ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRUNG TÂM PHÁT TRIỂN CÔNG NGHỆ THÔNG TIN
CITD

GIÁO TRÌNH

# LẬP TRÌNH CSDL VỚI VISUAL BASIC

Biên soạn: Nguyễn Văn Toàn
Nguyễn Thị Kim Phụng
Nguyễn Đình Loan Phương



NHÀ XUẤT BẢN
ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRUNG TÂM PHÁT TRIỂN CÔNG NGHỆ THÔNG TIN

---------o0o---------

# Giáo trình

# LẬP TRÌNH CƠ SỞ DỮ LIỆU với VISUAL BASIC

*Biên soạn*: Nguyễn Văn Toàn
Nguyễn Thị Kim Phụng
Nguyễn Đình Loan Phương

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA
TP HỒ CHÍ MINH – 2005

# LỜI NÓI ĐẦU

*Visual Basic là ngôn ngữ dùng để viết một ứng dụng nhanh chóng và hiệu quả trên Microsoft Windows. Thật vậy, dù bạn là lập trình viên chuyên nghiệp hay một người mới học lập trình, Visual Basic đều cung cấp cho bạn các công cụ hoàn chỉnh để đơn giản hóa việc tạo chương trình.*

*Thêm vào đó, Visual Basic cung cấp cho bạn các công cụ kết nối cơ sở dữ liệu một cách thật đơn giản. Trong giáo trình "**Lập trình cơ sở dữ liệu với Visual Basic**" này, chúng tôi cung cấp cho bạn đọc một số vấn đề cơ bản về lập trình trên Visual Basic nói chung và cơ sở dữ liệu nói riêng. Chúng tôi hy vọng sau khi tham khảo xong cuốn giáo trình này, bạn sẽ có thể tự mình xây dựng một ứng dụng vừa và nhỏ.*

*Tuy đã rất cố gắng trong công tác biên soạn, nhưng do lần đầu tiên xuất bản nên chắc chắn sẽ có ít nhiều khiếm khuyết. Chúng tôi rất trân trọng tiếp thu những ý kiến đóng góp của bạn đọc để hoàn thiện giáo trình trong các lần tái bản.*

Chương I

# TỔNG QUAN VỀ VISUAL BASIC

Visual Basic là một công cụ lập trình trực quan của Microsoft, giúp ta có thể xây dựng nhanh các ứng dụng trên Windows. Khác với các môi trường lập trình hướng thủ tục trước đây, Visual Basic là môi trường lập trình hướng sự kiện trên Windows.

Visual Basic cung cấp một bộ công cụ hoàn chỉnh để đơn giản hóa việc triển khai lập trình ứng dụng.

Có ba ấn bản VB6: Learning, Professional và Enterprise, trong đó ấn bản Learning không thông dụng lắm, ta tạm thời gác qua. Ấn bản Professional cung cấp đầy đủ những gì ta cần để triển khai một ứng dụng, nhất là các Control ActiveX, những bộ phận tiền chế và rất hữu dụng cho các chương trình ứng dụng. Ấn bản Enterprise là ấn bản Professional cộng thêm các công cụ Back Office (SQL Server, Microsoft Transaction Server, Internet Information Server).

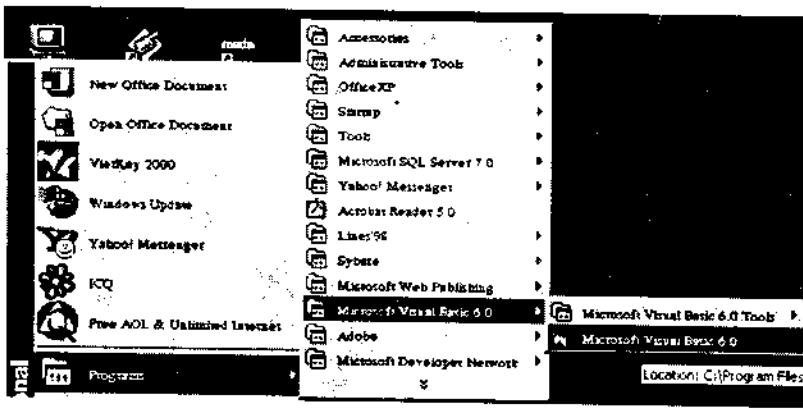
Khi viết một chương trình trong Visual Basic, ta phải qua hai bước: thiết kế giao diện và viết lệnh. Khi thiết kế giao diện, ta sử dụng các công cụ do Visual Basic cung cấp sẵn. Các công cụ này cho phép thiết kế giao diện bằng chuột lẫn bàn phím.

Trong chương này, ta chỉ cần nắm rõ cách sử dụng các công cụ sao cho thành thạo.

## I. KHỞI ĐỘNG CHƯƠNG TRÌNH VISUAL BASIC

Ta có thể khởi động chương trình Visual Basic theo cách sau:

**Nhấp nút Start → Programs → Microsoft Visual Studio 6.0 → Microsoft Visual Basic 6.0**




- Hình I.1 -

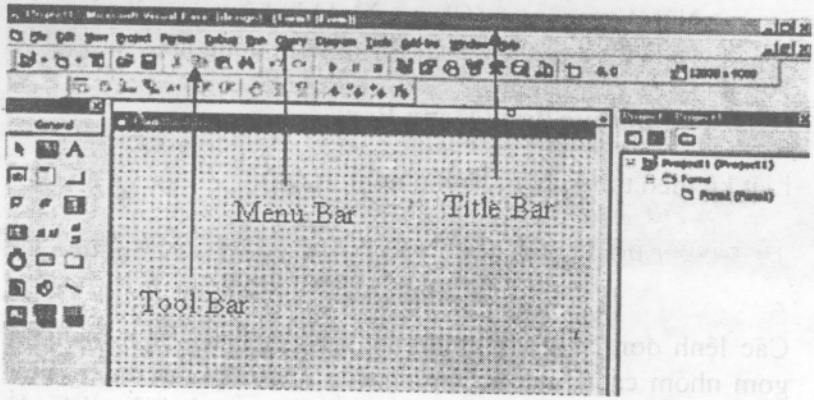
## II. TẠO THƯ MỤC LÀM VIỆC

Trước khi lập trình, ta cần tạo một thư mục chứa các chương trình làm việc.

Ví dụ: lưu các chương trình trong thư mục: **D:\VB**.

## III. GIỚI THIỆU GIAO DIỆN CỬA SỔ CỦA VISUAL BASIC

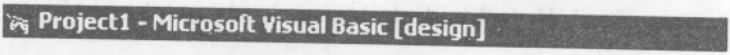
Sau khi khởi động, cửa sổ Visual Basic có dạng như hình bên dưới :




- Hình I.2 -

### III.1. Thanh tiêu đề (Title Bar)

Thể hiện tên của chương trình là Microsoft Visual Basic




- Hình I.3 -

Góc phải thanh tiêu đề có ba nút điều khiển 

- Nút thu nhỏ (minimize window)  : thu nhỏ cửa sổ làm việc

- Nút phóng to và phục hồi (maximize/restore window)  :

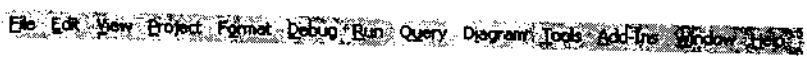
  + Nếu cửa sổ đang ở trạng thái bị thu hẹp kích thước, nút maximize (  ) giúp phóng to cửa sổ (cửa sổ ở trạng thái lớn nhất)

  + Nếu cửa sổ đang ở trạng thái lớn nhất về kích thước, nút Restore window () giúp ta có thể thu hẹp bớt kích thước cửa sổ.

- Nút đóng cửa sổ (Close) : khi nhấn vào nút này, cửa sổ làm việc của Visual Basic sẽ bị đóng lại.

### III.2. Thanh thực đơn (Menu Bar)

Liệt kê theo từng nhóm lệnh như hình dưới




- Hình I.4 -

Các lệnh đơn cung cấp một cách tiện dụng và nhất quán để gom nhóm các lệnh và giúp người dùng truy cập chúng một cách dễ dàng. Nó cung cấp cho người dùng một biện pháp để gọi các hành động khác nhau.

Thanh lệnh đơn (menu bar) xuất hiện ngay dưới thanh tiêu đề (title bar) và chứa một hoặc nhiều tùy chọn lệnh đơn, mỗi tùy chọn trên thanh lệnh đơn định danh một phạm trù các tác vụ.

Xuất hiện ngay dưới mỗi tùy chọn lệnh đơn có thể là một hoặc nhiều tùy chọn. Các tùy chọn xuất hiện trên thanh lệnh đơn được gọi là các tùy chọn lệnh đơn chính và những tùy chọn dưới chúng được gọi là các tùy chọn con.

Ví dụ vài lệnh đơn của Visual Basic:

+ File: có các tùy chọn con tạo thuận lợi cho các tác vụ điều quản tập tin như mở, đóng, lưu các tập tin

+ Edit: có các tùy chọn con tạo thuận lợi cho các tác vụ hiệu chỉnh như cắt, chép, dán văn bản.

+ Help: có các tùy chọn con trợ giúp trực tuyến cho nhiều công việc khác nhau trong ứng dụng.

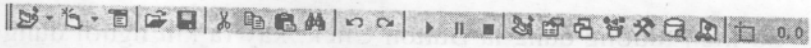
.....

### III.3. Thanh công cụ (Tool Bar)

Là các lệnh được thể hiện dưới dạng biểu tượng giúp người sử dụng có thể thi hành một lệnh nhanh hơn.

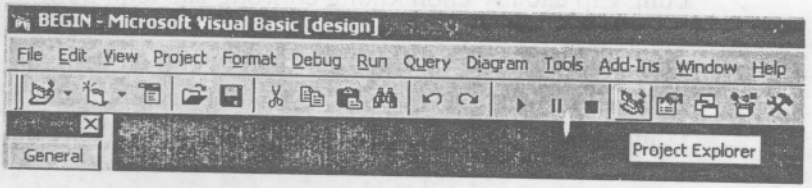
- Thanh công cụ chứa các nút thanh công cụ, cho phép nhanh chóng truy cập đến các lệnh thường dùng nhất.

- Các nút trong thanh công cụ cũng tương tự như các mục trong thanh thực đơn. Các nút trong thanh công cụ cung cấp cho người dùng một lối tắt đối với các hành động mà ta phải chọn qua nhiều bước bằng lệnh đơn.



- Hình I.5 -

- Các nút trong thanh công cụ có một ô gợi nhớ (gọi là Tool Tip) – là những cửa sổ nhỏ bật ra chứa một mô tả văn bản ngắn gọn về công dụng và tên gọi của nút trong thanh công cụ. Khi người dùng đưa chuột đến một nút, cửa sổ gợi nhớ sẽ bật ra văn bản nhắc nhở.




- Hình I.6 -

➢Các tính chất của thanh công cụ

Các tính chất duy nhất đối với một điều khiển thanh công cụ gồm:

+ Style: xác định hình dáng của điều khiển

+ TextAlignment: xác định vị trí của văn bản tương đối với nút

+ ToolTipText: ô gợi nhớ tên công cụ

➢ Các phương pháp của thanh công cụ

Dưới đây là vài phương pháp thường dùng:

+ Move

+ Drag

➢ Các sự kiện của thanh công cụ

Sự kiện thường dùng là Button Click: xảy ra khi người dùng nhấp lên một đối tượng Object trong một điều khiển ToolBar.

➢ Vài điểm cần lưu ý về thanh công cụ

- Một thanh công cụ cung cấp các lối tắt đối với các hành động (hành động đó có thể được gọi bằng cách dùng thanh lệnh đơn). Tuy nhiên, thanh công cụ có thể cung cấp các tuỳ chọn không có trong các lệnh đơn;

- Hầu hết các tuỳ chọn trên một thanh công cụ đều xuất hiện dưới dạng các nút, ngoài ra một tuỳ chọn thanh công cụ cũng có thể hiển thị dưới dạng một hộp liệt kê.

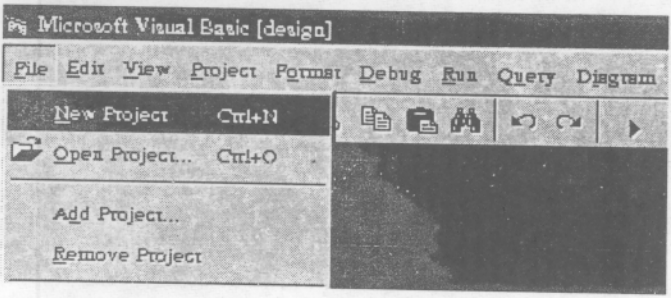
## IV. TẠO / LƯU PROJECT LÀM VIỆC

### IV.1. Tạo project

➢ Các bước thực hiện

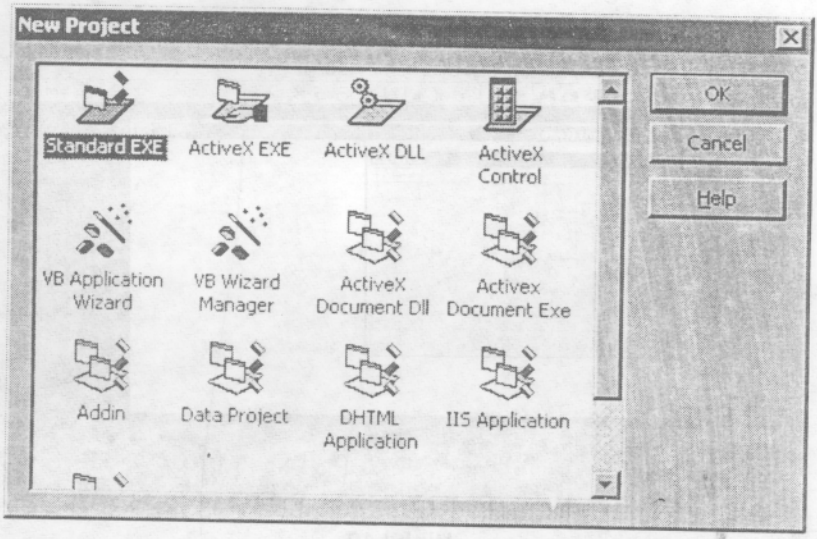
- Khởi động Visual Basic

- Chọn trình đơn **File** → chọn mục **New Project**



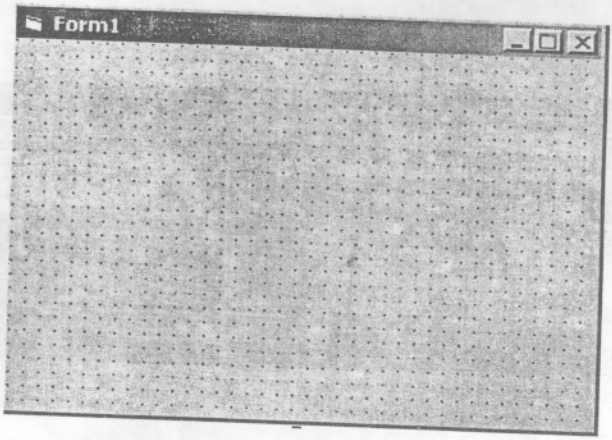

**- Hình I.7-**

- Trong hộp thoại New Project, chọn loại project muốn tạo, ở đây ta sẽ chọn StandardEXE như hình sau



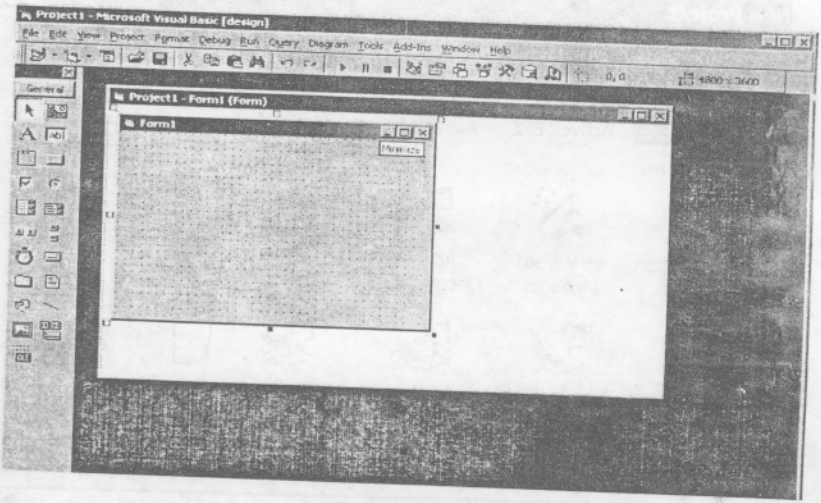

**- Hình I.8 -**

- Visual Basic sẽ hiện một cửa sổ trên màn hình. Đây là **Form** (cửa sổ) trống với tiêu đề (caption) là Form1.




- Hình I.9 -

Cửa sổ chương trình lúc này có dạng:




- Hình I.10 -

## IV.2. Lưu project

Một trong những vấn đề quan trọng khi viết chương trình là lưu lại những gì mình đã làm. Mặc dù chưa có thao tác gì thay

đổi trong form trống, ta cũng nên lưu form lên đĩa. Khi ta lưu project, có hai tập tin được lưu:

1. Tập tin chứa các thông tin Visual Basic cần trong việc xây dựng project. Tập tin này có phần mở rộng là VBP.

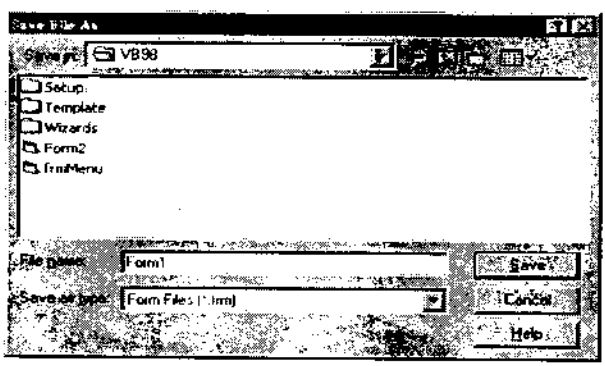
2. Tập tin chứa các thông tin về form. Tập tin này có phần mở rộng là FRM.

➢ Các bước thực hiện

- Chọn trình đơn **File** → Chọn mục **Save Project As**

- Visual Basic hiện hộp thoại (Dialog Box) **Save File As**

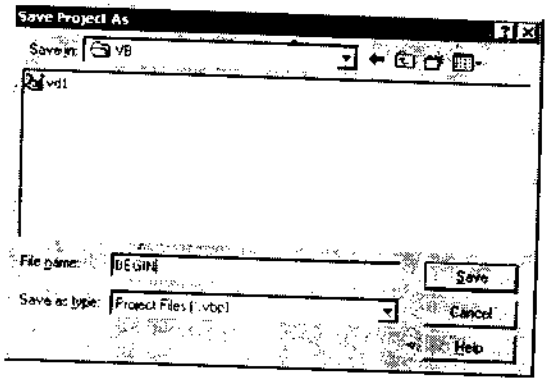
  Chọn thư mục **D:\VB** làm thư mục lưu và thay đổi tên mặc định của form trống từ **Form1.FRM** thành **BEGIN.FRM**




**- Hình I.11 -**

**Lưu ý:** *Không nên dùng tên mặc định Visual Basic đặt cho Form, ta nên đặt tên form gần gũi với các chức năng của Form bạn đang muốn xây dựng.*

- Lúc này, Visual Basic hiện tiếp hộp thoại **Save File As**, tập tin project có tên mặc định là **Project1.VBP**. Ta đổi lại tên cho tập tin Project là **BEGIN.VBP** và lưu vào thư mục **D:\VB**.




**- Hình I.12 -**

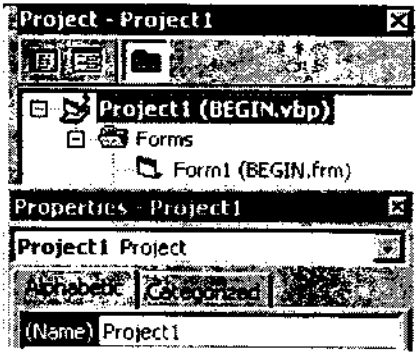
**Lưu ý:** *Không nên dùng tên mặc định Visual Basic đặt cho project, ta nên đặt tên p0roject gần gũi với chương trình ta đang muốn xây dựng.*

Như vậy, hai tập tin **BEGIN.FRM** và **BEGIN.VBP** đã được lưu vào thư mục **D:\VB** (lúc này chương trình sẽ phát sinh một file phụ là **BEGIN.VBW**).

## V. LÀM VIỆC VỚI CỬA SỔ PROJECT

Hiện tại, trong project chỉ chứa một form (được lưu thành tập tin BEGIN.FRM). Tuy nhiên, một project có thể có nhiều tập tin form. Visual Basic cung cấp một công cụ để quản lý tất cả các tập tin form mà project cần có.

Chọn trình đơn **View** → chọn mục **Project Explorer**. Cửa sổ project xuất hiện như hình I.13.




**- Hình I.13 -**

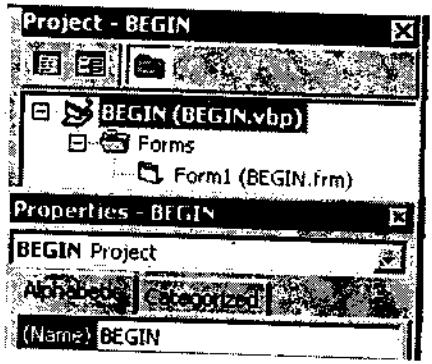
Tiêu đề của cửa sổ Project chính là tên project (thuộc tính Name).

Tên tập tin đầu tiên trong cửa sổ project là tên tập tin BEGIN.FRM.

Khi tạo Project, tiêu đề (caption) mặc định của project là Project1, Project2,... như hình I.13.

Do đó, lúc này, tên cửa sổ Project cũng là Project1 như ta đã thấy trong hình.

Muốn đổi tên hiển thị này của project, ta có thể chọn vào project trong cửa sổ Project, lúc này ta thấy bên dưới sẽ xuất hiện một cửa sổ phụ, đó là cửa sổ **Properties** của project, tại đây, ta có thể thay đổi thuộc tính **Name** của project.




- Hình I.14 –

Bây giờ, tên hiển thị của project đã được đổi lại thành BEGIN, nên tên của cửa sổ Project lúc này cũng là BEGIN.

## VI. GIỚI THIỆU TỔNG QUAN VỀ FORM

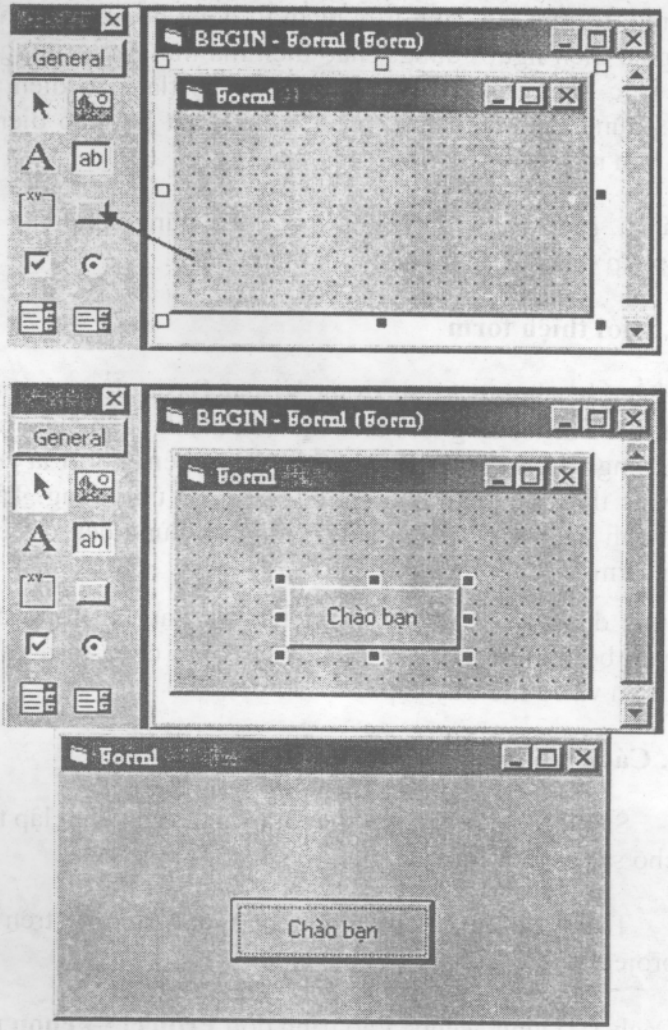
### VI.1. Lập trình trực quan

Lập trình trực quan nhắm đến việc cung cấp cho người dùng một giao diện trực quan và dễ dùng. Trong khi phát triển giao diện, ta cần sử dụng các tính năng thân thiện với người dùng như các cửa sổ, các lệnh đơn, các nút, các hộp liệt kê,..

