá trị các thuộc tính lại như trên để xác nhận rằng Recordset có thể nối kết lại để cập nhật về sau.

Sau đó, ta sẽ thiết lập một hàm nhận Recordset ngắt kết nối làm tham biến để tạo một đối tượng Recordset khác cập nhật dữ liệu.

```
Public Sub WriteData(rsDis As ADODB.Recordset)
        Dim rs As ADODB.Recordset
        Set rs = New ADODB.Recordset
        Set rs.ActiveConnection = cn

        rs.Open rsDis, cn
        rs.UpdateBatch
End Sub
```

Gọi thực thi thủ tục WriteData:

```
Private Sub cmdWrite_Click()
        WriteData GetList("Select * From THanghoa")
End Sub
```

III.3 Đối tượng Command

Đây là đối tượng được người lập trình sử dụng khi muốn thi hành các thủ tục lưu trữ sẵn hay những câu truy vấn có tham số.

Với đối tượng Command ta có thể thi hành một số công việc như sau:

- Sử dụng thuộc tính CommandText để định nghĩa các đoạn Text thi hành được. Thông thường thuộc tính này dùng để thiết lập một câu lệnh SQL hoặc một lời gọi thủ tục lưu trữ sẵn, hay những dạng khác mà trình cung cấp hỗ trợ
- Xây dựng chuỗi các đối số của câu truy vấn cũng như các tham số của các thủ tục lưu trữ sẵn thông qua đối tượng Parameter hoặc tập hợp Parameters.
- Thực hiện một câu truy vấn và trả về đối tượng Recordset thông qua phương thức Execute.
- Xác định kiểu của đối tượng Command để nâng cao hiệu quả thông qua thuộc tính CommandType.
- Xác định số giây mà trình cung cấp phải chờ khi thi hành một đối tượng Command thông qua thuộc tính CommandTimeOut.

Các kiểu của đối tượng Command được trình bày trong bảng dưới đây:

Hằng	Ý nghĩa
adCmdText	Định giá thuộc tính CommandText dưới dạng Text của một câu lệnh hoặc một lời gọi thủ tục lưu trữ sẵn.
adCmdTable	Định giá thuộc tính CommandText như là tên của một bảng khi tất cả các trường của bảng đó sẽ được trả về bởi câu lệnh truy vấn nội tại.
adCmdTableDirect	Định giá thuộc tính CommandText như là tên của một bảng khi mà tất cả các trường của bảng đó sẽ được trả về.
adCmdStoredProc	Định giá thuộc tính CommandText như là tên của một thủ tục lưu trữ sẵn.
adExecuteNoRecords	Chỉ định rằng thuộc tính CommandText là một câu lệnh hoặc một thủ tục lưu trữ sẵn không trả về bất kỳ dòng nào (ví dụ như lệnh thêm mới dữ liệu …). Cấu trúc này luôn bao hàm adCmdText, adCmdStoredProc.

Thuộc tính Parameter được xác lập thông qua hai phương thức CreateParameter và Append

Set parameter = command.CreateParameter (Name, Type, _
Direction, Size, Value)

- ✓ Name: tùy chọn, chuỗi xác định tên của đối tượng Parameter.
- ✓ Type, Direction: giá trị xác đin kiểu của đối tượng Parameter

✓ Size: giá trị xác định độ dài tối đa của giá trị đối tượng Parameter bằng ký tự hoặc Byte.

✓ Value: biến xác định giá trị của Parameter truyền.

Những giá trị có thể của thuộc tính Direction:

Hằng	Mô tả
adParamUnknown	Không biết chiều của Parameter.
adParamInput	Mặc định, xác định đây là tham số đầu vào.
adParamOutput	Tham số đầu ra.
adParamInputOutput	Vừa là tham số đầu vào vừa là tham số đầu ra.
adParamReturnValue	Đây là giá trị trả về.

Phương thức Append dùng để đưa đối tượng Parameter vừa tạo vào tập hợp. Chúng ta sẽ xét qua ví dụ dau đây:

```
Public Sub ActiveConnectionX()
      Dim cnn1 As ADODB.Connection
            Dim  cmdByRoyalty As ADODB.Command
            Dim  prmByRoyalty As ADODB.Parameter
            Dim  rstByRoyalty As ADODB.Recordset
            Dim  rstAuthors As ADODB.Recordset
            Dim  intRoyalty As Integer
            Dim  strAuthorID As String
            Dim  strCnn As String

   ' Định nghĩa 1 đối tượng command cho một thủ tục lưu trữ sẵn
            Set cnn1 =  New ADODB.Connection
            cnn1.Provider = "SQLOLEDB.1"
            cnn1.ConnectionString = "DATABASE=Pubs;" & _
                  "SERVER=(local);UID=user;PWD=user;"

            cnn1.Open

            Set cmdByRoyalty = New ADODB.Command
            Set cmdByRoyalty.ActiveConnection = cnn1
            cmdByRoyalty.CommandText = "byroyalty"
            cmdByRoyalty.CommandType = adCmdStoredProc
            cmdByRoyalty.CommandTimeout = 15

   ' Định nghĩa đối số đầu vào cho thủ tục lưu trữ
            intRoyalty = Trim(InputBox(  "Enter royalty:"))
            Set prmByRoyalty  = New ADODB.Parameter
```

```
        prmByRoyalty.Type = adInteger
        prmByRoyalty.Size = 3
        prmByRoyalty.Direction = adParamInput
        prmByRoyalty.Value = intRoyalty
        cmdByRoyalty.Parameters.Append prmByRoyalty

' Tạo một recordset bằng cách thi hành đối tượng Command.
        Set rstByRoyalty = cmdByRoyalty.Execute()

' Mở bảng Authors để lấy tên hiển thị
        Set  rstAuthors =  New ADODB.Recordset
        rstAuthors.Open "authors", cnn1, , , adCmdTable
        Debug.Print "Authors with " & intRoyalty & _
              " percent royalty"
        Do  While Not rstByRoyalty.EOF
              strAuthorID = rstByRoyalty!au_id
              Debug.Print , rstByRoyalty!au_id & ", ";
              rstAuthors.Filter = "au_id = '" & _
                    strAuthorID & "'"
              Debug.Print rstAuthors!au_fname & " - " & _
                    rstAuthors!au_lname
              rstByRoyalty.MoveNext
        Loop
        rstByRoyalty.Close
        rstAuthors.Close
        cnn1.Close
  End Sub
```

III.4 Đối tượng Field

Dùng đối tượng Field và tập hợp Fields khi ta muốn truy cập giá trị của một trường của một Recordset nào đó, kỹ thuật này tương tự như đối với DAO.

IV. Dịch vụ dữ liệu từ xa của ADO

Đây là kỹ thuật sử dụng thư viện Remote Data Service (RDS) để vận chuyển ADO Recordset từ server đến máy tính client Recordset kết quả được lưu ở máy client và chúng được ngắt kết nối đến server.

RDS là một phần của Microsoft Data Access Components (MDAC). Các thông tin về RDS có thể tìm thấy ở trang http://www.microsoft.com/data/. RDS gồm 2 phần chính:

- RDS 1.5 server: đi kèm khi cài đặt Internet Information Server (IIS) 4.0.
- RDS 1.5 client đi kèm khi cài đặt Internet Explorer (IE) 4.0.

Thư viện ADODB gồm các thành phần hoạt động chủ yếu phía server (server side) như các đối tượng Connection, Command, Error, Parameters ... Sẽ thật hiệu quả nếu sử dụng các thành phần này giao tiếp với cơ sở dữ liệu. Tuy nhiên trong trường hợp sử dụng các chức năng cần phải có ở phía client thì ta cần phải phân phối kèm theo

một số tập tin và sử dụng ODBC cho mỗi máy client. Đối tượng ADODB Recordset không thể phân phối với các thành phần của RDS Client. Thay vào đó thư viện đối tượng Microsoft ActiveX Data Objects RecordSet (ADOR) được sử dụng. Thư viện này gồm các thành phần hoàn toàn nằm ở phía client và cho phép ta có các thao tác trên một recordset thật sự phía client. ADOR không có các đối tượng Connection, Command, Error, hay Parameters. ADOR có các chức năng cho phép phân phối recordset với các thành phần RDS client.

Một ADO Recordset không thể vận chuyển thông qua giao thức http. Thay vào đó RDS được sử dụng để nhận và tương tác dữ liệu từ xa thông qua http. Một proxy RDS được sử dụng để kiểm soát từ xa một ADOR Recordset ngắt kết nối truyền thông qua giao thức http. Như vậy RDS là vật chứa (container) cho phép lưu trữ và truy cập từ xa các ADOR Recordset.