Một môi trường lập trình trực quan cung cấp tất cả các tính năng cần có để phát triển một giao diện người dùng đồ họa dưới các thành phần sẵn dùng.

Khi cần một tính năng giao diện người dùng cụ thể, ta chỉ việc chọn lựa thành phần có sẵn thích hợp mà môi trường lập trình trực quan cung cấp. Bên cạnh đó, ta có thể dời vị trí, điều chỉnh kích cỡ và đặt lại tên cho các thành phần này.

Ví dụ: Nếu ta cần dùng một nút, môi trường lập trình trực quan sẽ cung cấp cho ta nút đó. Tất cả những gì ta phải làm là lựa một nút rồi đặt nó lên màn hình tại vị trí thích hợp.




**- Hình I.15 -**

Thông thường, ta dùng chuột để lựa chọn và sắp đặt các thành phần cần thiết. Như vậy, môi trường lập trình trực quan cũng được gọi là môi trường nhấp thả.

Một môi trường lập trình trực quan tự động hóa tiến trình tạo một giao diện người dùng. Giao diện mà một môi trường lập trình trực quan cung cấp cho ta được gọi là giao diện trực quan. Dùng giao diện trực quan, ta sẽ thiết kế giao diện dễ dàng hơn mà không phải viết mã.

Ngoài ra, môi trường lập trình trực quan cũng cung cấp một biện pháp để kết hợp mã với mỗi thành phần.

## VI.2. Giới thiệu form

Hầu hết các chương trình trong Visual Basic đều có ít nhất một form. Form còn gọi là biểu mẫu, là thành phần quan trọng nhất trong số tất cả các thành phần giao diện. Form là cơ sở cho việc thiết kế giao diện người dùng của ứng dụng. Tất cả các thành phần tạo thành giao diện người dùng đều được đặt trong form.

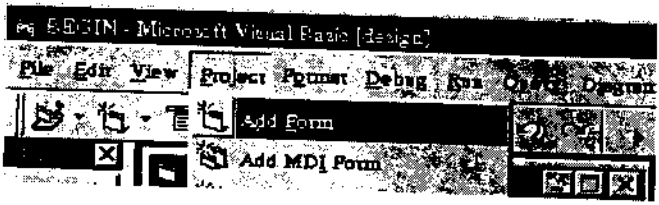
Khi ứng dụng chạy, nó mở ra một form. Sau khi form được nạp vào bộ nhớ, tất cả các thành phần hàm chứa trong nó sẽ hiện diện và có thể truy cập.

## VI.3. Cách tạo form

- Sau khi tạo project mới, môi trường triển khai lập trình cho sẵn ta một form.

- Trong trường hợp ta muốn tạo một form mới trên một project đã mở sẵn, ta có hai cách:

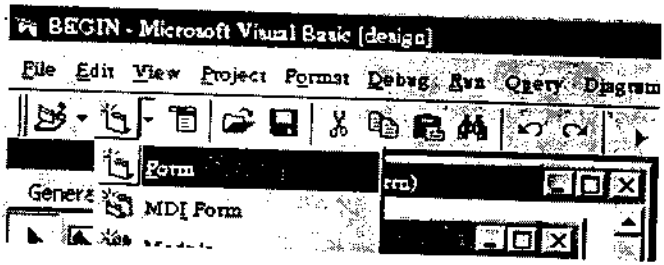
  Cách 1: Dùng Menu: Vào trình đơn **Project → chọn mục Add Form** như hình sau:




- Hình I.16 –

Một form mới sẽ được tạo ra và thêm vào trong project.

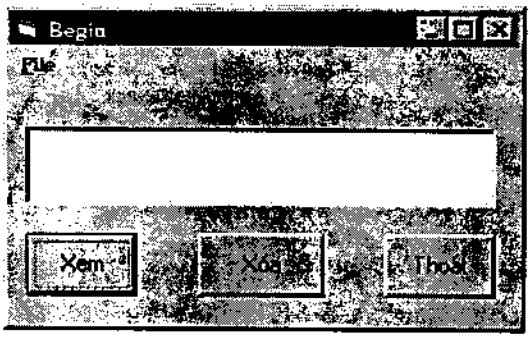
Cách 2: Dùng công cụ: Trên thanh công cụ, ta chọn vào công cụ **Add Form** như hình sau:




- Hình I.17 -

- Sau khi đã có form trống, ta đặt các thành phần lên form, thực hiện chỉnh sửa (phần này sẽ được nói kỹ trong chương sau).

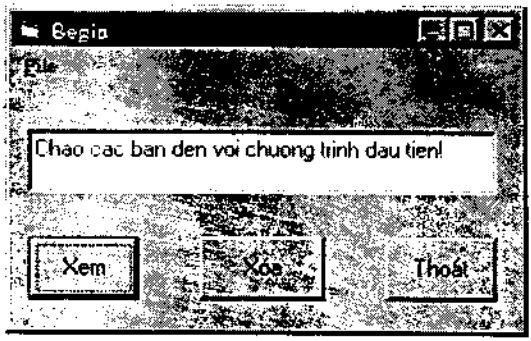
Ví dụ: Ta xem xét ví dụ tạo chương trình BEGIN như sau. Khi chương trình BEGIN chạy: một cửa sổ xuất hiện như hình bên dưới. Bên trong cửa sổ có ba nút lệnh (Command Button): Xem, Xóa và Thoát và một hộp soạn thảo (Text Box) tạm thời để trống.




**- Hình I.18 -**

➢ Hoạt động của chương trình

- Khi nhấn nút Xem, hộp soạn thảo sẽ xuất hiện lời chào: "Chào các bạn đến với chương trình đầu tiên!"




**- Hình I.19 -**

- Bấm nút Xóa, hộp soạn thảo bị xóa trắng như ban đầu

- Để kết thúc chương trình, ta nhấn nút Thoát.

### VI.4. MDI Form (Multiple Document Interface)

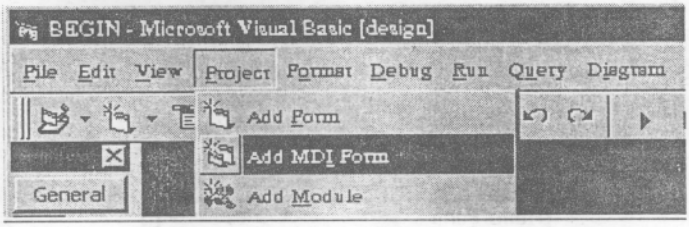
Đôi khi bạn muốn có một MDI Form, tức là một form có thể chứa nhiều form con bên trong. Dạng MDI Form này thường

được dùng trong các application như wordprocessor để có thể mở nhiều document cùng một lúc, mỗi document được hiển thị trong một form con.

➢ Cách tạo MDI Form

Để có một MDI Form ta có hai cách:

Cách 1: Dùng Menu: vào trình đơn **Project → chọn mục Add MDI Form** như hình sau:




**- Hình I.20 -**

Cách 2: Dùng công cụ: Trên thanh công cụ, ta chọn vào công cụ **Add Form** rồi chọn **MDI Form** như hình:



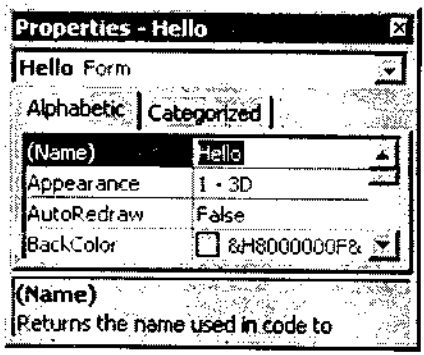

**- Hình I.21 -**

Mỗi project VB chỉ có thể có tối đa một MDI Form. Muốn một form trở thành một form con bạn **set property MDI Child** của nó thành **True**. Khi chạy (At run-time) bạn không thể dấu một MDIChild Form (hide - biến nó thành invisible), nhưng có thể thu nhỏ (minimize) nó. Nếu bạn thật sự muốn

dấu (hide) nó thì phải dùng một mẹo nhỏ là cho nó vị trí (top,left) số âm lớn hơn kích thước nó để nó nằm ngoài tầm hiển thị của form. Trong một chương trình dùng MDI Form, khi bạn chọn (Click) MDI Form nó không nhảy ra phía trước và che các form con, nó luôn luôn chứa các form con gói gọn trong cửa sổ của nó (bạn thử click minimize cửa sổ MDI Form sẽ thấy tất cả đều thu nhỏ thành một biểu tượng trên thanh Taskbar).

## VI.5. Lưu và chạy form

Tương tự như project, ta cũng có thể đổi tên hiển thị của form, bằng cách thay đổi thuộc tính Name trong cửa sổ Properties của form như hình:




- Hình I.22 -

Ví dụ ở đây ta đổi tên hiển thị của form là Hello (tên cũ là BEGIN), lúc này tên của cửa sổ Properties của form cũng trở thành Hello.

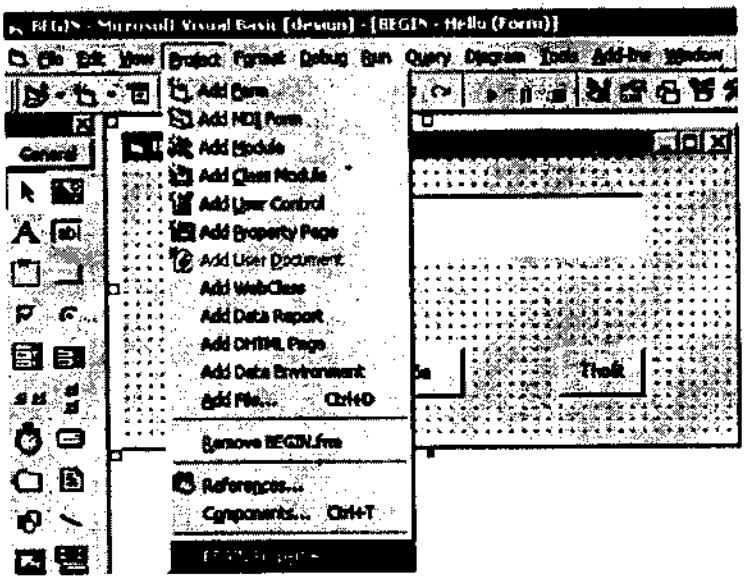
Sau khi tạo xong form, ta nên lưu form trước khi chạy. Cách lưu đã trình bày ở trên.

Để xem kết quả của Form. Ta tiến hành biên dịch và cho thực thi chương trình. Các bước được tiến hành như sau:

- Chọn form muốn chạy:

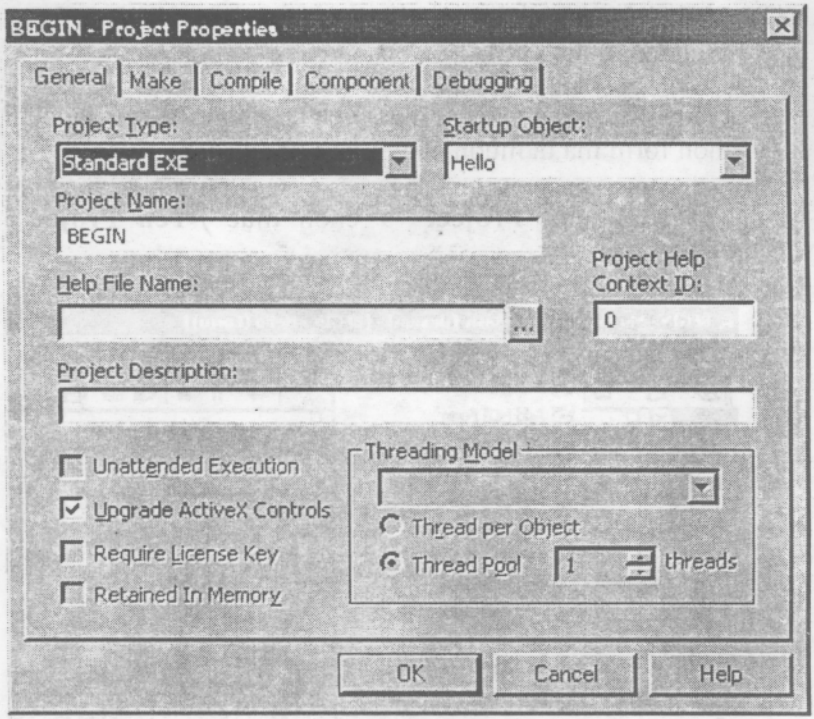
Trong trường hợp trong project có nhiều form, ta phải chọn form mà ta muốn chạy trước bằng cách:

Vào trình đơn **Project** → chọn mục **<Tên Project> Properties**



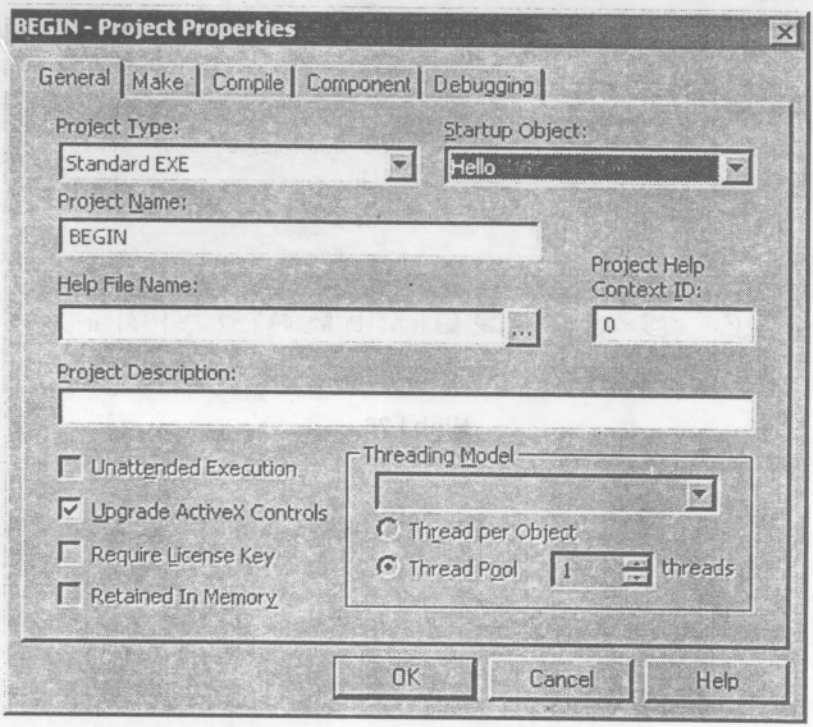

**- Hình I.23-**

Hộp thoại Project Properties xuất hiện



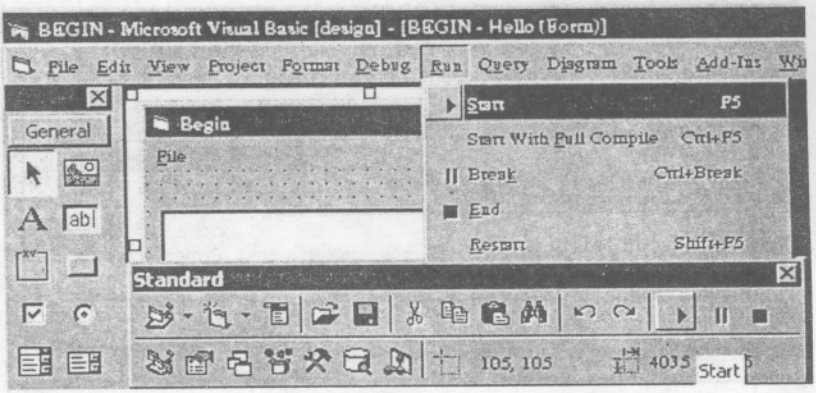

**- Hình I.24 -**

Trong mục **Startup Object** ta chọn form muốn chạy. Trong ví dụ này, ta chọn form có tên là Hello




**- Hình I.25 –**

- Cho thực thi form vừa chọn bằng cách dùng menu: chọn trình đơn **Run → Start**


BEGIN - Microsoft Visual Basic [design] - [BEGIN - Hello (Form)]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Wi
General
Begin
File
Start F5
Start With Full Compile Ctrl+F5
Break Ctrl+Break
End
Restart Shift+F5
Standard
105, 105
4035
Start

- Hình I.26 -

## Chương II

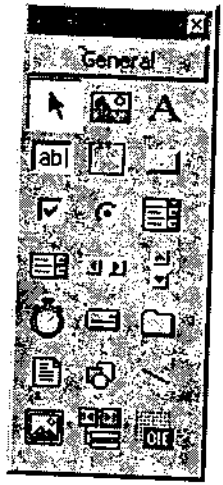
# FORM VÀ CÁC CONTROL TRONG VISUAL BASIC

## I. GIỚI THIỆU

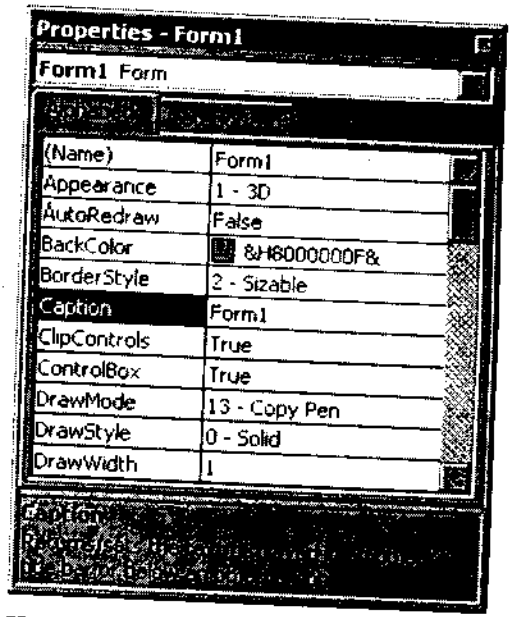
Bạn dùng các control để lấy các thông tin do người sử dụng nhập vào và hiển thị các thông tin lên màn hình. Một vài control thông dụng luôn xuất hiện trên toolbox như TextBox, CommandButton, ListBox ... gọi là các control thông dụng. Còn các control khác không có sẵn trên Toolbox gọi là các ActiveX control. Thông thường đây là các tập tin ActiveX (*.ocx). Việc tạo ra control riêng là một vấn đề rất lý thú trong lập trình. Bạn có thể viết riêng cho mình một control có các chức năng giống như các control mà Windows đã cung cấp sẵn cho chúng ta. Còn không thì có rất nhiều các control có các tính năng đặc biệt mà bạn có thể xin hay mua trên Internet. Bạn có thể tải về rồi sử dụng trong chương trình của mình như các control khác. Do đó bạn có thể update chương trình của mình mà không cần phải biên dịch lại chương trình.

Các thuật ngữ

- Hộp công cụ (**Toolbox**) cho phép chúng ta có thể thêm các điều khiển (**control**) vào form hay report.
- Cửa sổ thuộc tính (**Properties windows**): Danh sách các thuộc tính, các sự kiện ... của một đối tượng. Tùy thuộc vào đối tượng nào được chọn trên màn hình mà Properties windows sẽ thể hiện thuộc tính của đối tượng ấy. Các thuộc tính sẽ được chia thành các phần riêng biệt tùy vào loại của thuộc tính.




**Hình II. 1**: *Hộp Toolbox chứa các control*




**Hình II. 2**: *Properties windows*

Chúng ta có thể tác động đến một control thông qua các đặc tính sau:

- Thuộc tính (Properties): tập hợp các thuộc tính của control mà ta có thể thiết lập khi thiết kế hay khi chạy chương trình.
- Phương thức (Method): những gì control thực hiện được.
- Sự kiện (Event): những sự kiện mà control sẽ thông báo cho chương trình biết khi nó xảy ra với control. Khi một sự kiện xảy ra, chương trình sẽ tự động xử lý một hàm sự kiện (Event_Handler). Ví dụ : khi bạn click vào một CommandButton có tên Command1 thì chương trình tự động thực hiện các hành động trong hàm Command1_Click()

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

- **Form_Initialize**: sự kiện này xảy ra trước nhất. Trong chương trình sự kiện này chỉ xảy ra duy nhất một lần. Nếu khi thực hiện chương trình ta đóng mở form nhiều lần thì sự kiện này chỉ xuất hiện ở lần mở form đầu tiên.
- **Form_Load**: sự kiện này xảy ra mỗi khi ta khởi mở form. Trong một chương trình, nếu ta đóng mở form nhiều lần thì mỗi lần mở form, sự kiện Form_Load sẽ xuất hiện. Ta dùng sự kiện Form_Load để khởi tạo biến, control...

  Một form được coi như hình thành xong khi sự kiện Form_Load hoàn tất.