Ta có thể dùng đối tượng DataControl của RDS để nhận về đối tượng Recordset của ADO từ Internet.

Để có thể sử dụng kỹ thuật này, ta cần tham khảo các thuộc tính chủ yếu của đối tượng DataControl.

- *Thuộc tính Connect*:

DataControl.Connect = "DSN=DSNName;UID=usr;PWD=pw;"

- *Thuộc tính Server*: Xác định máy chủ Web chứa nguồn dữ liệu bao gồm tên và giao thức nối kết.

- *Thuộc tính SQL*: Là câu lệnh truy vấn để nhận về đối tượng Recordset

DataControl.SQL = "QueryString"

- *Thuộc tính ExecuteOptions*: xác định việc thi hành các câu lệnh truy vấn một cách đồng bộ hay không, các giá trị là một trong hai giá trị sau đây:

Hằng	Mô tả
adcExecSync	Thi hành đồng bộ
adcExecAsync	Mặc định, Thi hành không đồng bộ.

- *Thuộc tính ReadyState*: Xác định trạng thái của điều khiển.

Giá trị	Mô tả
adcReadyStateLoaded	Câu truy vấn hiện hành vẫn đang còn thực hiện và chưa có một dòng nào được trả về. Đối tượng Recordset của RDS.DataControl chưa thể sử dụng.
adcReadyStateInteractive	Tập hợp dòng ban đầu đã được trả về và chứa trong đối tượng Recordset, các dòng tiếp theo vẫn đang được trả về.
adcReadyStateComplete	Tất cả các dòng đều đã được chứa trong đối tượng Recordset.

- *Phương thức Refresh*: thi hành câu truy vấn.

- *Thuộc tính Recordset*: trả về Recordset kết quả.

Recordset = DataControl.Recordset

- *Phương thức DoEvents*: Đây là hàm của VB, nó sẽ trả điều khiển cho hệ điều hành thực hiện các quá trình khác.

Chương 12: MÔI TRƯỜNG DỮ LIỆU (DATA ENVIRONMENT)

Mục tiêu:

Chương này giới thiệu cách thức sử dụng môi trường dữ liệu (Data Environment), cách thức tạo các ứng dụng tương tác với cơ sở dữ liệu cách nhanh chóng trong VB 6.0.

Học xong chương này, sinh viên có thể:

- Sử dụng thành thạo môi trường dữ liệu của VB 6.0 để tạo các biểu mẫu nhập liệu.

- Sử dụng thành thạo môi trường dữ liệu để thiết kế các câu truy vấn dữ liệu cách trực quan, điều này làm cho việc thiết kế ứng dụng được tiện lợi hơn.

Kiến thức có liên quan:

- Thư viện đối tượng ADO.

- Câu lệnh SQL để truy vấn dữ liệu.

Tài liệu tham khảo:

- **Tự học Lập trình cơ sở dữ liệu với Visual Basic 6.0 trong 21 ngày** (T1) – Chương 9, trang 395 - **Nguyễn Đình Tê** (chủ biên) - **Nhà xuất bản Giáo dục** - **2001**.

I) GIỚI THIỆU VỀ TRÌNH DATA ENVIRONMENT DESIGNER (DED)

Giới thiệu

Công cụ DED là một giao diện trực quan rất mạnh của Visual Basic để xây dựng các form ràng buộc dữ liệu. DED cho phép ta thao tác với một vài hộp thoại để tạo kết nối đến nguồn dữ liệu (cơ sở dữ liệu) & các nguồn record (dataset hay recordset) một cách nhanh chóng.

Cấu trúc chi tiết của DED

Giao diện DED ActiveX Designer

DED sử dụng một lớp các đối tượng Visual Basic gọi là ActiveX Designers. ActiveX Designers được nạp vào môi trường soạn thảo VB cũng như các mục khác như Menu, Form, Modul… Tuy nhiên cách thức để thao tác trên nó khác với các mục này. Khi thao tác với DED ta sử dụng các mục trên menu của trình soạn thảo DED để xây dựng một tập hoàn chỉnh các kết nối (Connection) & các lệnh dữ liệu (Command) để sử dụng trong chương trình.

Trong lúc thao tác với DED, ta có sử dụng 2 đối tượng khác nhau:

Đối tượng kết nối dữ liệu (Connection): đối tượng này định nghĩa một kết nối giữa chương trình của ta & nguồn dữ liệu.

Đối tượng lệnh dữ liệu (Command): định nghĩa một tập các record lấy ra từ kết nối dữ liệu trên.

Khi xác định đối tượng Command, ta cần chỉ định đối tượng Command này lấy dữ liệu từ đâu trong cơ sở dữ liệu bằng việc xác lập nguồn dữ liệu (Data of Source). Nguồn dữ liệu này xác lập cách thức lấy dữ liệu từ cơ sở dữ liệu: lấy thông qua một TABLE, VIEW, STORED PROCEDURE, SQL…

Mỗi khi một đối tượng Command được tạo ra & được gọi thực thi, một RecordSet của đối tượng Command này cũng được kích hoạt. Lúc này tên của RecordSet tương ứng là:

rs + Tên Command

Ví dụ: Ta tạo một đối tượng Command có tên là comHH, lúc đó tên RecordSet tương ứng là: rscomHH.

Khi đối tượng Command được tạo ra, lúc này nếu muốn thao tác trên đối tượng Command này, ta sẽ thao tác trên RecordSet tương ứng của nó.

Thiết kế các Form ràng buộc dữ liệu không cần mã lệnh:

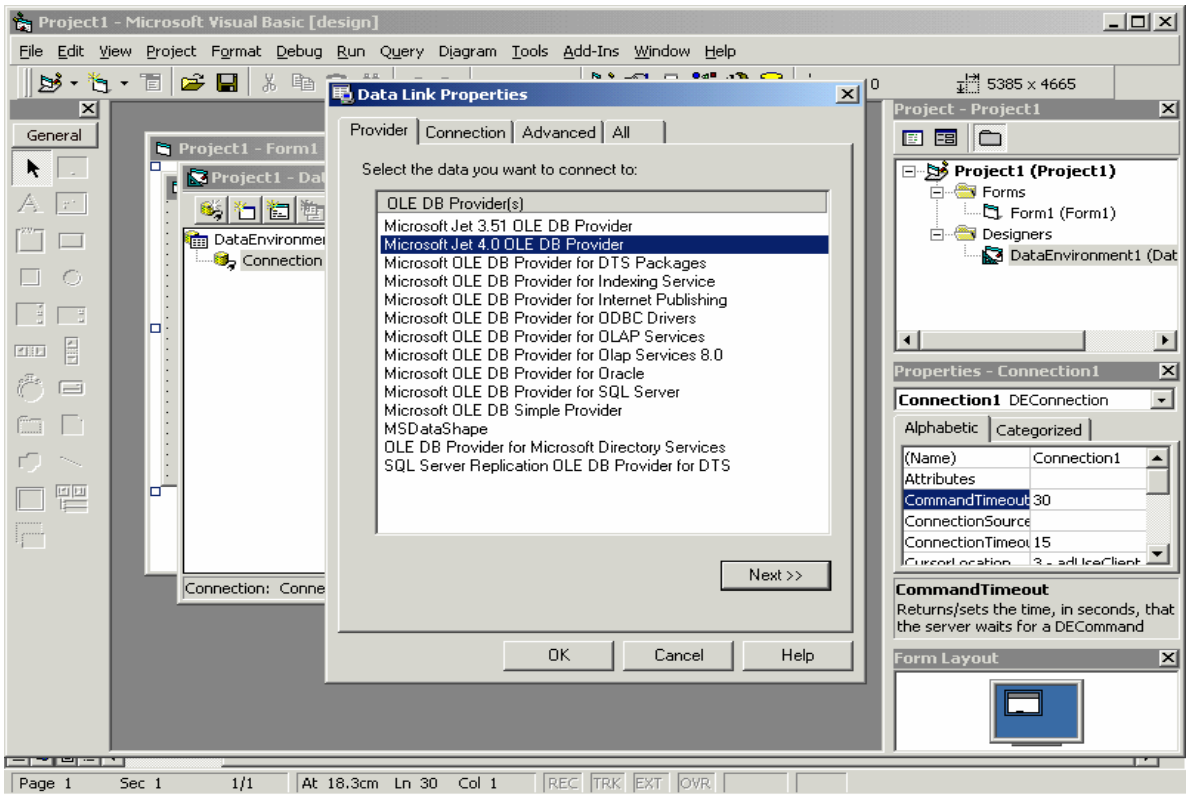
DED cho phép ta thiết kế các Form ràng buộc dữ liệu không cần mã lệnh.

Ví dụ: Với CSDL HangHoa.MDB; ta thiết kế một Form cho Table HANGHOA bằng cách sử dụng trình DED như sau:

Bước 1: Tạo nối kết đến cơ sở dữ liệu HangHoa.MDB:

Nếu mục Data Environment không có sẵn trong Project Explorer, ta chọn Project\Components…, đánh dấu vào mục Data Environment trong tùy chọn Designers, nhấp OK. Chọn Project\More ActiveX Designers… để thêm Data Environment vào môi trường soạn thảo.

Trong đối tượng Connection1, chọn Properties, một cửa sổ hiện lên:



Hình 12.1: Thiết lập nối kết dữ liệu

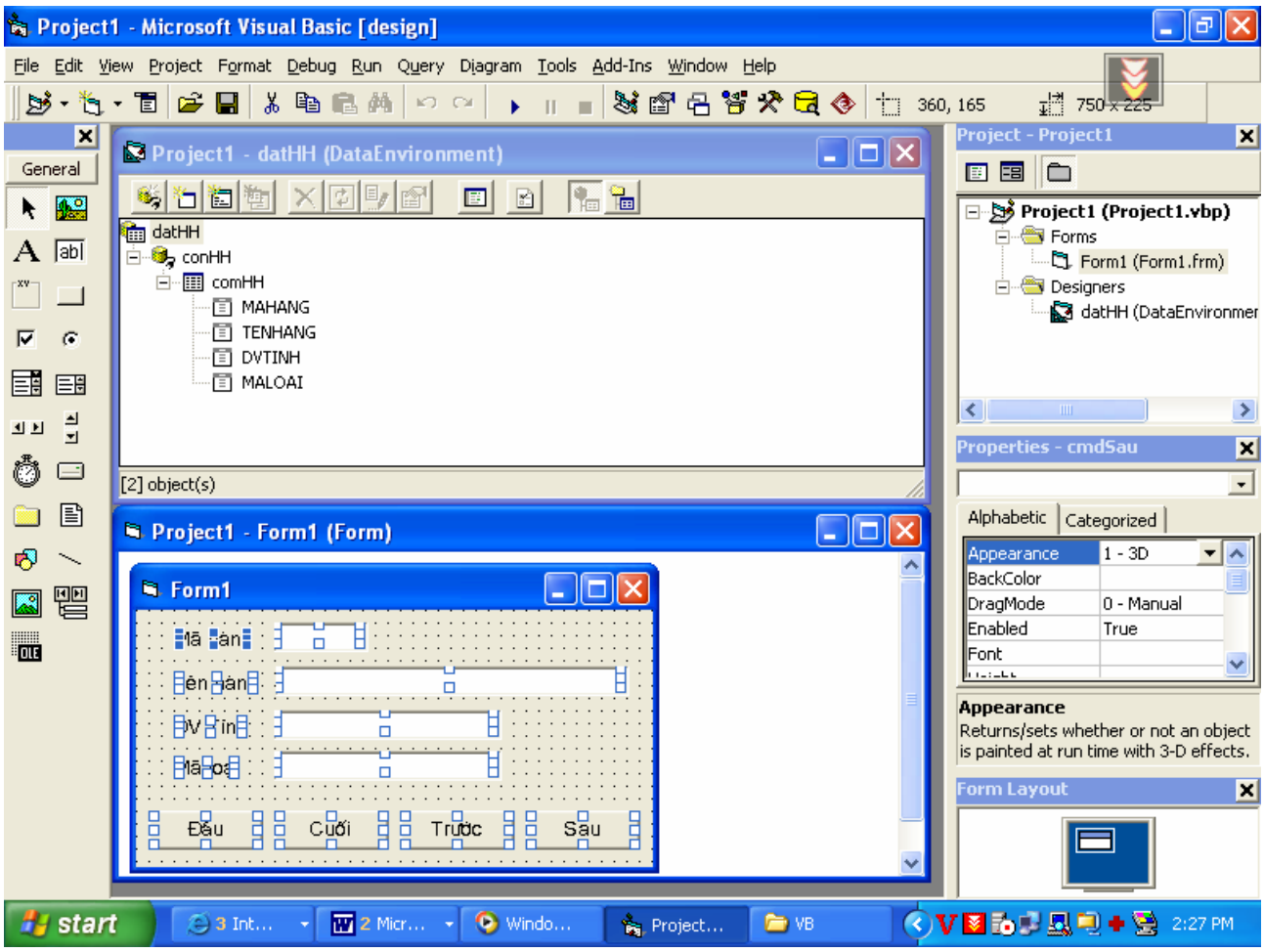
Chọn Microsoft Jet 4.0 OLE DB Provider; chọn Next. Chọn cơ sở dữ liệu ta muốn thao tác trong mục *Select or Enter a Database Name Box*; ở đây ta chọn CSDL HANGHOA.MDB (giả sử nằm trong đường dẫn D:\DED\HangHoa.Mdb). Nhấp nút Test Connection để kiểm tra nối kết với CSDL có bị lỗi hay không? Ta sẽ nhấp OK nếu nối kết này thành công (nếu không ta phải kiểm tra lại).

Bước 2: Thêm đối tượng Command vào DED:

Nhấp chuột phải vào Connection1 và chọn *ADD COMMAND* trên menu, một đối tượng command được tạo ra với tên là Command1 trong Data Environment. Nhấp chuột phải vào đối tượng mới tạo này, chọn Properties để chọn cách thức thao tác đối với đối tượng Command này.

Trong đó, mục Database Object: Table; ObjectName: THANGHOA.

Bước 3: DED cho phép ta kéo một tập các trường (Field) từ một đối tượng dữ liệu trong DED vào 1 Form chuẩn trong Windows và thả nó tại một nơi nào đó.



Hình 12.2: Dùng DED tạo Form ràng buộc dữ liệu

Bước 4: Lúc này khi chạy chương trình, ta thấy Form này đã hoạt động; tuy nhiên ta không thể thấy các nút nhấn điều khiển việc di chuyển các record (Đầu, Cuối, Trước, Sau); hay các nút nhấn hành động (Cập nhật, Thêm, Xóa); ta cần tự bổ sung.

Các thuận lợi của DED:

Xây dựng các Form ràng buộc dữ liệu cách dễ dàng.

Có nhiều tùy chọn để định nghĩa kết nối và các lệnh dữ liệu.

Ta có một giao diện thân thiện hơn để thao tác các kết nối & lệnh.

DED sử dụng ActiveX Data Objects (ADO) để truy cập dữ liệu.

II) SỬ DỤNG TRÌNH DATA ENVIRONMENT DESIGNER

Quá trình xây dựng một chương trình thao tác CSDL với VB thông qua DED bao gồm 3 bước:

- Chọn một trình cung cấp dữ liệu (ODBC hay OLE DB).
- Tạo một kết nối dữ liệu (file MDB, SQL Server…)
- Tạo một lệnh dữ liệu (đối tượng Command).

1) Các trình cung cấp dữ liệu (Data Provider)

- Data Provider là một thành phần điều khiển sự tương tác của chương trình của ta & nguồn dữ liệu. Một trình cung cấp rất quen thuộc là trình cung cấp ODBC (Open Database Connectivity: kết nối cơ sở dữ liệu mở). Giao diện này dựa trên ý tưởng là mọi nguồn dữ liệu có thể được thao tác với ngôn ngữ SQL.

- Một giao diện mới được đưa ra bởi Microsoft: giao diện OLE DB. Giao diện này không yêu cầu nguồn dữ liệu phải nhắm vào việc sử dụng ngôn ngữ truy vấn SQL; thay vào đó, giao diện OLE DB cho phép trình cung cấp dữ liệu chấp nhận ngôn ngữ truy vấn nào mà họ muốn hỗ trợ. Do vậy các nguồn dữ liệu được mở rộng ra từ các CSDL truyền thống: dBase, SQL Server…; đến các nguồn dữ liệu khác như các tập tin, thư mục của hệ điều hành…

- VB 6 gửi kèm với các trình cung cấp dữ liệu như sau:
  - ✓ Microsoft Jet 3.51 OLE DB Provider.
  - ✓ Microsoft Jet 4.0 OLE DB Provider.
  - ✓ Microsoft OLE DB Provider for SQL Server.
  - ✓ Microsoft OLE DB Provider for Oracle.
  - ✓ Microsoft OLE DB Provider for ODBC Drivers.