- **Form_Activate**: Sau khi form xử lý xong sự kiện Form_Load. Nếu không có gì thay đổi, form sẽ phát sinh ra sự kiện Activate.
- **Form_QueryUnload**: Khi người sử dụng nhấn vào nút X phía trên bên phải để đóng form thì nó tạo ra sự kiện QueryUnload như sau:

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

End Sub
```

trong đó tham số UnloadMode cho biết ai, hay tác vụ nào đóng form.

Ta có thể bắt sự kiện này để hỏi lại người sử dụng có chắc chắn đóng form hay không? Nếu người sử dụng chưa muốn đóng form thì ta thiết lập tham số Cancel bằng 1.

## II. CÁC THAO TÁC CƠ BẢN TRÊN FORM

### II.1. Thêm một control vào form

Muốn thêm một control vào form, chúng ta chỉ việc chọn biểu tượng của control tương ứng trên Toolbox, sau đó di chuyển chuột đến vị trí mong muốn trên form và kéo chuột vẽ một hình chữ nhật bằng đúng kích thước của control sẽ được tạo.

### II.2. Chọn nhiều control

Khi muốn di chuyển, xóa, thay đổi thuộc tính của nhiều control một lúc, đầu tiên chúng ta phải chọn nhiều control. Sau đó thực hiện các tác vụ trên. Bạn có thể chọn nhiều control một lúc bằng một trong các cách sau:

- Chọn từng đối tượng bằng cách click chuột vào các control trong khi nhấn giữ phím Shift.
- Sử dụng control Pointer trên ToolBox, sau đó kéo chuột tạo thành một hình chữ nhật, khi đó chúng ta sẽ chọn được các control trong phạm vi hình chữ nhật.

### II.3. Thay đổi kích thước, di chuyển, xóa ... control

Bạn có thể thay đổi kích thước, di chuyển, xóa ... một control bằng một trong các cách sau:

- Nhấp chọn control và sử dụng chuột để thay đổi kích thước, di chuyển, xóa ... các control bằng cách đưa chuột vào các cạnh của các control và kéo chuột (thay đổi kích thước, di chuyển) hay nhấn phím Del (xóa).
- Chọn control sau đó thay đổi các thuộc tính Left, Top, Width, Height... trong cửa sổ Properties.

### II.4. Thiết lập lại thuộc tính cho các control

Bạn có thể chỉnh lại font chữ, kích thước chữ, canh lề, màu sắc, bóng mờ, dòng chữ hiển thị ... của các control thông qua cửa sổ thuộc tính (Properties). Bạn có thể ẩn hay hiện cửa sổ này bằng menu View/ Properties Window hay phím F4 .

### II.5. Thứ tự tab của các control

Khi nhập dữ liệu vào form, bạn có thể di chuyển focus giữa các control bằng phím Tab. Muốn thiết lập thứ tự này bằng cách sử dụng thuộc tính Tab Order của control.

## III. CÁC THUỘC TÍNH CHUNG

### III.1. Thuộc tính Name

Mỗi control trong form có một thuộc tính Name (tên) để phân biệt với các control khác. Thuộc tính này còn được dùng để truy xuất đến control đó. Cách đặt tên cho control bạn nên tuân theo qui tắc đặt tên của Visual Basic (ví dụ: trên form có một control textbox cho người dùng nhập họ lót thì bạn đặt thuộc tính name cho textbox này là "txtHoLot").

### III.2. Thuộc tính định dạng

| Thuộc tính | Giải thích |
|---|---|
| Alignment | Canh lề cho dòng chữ (Text, Caption) trong control |
| Appearance | Xác định control là 3D hay không |
| ForeColor | Màu chữ trên control |
| BackColor | Màu nền trên control |
| Font | Font của control |
| Enable | Nếu thuộc tính này có giá trị FALSE, control sẽ |

| | |
|---|---|
| | không sử dụng được, dù có thể nhìn thấy. |
| Visible | Ẩn hay hiện control. |
| Top | Qui định góc trái trên của control |
| Left | |
| Width | Qui định kích thước control |
| Height | |
| ToolTextTip | Ghi chú nội dung của control. Dòng chữ này sẽ hiện ra khi ta dừng chuột ngay trên control. |

### III.3. Thuộc tính giá trị (Value)

Khi thao tác với các control, bạn muốn lấy hay gán giá trị cho một control. Tùy theo loại control mà chúng ta có thể lấy các giá trị của control qua các thuộc tính khác nhau. Bảng dưới đây mô tả thuộc tính giá trị cho mỗi control.

| Control | Thuộc tính giá trị |
|---|---|
| Label | Caption |
| TextBox | Text |
| OptionButton | Value |
| CheckBox | Value |
| Frame | Caption |
| ListBox | Text |
| ComboBox | Text |
| HScrollBar | Value |
| VScrollBar | Value |
| DriveListBox | Path |

| | |
|---|---|
| FileListBox | FileName |
| Line | Visiable |
| Shape | Shape |
| PictureBox | Picture |
| Image | Picture |

Tuy nhiên các thuộc tính này được sử dụng thường xuyên nên Visual Basic cho phép chúng ta có thể dùng tên của control để truy cập đến thuộc tính giá trị.

Ví dụ:
txtHoLot.Text = "Nguyễn Thùy" có thể được viết như sau
txtHoLot = "Nguyễn Thùy"

## IV. GIỚI THIỆU CÁC CONTROL CƠ BẢN TRONG TOOLBOX

### IV.1. Pointer (Đối tượng lựa chọn)




Đối tượng này cho phép chúng ta có thể chọn một hay nhiều đối tượng trên màn hình thiết kế. Sau khi chọn đối tượng xong, chúng ta có thể cho các đối tượng đó di chuyển, thay đổi các thuộc tính... Chúng ta có thể chọn nhiều đối tượng bằng cách kéo chuột từ góc trái trên đến góc phải dưới, khi đó, các đối tượng thuộc phạm vi hình chữ nhật đó xem như được chọn. Ngoài ra chúng ta có thể chọn lần lượt các control bằng cách giữ phím Shift.

### IV.2. Label




Mục đích của Label là hiển thị một đoạn văn bản lên form (Label.Caption=đoạn-văn-bản). Label khác với textbox ở chỗ nó không cho người sử dụng nhập một

đoạn văn bản vào như là textbox.

### IV.3. TextBox




TextBox được sử dụng để nhập vào một đoạn văn bản hay để hiển thị một đoạn văn bản.

#### IV.3.1. Thuộc tính Text

Thuộc tính này chứa nội dung của TextBox. Thông thường bạn có thể nhập vào tối đa 2048 ký tự vào TextBox. Nếu bạn thiết lập thuộc tính MultiLine bằng True, bạn có thể nhập vào 32k ký tự.

#### IV.3.2. Thuộc tính MultiLine

Khi thuộc tính MultiLine bằng True, Visual Basic sẽ tự động xuống hàng khi phần nội dung của TextBox quá dài.

#### IV.3.3. Thuộc tính Alignment

Thuộc tính này sẽ điều chỉnh chế độ canh lề của chuỗi là canh lề trái (0 - Left Justify), Giữa (1 – Center), hay Phải (2 - Right Justify).

#### IV.3.4. Chọn khối trong TextBox

Bạn có thể xác định và xử lý khối văn bản được chọn trong TextBox bằng các thuộc tính sau:

- **SelLength** — trả về hay thiết lập số ký tự trong khối văn bản được chọn.

- **SelStart** — trả về hay thiết lập vị trí bắt đầu của khối văn bản được chọn; xác định vị trí của điểm chèn nếu không có văn bản nào được chọn.

- **SelText** — trả về hay thiết lập nội dung của đoạn văn bản được chọn. Nếu không có ký tự nào được chọn, giá trị của thuộc tính này là chuỗi rỗng("").

Các thuộc tính này không có giá trị khi form đang ở chế độ thiết kế.

Đôi khi ba thuộc tính này dùng với đối tượng Clipboard của Windows. Đối tượng Clipboard có các thuộc tính và phương thức sau:

- **SetText**: gán giá trị vào Clipboard.
- **GetText**: có giá trị là chuỗi ký tự đang có trong Clipboard.
- **Clear**: xóa nội dung có trong Clipboard.

Ví dụ:
```
Clipboard.SetText  Text1.SelText
Text1.SelText = Clipboard.SetText
Clipboard.Clear
```

### IV.3.5. Tạo một TextBox để nhập password

Một TextBox cho phép người sử dụng đánh password vào, các ký tự được nhập vào sẽ được thay thế bằng một ký tự nào đó, ký tự * chẳng hạn.

Để thực hiện điều này bạn set thuộc tính PasswordChar trong cửa sổ Properties bằng một ký tự (ví dụ ký tự *) (ký tự này sẽ hiển thị lên TextBox khi người dùng nhập password). Thuộc tính MaxLength, xác định số ký tự nhiều nhất mà bạn có thể

nhập vào ô TextBox, nếu bạn nhập nhiều hơn chiều dài qui định, Visual Basic sẽ phát ra tiếng Beep và không nhận thêm ký tự nào nữa.

### IV.3.6. Lọc phím trong TextBox

Bạn có thể dùng sự kiện KeyPress để điều khiển dữ liệu nhập vào trong TextBox. Thủ tục xử lý sự kiện KeyPress dùng một tham số là **KeyAscii**. Tham số này là một số nguyên tượng trưng cho một mã số ASCII của ký tự được đánh vào.

Ví dụ: Bạn muốn người sử dụng chỉ được nhập số vào trong TextBox. Khi đó bạn sử dụng phương pháp như sau: nếu ký tự được nhập không thuộc phạm vi số thì gán tham số KeyAscii là 0. TextBox trong ví dụ này có thuộc tính name là txtEnterNums.

```
Private Sub txtEnterNums_KeyPress (KeyAscii As Integer)
    If KeyAscii < Asc("0") Or KeyAscii > Asc("9") Then
        KeyAscii = 0   ' Hủy bỏ phím được nhập vào.
        Beep           ' Phát âm thanh beep báo lỗi.
    End If
End Sub
```

Tuy nhiên sự kiện KeyPress chỉ phát sinh khi có các ký tự được đánh vào, còn các ký tự điều khiển thì sự kiện này không phát sinh. Muốn bắt được sự kiện nhấn các ký tự điều khiển bạn sử dụng sự kiện KeyDown, KeyUp.

Thủ tục xử lý sự kiện KeyDown và KeyUp có hai tham số là KeyCode và Shift. Tham số KeyCode sẽ xác định phím nào được nhấn, còn tham số Shift sẽ có giá trị như sau:

| Hằng | Giá trị | Mô tả |
|---|---|---|
| vbShiftMask | 1 | Mặt nạ bit phím SHIFT |
| VbCtrlMask | 2 | Mặt nạ bit phím CTRL |
| VbAltMask | 4 | Mặt nạ bit phím ALT |

Ví dụ:

Khi người sử dụng nhấn {F1} thì Key Code =111 và Shift=0

Khi người sử dụng nhấn {Ctrl+F1} thì Key Code = 111 và Shift = 2

Khi người sử dụng nhấn {Ctrl+Shift+F1} thì Key Code = 111 và Shift = 3 (1+2=3)

### IV.3.7. Tạo một TextBox chỉ hiển thị

Khi bạn muốn người sử dụng không thể nhập hay thay đổi dữ liệu trong TextBox, bạn thiết lập thuộc tính Locked bằng True. Với thuộc tính Locked bằng True, lệnh Copy vẫn hoạt động trong khi lệnh Cắt (Cut)và Dán (Paste) không hoạt động với TextBox. Thuộc tính chỉ có hiệu quả về mặt giao diện người dùng (*user interaction* ) khi chạy chương trình. Bạn vẫn có thể thay đổi nội dung của thuộc tính Text của the TextBox bằng lệnh.

### IV.3.8. In dấu nháy kép (") trong TextBox

Thỉnh thoảng bạn có thể đưa lên TextBox một dòng có dấu nháy kép như sau.

She said, "You deserve a treat!"

Bởi vì trong Visual Basic chuỗi được xác định là các ký tự được bao bởi dấu nháy kép (") nên chúng ta khó hiển thị dấu nháy kép (") vào trong TextBox với phương pháp bình thường. Muốn hiển thị dấu nháy kép trong chuỗi thì ta sử dụng hai dấu nháy kép liên tục. Ví dụ, muốn gán câu trên cho một TextBox ta viết như sau:

```
Text1.Text = "She said, ""You deserve a treat!"" "
```

Hay bạn có thể dùng ký tự thứ 34 trong bảng mã ASCII để thể hiện dấu nháy kép:

```
Text1.Text = "She said, " & Chr(34) + "You deserve a treat!" & Chr(34)
```

### IV.4. CommandButton




CommandButton là một nút ta có thể nhấn vào (sự kiện Click). Khi đó nó sẽ thực hiện một hành động nào đó kéo theo.

### IV.5. OptionButton




OptionButton còn gọi là RadioButton. Các OptionButton thường được nhóm thành một nhóm. Điều này bởi vì trong một nhóm, chỉ có duy nhất một OptionButton được chọn. Khi một OptionButton được chọn thì các OptionButton còn lại trong nhóm sẽ mất chọn. Nhóm này thường được bao bởi một Frame. Muốn đặt OptionButton (RadioButton) lên một Frame ta chỉ việc cắt (cut) một đối tượng OptionButton (RadioButton) và dán (paste) lên Frame trên. Hay ta có thể chọn Frame, sau đó vẽ một OptionButton lên đó.

Muốn biết một OptionButton có nằm trong Frame nào đó hay không ta chỉ việc di chuyển Frame, nếu OptionButton nào bị kéo theo thì OptionButton thuộc Frame đó.
Giá trị trả về của một OptionButton có thể được lấy thông qua thuộc tính Value của nó. Nếu OptionButton được chọn (Checked) thì thuộc tính Value sẽ có giá trị là True. Ngược lại nếu OptionButton không được chọn (Uncheck) thì thuộc tính Value sẽ có giá trị là False.
Thuộc tính Value có thể được sử dụng để gán giá trị cho OptionButton.

## IV.6. CheckBox




Các thuộc tính và cách sử dụng của CheckBox cũng giống như OptionButton. Các giá trị trả về cũng giống như OptionButton. Tuy nhiên chúng ta chú ý một số điểm khác nhau giữa CheckBox và OptionButton như sau:

CheckBox là control cho phép chúng ta có thể chọn hay không chọn nhiều CheckBox. Đặc điểm của CheckBox này khác với OptionButton.

Điểm khác biệt thứ hai so với Option là thuộc tính Value của CheckBox có đến ba giá trị trạng thái có thể có là:

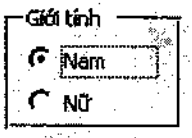
| Trạng thái | Giá trị | Hằng |
|---|---|---|
| Unchecked | 0 | vbUnchecked |
| Checked | 1 | vbChecked |
| Unavailable | 2 | vbGrayed |

Trạng thái Grayed của CheckBox được dùng theo chủ ý riêng của người lập trình.

## IV.7. Frame




Control này cho phép chúng ta thể hiện một nhóm các lựa chọn. Control được dùng để gom nhóm một số control khác. Thông thường Frame dùng để gom các OptionButton lại thành một nhóm. Khi đó chúng ta chỉ có thể chọn duy nhất một OptionButton trong nhóm mà thôi. Nếu chúng ta đã chọn một OptionButton, sau đó chọn OptionButton khác thì OptionButton đã được chọn trước đó sẽ không được chọn nữa. Ví dụ một cách dùng của Frame:




## IV.8. ListBox




ListBox được sử dụng để hiển thị một loạt các thông tin dưới dạng các hàng, mỗi hàng được gọi là một Item. Người sử dụng chỉ có thể chọn mà không được phép nhập giá trị vào trong ListBox. Chúng ta có thể chọn nhiều item trong listbox.

Sau đây là các thuộc tính, phương thức và cách sử dụng quan trọng của ListBox:

### IV.8.1. Thuộc tính Style

Thuộc tính Style có hai giá trị tương ứng với hai loại ListBox như sau:

- **Standard** (Style = 0): Đây là giá trị mặc định cho ListBox.
- **Check Box** (Style = 1): Loại này có thêm một check box bên trái mỗi item trong danh sách (list).

### IV.8.2. Thuộc tính List

Đây là tập hợp các Item trong ListBox. Đơn giản hơn có thể xem đây là mảng các chuỗi hiển thị trong ListBox. Chỉ số của mảng này bắt đầu từ 0. Để truy xuất giá trị của Item thứ index ta sử dụng cú pháp sau:
**<Tên ListBox>.List(<index>)**

Ví dụ: bạn muốn lấy giá trị của Item thứ ba (index=2) trong ListBox và gán cho TextBox, bạn viết lệnh như sau:

```
Text1.Text = List1.List(2)
```

### IV.8.3. Thuộc tính ListIndex

Thuộc tính ListIndex cho biết vị trí item được chọn trên list. Thuộc tính này dùng để lấy hay gán chỉ số (index) của item đang được chọn của ListBox. Thuộc tính này chỉ có tác dụng khi chương trình đang chạy (run time). Khi chúng ta thiết lập thuộc tính ListIndex cho ListBox, Visual Basic cũng đồng thời tạo ra sự kiện Click cho ListBox.

Giá trị của thuộc tính này bằng 0 nếu item đầu tiên được chọn, 1 nếu item kế tiếp được chọn.... ListIndex bằng – 1 nếu không có item nào được chọn.

**Chú ý:** *Thuộc tính NewIndex cho phép chúng ta có thể biết được item cuối cùng nào được thêm vào ListBox. Thuộc tính này rất hữu ích khi chúng ta chèn một item vào một list có thứ tự (sorted list).*

### IV.8.4. Thuộc tính ListCount

Thuộc tính này xác định số lượng các Item trong ListBox. Ví dụ đoạn lệnh sau sẽ cho chúng ta biết số lượng của các item trong ListBox thông qua TextBox:

```
Text1.Text = "Bạn có " & List1.ListCount & " Item trong ListBox"
```

### IV.8.5. Thuộc tính Text

Thuộc tính Text sẽ cho biết giá trị một Item mà người sử dụng đang chọn. Thông thường, cách dễ nhất để lấy giá trị của item đang được chọn trong ListBox bằng thuộc tính Text.

### IV.8.6. Thuộc tính ItemData

Thuộc tính ItemData là một mảng số long. Số phần tử của mảng ItemData bằng số phần tử trong List. Thuộc tính này dùng để thêm các dữ liệu phụ bên trong ListBox và ComboBox.

### IV.8.7. Sự kiên Click và Double-Click

Sự kiện Click sẽ chọn một Item trên ListBox.

Chúng ta giả sử có một ListBox và một CommandButton dùng với ListBox trên cùng trên một form. Thủ tục xử lý sự kiện Click của CommandButton sẽ làm xuất hiện một MessageBox thông báo item nào trong ListBox được chọn.

Bạn muốn khi Double-clicking trên list có tác dụng tương đương với việc chọn một Item và click vào CommandButton. Để có thể thực hiện được công việc trên bạn viết sự kiện Click của ListBox như sau:

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

### IV.8.8. Thêm một item vào ListBox

Khi thiết kế form, chúng ta có thể đặt các item vào trong ListBox bằng cách sử dụng thuộc tính List. Ta nhập từng item vào ô ListBox, mỗi item cách nhau bởi một hàng. Để xuống hàng, ta sử dụng Ctrl+Enter.

Để thêm một Item vào một ListBox khi chương trình đang chạy (run-time), chúng ta dùng phương thức AddItem:

**<Tên ListBox>.AddItem *item*[, *chỉ số*]**

| Tham số | Mô tả |
|---|---|
| Item | Chuỗi cần thêm vào list. |
| Chỉ số | Chỉ định vị trí item mới sẽ được thêm vào list. Nếu ta không chỉ định tham số này, item sẽ được thêm vào cuối (hay vị trí thích hợp nếu list được sắp xếp). |

Đoạn lệnh sau sẽ thêm tên bốn quốc gia "Germany," "India," "France," và "USA" vào ListBox có tên List1 khi form chứa ListBox được khởi tạo:

```
Private Sub Form_Load ()
  List1.AddItem "Germany"
  List1.AddItem "India"
  List1.AddItem "France"
  List1.AddItem "USA"
End Sub
```

### IV.8.9. Thuộc tính Sorted

Bạn có thể chỉ định cho các Item đã có trong ListBox được sắp xếp theo thứ tự alphabet bằng cách thiết lập thuộc tính Sorted thành True. Việc sắp thứ tự này sẽ không phân biệt chữ hoa và chữ thường, ví dụ, từ "japan" và "Japan" là như nhau.

### IV.8.10. Xóa một item trong ListBox

Bạn có thể dùng phương thức RemoveItem để xóa một item trong ListBox. RemoveItem cần một tham số là *index* (vị trí nào sẽ bị xóa):

**<Tên ListBox>.RemoveItem *index***

Các tham số này cũng như trong phần AddItem

Ví dụ: để xóa Item đầu tiên trong ListBox ta có đoạn lệnh sau:

```
List1.RemoveItem 0
```

Để xóa tất cả các Item trong ListBox chúng ta sử dụng phương thức Clear.

Ví dụ: List1.Clear

Khi xóa một item trong ListBox, chỉ số index của các Item sẽ thay đổi. Ví dụ: Nếu chúng ta xóa item có index bằng 0, thì Item có index bằng 1 sẽ chuyển sang index mới là 0...

### IV.8.11. Tạo ListBox có nhiều cột và ListBox có nhiều sự lựa chọn (Multiple-Selection)

Thuộc tính Columns cho phép bạn xác định số cột trong một ListBox. Thuộc tính này có thể có các giá trị sau.

| Giá trị | Giải thích |
|---|---|
| 0 | ListBox một cột với thanh cuộn dọc |
| 1 | ListBox một cột với thanh cuộn ngang |
| >1 | ListBox nhiều cột với thanh cuộn dọc |

Bạn có thể cho phép người sử dụng có thể chọn nhiều item trong ListBox bằng cách thiết lập thuộc tính MultiSelect mang một trong các giá trị sau.

| Giá trị | Hằng chuỗi | Giải thích |
|---|---|---|
| 0 | None | ListBox bình thường |
| 1 | Simple multiple selection | Click hay nhấn Spacebar để chọn hay không chọn một item trong list. |
| 2 | Extended multiple selection | Dùng Shift+ click hay Shift+ phím mũi tên để chọn các Item ở giữa vị trí chọn trước đó và vị trí chọn hiện thời. Ctrl+ click để chọn hay không chọn một item trong list. |

### IV.8.12. Xác định mục được chọn trong một MultiSelect ListBox

Bạn có thể sử dụng thuộc tính Selected của ListBox để kiểm tra xem một mục nào đó có được chọn hay không? Nếu mục nào được chọn thì giá trị **<Tên ListBox>.Selected (index)** có giá trị là True, ngược lại sẽ có giá trị là False.

## IV.9. ComboBox




— ComboBox được sử dụng để hiển thị một loạt các thông tin dưới dạng các hàng, mỗi hàng được gọi là một Item. Tuy nhiên ComboBox chỉ hiện ra một dòng, bạn muốn xem các dòng khác thì click vào nút mũi tên

ở bên cạnh. Khi đó sẽ có một **list** hiện ra bên dưới. ComboBox cho phép người sử dụng nhập giá trị vào.
Một số thuộc tính và phương thức đáng chú ý:

### IV.9.1. Thuộc tính Style

Thuộc tính Style có ba giá trị tương ứng với ba loại ComboBox như sau:

- **Drop-down ComboBox** (Style = 0 ): Đây là giá trị mặc định cho ComboBox, một drop-down ComboBox là một ComboBox bình thường. Người sử dụng có thể hoặc đánh chuỗi văn bản trực tiếp vào (giống như textbox) hay chọn vào mũi tên chỉ xuống ở bên phải của ComboBox để chọn từ danh sách.
- **Simple ComboBox** (Style = 1): Loại này giống Drop-Down ComboBox, tuy nhiên có thêm một ListBox bên dưới. Tuy nhiên ListBox này luôn hiện ra, không cần người sử dụng phải click vào dấu mũi tên. Người sử dụng có thể nhập chuỗi trực tiếp hay chọn từ ListBox bên dưới. Cũng giống như Drop-Down ComboBox, Simple ComboBox có thể cho người sử dụng nhập vào một giá trị không có trong ListBox.
- **Drop-down List** (Style = 2) loại này giống như một ListBox ở chỗ chỉ cho phép người sử dụng chọn các giá trị có trong List. Tuy nhiên, List trong Drop-down ListBox chỉ hiện ra khi người sử dụng nhấn vào mũi tên ở bên phải ComboBox.

### IV.9.2. Sự kiện Change

Sự kiện này xảy ra khi người sử dụng nhập một giá trị vào ComboBox. Chú ý: khi người sử dụng chọn một item trong phần list thì nội dung của ComboBox sẽ thay đổi nhưng sự kiện Change sẽ không phát sinh.

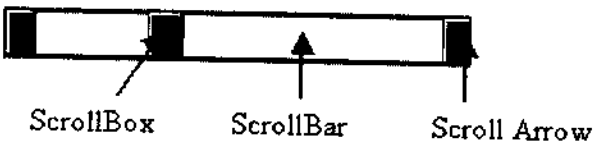
Các thuộc tính và sự kiện khác giống như ListBox.

## IV.10. VScrollBar và HScrollBar




Đối tượng này hơi khác với các đối tượng mà chúng ta đã khảo sát ở trên. Các đối tượng này được tự động thêm vào TextBox, ComboBox, MDI Form... khi kích thước dữ liệu lớn hơn kích thước định sẵn trên màn hình. Hai control này được gọi tên chung là ScrollBar.

Trong ScrollBar có các thành phần như sau:




ScrollBar có các sự kiện như sau:

| Event | Description |
|---|---|
| Change | Xảy ra sau khi scrollbox được di chuyển |
| Scroll | Xảy ra khi scrollbox được di chuyển. sự kiện này không xảy ra khi hai mũi trên (scroll arrow) hay thanh trống (scroll bar) bị nhấn. |

## IV.11. Timer




Control dùng để hiển thị tự động thực hiện một công việc nào đó sau một khoảng thời gian nhất định.

### IV.12. DriveListBox




Control dùng để hiển thị và chọn các ổ đĩa của máy.

### IV.13. FileListBox




Control dùng để hiển thị và chọn các tập tin trong một thư mục.

### IV.14. Line




Control này dùng để vẽ một đường thẳng trên report hay form. Mục đích của control này là để trang trí.

### IV.15. Shape




Control này dùng để vẽ một đa giác trên report hay form. Mục đích của control này là để trang trí.

### IV.16. PictureBox




Control này dùng để hiển thị hình vẽ lên form.

Gán một ảnh vào control PictureBox: Hình ảnh có thể được gán (load) vào PictureBox khi chúng ta đang thiết kế bằng cách thiết lập thuộc tính Picture trên cửa sổ Properties, hay khi chạy chương trình (run time) bằng cách sử dụng thuộc tính Picture và hàm LoadPicture.