2) Tạo một kết nối dữ liệu với DED

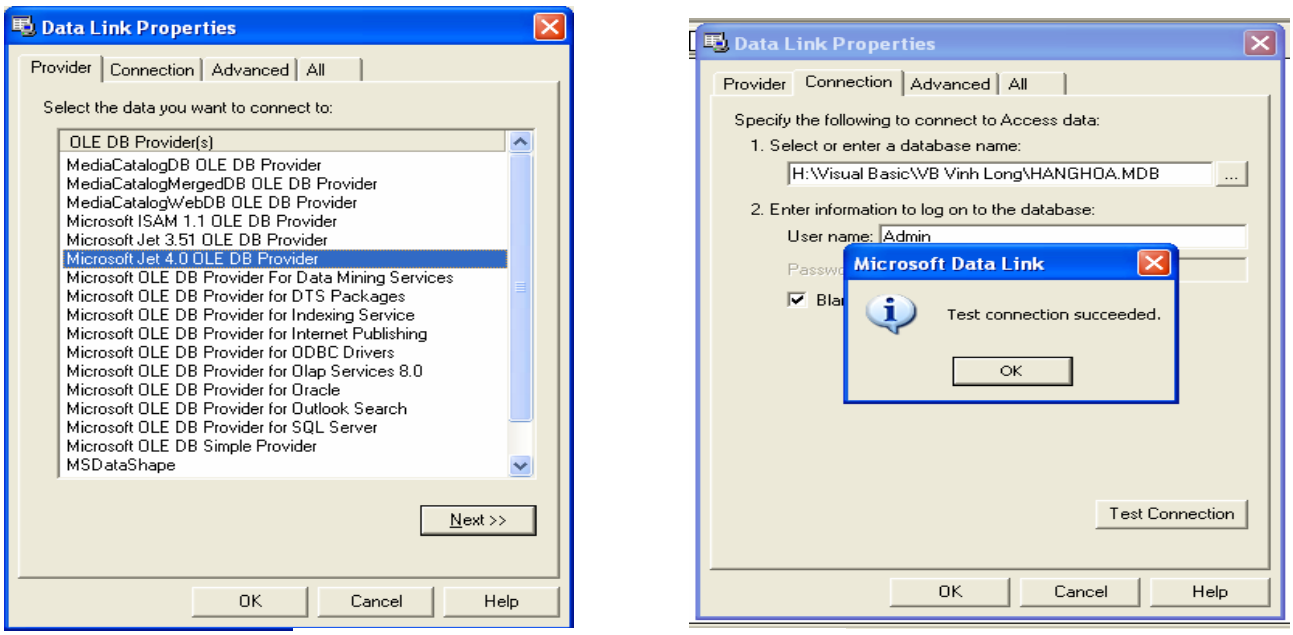
o Tạo một dự án mới; bổ sung Data Enviroment vào dự án của ta nhờ chọn Project/Add Data Environment. Khi lựa chọn mục này, môi trường DED sẽ hiển thị; sử dụng cửa sổ Properties để thiết lập thuộc tính Name là: datHH. Ở đây, ta sẽ sử dụng DED để kết nối với CSDL HANGHOA.MDB.

o Sửa lại thuộc tính Name của Connection1 là conHH; sau đó chuột phải lên conHH, chọn Properties.

Ở hộp thoại đầu tiên, ta phải chọn một trình cung cấp dữ liệu, ở đây chọn *Microsoft Jet 4.0 OLE DB Provider*, nhấn Next để tiếp tục.

Tiếp theo ta cần nhập chính xác đường dẫn đến tập tin CSDL, chẳng hạn ở đây là: H:\Visual Basic\HangHoa.Mdb.

Cuối cùng, nhấn nút Test Connection để kiểm tra việc nối kết dữ liệu chính xác hay không?



Hình 12.3: Đối tượng Connection

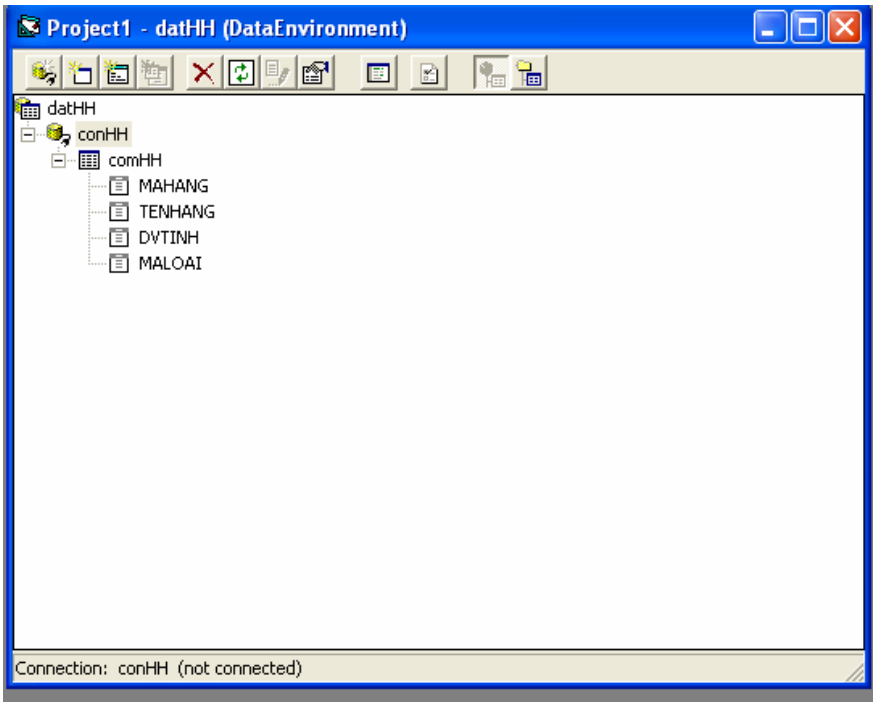
3) Tạo đối tượng Command:

o Xây dựng một đối tượng Command kết nối trực tiếp với Table (bảng) THANGHOA trong file dữ liệu HangHoa.mdb.

o Nhấp chuột phải trên kết nối dữ liệu conHH & chọn Add Command; sửa Command Name là: comHH; chọn Table từ Combo Box Database Object, chọn THANGHOA từ Combo Box Object Name.

o Trước khi đóng hộp thoại này, ta chuyển qua nhãn Advanced & thiết lập LockType là 3 – Optimistic (mặc nhiên là 1 – Read Only); Cursor Location: Use client-side cursor. Nhờ vậy ta mới có thể cập nhật Record Set từ chương trình của ta.

o Trở lại giao diện DED, ta được:



Hình 12.4: Đối tượng Command

4) Tạo một ứng dụng nhập liệu với DED

o Ở môi trường DED, ta kéo các trường của Command comHH vào Form1, chỉnh sửa lại cho thích hợp.

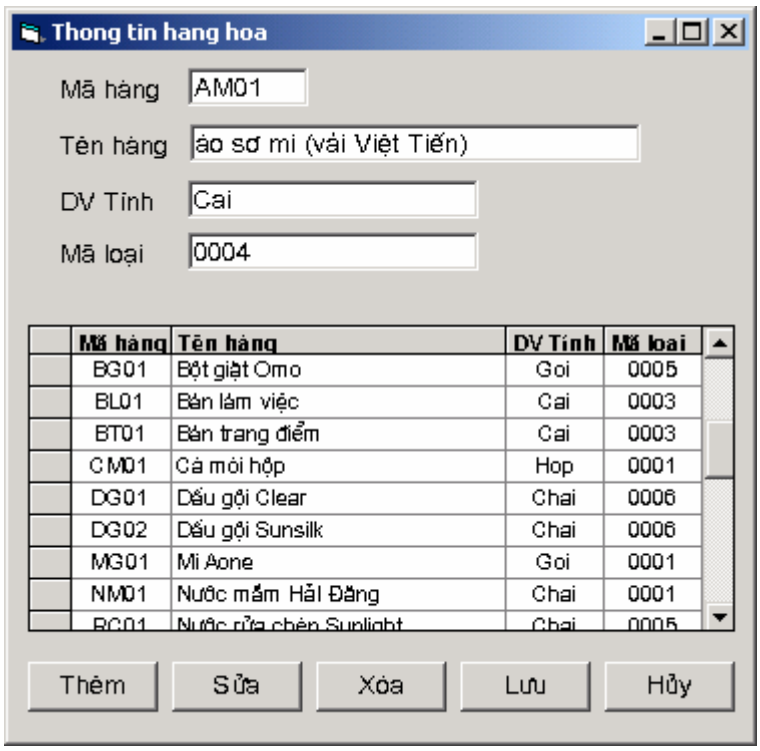
o Ở đây ta có sử dụng một lưới để hiển thị dữ liệu; do vậy ta chọn Project\Component; chọn Microsoft DataGrid Control 6.0 (OLE DB); sau đó kéo điều khiển này vào Form, thiết lập các thuộc tính cho thích hợp.