```
Set Picture1.Picture =
LoadPicture("c:\Windows\Winlogo.cur")
```

Để xóa một hình khỏi control PictureBox, bạn dùng hàm LoadPicture mà không có tham số tên file. Ví dụ:

```
Set Picture1.Picture = LoadPicture
```

Bạn có thể cắt và dán hình từ một chương trình xử lý ảnh nào đó lên một đối tượng PictureBox.

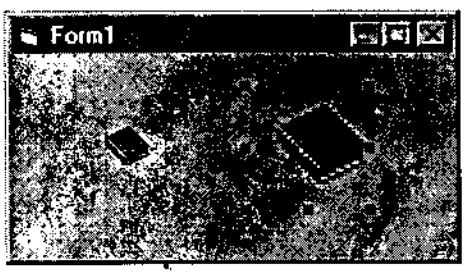
### IV.17. Image

Control này dùng để hiển thị hình vẽ lên form.




Cách sử dụng của đối tượng này giống như PictureBox.

Sự khác nhau giữa control Image với PictureBox là control Image sử dụng ít tài nguyên hệ thống hơn và nó vẽ lại nhanh hơn PictureBox, tuy nhiên nó hỗ trợ ít các thuộc tính, sự kiện, phương thức hơn PictureBox. Cả hai loại control đều hỗ trợ cùng các format file ảnh như nhau. Tuy nhiên bạn chỉ có thể kéo dãn (gán thuộc tính Stretch = True) ảnh trong Image bằng với kích thước bạn thiết kế trên form, còn đối tượng Picture thì không được.




**Hình II. 3**: *Một đối tượng Image được kéo (Stretch)*

## IV.18. Data




Control này cho phép kết nối với một CSDL, khi đó control này tương ứng với một recordset. Sau đó kết buộc dữ liệu (Data binding) các control khác trên form với control này. Lúc đó các control được kết buộc sẽ hiển thị giá trị tương ứng trong CSDL.

## Chương III

# THIẾT KẾ MENU

Các ứng dụng có thể thực hiện một số công việc. Để sử dụng ứng dụng một cách hiệu quả, nó phải:

- Thông báo cho người dùng các công việc sẵn có.
- Cung cấp cho người dùng một biện pháp để chỉ định công việc sẽ được thực hiện.
- **Một ứng dụng thực hiện các công việc trên thông qua một lệnh đơn (menu).**

Các lệnh đơn cung cấp một cách tiện dụng và nhất quán để gom nhóm các lệnh và giúp người dùng truy cập chúng một cách dễ dàng. Nó cho phép một ứng dụng đưa ra cho người dùng một cách thức để gọi các hành động khác nhau.

Ví dụ: Thanh lệnh đơn của chương trình Visual Basic:




- Hình III.1 -

Việc thiết kế menu gồm hai quá trình:

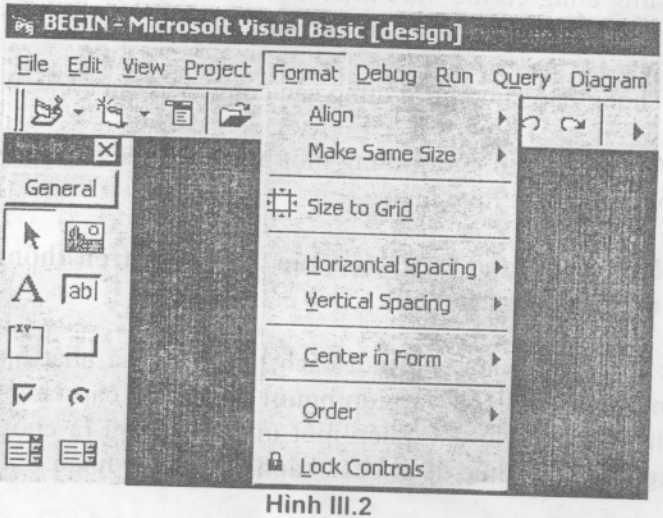
(1) Thiết kế giao diện

(2) Viết lệnh cho các thủ tục đáp ứng biến cố khi chọn mục menu.

Thiết kế đúng loại lệnh đơn sẽ khiến ứng dụng thân thiện với người dùng hơn. Visual Basic cho phép xây dựng sáu cấp lệnh đơn. Nhưng nếu dùng quá nhiều cấp lệnh đơn, có thể khiến

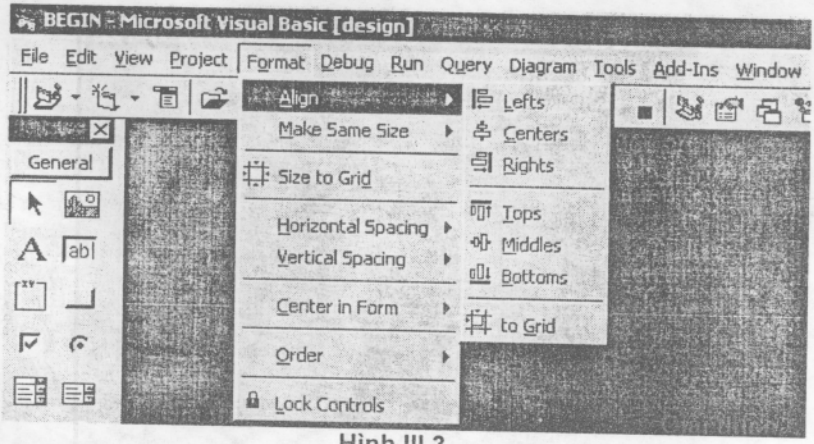
ứng dụng trở nên phức tạp và dễ gây lẫn lộn đối với người sử dụng. Thường thì ta thường dùng hai đến ba cấp.

Ký hiệu ▶ theo sau mục lệnh đơn nêu rõ có một lệnh đơn con tồn tại dưới mục lệnh đơn đó.




Hình III.2

Có hai loại menu ta thường gặp: **(drop-down)** menu và **(pop-up)** menu (là menu **hiện lên**). Ta dùng drop-down menu làm Menu chính cho chương trình. Thông thường nó nằm ở phía trên chóp màn hình. Nằm dọc theo chiều ngang là Menu Bar, nếu ta click lên một lệnh (menu command) trong thanh lệnh đơn (menu bar) thì chương trình sẽ thả xuống một menu với những lệnh con (MenuItems) nằm dọc theo chiều thẳng đứng. Nếu ta click chuột lên MenuItems nào có dấu hình tam giác nhỏ bên phải ▶ thì chương trình sẽ thả xuống một menu như trong hình dưới đây:




**Hình III.3**

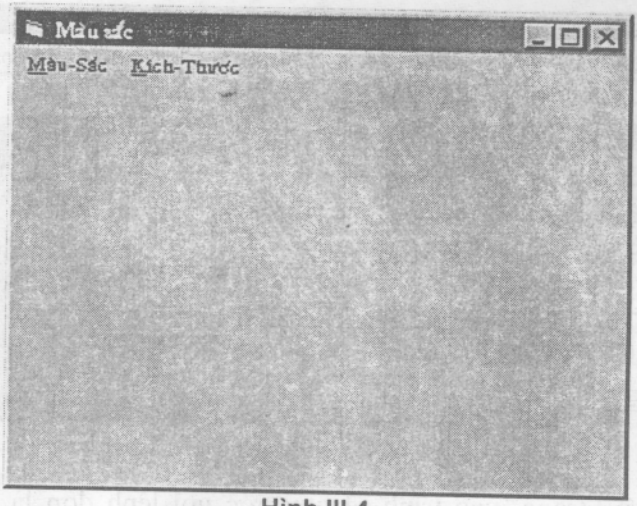
Từ đây, trong giáo trình sẽ quy ước gọi lệnh đơn là **Menu command** và lệnh con là **MenuItems**

Ta sẽ minh họa phần này bằng chương trình sau:

Chương trình sẽ cung cấp một Menu gồm hai trình đơn: "Màu-Sắc" và "Kích-Thước". Trình đơn "Màu-Sắc" cho phép ta thay đổi màu nền form và trình đơn "Kích-Thước" cho phép ta thay đổi kích thước form.

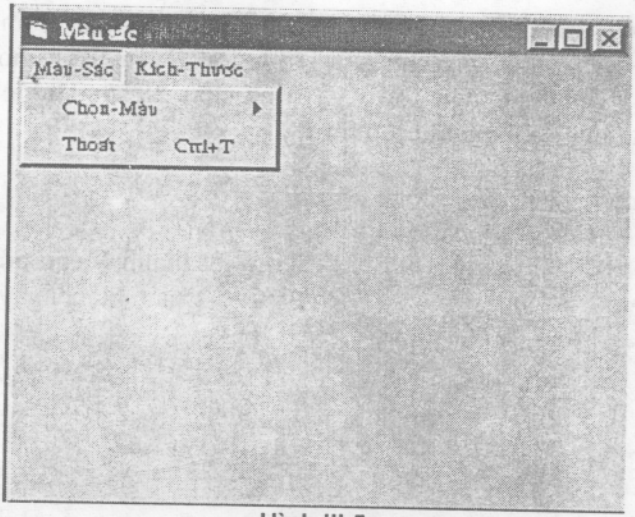
Chương trình sẽ hoạt động như sau:

- Khi thực hiện chương trình, một thanh Menu gồm hai mục: "Màu-Sắc" và "Kích-Thước" xuất hiện ngay dưới thanh tiêu đề của cửa sổ



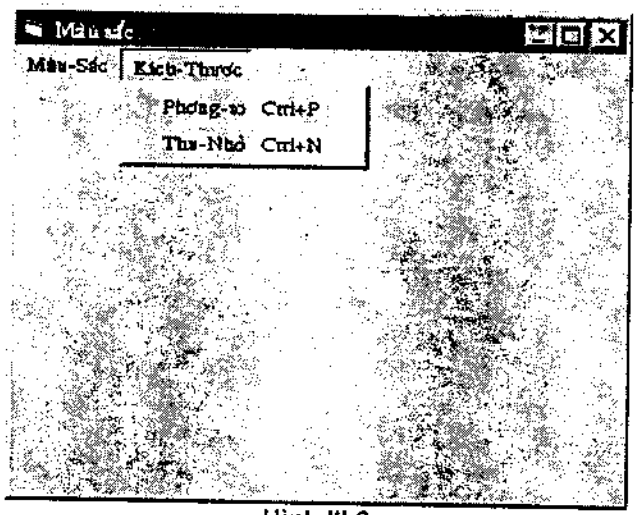

Hình III.4

- Trong trình đơn ""Màu-Sắc" có hai mục: "Chọn-màu" và "Thoát"



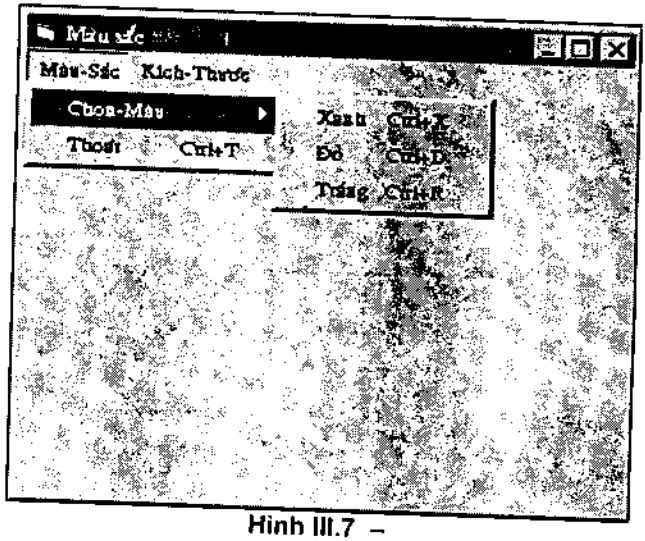

Hình III.5

- Trong trình đơn "Kích-thước" có hai mục: "Phóng-to" và "Thu-nhỏ"




Hình III.6

- Khi chọn mục "Chọn-Màu" trong trình đơn "Màu-Sắc", một trình đơn con (Menu Popup) xuất hiện tiếp với ba mục: "Xanh", "Đỏ", "Trắng". Đây là các màu ta có thể chọn làm màu nền form.




Hình III.7 –

- Khi ta chọn một trong hai mục trong trình đơn "Kích-Thước", kích thước form thay đổi.

- Khi chọn mục "Thoát" trong trình đơn "Màu-Sắc", chương trình kết thúc.

## I. THIẾT KẾ GIAO DIỆN

Trong Visual Basic, ta dùng bộ soạn thảo lệnh đơn (Menu Editor) để thiết kế các lệnh đơn cho chương trình.

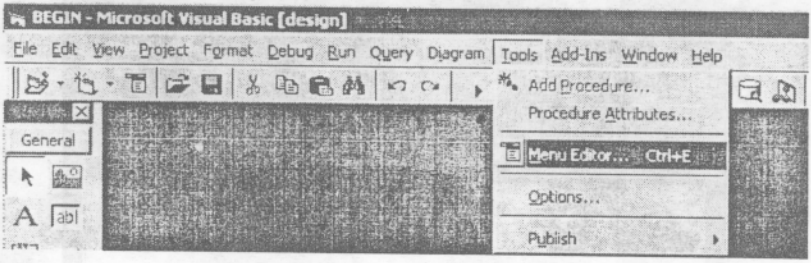
### I.1. Các bước tiến hành

(1) Menu thuộc về một form, do đó, trước hết ta chọn một form để làm việc (ở đây ta làm việc trong phần Design của form chứ không làm việc trên phần Code).

Ví dụ: ta minh họa phần này trên form có thuộc tính name là **frmMenu**, thuộc tính caption là "**Màu sắc**"

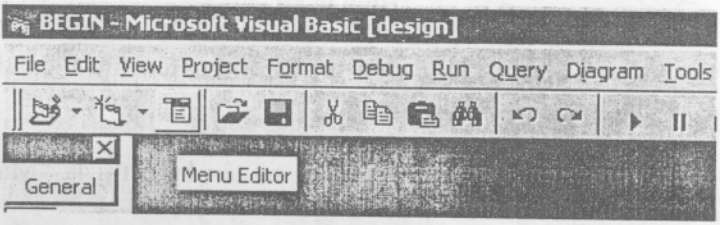
(2) Gọi **Menu Editor** bằng cách hai cách:

Cách 1: Dùng menu: Vào trình đơn **Tools** -> chọn mục **Menu Editor** như hình sau:




**Hình III.8**

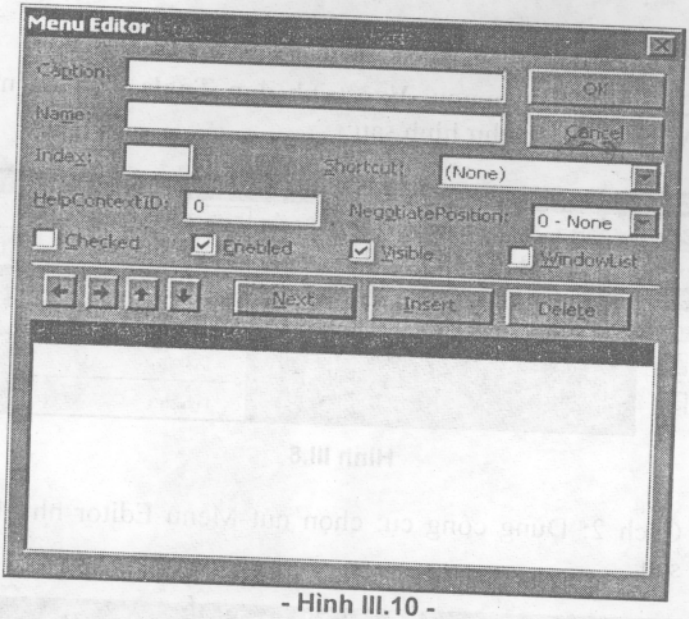
Cách 2: Dùng công cụ: chọn nút Menu Editor như hình sau




**Hình III.9**

(3) Tạo các đối tượng Menu:

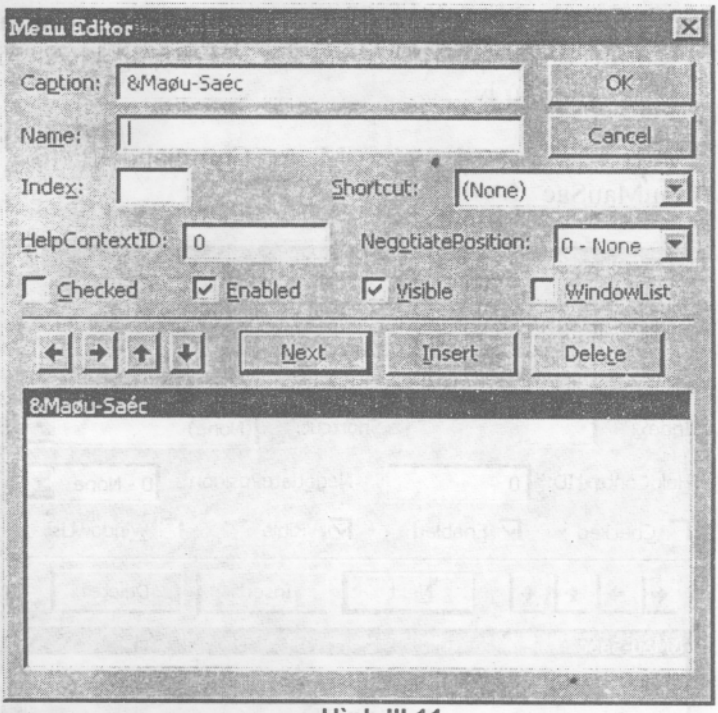
Ta quan sát hình sau:




**- Hình III.10 -**

- Mục Caption: Nơi hiển thị Caption (tên lúc hiển thị) của các đối tượng Menu.

Ví dụ: Ta có một Menu Command có tên hiển thị là "Màu-Sắc" như hình vẽ




**- Hình III.11 -**

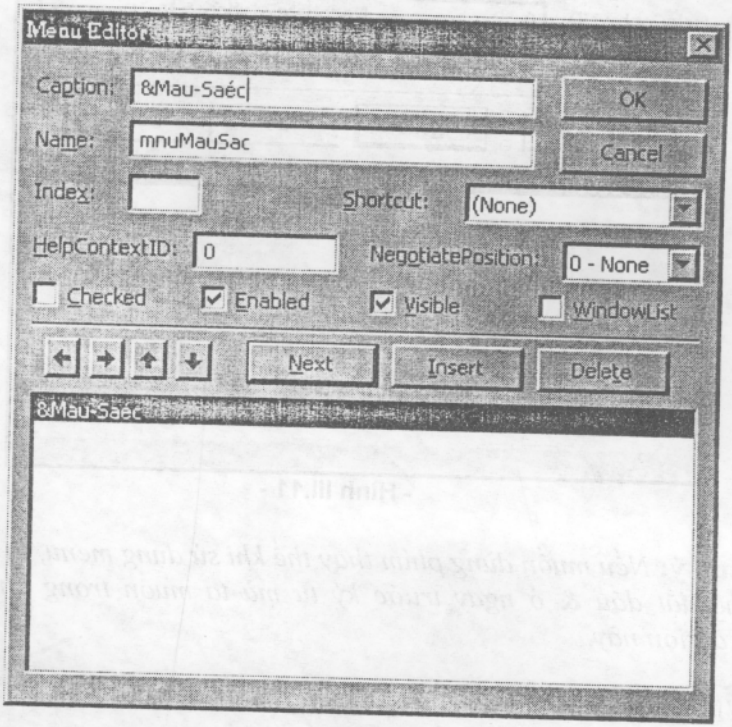
**Lưu ý:** *Nếu muốn dùng phím thay thế khi sử dụng menu, ta có thể đặt dấu & ở ngay trước ký tự mà ta muốn trong phần Caption này.*

Ví dụ: **ALT-M** sẽ cho chọn Menu Command **Màu-Sắc**

- Name: nơi đặt tên của các đối tượng menu. Thường thì ta đặt cho Menu Command một cái tên, nhưng ta ít khi dùng đến nó, trừ trường hợp muốn nó hiện ra hoặc biến mất (Visible/Invisible), thường thì ta hay dùng tên của MenuItems nhiều hơn.

Ta nên đặt tên cho các đối tượng menu có chữ **mnu** ở trước để dễ phân biệt với các đối tượng khác trong form khi xử lý

Ví dụ: ta đặt tên cho Menu Command ở trên là mnuMauSac

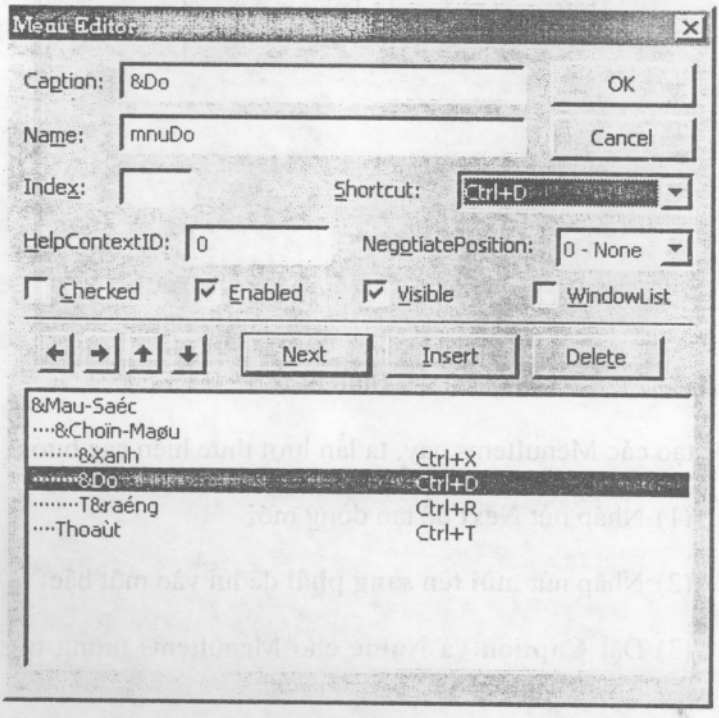



**- Hình III.12 -**

Ta có thể thêm vào các Menu Command khác bằng cách nhấp nút Next để sang một dòng mới.

Tương tự như vậy, tiếp theo ta tạo Menu Command "Kích-Thước"

- Trong các Menu Command thường có các Menultems. Tất cả các Menultems của Menu Command đều nằm thụt qua bên phải với bốn dấu chấm (....) ở phía trước. Khi ta nhấp chuột vào dấu mũi tên chỉ qua phải trong Menu Editor thì Menultems ta đang chỉnh sửa sẽ có thêm bốn dấu chấm, tức là thụt một bậc trong menu như hình minh họa sau:

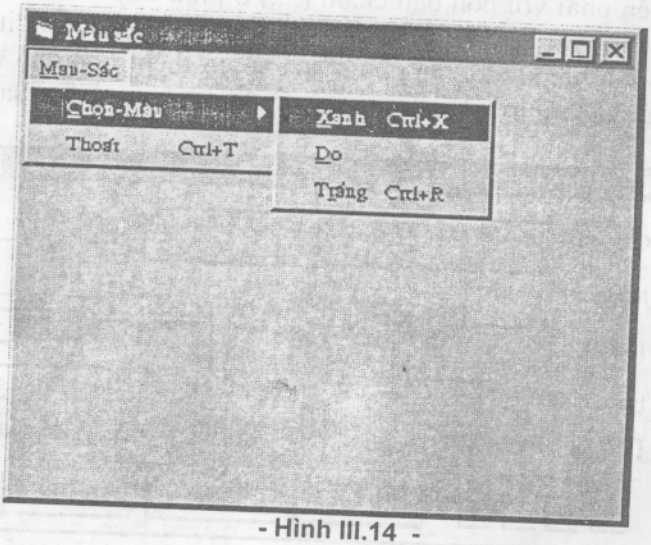



**Hình III.13**

Nếu muốn tạo Menultems có cấp thấp hơn nữa, ta lại tiếp tục nhấp mũi tên qua phải, Menultems lại thụt tiếp một bậc nữa.

Tương tự, khi ta nhấp mũi tên qua trái thì Menultems đang chỉnh sửa sẽ mất bốn dấu chấm phía trước, tức là nó đã được trồi ra một bậc.

Ví dụ : Trong Menu Command "Màu-Sắc" có các MenuItems như hình vẽ:




- Hình III.14 -

Để tạo các MenuItems này, ta lần lượt thực hiện các bước:

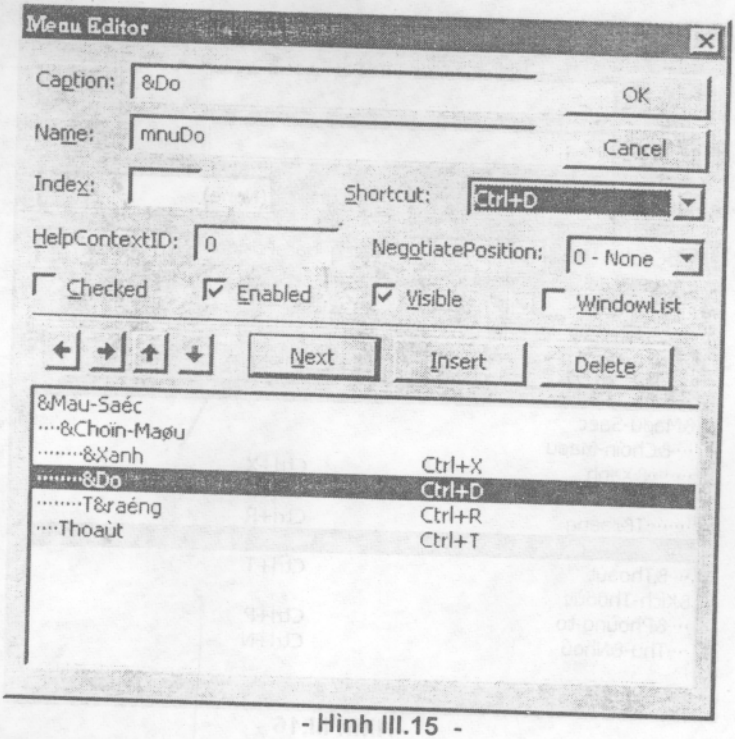
(1) Nhấp nút **Next** để tạo dòng mới

(2) Nhấp nút **mũi tên sang phải** để lùi vào một bậc

(3) Đặt **Caption** và **Name** cho MenuItems tương tự như Menu Command.

(4) Tuy nhiên, MenuItems có thêm một tính năng nữa, đó là ta có thể tạo phím tắt cho nó trong mục **Shortcut** .

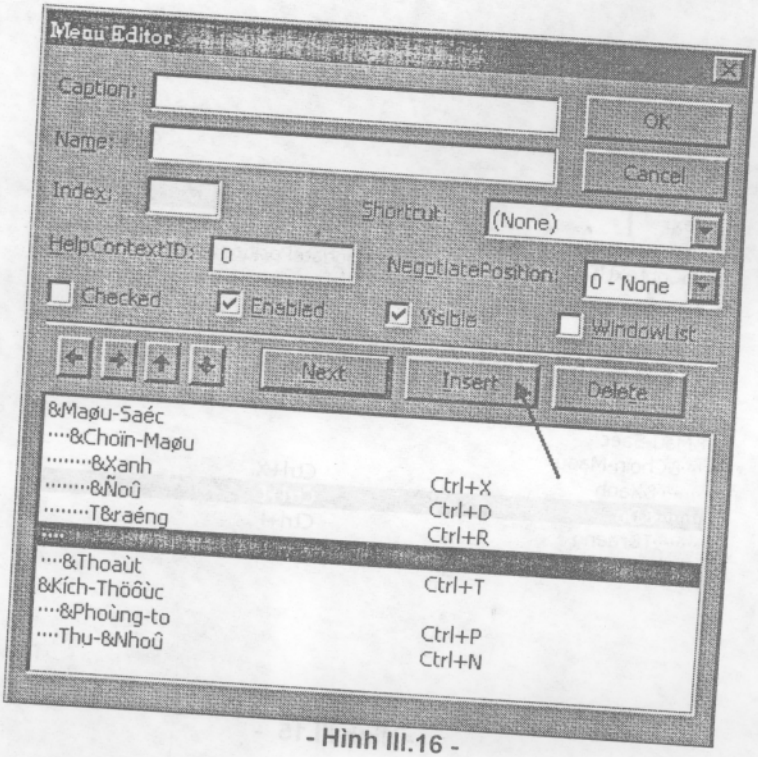
Quá trình này có thể được minh họa trong hình sau:




**- Hình III.15 -**

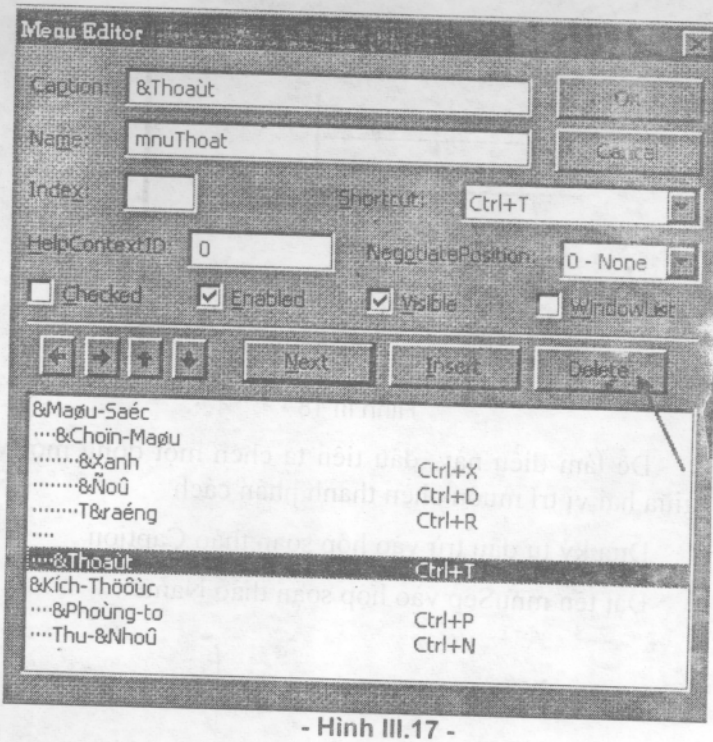
Ở đây, ta nhận thấy có thể dùng phím tắt CTRL+D để thay cho việc chọn lệnh đơn "Đỏ" trong Menu "Màu-Sắc"→"Chọn-Màu".

**Lưu ý:** *Để chèn một dòng mới vào menu đã có sẵn, ta đặt thanh sáng ở dòng muốn chèn vào trước nó rồi nhấp vào nút* ***Insert***




**- Hình III.16 -**

*Để xóa một dòng trong Menu, ta chọn dòng muốn xóa rồi nhấp nút* ***Delete.***

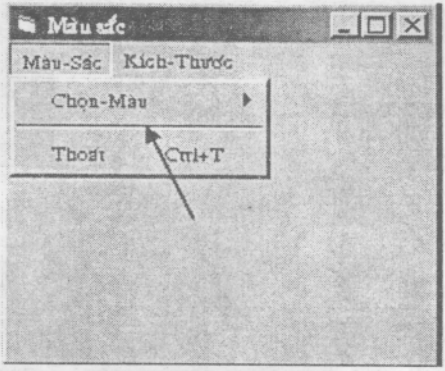



- Hình III.17 -

## I.2. Một số hiệu ứng đặc biệt trên menu

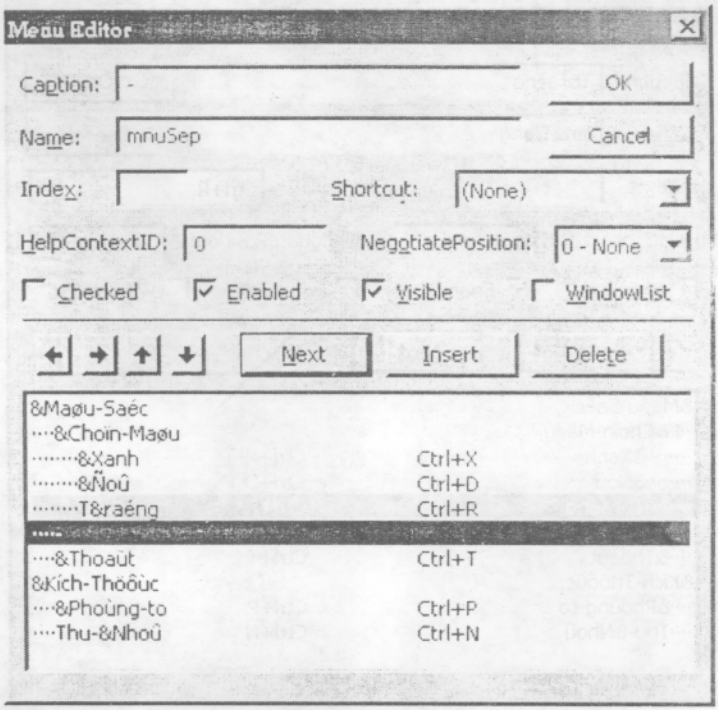
### I.2.1. Thêm thanh phân cách

Thanh phân cách là đoạn thẳng nằm giữa các mục menu. Thanh có tác dụng bài trí menu cho rõ ràng, đẹp mắt. Hình dưới đây cho thấy có một đường phân cách giữa các mục menu khác và mục Exit.




Hình III.18

- Để làm điều này, đầu tiên ta chèn một dòng mới vào giữa hai vị trí muốn chèn thanh phân cách
- Đưa ký tự dấu trừ vào hộp soạn thảo Caption
- Đặt tên mnuSep vào hộp soạn thảo Name

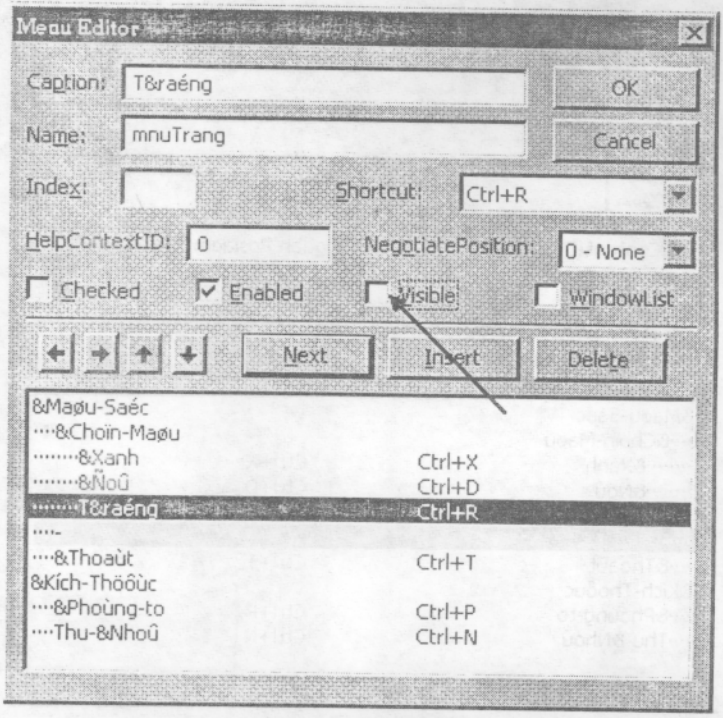



**- Hình III.19 -**

Ký tự dấu – trong hộp soạn thảo Caption chính là ký hiệu đặc tả thanh phân cách. Do Visual Basic yêu cầu tất cả các đối tượng menu phải có tên. Nên ta có thể đặt cho thanh phân cách một tên tùy ý. Một điều cần lưu ý là ta không thể chọn thanh phân cách khi chương trình thực hiện.

### I.2.2. Ẩn/Hiện đối tượng Menu

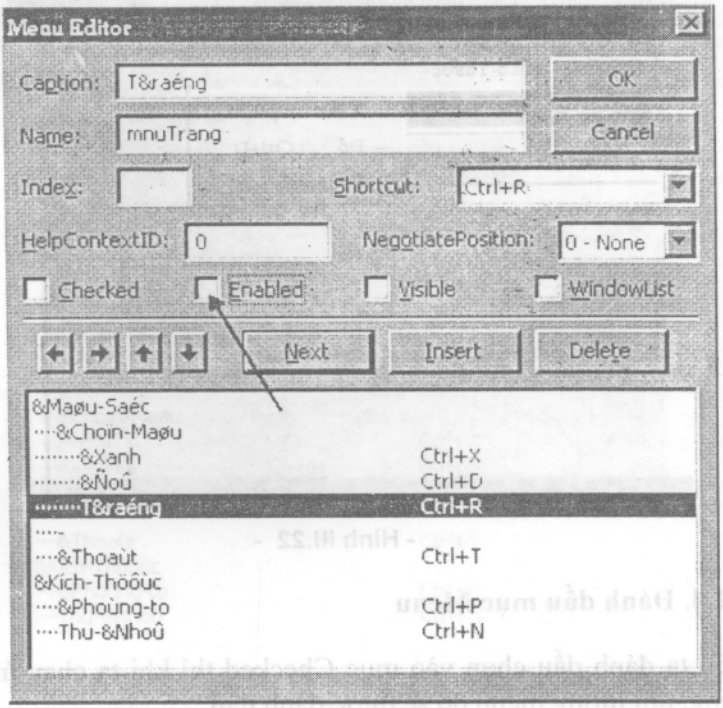
Ta có thể cho hiện/ẩn một đối tượng Menu bằng cách chọn/bỏ chọn mục **Visible** của đối tượng trong Menu Editor. Trường hợp mặc định là mục Visible được chọn. Trong trường hợp ta muốn đối tượng ẩn đi thì ta bỏ chọn mục này
Ví dụ:




**- Hình III.20 -**

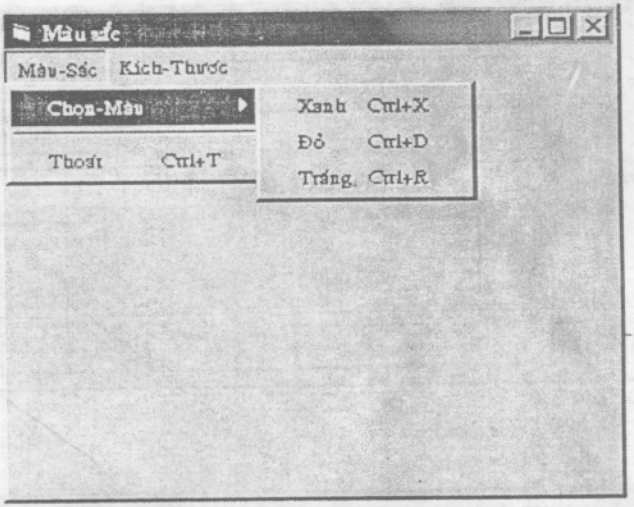
### I.2.3. Kích hoạt/vô hiệu hoá đối tượng menu

Ta có thể che (không cho phép chọn) một đối tượng menu thông qua thuộc tính Enabled. Khi mục Enabled của một đối tượng menu không được đánh dấu, thì đối tượng menu đó vẫn được nhìn thấy, nhưng ta không thể chọn nó được. Trường hợp mặc định thì mục này được chọn, nghĩa là MenuItems có khả năng hoạt động.




**- Hình III.21 -**

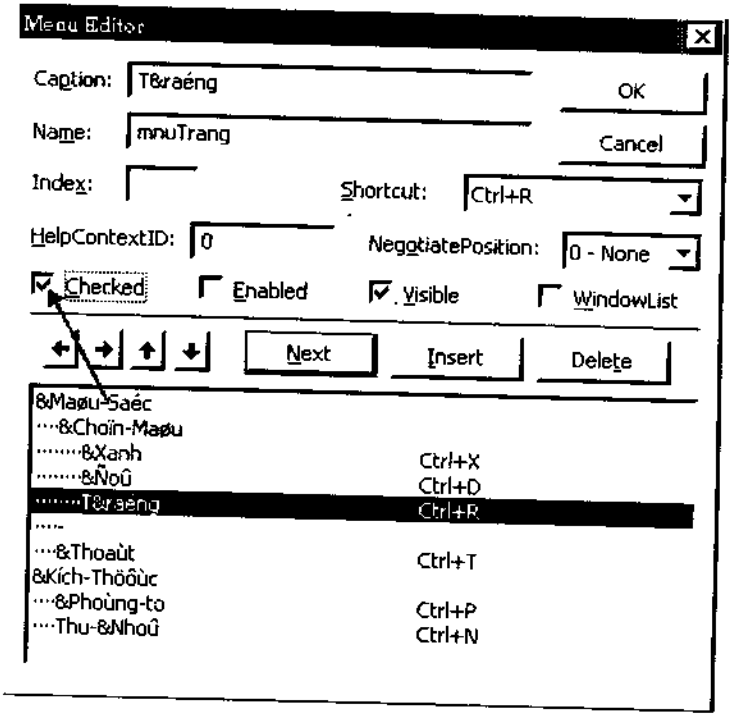
Kết quả là đối tượng menu này sẽ bị mờ đi khi ứng dụng được chạy và ta không thể chọn nó được.



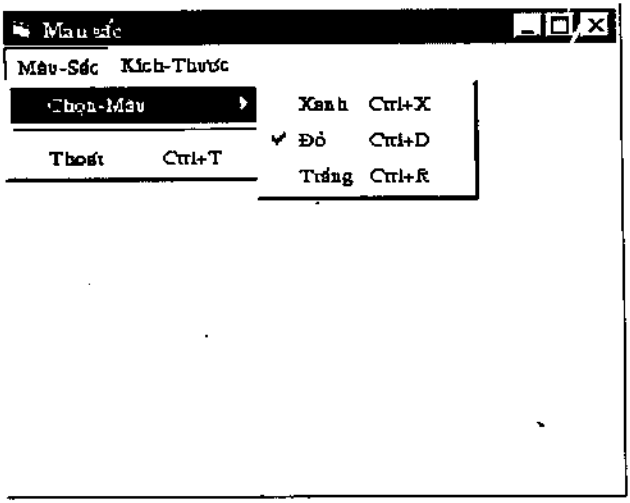

- Hình III.22 -

## I.2.4. Đánh dấu mục Menu

Nếu ta đánh dấu chọn vào mục Checked thì khi ta chạy ứng dụng, đối tượng menu đó sẽ được đánh dấu.


Menu Editor
Caption: T&raéng
OK
Name: mnuTrang
Cancel
Index:
Shortcut: Ctrl+R
HelpContextID: 0
NegotiatePosition: 0 - None
Checked
Enabled
Visible
WindowList
Next
Insert
Delete
&Maøu-Saéc
····&Choïn-Maøu
·····&Xanh
·····&Ñoû
·····T&raéng
····-
····&Thoaùt
&Kích-Thöôùc
····&Phoùng-to
····Thu-&Nhoû
Ctrl+X
Ctrl+D
Ctrl+R
Ctrl+T
Ctrl+P
Ctrl+N

**- Hình III.23 -**




Hình III.24

## II. VIẾT LỆNH CHO CÁC THỦ TỤC ĐÁP ỨNG BIẾN CỐ CHỌN MỤC MENU

Mỗi đối tượng menu có một thủ tục đáp ứng biến cố **chọn** (Click). Ví dụ, khi ta chọn mục "Thoát" trong trình đơn "Màu-Sắc", thủ tục mnuExit_Click() được thực hiện. Tên thủ tục vẫn tuân theo qui tắc: tiền tố mnu giúp nhận diện loại đối tượng (là Menu), tên đối tượng (là Exit) và cuối cùng là tên biến cố (là Click).

Có hai cách để xem thủ tục đáp ứng biến cố chọn Menu:

(1) Khi thiết kế giao diện, muốn xem thủ tục của đối tượng menu nào, ta chỉ cần chọn mục menu đó.

(2) Hiện cửa sổ Code và chọn thủ tục cần xem trong hộp Object ComboBox.

Để minh họa cho phần này, ta sẽ lần lượt viết lệnh cho các thủ tục đáp ứng biến cố chọn của các đối tượng menu trong chương trình trên.

**BẢNG HỆ THỐNG MENU CỦA FORM FRMMENU**

| Caption | Name |
|---|---|
| &Màu-Sắc | MnuMauSac |
| ...&Chọn-Màu | MnuChonMau |
| .....&Xanh | MnuXanh |
| .....&Đỏ | MnuDo |
| .....T&rắng | MnuTrang |
| ...&Thoát | MnuThoat |
| &Kích-Thước | MnuKichThuoc |
| ...&Phóng-To | MnuPhongTo |
| ...&Thu-Nhỏ | MnuThuNho |

➢ Thủ tục Form_Load(): thủ tục này thực hiện ngay khi ta khởi động chương trình Menu. Thủ tục làm nhiệm vụ cho ẩn đi hai mục "Trắng" và "Thu-Nhỏ" bằng cách đặt thuộc tính Enabled thành False

```
Sub Form_Load()
' Che mnuTrang vì màu trắng đang là màu nền Form
   mnuTrang.Enable = False
' Che mnuThuNho vì cửa sổ Form đang ở kích thước nhỏ
   mnuThuNho.Enable = False
End Sub
```

➢ Thủ tục mnuXanh_Click(): Thủ tục được thực hiện khi ta chọn lệnh "Xanh". Thủ tục có nhiệm vụ đổi màu nền của form thành màu xanh và cho ẩn đi (không cho chọn) mục "Xanh", đồng thời bật các mục "Đỏ" và "Trắng" lên cho phép ta chọn.

```
Sub mnuXanh_Click()
' Đặt lại màu nền là màu xanh
    frmMenu.BackColor = QBColors(1)
```

```
    ' Che mnuXanh vì màu xanh đang là màu nền form
      mnuXanh.Enable = False
    ' Cho phép chọn màu đỏ và màu trắng
      mnuDo.Enable  =  True
      mnuTrang.Enable =  True
    End Sub
```

➢ Thủ tục mnuDo_Click(): Thủ tục được thực hiện khi ta chọn lệnh "Đỏ". Thủ tục có nhiệm vụ đổi màu nền của form thành màu đỏ và cho ẩn đi (không cho chọn) mục "Đỏ", đồng thời bật các mục "Xanh" và "Trắng" lên cho phép ta chọn.

```
Sub mnuDo_Click()
' Đặt lại màu nền là màu đỏ
  frmMenu.BackColor = QBColors(4)
' Che mnuDo vì màu đỏ đang là màu nền Form
  mnuDo.Enable = False
        ' Cho phép chọn màu xanh và màu trắng
  mnuXanh.Enable  =  True
  mnuTrang.Enable =  True
End Sub
```

➢ Thủ tục mnuTrang_Click(): Thủ tục được thực hiện khi ta chọn lệnh "Trắng". Thủ tục có nhiệm vụ đổi màu nền của form thành màu trắng và cho ẩn đi (không cho chọn) mục "Trắng", đồng thời bật các mục "Đỏ" và "Xanh" lên cho phép ta chọn.

```
Sub mnuTrang_Click()
' Đặt lại màu nền là màu trắng
  frmMenu.BackColor = QBColors(15)
' Che mnuTrang vì màu trắng đang là màu nền Form
  mnuTrắng.Enable = False
' Cho phép chọn màu xanh và màu đỏ
  mnuXanh.Enable  =  True
  mnuDo.Enable =  True
End Sub
```

➢ Thủ tục mnuPhongTo_Click(): Thủ tục thực hiện khi ta chọn mục "Phóng-To". Thủ tục làm nhiệm vụ đưa cửa sổ form về dạng lớn và che mục "Phóng-To" đi, đồng thời bật mục "Thu-Nhỏ".

```
Sub mnuPhongTo_Click()
  ' Đặt cửa sổ Form ở dạng lớn
    frmMenu.WindowState = 2
  ' Che mnuPhongTo vì cửa sổ Form đang ở kích thước lớn
    mnuPhongTo.Enable = False
  ' Cho phép đổi sang dạng cửa sổ có kích thước nhỏ
    mnuThuNho.Enable  =  True
End Sub
```

➢ Thủ tục mnuThuNho_Click(): Thủ tục thực hiện khi ta chọn mục "Thu-Nhỏ". Thủ tục làm nhiệm vụ đưa cửa sổ form về dạng nhỏ và che mục "Thu-Nhỏ" đi, đồng thời bật mục "Phóng-To".