Name: grdHH.
DataSource: datHH
DataMember: comHH

o Nhấp chuột phải lên điều khiển DataGrid, chọn Retrieve Structure. Sau đó, lưu dự án & chạy chương trình ta được:

Hình 12.5: Form hiển thị table THangHoa

o Thêm các nút hành động (Thêm, Sửa, Xóa,…). Chẳng hạn các sự kiện cmd_Them_Click, cmdXoa_Click, cmdLuu_Click, cmdHuy_Click được xử lý:



Hình 12.6: Form nhập hoàn chỉnh cho table THangHoa

```
Private Sub cmdThem_Click()
    With datHH.rscomHH
        .AddNew
    End With
End Sub


Private Sub cmdXoa_Click()
    With datHH.rscomHH
        .Delete
       .Update
        Me.Refresh
    End With
End Sub

Private Sub cmdHuy_Click()
    With datHH.rscomHH
        .CancelUpdate
        Me.Refresh
    End With
End Sub

Private Sub cmdLuu_Click()
    On Error GoTo Xuly
    With datHH.rscomHH
        .Update
    End With
    Me.Refresh
```

```
    Exit Sub
Xuly:
    MsgBox Err.Description, vbCritical + vbSystemModal,
"Error"
End Sub
```

Như vậy, ta đã thiết kế xong một Form cho phép hiển thị thông tin các hàng hóa, Form này cho phép sửa đổi, thêm mới các mẩu tin trong bảng THANGHOA của CSDL HANGHOA.MDB.

5) Đối tượng Command sử dụng câu lệnh SQL

- Ta có thể thiết kế các câu lệnh SQL cách nhanh chóng nhờ sử dụng trình SQL Builder.

- Với trình DataEnvironment trên, ta thêm một đối tượng Command nữa; nhưng đối tượng Command này lấy dữ liệu từ câu lệnh SQL. Ở đây, ta cần hiển thị thông tin về các loại hàng hóa & tên loại hàng tương ứng.

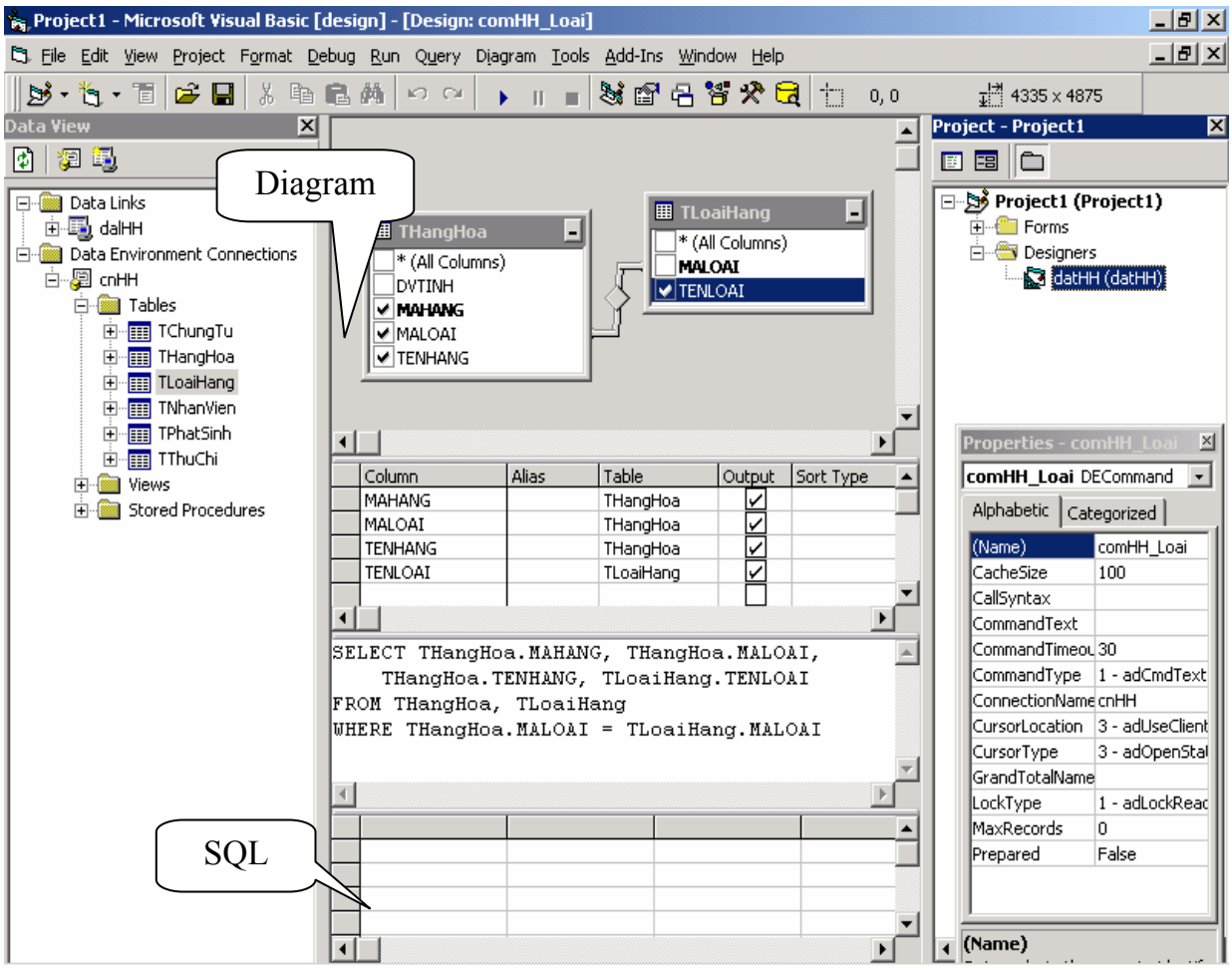
- Sau khi thêm một đối tượng Command vào, ta sửa các thuộc tính như sau:

  Name: comHH_Loai

  Source of Data : SQL Statement.

- Sau đó chọn SQL Builder trong cửa sổ thuộc tính; một cửa sổ hiện ra. Kéo bảng THangHoa & TLoaiHang trong cửa sổ DataView vào khung Diagram của trình SQL Builder. Check chọn các trường tương ứng (MaHang, TenHang, DVTinh, TenLoai). Ta được kết quả sau (hình dưới).

- Đóng cửa sổ này lại; ta được một đối tượng Command mới.



Hình 12.7: Cửa sổ SQL Builder

Chương 13: THIẾT LẬP BÁO CÁO

Mục tiêu:

Chương này giới thiệu cách thức để tạo báo cáo bao gồm hiển thị dữ liệu cũng như sắp xếp và phân nhóm dữ liệu.

Học xong chương này, sinh viên có thể:

- Sử dụng tính năng Report của Microsoft Access trong các ứng dụng nhỏ.
- Sử dụng Data Report để tạo báo biểu.
- Sử dụng Crystal Report, công cụ mạnh để tạo báo biểu.

Kiến thức cần thiết:

- Thư viện đối tượng ActiveX Data Objects (ADO).
- Môi trường dữ liệu Data Environment.

Tài liệu tham khảo:

Visual Basic 6.0 và Lập trình cơ sở dữ liệu - Chương 21, trang 637 - **Nguyễn Thị Ngọc Mai** (chủ biên) – **Nhà xuất bản Giáo dục** - **2001**.

I. SỬ DỤNG MICROSOFT ACCESS ĐỂ LẬP BÁO CÁO

Có hai kỹ thuật để thi hành một báo cáo Access từ ứng dụng VB:

- Sử dụng Automation để phóng một thể hiện (instance) của Microsoft Access, thi hành báo cáo trực tiếp từ trong ứng dụng. Automation là một kỹ thuật cho phép giao tiếp giữa các ứng dụng trên Windows. Ở đây Microsoft Access sẽ làm Automation Server.

- Dùng VSREPORTS của VideoSoft cho phép người sử dụng VB thi hành báo cáo của Microsoft Access bất kể máy của họ có cài đặt Microsoft Access hay là không. Đây là một điều khiển ActiveX chuyển đổi báo cáo từ tập tin MDB thành một định dạng mà ta có thể cung cấp cùng ứng dụng.