```
Sub mnuThuNho_Click()
  ' Đặt cửa sổ Form ở dạng nhỏ
    frmMenu.WindowState = 0
  ' Che mnuThuNho vì cửa sổ Form đang ở kích thước nhỏ
    mnuThuNho.Enable = False
  ' Cho phép đổi sang dạng cửa sổ có kích thước lớn
    mnuPhongTo.Enable  =  True
End Sub
```

**Lưu ý:** *Kích thước form phụ thuộc vào thuộc tính WindowsState. Khi WindowState=0: Form có kích thước như lúc thiết kế giao diện (Normal), WindowState=2: Form chiếm toàn bộ màn hình (Maximum).*

Chương IV

# SỰ KIỆN TRÊN FORM

Phần lớn các tương tác giữa con người trong cuộc sống hàng ngày đều theo dạng các sự kiện và các đáp ứng đối với các sự kiện.

Ví dụ: khi chuông điện thoại reo, ta nhấc ống nghe. Ở đây, tiếng chuông điện thoại chính là sự kiện và hành động nhấc ống nghe là đáp ứng đối với sự kiện.

Một sự kiện là một hành động bất kỳ của người dùng hướng vào ứng dụng.

Các thành phần giao diện có khả năng đáp ứng các sự kiện. Ví dụ khi ta nhấp lên một nút, nó được nhận ra dưới dạng một sự kiện bởi nút. Cũng vậy, một hộp văn bản nhận ra các sự kiện khi ta nhấn phím... Khả năng nhận ra các sự kiện được cài sẵn trong một thành phần giao diện. Các sự kiện mà một thành phần giao diện nhận ra thường khác nhau theo từng kiểu thành phần. Một ứng dụng điều khiển theo sự kiện hoạt động bằng cách đáp ứng các sự kiện người dùng.

Khi một thành phần giao diện được dùng, ta cần quyết định hành động cần có của thành phần để đáp lại một sự kiện cụ thể và viết mã sẽ được thi hành khi sự kiện xảy ra. Như vậy, đáp ứng của các thành phần giao diện đồng nhất đối với một sự kiện có thể khác nhau trong các tình huống khác nhau.

## I. SỰ KIỆN TRÊN FORM

Form cũng có thể đáp ứng các sự kiện do người dùng khởi phát hoặc do hệ thống ứng tác. Dưới đây là một số sự kiện của form:

(1) Load: xảy ra khi biểu mẫu được nạp vào bộ nhớ.

(2) Unload: xảy ra khi form đóng lại

(3) Activate: xảy ra khi form trở thành hoạt động.

(4) DeActivate: xảy ra khi form khác được kích hoạt, Form trở thành DeActivate.

(5) Click: xảy ra khi người dùng nhấn rồi thả chuột.

(6) MouseDown: xảy ra khi người dùng nhấn chuột.

(7) MouseUp: xảy ra khi người dùng nhả nút chuột.

(8) MouseMove: xảy ra khi người dùng di chuyển chuột.

(9) Resize: xảy ra khi form bị thay đổi kích thước.

Ví dụ: Ta có form có thuộc tính name là "frmEvents" (bạn set thuộc tính này trong cửa sổ properties của form).

Vd1: Xây dựng thủ tục đáp ứng sự kiện Form_Load, khi form được mở, ta định kích thước thu nhỏ cho form và đặt màu nền là màu xanh.

```
Private Sub Form_Load()
' Đặt cửa sổ Form ở dạng nhỏ
  frmEvents.WindowState = 0
' Đặt lại màu nền là màu xanh
  frmEvents.BackColor = QBColors(1)
End Sub
```

Vd2: Xây dựng thủ tục đáp ứng sự kiện Form_UnLoad, khi Form được đóng, xuất hiện hộp thoại hỏi bạn có thật sự muốn đóng form hay không. Nếu có, form sẽ được đóng.

```
Private Sub Form_Unload(Cancel As Integer)
   MsgBox "Ban co muon dong khong?"
End Sub
```

Vd3: Xây dựng thủ tục đáp ứng sự kiện form_Activate, khi Form trở thành Activate, thủ tục này sẽ đưa tiêu đề của form vào TextBox có tên là "txtTieudeForm".

```
Private Sub Form_Activate ()
    txtTieudeForm.Text = Me.Caption
End Sub
```

Vd4: Xây dựng thủ tục đáp ứng sự kiện form_Click, khi ta nhấp chuột vào form

```
Private Sub Form_Click()
' Hiện ra câu thông báo: "Bạn vừa nhắp chuột vào Form"
    MsgBox "Bạn vừa nhắp chuột vào Form"
End Sub
```

Vd5: Xây dựng thủ tục đáp ứng sự kiện khi ta thay đổi kích thước form, hiện ra hộp thông báo kích thước chiều cao và chiều rộng form hiện tại.

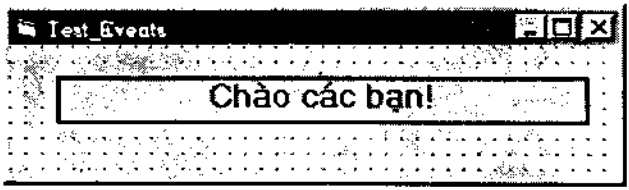
```
Private Sub Form_Resize()
' Hộp thoại thông báo kích thước Form: chiều cao x chiều rộng
        MsgBox Me.Height & "x" & Me.Width
End Sub
```

## II. SỰ KIỆN TRÊN NHÃN (LABEL)

Nhãn hỗ trợ các sự kiện như:

1. Click: xảy ra khi người dùng nhấp chuột lên nhãn
2. Change: xảy ra khi văn bản của nhãn được ứng dụng thay đổi
3. DblClick: xảy ra khi người dùng nhấp đúp trên nhãn.

Ví dụ: Trên form "frmEvents", ta có một nhãn có tên là "lblTest" như hình sau:




-Hình IV.1-

Xây dựng thủ tục đáp ứng sự kiện khi ta Click chuột vào nhãn "lblTest", nhãn được đổi thành "Bạn vừa đổi nhãn mới"

```
Private Sub lblTest_Click()
    lblTest.Caption = " Bạn vừa đổi nhãn mới"
End Sub
```

## III. SỰ KIỆN TRÊN HỘP VĂN BẢN (TEXTBOX)

Dưới đây là một số sự kiện hộp văn bản thường dùng:

1. KeyPress: xảy ra khi người dùng nhấn và thả phím.

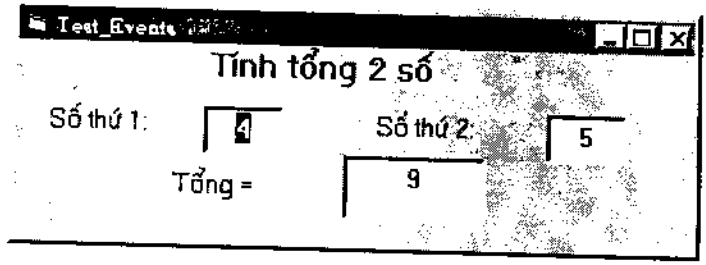
2. Click: xảy ra khi người dùng nhắp chuột vào hộp văn bản.

3. Change: xảy ra khi văn bản trong hộp văn bản thay đổi.

4. LostFocus: xảy ra khi một điều khiển mất tiêu điểm (focus). Lúc này con trỏ không còn nhấp nháy trong hộp văn bản nữa, ngược lại khi con trỏ nhấp nháy trong hộp văn bản để cho người dùng nhập giá trị vào lúc này mới xuất hiện sự kiện GotFocus).

Ví dụ: Form "frmEvents" làm nhiệm vụ tính tổng hai số, số thứ nhất để trống chờ người dùng nhập vào, số thứ hai luôn cố định là 5. Trong form "frmEvents" ta có ba TextBox: txtSo1 (chứa số thứ nhất), txtSo2 (chứa số thứ hai) và txtTong (chứa

tổng của hai số này). Ta xây dựng thủ tục đáp ứng sự kiện khi ta thay đổi số trong ô thứ nhất thì tổng hai số cũng thay đổi với giá trị tương ứng. Kết quả được minh họa trong hình sau:



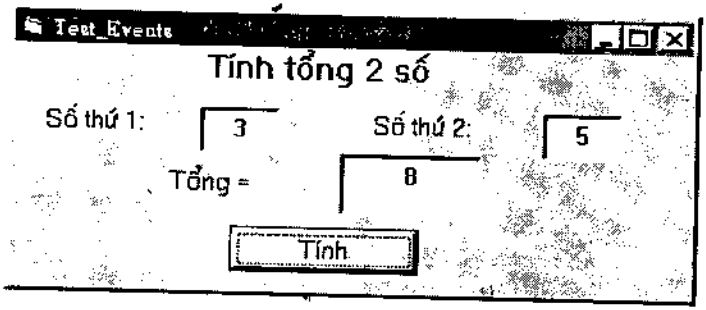

-Hình IV.2-

```
Private Sub txtSo1_Change()
   txtTong.Text = Val(txtSo1.Text) + Val(txtSo2.Text)
End Sub
```

## IV. SỰ KIỆN TRÊN NÚT (BUTTON)

Nút chỉ có một sự kiện thường dùng nhất, đó là Click: xảy ra khi người dùng nhấp chuột trên nút.

Ví dụ: Ta thay đổi lại ví dụ trong phần III một chút, ở đây ta cho người dùng nhập vào cả hai số 1 và 2. Khi nhấn nút Tính (cmdTinh) thì ta sẽ tính tổng của hai số đó và hiện kết quả ở hộp văn bản "Tổng"(txtTong)




-Hình IV.3-

```
Private Sub cmdTinh_Click()
   txtTong.Text = Val(txtSo1.Text) + Val(txtSo2.Text)
End Sub
```

## V. SỰ KIỆN TRÊN HỘP KIỂM TRA (CHECK BOX)

Click: xảy ra khi người dùng nhấp chuột trên hộp kiểm tra

Ví dụ: Ta có form gồm các đối tượng chính như sau:

- Frame "fra1": chứa phần nhập học viên

- Frame "fra2": chứa phần xem danh sách học viên

- Label "lblQl": chứa tiêu đề quản lý học viên

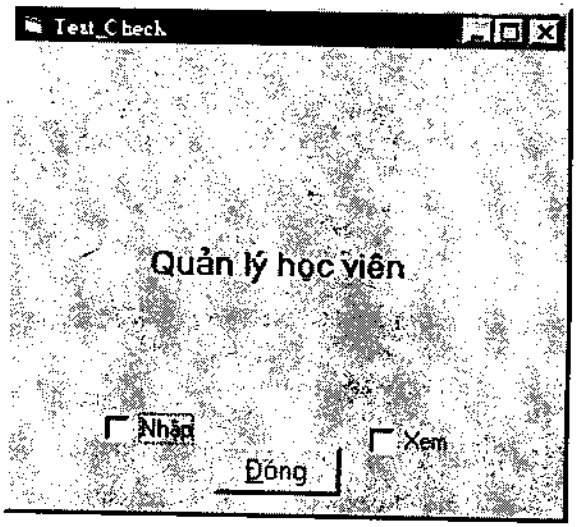
- Hai Check Box:

  + "chkNhap" có caption là "Nhập"

  + "chkXemNhap" có caption là "XemNhập"

Chương trình hoạt động như sau:

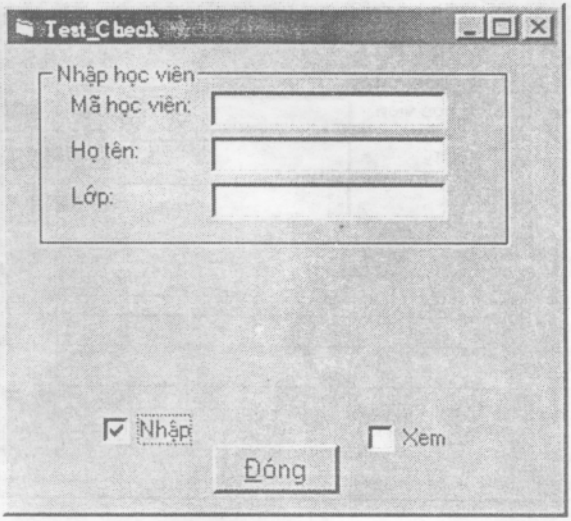
- Khi Form được nạp, chưa có check vào hai CheckBox «Nhập» và «Xem» chương trình có dạng như hình:



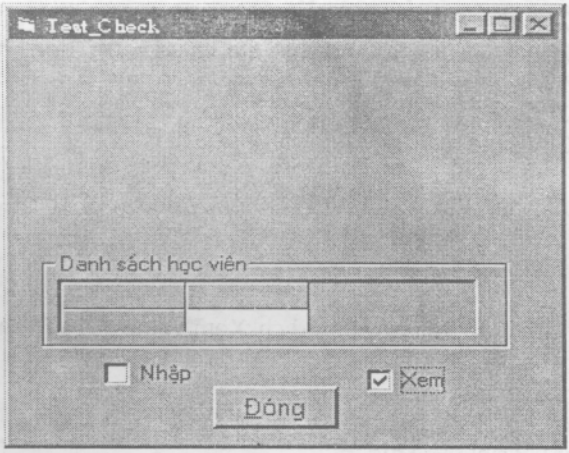

-Hình IV.4-

```
Private Sub Form_Load()
        fra1.Visible = False
        fra2.Visible = False
        lblQl.Visible = True
End Sub
```

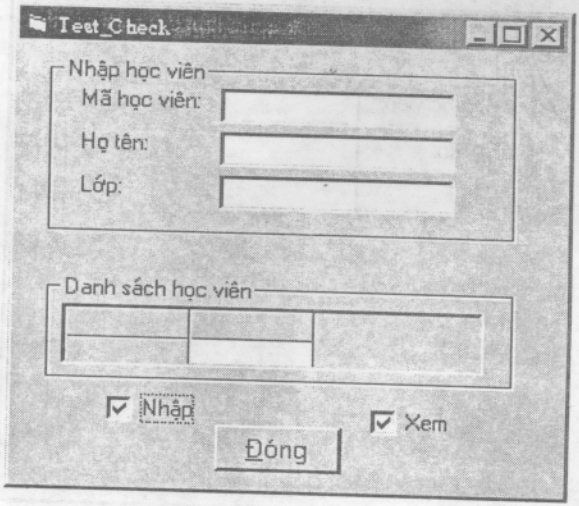
Khi Click vào Check Box "chkNhap" để chọn, phần nhập học viên hiển thị cho phép ta nhập học viên, nếu ta Click lại lần nữa để bỏ chọn, phần này sẽ biến mất, tương tự như vậy cho Check Box "chkXem"




-Hình IV.5-




-Hình IV.6-




-Hình IV.7-

```
Private Sub chkNhap_Click()
    lblQl.Visible = False
    If (chkNhap.Value = 0 And chkXem.Value = 0) Then
       fra1.Visible = False
       fra2.Visible = False
       lblQl.Visible = True
    Else
      If (chkNhap.Value = 1 And chkXem.Value = 0) Then
       fra1.Visible = True
       fra2.Visible = False
      Else
       If (chkNhap.Value = 0 And chkXem.Value = 1) Then
        fra1.Visible = False
        fra2.Visible = True
       Else
        fra1.Visible = True
        fra2.Visible = True
       End If
      End If
    End If
End Sub
```

**Hàm xử lý sự kiện khi ta click chuột vào CheckBox "chkXem" cũng tương tự như trên**

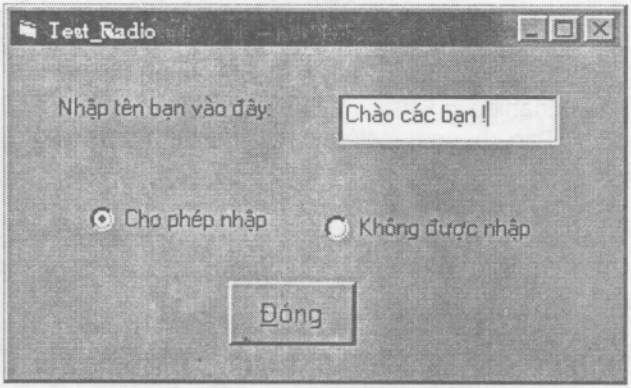
## VI. SỰ KIỆN TRÊN NÚT TÙY CHỌN (OPTION BUTTON)

Tuy các nút tùy chọn hỗ trợ vài sự kiện, nhưng sự kiện mà nút tùy chọn thường dùng nhất là Click: xảy ra khi người dùng nhấp chuột lên nút tùy chọn

Ví dụ: Ta có form như hình bên dưới, form gồm có các đối tượng chính như sau:

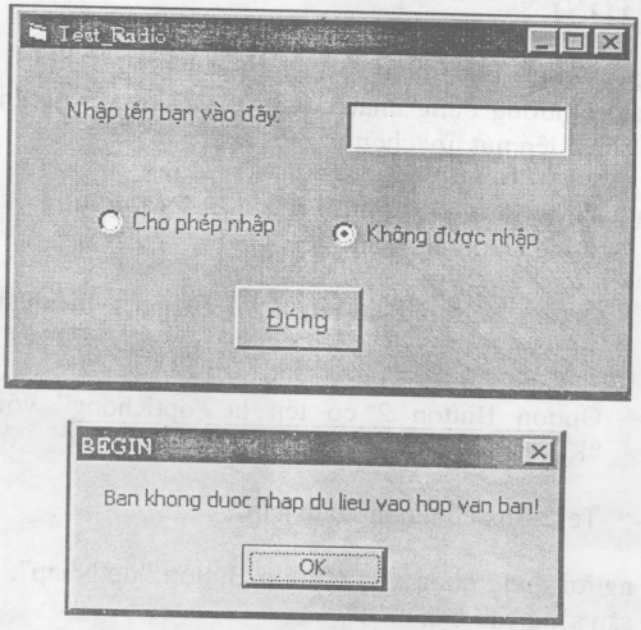
- Option Button 1 có tên là "optNhap" với nhãn "Cho phép nhập"
- Option Button 2 có tên là "optKhong" với nhãn "Không được nhập"
- Text Box có tên là "txtNhap"

Khi người dùng chọn vào Option Button "optNhap", ô Text Box sẵn sàng cho bạn nhập vào




**-Hình IV.8-**

Khi người dùng chọn vào Option Button "optKhong", ô Text Box bị vô hiệu hóa, ta không thể nhập vào ô này được, đồng thời xuất hiện hộp thoại "Bạn không được nhập dữ liệu vào hộp văn bản".




**- Hình IV.9-**

```
Private Sub optKhong_Click()
   txtNhap.Enabled = False
   MsgBox "Ban khong duoc nhap du lieu vao hop van ban!"
End Sub

Private Sub optNhap_Click()
   txtNhap.Enabled = True
End Sub
```

## VII. SỰ KIỆN TRÊN HỘP LIỆT KÊ (LIST BOX)

Sau đây là vài sự kiện thường dùng của hộp liệt kê:

- Click: xảy ra khi người dùng nhấp chuột lên hộp liệt kê

- Scroll: xảy ra khi người dùng cuộn qua danh sách trong hộp liệt kê

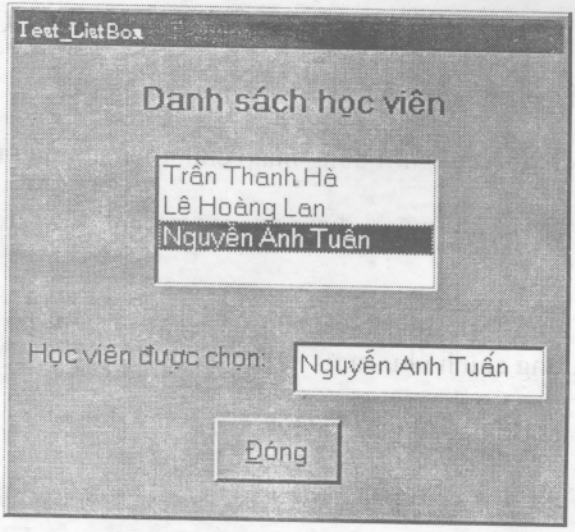
Ví dụ: Ta có form như hình bên dưới, form gồm có các đối tượng chính như sau:

- List Box có tên là "lstHv"

- Text Box có tên là "txtHv"

Khi người dùng chọn vào một trong các tên học viên ở List Box "lstHv" thì mục đó được hiển thị trong Text Box "txtHv" (tức học viên được chọn)

```
Private Sub lstHv_Click()
  txtHv.Text = lstHv.Text
End Sub
```




-Hình IV.10-

## VIII. SỰ KIỆN TRÊN HỘP LIỆT KÊ THẢ (COMBOBOX)

Hộp ComboBox hỗ trợ tất cả các sự kiện của hộp liệt kê:

- Click: xảy ra khi người dùng nhấp chuột lên hộp ComboBox

- Scroll: xảy ra khi người dùng cuộn qua danh sách trong hộp ComboBox

Ngoài ra, hộp ComboBox còn có các sự kiện giống như hộp văn bản:
- Change: xảy ra khi người dùng thay đổi văn bản trong phần hộp văn bản của hộp ComboBox

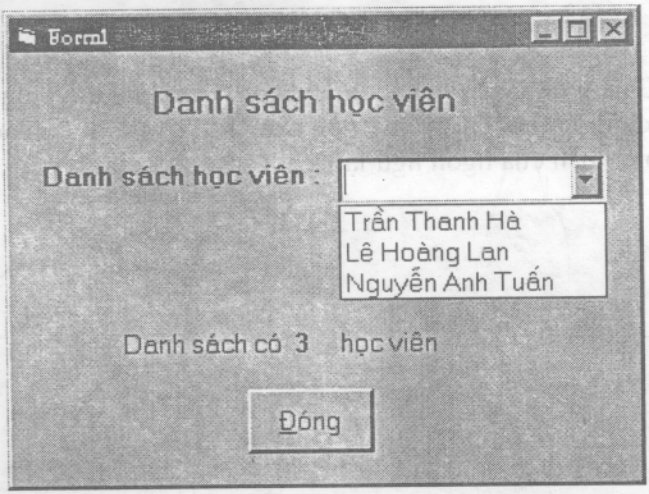
Ngoài ra, hộp ComboBox còn có sự kiện riêng của nó:
- DropDown: xảy ra khi người dùng nhấp chuột lên mũi tên để hiển thị danh sách

Ví dụ: Ta có form như hình IV.11, form gồm có các đối tượng chính như sau:

- Combo Box có tên là "cboHv"
- Label có tên là "lblSohv"

Khi người dùng nhấp vào nút DropDown của hộp Combo Box, ta sẽ đếm số học viên có trong danh sách đó và hiển thị kết quả ở "lblSohv"

```
Private Sub cboHv_DropDown()
        lblSohv.Caption = cboHv.ListCount
End Sub
```




**- Hình IV.11 -**

## Chương V

# VIẾT LỆNH

## I. GIỚI THIỆU

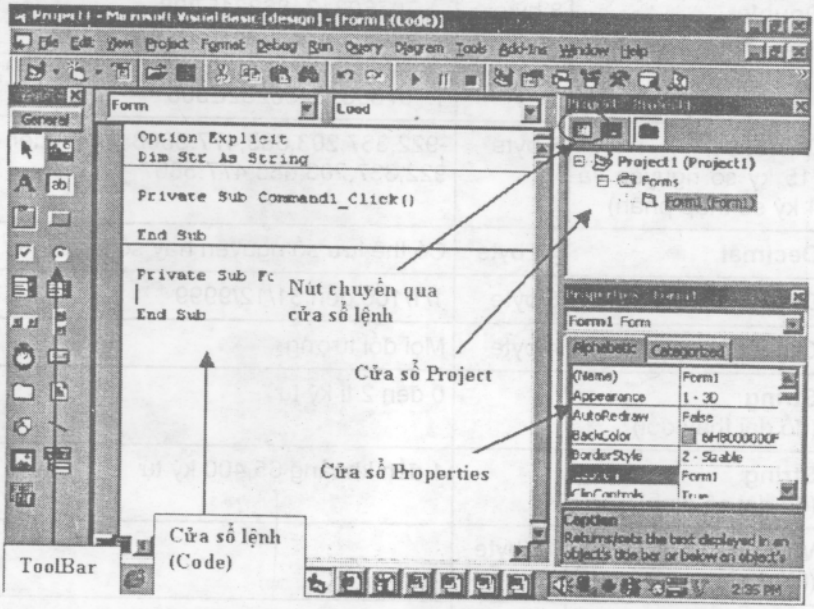
### I.1. Ý nghĩa

Như chương III đã trình bày, một chương trình Visual Basic bao gồm các thành phần sau:

- Form
- Control
- Thủ tục xử lý sự kiện

Chúng ta phải viết các thủ tục xử lý sự kiện theo mục đích của chương trình. Chương này chúng ta sẽ tìm hiểu các khái niệm cơ bản của ngôn ngữ lập trình Visual Basic.

## I.2. Các thành phần và khái niệm




**Hình V.1**: *Các thành phần và khái niệm trong Visual Basic*

# II. CÁC KIỂU DỮ LIỆU TRONG VISUAL BASIC

Visual Basic cung cấp cho chúng ta các kiểu dữ liệu sau:

| Kiểu dữ liệu | Kích thước | Phạm vi |
|---|---|---|
| Byte | 1 byte | 0 đến 255 |
| Boolean | 2 byte | True hoặc False |
| Integer | 2 byte | -32,768 đến 32,767 |
| Long | 4 byte | -2,147,483,648 đến 2,147,483,647 |
| Single | 4 byte | -3.402823E38 đến -1.401298E-45 |

| | | |
|---|---|---|
| (Số thực) | | 1.401298E-45 đến 3.402823E+38 |
| **Double** (Số thực) | 8 byte | -1.79769313486231E308 đến -4.94065645841247E-324<br>4.94065645841247E-324 đến 1.79769313486232E308 |
| **Currency** (15 ký số nguyên và 4 ký số thập phân) | 8 byte | -922,337,203,685,477.5808 đến 922,337,203,685,477.5807 |
| **Decimal** | 14 byte | Có thể lưu số nguyên hay số thực |
| **Date** | 8 byte | 1/1/100 đến 31/12/9999 |
| **Object** | 4 byte | Mọi đối tượng |
| **String** (độ dài thay đổi) | | 0 đến 2 tỉ ký tự |
| **String** (độ dài cố định) | | 1 đến khoảng 65,400 ký tự |
| **Variant** (kiểu số) | 16 byte | |
| **Variant** (kiểu ký tự) | | |
| Người sử dụng định nghĩa (khai báo **Type**) | | |

## II.1. Kiểu dữ liệu Variant

Đây là kiểu dữ liệu mà biến của kiểu dữ liệu này có khả năng lưu trữ nhiều loại dữ liệu khác nhau.

Ví dụ:

```
Dim Var
Var = "123"     ' biến Var sẽ có giá trị là chuỗi "123"
Var = 12        ' biến Var sẽ có giá trị là số 12
```

Tuy nhiên khi thực hiện các phép toán trên các giá trị kiểu Variant thì chúng ta có một số lưu ý thứ tự thực hiện của các phép toán như sau:

- Đối với các toán hạng trong biểu thức là cùng kiểu thì kết quả sẽ là kiểu dữ liệu của các toán hạng

- Nếu các toán hạng có các kiểu dữ liệu khác nhau thì kết quả sẽ phụ thuộc vào toán tử. Nếu toán tử trong biểu thức áp dụng cho kiểu dữ liệu nào thì kết quả sẽ có kiểu dữ liệu đó.

Ví dụ:

```
Dim X, Y, Z
X = "1"
Y = "2"
Z = X + Y       ' Z sẽ có giá trị là "12"
Z = X & Y       ' Z sẽ có giá trị là "12"
X = 1
Y = 2
Z = X + Y       ' Z sẽ có giá trị là 3
Z = X & Y       ' Z sẽ có giá trị là 12
X = "1"
Y = 2
Z = X + Y       ' Z sẽ có giá trị là 3
Z = X & Y       ' Z sẽ có giá trị là "12"
```

Nếu việc chuyển đổi kiểu không thành công thì Visual Basic sẽ phát sinh ra lỗi "Type mismatch".

Visual Basic cũng cung cấp cho chúng ta các hàm để chúng ta có thể xác định kiểu dữ liệu của biến kiểu **Variant.**

**IsNumeric(biến)**: xác định biến có phải là kiểu số hay không.

**IsDate(biến)**: xác định biến có phải là kiểu ngày hay không.

...

Các hàm trên sẽ trả về giá trị **Boolean.**

Còn một điểm đáng lưu ý của kiểu dữ liệu **Variant** là: giá trị rỗng và giá trị NULL

### II.2. Giá trị rỗng

Khi một biến thuộc kiểu **Variant** được khai báo thì nó sẽ được gán giá trị rỗng. Cần phân biệt giá trị rỗng và giá trị số 0 hay giá trị chuỗi rỗng (chuỗi có độ dài bằng 0) hay giá trị NULL. Chúng ta có thể kiểm tra một biến có chứa giá trị rỗng hay không bằng hàm: **BOOLEAN IsEmpy(biến)**

Nếu biến chứa giá trị rỗng thì hàm này trả về giá trị là TRUE.

### II.3. Giá trị NULL

Ngoài ra biến có kiểu Variant còn có chứa một giá trị đặc biệt nữa là giá trị NULL. NULL là một giá trị biểu thị dữ liệu bị thất lạc hay không xác định được. Để kiểm tra một biến có chứa giá trị NULL hay không bằng hàm: **BOOLEAN IsNull(biến)**

Nếu biến chứa giá trị NULL thì hàm này trả về giá trị là TRUE.

Có một vài chú ý đối với giá trị NULL như sau:

Một biểu thức có chứa một giá trị NULL thì kết quả là NULL.

Giá trị NULL có thể được gán cho một biến kiểu **Variant.**

```
Dim X
X = NULL
```

Và chỉ có biến **Variant** mới có thể được gán cho giá trị NULL mà thôi. Các biến khác khi gán như vậy sẽ bị báo lỗi.

### II.4. Kiểu dữ liệu Chuỗi

Khi chúng ta khai báo một biến kiểu chuỗi thì chiều dài của biến có thể thay đổi tùy thuộc vào giá trị dữ liệu mà nó đang chứa. Tuy nhiên, chúng ta có thể khai báo một biến có độ dài cố định, có giới hạn. Visual Basic sẽ tự động thêm vào các ký

tự trắng hay xóa các ký tự dư thừa để độ dài của biến bằng như khi khai báo.

Cú pháp:

**Dim <biến> As String * [độ dài]**

Ví dụ:

```
Dim Str1 As String
Dim Str2 As String * 6
Dim Str3 As String * 12

Str1 = "Visual Basic"
Str2 = Str1
Str3 = Str2
```

**Str2** sẽ có giá trị là **"Visual"**
**Str3** sẽ có giá trị là **"Visual "**

Để xóa đi các khoảng trắng dư thừa ở đầu hay cuối một biến kiểu `String` thì chúng ta có thể sử dụng hai hàm sau

**Trim** : xóa các khoảng trắng ở đầu và cuối biến
**LTrim** : xóa các khoảng trắng ở đầu (bên trái) biến
**RTrim** : xóa các khoảng trắng ở cuối (bên phải) biến

## II.5. Kiểu dữ liệu Ngày

Hằng giá trị ngày phải được đặt trong hai dấu thăng.

Ví dụ:

```
Dim DayBegin As Date
DayBegin = #1/1/2003#
```

## II.6. Kiểu dữ liệu do người dùng định nghĩa

Đây là cấu trúc giống như kiểu dữ liệu struct của ngôn ngữ C hay record của Pascal). Nó cho phép chúng ta có thể khai báo nhiều loại thông tin với các kiểu dữ liệu khác nhau trong một biến.

Cú pháp chung:

```
[Private | Public] Type <Tên Kiểu Dữ Liệu >
        <Thành phần 1> As <Kiểu dữ liệu 1>
        <Thành phần 2> As <Kiểu dữ liệu 2>
        ...
End Type
```

Ví dụ:

```
Public Type CV
        Name As String
        Sex As Boolean
        Birthday As Date
End Type
```

Ta có thể truy cập các thành phần trong cấu trúc Type theo cú pháp sau

**<Tên Biến>.<Tên Thành Phần>**

hay

```
With <TênBiến>
        .<Tên Thành Phần>
End With
```

Ví dụ:

```
Dim Person1, Person2 As CV
Person1.Name = "Nguyễn Thị B"
Person1.Birthday = #31/12/1968#
```

hay

```
With Person1
        .Name = "Nguyễn Thị B"
        .Birthday = #31/12/1968#
End With
```

Ta có thể gán hai biến dạng này với nhau (chúng cùng kiểu dữ liệu).

Ví dụ:

```
Person2 = Person1
```

## II.7. Hằng số

Cú pháp định nghĩa một hằng số

**[Public | Private] Const <Tên Hằng> [As <Kiểu dữ liệu >] = biểu thức**

Ví dụ:

```
Const PI = 3.14159
Const DepDate As Date = #30/4/75#
```

## II.8. Mảng

Đây là kiểu dữ liệu quen thuộc trong các ngôn ngữ lập trình. Mảng dùng để lưu một dãy các biến có cùng kiểu dữ liệu. Chúng được xác định bởi chỉ số của mảng.

Khai báo:

**Dim <Tên Mảng>(Chỉ số[,Chỉ số 2,... ]) As <Kiểu Dữ Liệu>**

Chỉ số có thể là một số nguyên N. Khi đó Mảng sẽ có chỉ số từ 0 đến N. Chú ý rằng chỉ số bắt đầu từ 0 hay từ 1 cũng có thể được chúng ta thiết lập thông qua lệnh

**Option Base 0**

hay

**Option Base 1**

Tuy nhiên chúng ta có thể giới hạn lại phạm vi của chỉ số bằng cách dùng từ khóa **To.**

Ví dụ:

```
Dim A(10) As Integer   ' Mảng A sẽ có 11 phần từ từ 0 → 10
Dim A(2 to 10) As Integer ' Mảng A sẽ có 9 phần từ từ 2 → 10
Dim A(8, 8) As Integer ' Mảng A là ma trận 2 chiều có kích thước 8*8
```

Ví dụ:

Chương trình In ra Tổng của mảng 10 phần tử

```
Dim A[9] As Integer , i, Sum As Integer
Sum = 0
For i = 0 to 9
```

```
        Sum=Sum + A(i)
Next i
```

Nếu chỉ số không được chỉ rõ thì một mảng động sẽ được tạo ra. Khi đó, trong chương trình chúng ta có thể thiết lập kích thước mảng bằng cách dùng thủ tục

**ReDim [Preserve] <Tên Mảng>(<kích thước>)**

Chúng ta có thể ReDim một mảng nhiều lần. Nếu có từ khóa **Preserve** thì mảng cấp phát lại sẽ không xóa dữ liệu đã có trước đó trên mảng.

Ví dụ:

```
Dim MyArray() As Integer    ' Khai báo mảng động.
Redim MyArray(5)    ' Cấp phát 5 phần tử.
For I = 1 To 5
   MyArray(I) = I
Next I
Redim Preserve MyArray(15)
...
```

## III. KHAI BÁO BIẾN

### III.1. Khai báo biến

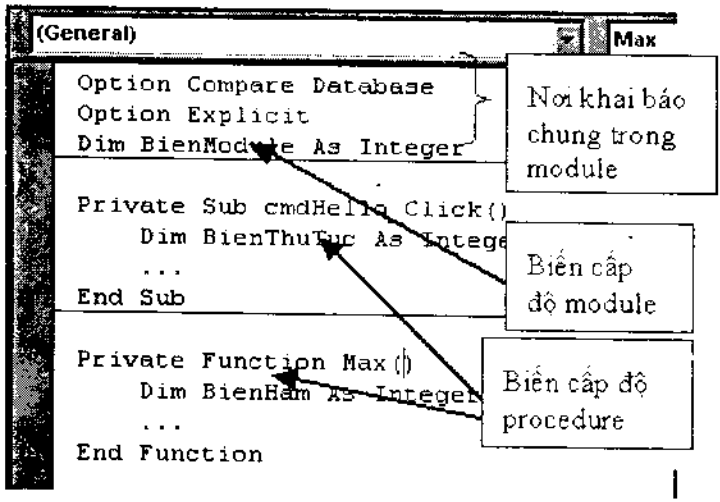
Cú pháp khai báo biến:

**Dim <biến> [As <kiểu dữ liệu>]**

Ví dụ:

```
Dim Str As String
Dim i As Integer
```

Một biến có thể được khai báo bên trong một thủ tục/hàm (biến này gọi là biến cấp độ procedure) hay khai báo ở phần đầu của mỗi module, trong phần khai báo General (biến này gọi là biến cấp độ module).




**Hình V.2**: *Các vị trí khai báo biến*

Nếu biến được khai báo trong thủ tục/hàm thì nó chỉ có thể sử dụng trong thủ tục/hàm đó. Nếu biến được khai báo trong phần khai báo của khai báo của module thì biến đó sẽ có tác dụng cho tất cả các thủ tục/hàm trong module đó; nhưng với các thủ tục/hàm trong cùng một project mà khác module thì biến đó không có tác dụng. Để cho biến được khai báo trong phần khai báo chung của module có tác dụng toàn bộ project thì khi khai báo biến đó, ta thay thế từ khóa **Dim** bằng từ khóa **Public**.

Ví dụ:

**Public strName As String**

Chúng ta có thể không cần định nghĩa kiểu dữ liệu của biến khi khai báo, lúc đó biến này sẽ được gán kiểu là **Variant**. Bạn có thể khai báo nhiều biến trên một câu lệnh. Tuy nhiên, khi đó bạn phải xác định loại dữ liệu cho mỗi biến.

Ví dụ:

**Dim intX As Integer, intY As Integer, intZ As Integer**

Với khai báo trên thì ta sẽ có ba biến **intX, intY, intZ** thuộc kiểu **Integer**; nhưng khi chúng ta khai báo:

```
Dim intX, intY, intZ As Integer
```

thì ta sẽ tạo ra có hai biến **intX, intY** thuộc kiểu **Variant** và biến **intZ** thuộc kiểu **integer.**

### III.2. Từ khóa Public

Bạn có thể dùng từ khóa **Public** để khai báo một biến **Public** cấp độ module.

Ví dụ:

```
Public strName As String
```

Một biến **Public** thì có thể được dùng trong mọi hàm/thủ tục trong project. Nếu một biến **Public** được khai báo trong một **standard module** hay **class module** thì nó sẽ có tác dụng trong mọi project mà có tham khảo đến project mà biến được khai báo.

### III.3. Từ khóa Private

Bạn có thể dùng từ khóa **Private** để khai báo một biến **Private** cấp độ module.

Ví dụ:

```
Private MyName As String
```

Biến **Private** có thể được dùng chỉ bởi các thủ tục/hàm trong cùng một mudule. Trong cấp độ module, từ khóa **Dim** tương đương với từ khóa **Private.**

### III.4. Từ khóa Static

Khi bạn dùng từ khóa **Static** để thay thế cho từ khóa **Dim**, thì sau mỗi lần sử dụng biến, giá trị của biến sẽ được nhớ lại.

Ví dụ:

```
Function KeepTotal(Number)
        Static Accumulate
        Accumulate = Accumulate + Number
        KeepTotal = Accumulate
End Function
```

Với đoạn chương trình trên, sau mỗi lần gọi làm, khi thoát khỏi hàm **KeepTotal**, biến **Accumulate** vẫn giữ lại giá trị của mình.

### III.5. Từ khóa Option Explicit

Visual Basic cũng cung cấp cho chúng ta cách sử dụng biến một cách linh hoạt. Ta có thể không cần khai báo biến trước khi sử dụng. Tuy nhiên để chương trình của chúng ta rõ ràng và tránh lỗi, chúng ta nên khai báo biến trước khi sử dụng. Thật vậy, nếu chúng ta không khai báo biến trước đôi khi dẫn đến trường hợp chúng ta tham chiếu đến một tên biến sai, khi đó chương trình sẽ hiểu là tên một biến mới. Để bắt buộc người lập trình khai báo biến trước khi sử dụng chúng ta sử dụng thêm từ khóa **Option Explicit** trong phần khai báo của module.

## IV. CÁC CẤU TRÚC ĐIỀU KHIỂN TRONG VISUAL BASIC

Đây là các câu lệnh cho phép người lập trình có thể điều khiển thứ tự thi hành của các câu lệnh trong chương trình.

### IV.1. Cấu trúc If ...then

Cấu trúc này dùng để kiểm tra một điều kiện đúng hay sai, khi đó chương trình sẽ có các hành động tương ứng.

Cú pháp:

```
If <điều kiện> Then
                <Các lệnh 1>
```

```
Else
                    <Các lệnh 2>
End If
```

Nếu điều kiện đúng thì chương trình thi hành các lệnh 1, ngược lại sẽ thi hành các lệnh 2. Điều kiện là một biểu thức mà ở đó giá trị trả về là khác 0 hay bằng 0 (tương đương với điều kiện có giá trị là TRUE hay FALSE).

Ví dụ: chương trình kiểm tra số chẵn, lẻ

```
If (Num Mod 2) = 0 Then
                    MsgBox "Số chẵn"
Else
                    MsgBox "Số lẻ"
End If
```

Tuy nhiên, trong một số trường hợp đặc biệt thì ta có thể có các cấu trúc sau:

- **If <điều kiện> Then <Một lệnh >**

Nếu điều kiện đúng thì thực hiện duy nhất một lệnh. Lúc này ta không cần để End If

Ví dụ:

```
If Max < a Then Max = a
```

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

Nếu điều kiện đúng thì thực hiện các câu lệnh (phải có End If).

Ví dụ:

```
If Max < a Then
        Max = a
        a = 100
End If
```

Như đã nói ở trên trong VBA chúng ta có thể không cần khai báo kiểu dữ liệu của biến. Lúc đó VBA sẽ tự động xác định kiểu dữ liệu của biến. Trong cấu trúc lệnh If...Else...End If

chúng ta có thể kiểm tra kiểu dữ liệu của một biến bằng cú pháp sau:

```
If TypeOf <Tên biến> Is <Kiểu dữ liệu> Then
        <Các lệnh 1>
End If
```

Ta cũng có thể lồng các lệnh If ... Else ... End If vào với nhau.

Ví dụ: chương trình kiểm tra số âm, dương và số 0

```
If Num > 0 Then
        MsgBox "Số dương"
Else
        If Num < 0 Then
                MsgBox "Số âm"
        Else
                MsgBox "Số không"
        End If
End If
```

## IV.2. Cấu trúc Select Case

Cấu trúc này cũng giống như cấu trúc **If ...Else ... End If** . Tuy nhiên, cấu trúc này có nhiều sự chọn lựa hơn.

Cú pháp:

```
Select Case <Biểu thức>
Case <Giá Trị 1>
        <Các lệnh 1>
[Case <Giá Trị 2>
        <Các Lệnh 2>
Case Else
        <Các lệnh n>
]
End Select
```

trong đó, giá trị có thể là một giá trị cụ thể hoặc là miền giá trị hoặc một biểu thức.

- **<Biểu thức 1> To <Biểu thức 2>**: miền giá trị từ giá trị **<Biểu thức 1>** đến **<Biểu thức 2>**

• **Is <Toán tử so sánh> <biểu thức>** (hay **<Biểu thức> <Toán tử so sánh> <Biểu thức>**)

Ví dụ: xác định Mùa trong năm dựa vào tháng

```
Select Case Month
        Case  Is <4
                MsgBox "Mùa Xuân"
        Case 4,5,6
                MsgBox "Mùa Hè"
        Case 7 To 9
                MsgBox "Mùa Thu"
        Case Else
                MsgBox "Mùa Đông"
End Select
```

## IV.3. Cấu trúc lặp For

Cú pháp:

```
For <biến> = <Biểu thức 1>  To <Biểu thức 2>
[Step = <Biểu thức 3>]
        <Các lệnh 1>
        [Exit For]
        <Các lệnh 2>
Next [biến]
```

trong đó:

**Biểu thức 1** và **biểu thức 2** là hai chỉ số đầu và cuối của **biến**

**Biểu thức 3**: có thể có giá trị là âm hay dương. **Step** nhằm để tăng hay giảm **biến** sau mỗi lần lặp. Nếu không xác định rõ thì **Step** sẽ có giá trị là 1.

**Exit For**: sẽ nhảy khỏi vòng lặp mà không cần phải thực hiện đến cuối vòng lặp.

Ví dụ: tính tổng các số từ 1 đến 9

```
Dim Sum, i As Integer
Sum = 0
```

```
For i = 1  To 10
        Sum = Sum + i
Next i
```

hoặc cú pháp khác:

```
For Each <phần tử> In <nhóm>
        <Các lệnh 1>
        [Exit For]
        <Các lệnh 2>
Next [phần tử]
```

trong đó nhóm có thể là mảng hay một tập hợp.

Ví dụ:

```
Dim TestArray(10) As Integer, I As Variant
For Each I In TestArray
   TestArray(I) = I
Next I
```

## IV.4. Cấu trúc lặp Do ... Loop

Cú pháp:

```
Do <While | Until> <điều kiện lặp>
<Các lệnh 1>
[Exit Do]
<Các lệnh 2>
[Loop]
```

hoặc

```
Do
<Các lệnh 1>
[Exit Do]
<Các lệnh 2>
Loop <While | Until> <điều kiện lặp>
```

hoặc

```
While <điều kiện lặp>
        <Các lệnh>
Wend
```

**Điều kiện lặp**: là một biểu thức logic để xác định có nên thực hiện vòng lặp tiếp tục hay không.

**Exit Do**: sẽ nhảy khỏi vòng lặp mà không cần phải thực hiện đến khi điều kiện lặp bị vi phạm.

*Giải thích*

Lệnh **Do While <điều kiện lặp>** hoặc **While <điều kiện lặp>** hoặc **Loop While <điều kiện lặp>**: Khi điều kiện lặp vẫn còn đúng thì vòng lặp sẽ tiếp tục được thực hiện.

Lệnh **Do Until <điều kiện lặp>** hoặc **Loop Until <điều kiện lặp>**: lặp cho đến khi nào điều kiện lặp vẫn còn sai (điều kiện lặp ở đây trái ngược với lệnh **While**)

Chú ý là chúng ta có đến hai vị trí kiểm tra điều kiện lặp, vị trí thứ nhất ở trên các lệnh 1 và các lệnh 2 (như các lệnh **Do While/Until ... Loop** ). Khi đó điều kiện lặp sẽ được kiểm tra trước khi thực hiện các lệnh trong vòng lặp. Vị trí thứ hai ở dưới các lệnh 1 và các lệnh 2 (như các lệnh **Do ... Loop While/Until** ). Khi đó các lệnh 1 và các lệnh 2 sẽ được thực hiện rồi chương trình mới kiểm tra điều kiện lặp.

Ví dụ: tính tổng các số từ 1 đến 10

```
' Dùng lệnh While
Dim Sum, i As Integer
i = 0
Sum = 0
Do While i <= 10
        Sum = Sum + i
        i = i + 1
Loop

' Dùng lệnh Until
Dim Sum, i As Integer
i = 0
Sum = 0
Do Until i > 10
        Sum = Sum + i
        i = i + 1
Loop
```

Chúng ta có thể lồng các vòng lặp For, While, Until ... vào nhau.

# V. HÀM VÀ THỦ TỤC

## V.1. Giới thiệu

Khái niệm hàm và thủ tục của Visual Basic cũng giống như với các ngôn ngữ lập trình khác. Trong Visual Basic, bạn có thể đơn giản hóa các tác vụ trong chương trình của bạn bằng cách chia chương trình thành các phần nhỏ hơn. Các phần nhỏ hơn này, được gọi là thủ tục, làm cho chương trình bạn có thể dễ dàng được mở rộng về sau này. Thủ tục được sử dụng trong các tác vụ lặp lại hay chia sẻ như các hàm xử lý chuỗi, dữ liệu, các hàm tính toán...

Có hai tác dụng chính khi lập trình với thủ tục:

- Thủ tục cho phép bạn chia nhỏ chương trình thành các đơn vị riêng biệt nhỏ hơn, do đó mỗi khi có lỗi bạn có thể debug mỗi phần này dễ dàng hơn khi thực hiện trên toàn bộ chương trình.
- Bạn có thể tận dụng, chia sẻ các thủ tục trong một chương trình hay giữa các chương trình với nhau.

Có nhiều loại thủ tục được dùng trong Visual Basic.:

- Thủ tục **Sub** (gọi tắt là thủ tục): không trả về giá trị.
- Thủ tục **Function** (gọi tắt là hàm): trả về một giá trị.
- Thủ tục **Property** có thể trả về và gán nhiều giá trị. Thủ tục loại này có thể thiết lập một tham chiếu đến một đối tượng.

## V.2. Khai báo

Cú pháp khai báo một hàm hay thủ tục trong Visual Basic

```
[Public / Private] [Static] Sub <Tên Thủ Tục> ([Danh sách tham số])
        <các lệnh>
[End Sub]

[Public / Private] [ Static] Function  <Tên Hàm> ([Danh sách tham số]) As <Giá trị trả về>
        <các lệnh>
        Tên Hàm = Giá trị trả về
[End Function]
```

## V.3. Tham số của hàm và thủ tục

Danh sách tham số có dạng sau

```
[Optional][ByVal / ByRef ] <Tên biến> As <Kiểu Dữ liệu> [=Giá trị mặc định]
```

**ByVal**: Tham số được truyền vào theo kiểu tham trị

**ByRef**: Mặc định. Tham số được truyền vào theo kiểu tham biến

Ví dụ: Thủ tục hoán đổi giá trị hai số.