Trong bài giảng này, chúng tôi chỉ trình bày cách thứ nhất mặc dù cách này có nhiều hạn chế. Đối với cách thứ hai, để có thể thực hiện được ta cần phải cài đặt một số thư viện liên kết động (DLL). Các thư viện này tương đối khó tìm và nhất là chúng đòi hỏi bản quyền.

Bất lợi của kỹ thuật dùng Automation là buộc người dùng phải chạy một thể hiện (instance) của Microsoft Access cũng như phải cài đặt Microsoft Access trên máy. Để lập trình theo kỹ thuật này, ta tiến hành theo các bước sau:

- Tham chiếu đến Microsoft Access bằng cách từ menu Project chọn Preferences -> Microsoft Access 9.0 Object Library.

- Sau đó ta tạo một đối tượng như là đối tượng ứng dụng của Access như sau:

```
Dim MSAccess As Access.Application
```

- Sau đó ta cần tạo mới đối tượng này cũng như tạo một tham chiếu đến cơ sở dữ liệu chứa báo cáo:

```
Set MSAccess = New Access.Application
MSAccess.OpenCurrentDatabase("Database Name")
```

- Sử dụng thuộc tính DoCmd để thi hành báo cáo:

```
MSAccess.DoCmd.OpenReport "Report Name",acViewNormal
```

- Đóng cơ sở dữ liệu:

```
MSAccess.CloseCurrentDatabase
```

Lưu ý: Tránh dùng ràng buộc trễ với Automation

Phiên bản cũ của Automation là OLE Automation, dùng trong VB 3.0 và Microsoft Access 2.0.

Trong VB 3.0, ta có thể viết chương trình như sau:

```
Dim MSAccess As Object
Set MSAccess = CreateObject("Access.Application")
```

Đoạn chương trình trên hoạt động tốt đối với VB 3.0 nhưng có một cách khác tốt hơn. Thay vì dùng kiểu Object, ta nên chỉ rõ kiểu dữ liệu đối tượng mà Automation Server cung cấp (chẳng hạn Access.Aplication nếu là Access). Bởi vì khi đó, VB không cần thi hành câu truy vấn trên Automation Server mỗi

khi ta truy cập nó để xác định kiểu đối tượng cần tạo. Kỹ thuật này gọi là **ràng buộc trễ**, giờ đây chỉ phù hợp với 2 tình huống:

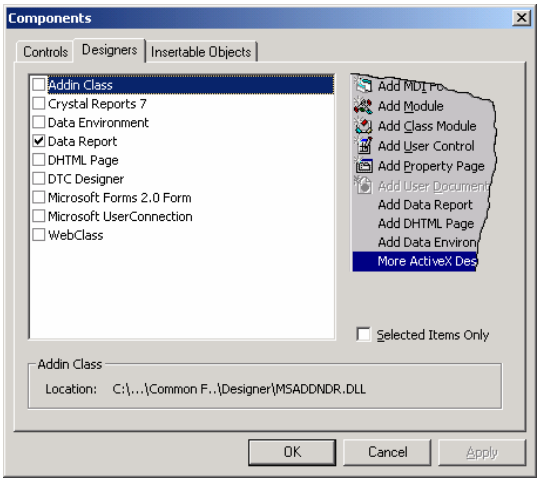
- Ta không biết trước kiểu đối tượng Automation Server.
- Ta đang sử dụng một môi trường phát triển ứng dụng không hỗ trợ ràng buộc sớm, như VBScript hay ASP.

II. SỬ DỤNG THIẾT KẾ DATA REPORT

Thiết kế báo cáo dùng DataReport là điểm mới trong VB6, đây là một công cụ được hỗ trợ bởi VB6, cung cấp một cách trực quan về thiết kế báo cáo và có ưu điểm là rất dễ dùng.

II.1 Thiết kế với DataReport

- Chọn Project -> Components.
- Chọn Tab Designers, đánh dấu chọn Data Report.



Hình 13.1 Đưa thiết kế báo cáo về đề án

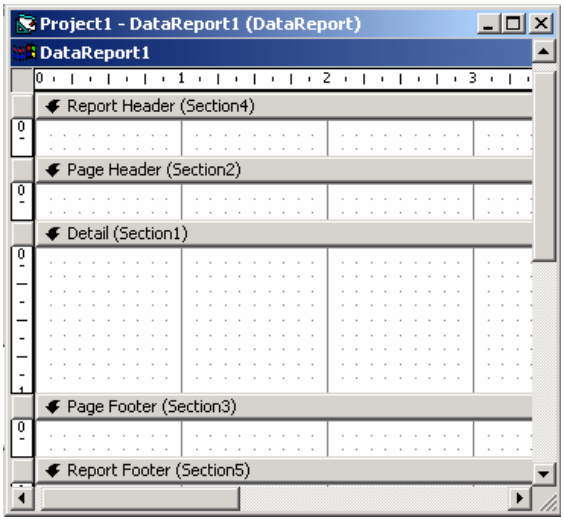
Các thành phần của một báo cáo như sau:

- Report Header: Hiển thị một lần ở đầu báo cáo.
- Report Footer: Hiển thị một lần ở cuối báo cáo.
- Page Header: Hiển thị tại đầu mỗi trang.
- Page Footer: Hiển thị tại cuối mỗi trang.
- Detail Section: Hiển thị các dòng dữ liệu.
- Một hoặc nhiều nhóm đầu cuối hiển thị tại đầu và cuối mỗi phân nhóm.

Các điều khiển của thiết kế Data Report như sau:

- Điều khiển nhãn (Rpt Label).
- Điều khiển hộp văn bản (Rpt Textbox).
- Điều khiển ảnh (Rpt Image).
- Điều khiển hình dạng (Rpt Shape).

- Điều khiển các hàm tính toán (Report Function: rptFuncSum, rptFuncAve, rptFuncMin, rptFuncMax…).

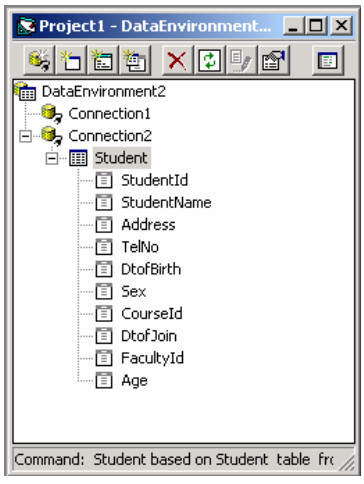


Hình 13.2 Cửa sổ Data Report

Các điều khiển của Data Report cũng giống như là các điều khiển chuẩn trên biểu mẫu, chúng có thể ràng buộc với nguồn dữ liệu. Tuy nhiên, ta có một cách thức khác dễ dàng hơn đó là sử dụng môi trường dữ liệu (được giới thiệu ở chương trước).

Sử dụng DataEnvironment trong việc tạo DataReport:
Quá trình thực hiện trải qua các bước sau:
- Tạo đối tượng Command.
- Kéo thả các trường của đối tượng Command này vào thiết kế của Report.
- Thêm các tiêu đề đầu trang & cuối trang.

Ví dụ: Tạo báo cáo về các sinh viên trong bảng STUDENT thuộc cơ sở dữ liệu Student.



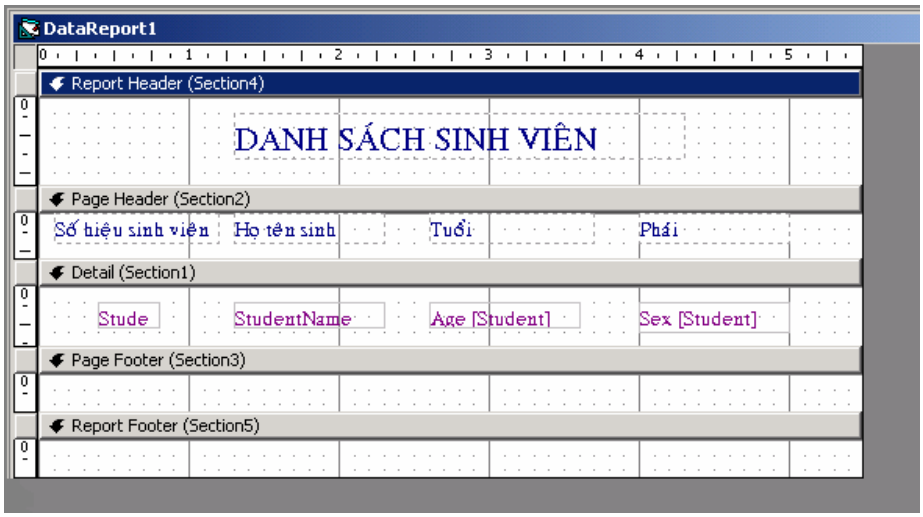
Hình 13.3 Thiết lập Data Environment

- Bước 1: Tạo một nối kết đến CSDL Student trong trình Data Environment, thêm một đối tượng Command cho phép lấy dữ liệu từ bảng Student.

- Bước 2: Kéo thả các trường cần hiển thị vào báo cáo tại mục Detail, chỉ giữ lại trường liên quan đến thông tin dữ liệu (đặt trong phần Detail Section). Thiết

lập tên trường dưới dạng tiếng Việt tại phần Page Header.

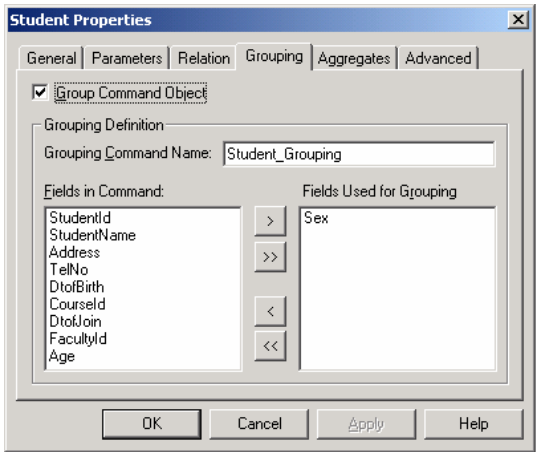
- Bước 3: Cung cấp các thông tin cho phép DataReport nhận dữ liệu từ đâu bằng cách xác lập: DataSource: DataEnvironment1, DataMember: Student.



Hình 13.4: Report khi đã kéo thả các trường

Thiết kế báo biểu có phân nhóm dữ liệu

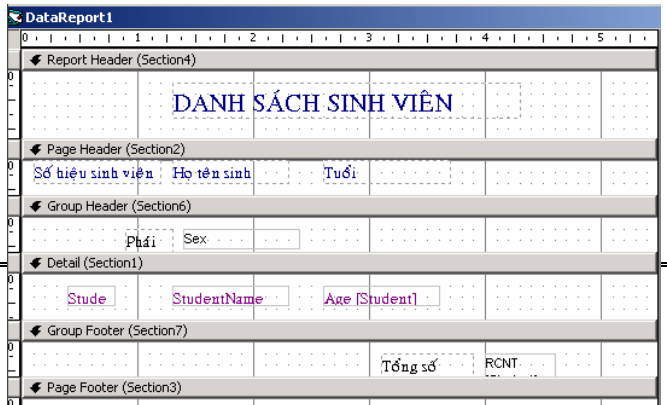
- Chọn đối tượng Command của trình DataEnvironment cần nhóm cơ sở dữ liệu.



Hình 13.4: Nhóm dữ liệu

- Hiển thị trang thuộc tính, chọn Tab Grouping.
- Chọn tùy chọn Group Command Object.
- Đặt tên cho nhóm cũng như chọn các trường tham gia vào nhóm dữ liệu.
- Đặt lại giá trị cho thuộc tính Data Member chỉ đến nối kết mới đã nhóm dữ liệu.
- Chọn báo cáo thiết kế, ấn chuột phải, chọn Insert Group Header/Footer.
- Chọn tên trường nhóm dữ liệu đưa vào đoạn Group Header.

Khi đó báo cáo được thiết kế như sau:



II.2 Xem và xuất Data Report

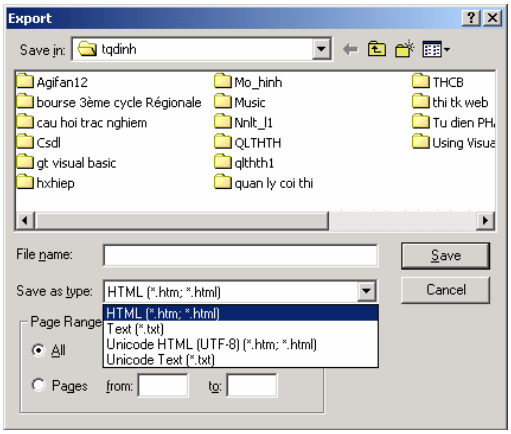
Ta có thể xem thông tin và in báo cáo trên một cửa sổ riêng biệt sử dụng chế độ Print Preview bằng cách thi hành phương thức Show.

Khi đó báo cáo sẽ được hiển thị như sau:

Hình 13.6: Thi hành báo cáo trong VB

Khi đó người sử dụng có thể duyệt qua các trang nếu như báo cáo có nhiều trang, cũng như chọn một trang báo cáo nào đó để in.

Ngoài ra người dùng có thể chọn Export báo cáo của mình ra tập tin có định dạng khác, các loại định dạng ở đây có thể là tập tin văn bản, tập tin HTML. Ta có thể chọn lựa xuất một số trang cụ thể nào đó hoặc toàn bộ báo cáo.



Hình 13.7: Hộp thoại xuất báo cáo

III. SỬ DỤNG CRYSTAL REPORT ĐỂ LẬP BÁO CÁO

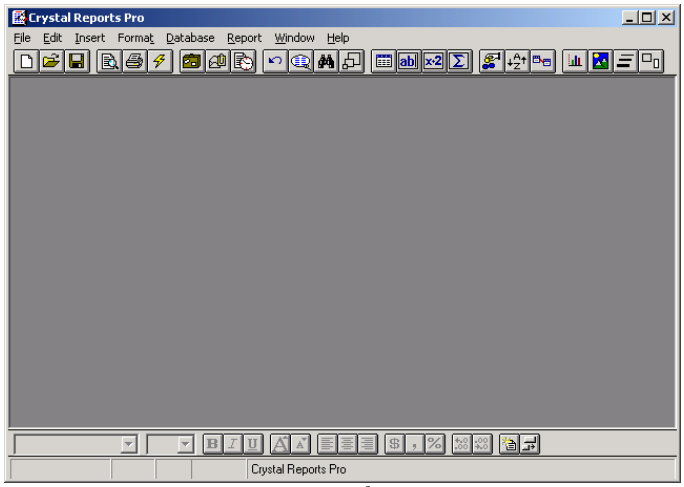
Crystal Report cho phép tạo báo cáo cơ sở dữ liệu trong ứng dụng viết bằng VB. Nó gồm 2 phần chủ yếu:

- Trình thiết kế báo cáo xác định dữ liệu sẽ đưa vào báo cáo và cách thể hiện của báo cáo.
- Một điều khiển ActiveX cho phép thi hành, hiển thị, điều khiển và in báo cáo khi thi hành ứng dụng.

Crystal Report không có sẵn khi cài VB6, ta cần cài đặt thêm. Chương trình cài đặt Crystal Report chỉ có trên bản Professional. Chạy tập tin Crystl32.exe trong thư mục \COMMON\TOOLS\VB\CRYSREPT.

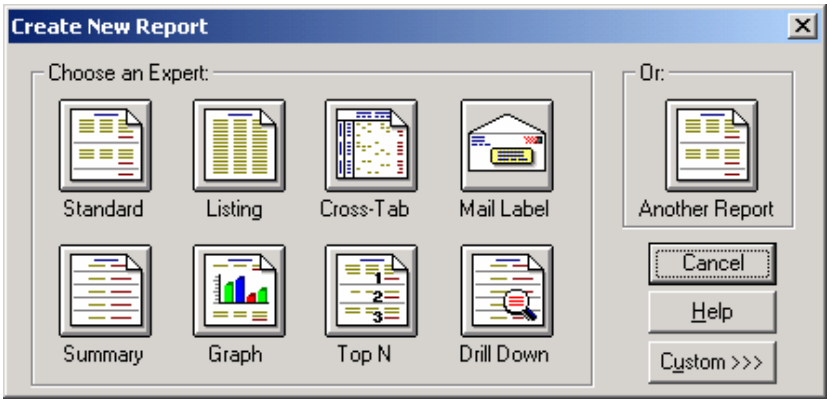
III.1 Thiết kế báo cáo

Một điểm khác biệt khi dùng Crystal Report là ta không thiết lập báo cáo đi đôi với ứng dụng cụ thể. Ta sẽ xây dựng báo cáo trước và sau đó sẽ gọi thi hành báo cáo từ phía ứng dụng, báo cáo không phải là một bộ phận thuộc ứng dụng. Cửa sổ thiết kế Crystal Report như hình bên dưới:



Hình 13.8 Cửa sổ Crystal Report

Khi ta chọn tạo một báo cáo mới, Crystal Report trình bày một hộp thoại cho phép lựa chọn một trong nhiều những khuôn mẫu báo cáo đã định sẵn.



Hình 13.9 Hộp thoại chọn các mẫu

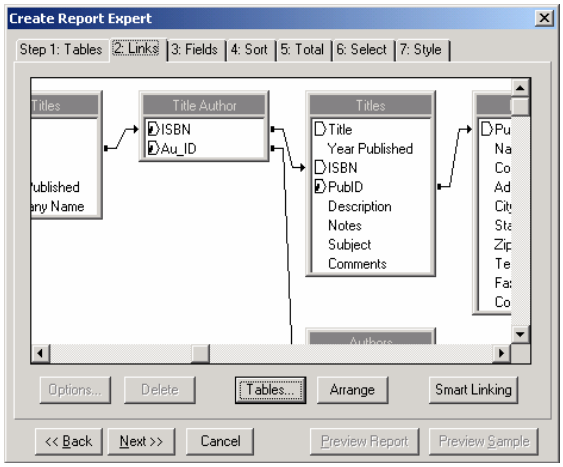
Kiểu báo cáo	**Mô tả**
Standard	Báo cáo sắp xếp thông tin theo dòng và cột, cho phép nhóm dữ liệu.
Listing	Báo cáo là danh sách dữ liệu liên tục không có tổng kết hay trường tổng cộng..
Cross-Tab	Sắp xếp dữ liệu theo hai chiều.
Mail label	Báo cáo được thiết kế để in dữ liệu theo cột cho nhãn thư.
Summary	Báo cáo chỉ hiển thị thông tin tổng quát, không chứa dữ liệu chi tiết.
Graph	Báo cáo thể hiện dữ liệu một cách trực quan bằng biểu đồ
Top N	Báo cáo cho phép chỉ hiển thị một số mẩu tin được chọn
Drill Down	Báo cáo cho phép nhấn đúp chuột lên dữ liệu tổng quát để hiển thị dữ liệu chi tiết.
Another	Các báo cáo có khuôn mẫu do người dùng định nghĩa trước đó.

Chúng ta xét qua một ví dụ sử dụng Crystal Report để lập báo cáo

- Khởi động Crystal Report và chọn New, chọn kiểu báo cáo là Standard.

- Tiếp theo chọn Data File.

- Trong hộp thoại chọn tập tin cơ sở dữ liệu, ta chỉ đến một tập tin cơ sở dữ liệu, sau đó ấn nút Done. Ta sẽ thấy các bảng cũng như các quan hệ giữa các bảng được hiển thị.



Hình 13.10 Hộp thoại quan hệ giữa các bảng

Quan hệ giữa các bảng đã được xác định ở mức thiết kế cơ sở dữ liệu nên ta không cần phải thay đổi những mối liên kết này.

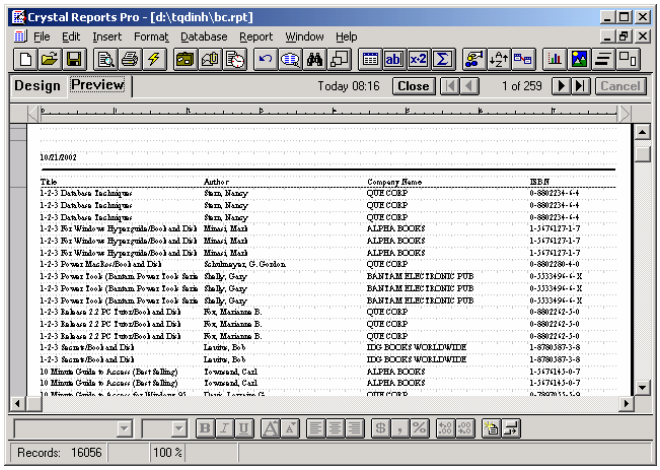
- Nhấn nút Next qua bước tiếp theo, ta sẽ chọn những trường tham gia vào báo cáo.

- Bước kế tiếp ta chọn qua Tab Sort để thực hiện việc sắp xếp dữ liệu.

- Tab Style cho phép chọn các dạng khác nhau của báo cáo.

- Sau khi đã thiết kế xong, ta ấn Save để lưu lại báo cáo.

Khi mở lại báo cáo đã thiết kế, ta thấy Crystal Report hiển thị báo cáo ở hai mức, thiết kế và duyệt trước.



Hình 13.11: Cửa sổ xem trước báo cáo và thiết kế báo cáo

III.2. Thi hành báo cáo trong ứng dụng thông qua điều khiển ActiveX của Crystal Report

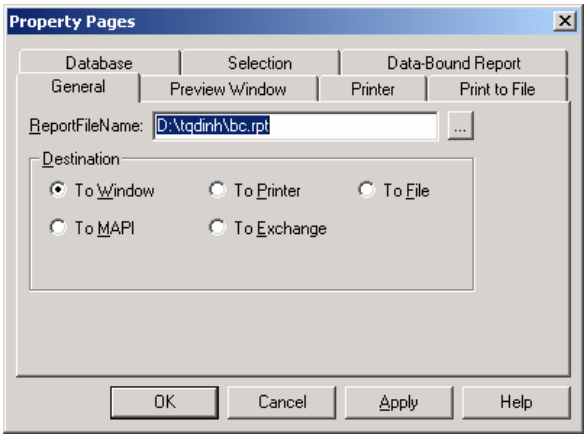
Bước đầu tiên để có thể thi hành báo cáo Crystal Report, ta cần tham khảo đến điều khiển ActiveX của Crystal Report bằng cách thêm công cụ Crystal Report vào đề án của chúng ta.

Chọn công cụ Crystal Report và đưa vào ứng dụng, biểu tượng trên hộp công cụ như sau. Trong sự kiện Click của một nút lệnh, ta viết đoạn mã sau:

```
Private Sub Command1_Click()
        CrystalReport1.ReportFileName = "d:\VB\bc.rpt"
        CrystalReport1.PrintReport
End Sub
```

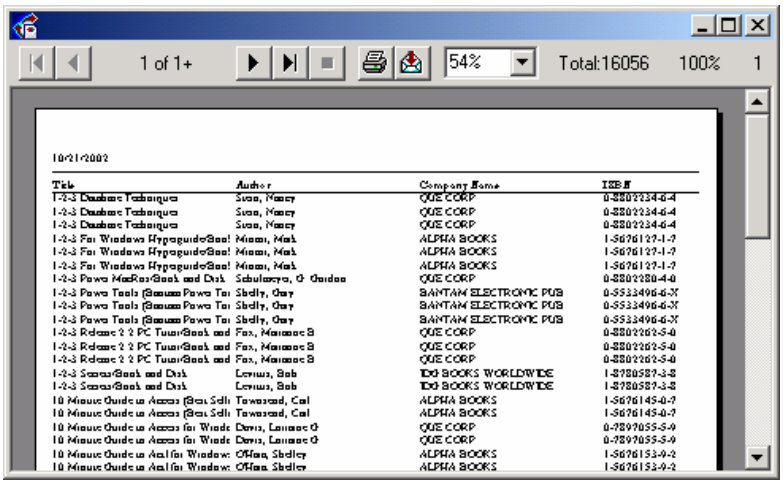
Thuộc tính ReportFileName xác định đường dẫn cũng như tên tập tin báo cáo.

Việc thi hành báo cáo được thực hiện nhờ vào thuộc tính PrintReport. Ngoài ra, báo cáo có thể thi hành bằng cách hiển thị trên một cửa sổ khác hoặc là xuất ra thẳng trên máy in, … Ta thiết đặt thuộc tính đó qua hộp thoại thuộc tính.



Hình 13.11 Hộp thoại xác lập thuộc tính

Báo cáo thi hành trên một cửa sổ riêng biệt, ta có thể lựa chọn nhiều công việc như xem qua các trang, in ấn báo cáo, phóng to thu nhỏ …



Hình 13.12: Báo cáo Crystal Report

LỜI KẾT

Chương Thiết lập báo cáo cũng là chương kết thúc của giáo trình Visual Basic. Tuy nhiên *lập trình sự kiện và lập trình cơ sở dữ liệu* với VB chỉ là một phần trong những khả năng mà VB mang lại. Hy vọng chúng tôi sẽ gặp lại bạn đọc trong những chuyên đề khác của VB.