```
Private Sub Swap(ByVal a As Integer, ByVal b As Integer )
        Dim c As Integer
        c = a
        a = b
        b = c
End Sub

Private Sub Test()
        Dim a, b As Integer
        a = 5
        b = 6
        Swap(a,b)
        ' Lúc này a vẫn sẽ có giá trị là 5
        ' Lúc này b vẫn sẽ có giá trị là 6
```

**End Sub**

Khi thực hiện xong thủ tục Test ta thấy a, b vẫn giữ lại giá trị cũ chứ không có giá trị mới. Điều này là do ở hàm Swap ta sử dụng hai tham số được truyền vào theo kiểu tham trị. Muốn sau khi thực hiện hàm Swap, hai biến a, b hoán đổi giá trị cho nhau thì chúng ta phải sửa lại khai báo hàm Swap như sau

```
Private Sub Swap(ByRef a As Integer, ByRef b As Integer )
        Dim c As Integer
        c = a
        a = b
        b = c
End Sub
```

Chúng ta bắt buộc phải đưa đầy đủ tham số cho thủ tục như khi khai báo, nếu không hệ thống sẽ báo lỗi và không thực hiện. Ví dụ như trong các hàm Swap như trên: ta phải đưa vào hai tham số là a và b. Tuy nhiên, với VB chúng ta có thể cho phép tham số nào có thể bỏ qua bằng cách khai báo thêm từ khóa Optional phía trước các khai báo tham số. Lúc đó VB sẽ tự động lấy giá trị mặc định cho tham số đó, giá trị mặc định này do người lập trình thiết lập khi khai báo Thủ tục/Hàm. Nếu người sử dụng không khai báo giá trị mặc định thì VB sẽ tự động phát sinh ra giá trị này tùy thuộc vào kiểu dữ liệu tương ứng. Cụ thể là VB sẽ gán cho tham số giá trị 0 nếu tham số đó kiểu số, còn nếu tham số có kiểu là chuỗi thì tham số sẽ được gán giá trị là chuỗi rỗng.

Ví dụ:

```
Private Function Inc(ByVal Num As Integer,Optional Byval
Delta As Integer = 1) As Integer
        Num = Num + Delta
        Inc = Num
End Function
```

Để sử dụng hàm trên, chúng ta sẽ sử dụng lệnh:

```
C = Inc(A,5) 'tương đương với C = A + 5
```

Hay
**C = Inc(A)** 'tương đương với C = A + 1

Tuy nhiên, có nhiều lúc chúng ta không biết trước số lượng tham số trong một hàm/ thủ tục. Muốn làm như vậy ta sử dụng từ khóa **ParamArray**. Lúc đó, chúng ta có thể sử dụng hàm **IsMissing** bên trong thủ tục/hàm để xác định giá trị của một tham số có được truyền vào trong thủ tục/hàm hay không? Hàm này trả về giá trị TRUE nếu tham số không được truyền vào trong thủ tục/hàm. Chú ý rằng, ta chỉ sử dụng từ khóa này với các tham số có kiểu là **Variant**.

Ví dụ: viết hàm tính tổng của một dãy số được gởi vào, nhưng ta không biết trước số lượng của dãy số này

```
Function Sum(ParamArray ArrayNums As Variant) As Integer
        Dim Total As Integer
        Dim Atom As Variant
        For Each Atom in ArrayNums
                Total = Total + Atom
        Next Atom
        Sum = Total
End Function
```

Khi đó chúng ta có thể gọi hàm như sau:

```
        nSum = Sum(1,3,5)
```

hay

```
        nSum = Sum(1,3,5,7,9,11)
```

### V.4. Gọi một hàm hay thủ tục trong cùng Module

Cú pháp khi gọi một thủ tục
**<Tên Thủ Tục> Tham Số 1, Tham Số 2 ,...**
hay
**Call <Tên Thủ Tục>(Tham Số 1, Tham Số 2 ,...)**

Cú pháp khi gọi một hàm
**<Tên Biến> = <Tên Thủ Tục> (Tham Số 1, Tham Số 2 ,...)**

## V.5. Gọi một hàm hay thủ tục khác Module

### V.5.1. Hàm và thủ tục của Form

Tất cả các lời gọi thủ tục hay hàm của một form khác thì phải chỉ rõ form chứa hàm hay thủ tục đó. Giả sử bạn gọi thủ tục SomeSub trong một form có tên là Form1 như sau:

```
Call Form1.SomeSub(arguments)
```

### V.5.2. Hàm và thủ tục trong Class Module

Giống như lời gọi hàm hay thủ tục trong form, lời gọi hàm hay thủ tục trong một class module cần phải đi kèm với một con trỏ chỉ đến một thể hiện của lớp.

Ví dụ: DemoClass là một thể hiện của lớp tên là Class1:

```
Dim DemoClass as New Class1
DemoClass.SomeSub
```

### V.5.3. Hàm và thủ tục trong Standard Module

Nếu hàm hay thủ tục được gọi ở trong Standard Module thì bạn không cần đi kèm tên của Standard Module trong lời gọi hàm. Tuy nhiên, nếu có hiện tượng hàm được gọi trùng tên trong cả hai Standard Module thì bạn phải chỉ rõ module nào chứa hàm đó.

Ví dụ: chúng ta có thủ tục tên CommonName trong Module1 và Module2, một lời gọi hàm CommonName trong Module2 sẽ thực hiện thủ tục CommonName trong Module2. Nếu bạn muốn gọi thủ tục CommonName trong Module1, bạn phải sử dụng:

```
Module2.CommonName([tham số])
```

## Chương VI

# DEBUG

## I. GIỚI THIỆU

### I.1. Ý nghĩa

Bug là những lỗi của chương trình mà ta phát hiện khi chạy nó. Debug là công việc loại tất cả những lỗi trong chương trình để nó chạy ổn định trong mọi hoàn cảnh.

Thông thường muốn debug một lỗi trước hết ta phải tìm hiểu lý do khiến lỗi xuất hiện. Visual Basic cung cấp rất nhiều công cụ hỗ trợ chúng ta phân tích cách vận hành của chương trình. Những công cụ hỗ trợ debug giúp bạn có thể tìm ra lỗi, nhưng chúng cũng có thể là công cụ giúp bạn hiểu hơn về cách làm việc của chương trình.

Chương này sẽ giúp các bạn các phương pháp hạn chế lỗi và cách sử dụng các công cụ để debug chương trình.

### I.2. Các loại lỗi

Có ba loại lỗi trong chương trình mà bạn có thể gặp phải:

- Lỗi Compile (Lỗi biến dịch)
- Lỗi Run-time (Lỗi phát sinh trong lúc chạy chương trình).
- Lỗi Logic.

### I.2.1. Lỗi compile

*Lỗi compile* là kết quả của sự không tuân thủ cú pháp của chương trình. Ví dụ như bạn quên một dấu nháy kép (") khi khai báo một hằng chuỗi hay viết một câu lệnh Next mà không có lệnh For phía trước. May mắn cho chúng ta là Visual Basic tự động phát hiện ra các lỗi này ngay lúc chúng ta viết và biên dịch chương trình.

Xét câu lệnh sau:

```
Left
```

Left là một từ đúng trong ngôn ngữ lập trình Visual Basic, nhưng từ này đứng một mình mà không có đối tượng nào ở phía trước, cú pháp đúng của câu lệnh này là **<đối tượng>.Left**. Nếu bạn chọn chức năng Auto Syntax Check, Visual Basic sẽ hiển thị thông báo lỗi khi bạn viết chương trình trong màn hình cửa sổ lệnh (Code window).

**Để thiết lập chức năng Auto Syntax Check**

1. Từ menu **Tools**, chọn **Options**, sau đó chọn tab **Editor** trên dialog **Options**.

2. Chọn **Auto Syntax Check**.

### I.2.2. Run-Time Errors

*Lỗi Run-time* xảy ra khi ứng dụng đang chạy (và được phát hiện bởi Visual Basic). Lúc đó trong chương trình có một câu lệnh không thể thực hiện được. Ví dụ, phép chia cho 0 là không thể thực hiện được. Chúng ta hãy xem câu lệnh:

```
TocDo = QuangDuong / ThoiGian
```

Nếu biến ThoiGian bằng zero, phép chia là một câu lệnh không hợp lệ, mặc dù câu lệnh này đúng cú pháp. Ứng dụng phải chạy trước khi nó có thể phát hiện ra lỗi này.

### I.2.3. Lỗi Logic

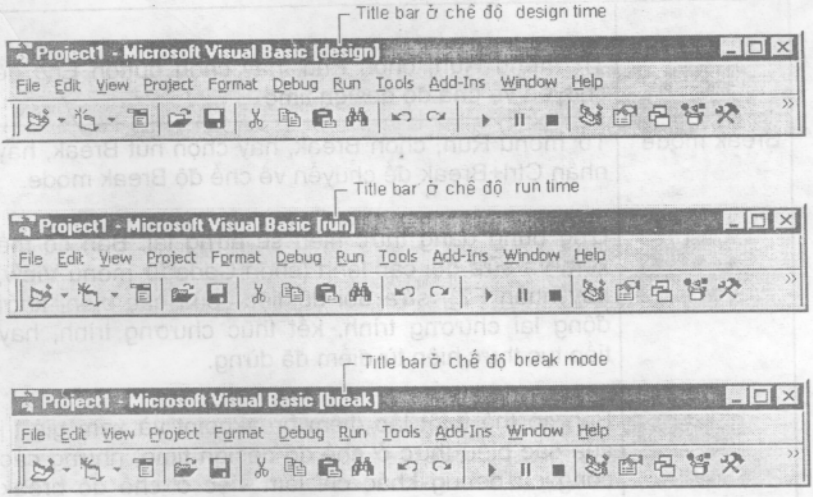
*Lỗi Logic* xảy ra khi ứng dụng không thể hoạt động theo ý định. Một chương trình không có lỗi cú pháp nhưng chạy ra kết quả sai. Đây là lỗi do người lập trình vô tình không viết lệnh đúng. Chỉ bằng cách kiểm tra ứng dụng và phân tích kết quả thu được bạn mới có thể tìm ra được nguyên nhân của lỗi này.

## II. DESIGN TIME, RUN TIME, VÀ BREAK MODE

Để kiểm tra và debug chương trình thì bạn phải chú ý đến ba chế độ làm việc trong Visual Basic. Đầu tiên, bạn sử dụng chế độ design time để thiết kế chương trình, và chế độ run time để thực hiện chương trình. Còn chế độ break (*break mode*), là chế độ dừng thực hiện chương trình để bạn kiểm tra giá trị của các biến và cách thức thực hiện các lệnh.

### II.1. Xác định trạng thái hiện thời

Title bar của Visual Basic cho bạn biết bạn đang ở chế độ nào.




**Hình VI.1**: *Chế độ làm việc hiển thị lên Title Bar*

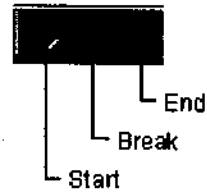
Đặc điểm của ba chế độ này được liệt kê ở bản sau:

| Chế độ | Mô tả |
|---|---|
| Design time | Hầu hết các công việc tạo ứng dụng được thực hiện ở chế độ design time. Bạn có thể thiết kế form, control, viết lệnh, và sử dụng cửa sổ Properties để thiết lập hay xem các thuộc tính của các control. Bạn không thể dùng các công cụ debug, ngoại trừ việc thiết lập các breakpoint và tạo các biểu thức trong cửa sổ Watch.<br><br>Từ menu Run, chọn Start, hay chọn nút Run để chuyển sang chế độ run time.<br><br>Nếu ứng dụng của bạn chứa lệnh. Chọn Step từ menu Run (hay nhấn F8) để chuyển sang chế độ break mode và thực hiện câu lệnh đầu tiên. |
| Run time | Khi ứng dụng đang chạy (ở chế độ run time), bạn có thể xem các lệnh nhưng bạn không thể thay đổi nó. |

| | |
|---|---|
| | Từ menu Run, chọn End, hay chọn button End để chuyển về chế độ design time. |
| Break mode | Từ menu Run, chọn Break, hay chọn nút Break, hay nhấn Ctrl+Break để chuyển về chế độ Break mode.<br><br>Ứng dụng đang thực hiện sẽ dừng lại. Bạn có thể xem và sửa đổi các lệnh (chọn Code từ menu View, hay nhấn F7), sửa đổi dữ liệu của các biến, khởi động lại chương trình, kết thúc chương trình, hay tiếp tục thực hiện từ điểm đã dừng.<br><br>Bạn có thể thiết lập điểm breakpoint và xem giá trị của các biểu thức ở chế độ design time, nhưng các công cụ debug khác chỉ làm việc ở chế độ break mode. |

## II.2. Dùng Toolbar để thay đổi chế độ làm việc

Toolbar cung cấp ba nút để bạn có thể nhanh chóng chuyển sang chế độ khác một cách nhanh chóng.




**Hình VI. 2**: *Ba nút Start, Break, End trong Toolbar*

Các nút này có hiệu lực tùy theo ứng dụng của bạn đang ở trong được chế độ run-time, design-time, hay break mode. Cụ thể:

| Chế độ | Các nút có hiệu lực |
|---|---|
| Design time | Start |
| Run time | Break, End |
| Break | Continue, End (trong chế độ break mode, nút Start trở thành nút Continue) |

Chạy từng phần trong ứng dụng của bạn.

Thông thường bạn không thể biết được câu lệnh nào gây ra lỗi mà chỉ biết được một đoạn lệnh phát sinh lỗi. Trong trường hợp này, bạn dùng breakpoint để ứng dụng chuyển sang chế độ BreakMode tại vùng phát sinh lỗi, sau đó bạn có thể sử dụng các lệnh Step Into, Step Over để chạy từng bước các lệnh. Nếu cần thiết, bạn có thể nhảy bước qua các câu lệnh.

| Chế độ | Mô tả |
|---|---|
| Step Into (F8) | Thực hiện dòng lệnh tiếp theo trong ứng dụng và nhảy vào trong thủ tục. |
| Step Over | Step Over tương tự như Step Into, ngoại trừ trường hợp câu lệnh hiện tại là câu lệnh có chứa một lời gọi thủ tục. Không giống như Step Into, sẽ nhảy vào thực hiện các câu lệnh trong thủ tục được gọi, Step Over sẽ thực hiện các lệnh gọi thủ tục như là các lệnh đơn và sau đó nhảy đến câu lệnh kế tiếp trong thủ tục hiện hành. |
| Step Out | Thực hiện thủ tục hiện thời và dừng lại ở dòng kế sau dòng gọi thủ tục hiện thời. |
| Run To Cursor | Bạn có thể dùng command Run To Cursor để chọn một câu lệnh trong đoạn lệnh của bạn. Khi đó ứng dụng sẽ thực hiện, dừng lại và chuyển sang chế độ BreakMode dừng ngay tại dòng lệnh bạn chọn. Biện pháp này được sử dụng khi bạn không muốn thực hiện từng bước các lệnh không quan trọng hay các vòng lặp lớn. |

## III. CÁCH SỬ DỤNG CÁC CÔNG CỤ DEBUG

Các công cụ Debug được thiết kế để giúp chúng ta tránh các lỗi:

- Lỗi Logic và lỗi run-time.
- Xem thứ tự và kết quả thực hiện các câu lệnh.

Ví dụ một kết quả sai có thể được tạo ra sau một loạt các tính toán ở phía trước. Trong Debug, chúng ta đi xác định xem tác vụ đó phát xuất từ đâu, ở đâu xảy ra lỗi. Có thể là do bạn không khởi tạo biến, chọn sai toán tử, chọn sai công thức...

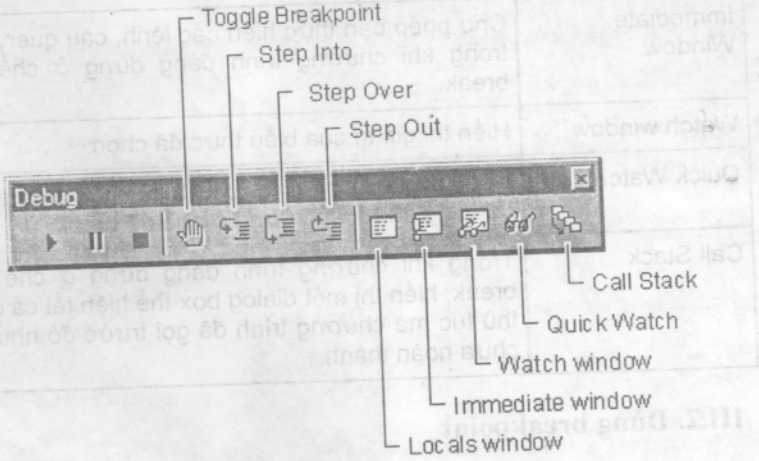
Không có một công cụ thông minh để tìm ra lỗi cho chúng ta. Các công cụ Debug chỉ giúp bạn hiểu cái gì hoạt động trong chương trình của bạn. Chúng giúp bạn chụp lại trạng thái của các biến, control trong ứng dụng như:

- Sự hiển thị của giao diện người dùng (UI, user interface).
- Giá trị của biến, biểu thức, và thuộc tính.
- Các thủ tục đang được kích hoạt.

Tóm lại, bạn càng hiểu chương trình của mình làm việc như thế nào, bạn càng nhanh chóng tìm ra lỗi.

### III.1. Debug Toolbar

Trong Toolbar Debug, Visual Basic cung cấp nhiều chức năng như hình bên dưới.




**Hình VI. 3**: *Debug ToolBar*

Để hiển thị Debug toolbar, bạn right-click trên Visual Basic toolbar và chọn chức năng Debug.

Bảng sau mô tả mục đích của từng Button trên ToolBar

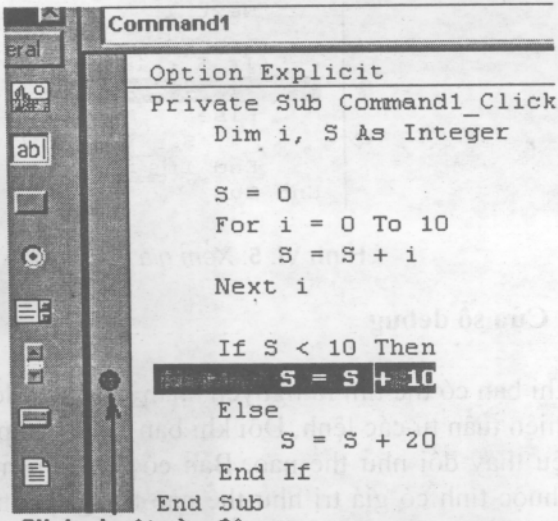
| Công cụ Debug | Mục đích |
|---|---|
| Breakpoint | Chỉ định một dòng trong cửa sổ lệnh (Code window) nơi mà Visual Basic sẽ dừng thực hiện ứng dụng khi chạy đến câu lệnh này. Khi đó chúng ta có thể xem trạng thái hiện hành cửa chương trình. |
| Step Into | Thực hiện dòng lệnh tiếp theo trong ứng dụng và nhảy vào trong thủ tục. |
| Step Over | Thực hiện dòng lệnh tiếp theo trong ứng dụng mà không nhảy vào thủ tục. |
| Step Out | Thực hiện thủ tục hiện thời và dừng lại ở dòng kế sau dòng gọi thủ tục hiện thời. |
| Locals Window | Hiển thị giá trị hiện thời của các biến cục bộ |

| | |
|---|---|
| Immediate Window | Cho phép bạn thực hiện các lệnh, câu query, ... trong khi chương trình đang dừng ở chế độ break. |
| Watch window | Hiển thị giá trị của biểu thức đã chọn. |
| Quick Watch | Liệt kê giá trị hiện tại cảu biểu thức trong khi chương trình đang dừng ở chế độ break. |
| Call Stack | Trong khi chương trình đang dừng ở chế độ break, hiển thị một dialog box thể hiện tất cả các thủ tục mà chương trình đã gọi trước đó nhưng chưa hoàn thành. |

### III.2. Dùng breakpoint

Nhiều khi chúng ta muốn chương trình chuyển sang chế độ BreakMode tại một hay nhiều nơi nào đó trong đoạn code chương trình. Cách hay nhất và trực quan nhất là bạn sử dụng breakpoint. Khi chương trình chuyển sang chế độ BreakMode, bạn có thể xem giá trị các biến ... tại thời điểm đó.

Cách đặt breakPoint trong chương trình rất đơn giản. Trước tiên, bạn đặt con trỏ vào vị trí bạn muốn đặt breakpoint. Sau đó, bạn click chuột vào hàng bên trái của đoạn mã lệnh muốn đặt.

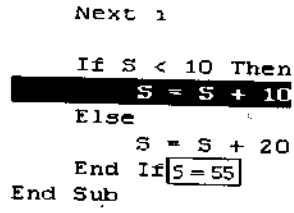



**Hình VI. 4**: *Chọn vị trí đặt breakpoint*

Khi đó, một dấu tròn đỏ sẽ xuất hiện như hình trên. Muốn hủy bỏ breakpoint, bạn chỉ việc click chuột vào điểm đỏ trên. Bạn cũng có thể hủy nhiều breakpoint một lúc bằng cách sử dụng menu **Debug** và chọn **Clear All Breakpoints.**

Bạn cũng có thể dùng phím F9 để thực hiện việc đặt và hủy các breakpoint.

Chương trình chuyển sang chế độ BreakMode khi gặp breakpoint, bạn có thể xem giá trị các biến ở các cửa sổ Debug (bạn có thể xem chi tiết ở phần sau). Tuy nhiên, bạn có thể xem nhanh giá trị của một biến, hay biểu thức bằng cách để chuột trên biến hay biểu thức đó. Khi đó một tooltip sẽ hiện ra và cho chúng ta biết giá trị của biến hay biểu thức đó.



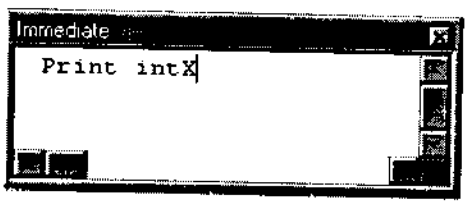

**Hình VI. 5**: *Xem giá trị của biến*

## III.3. Cửa sổ debug

Đôi khi bạn có thể tìm ra nguyên nhân phát sinh lỗi bằng cách thực hiện tuần tự các lệnh. Đôi khi bạn cũng cần phân tích các dữ liệu thay đổi như thế nào. Bạn có thể xem một biến hay một thuộc tính có giá trị như thế nào trong quá trình chương trình thực hiện, bạn cũng có thể gán lại giá trị một biến ... Tất cả các công đoạn trên bạn có thể thực hiện được bởi có sự hỗ trợ của các cửa sổ debug.

Với cửa sổ debug, bạn có thể xem giá trị của biểu thức và các biến trong khi thực hiện từng câu lệnh trong ứng dụng của bạn. Có ba cửa sổ debug là: cửa sổ Immediate, cửa sổ Watch, và cửa sổ Local.

- Cửa sổ *Immediate* hiển thị các thông tin về kết quả thực hiện các câu lệnh debug trong chương trình của bạn, hay bạn có thể gõ trực tiếp yêu cầu của mình vào cửa sổ.



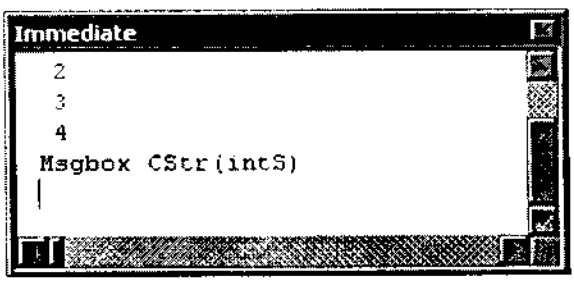

**Hình VI. 6**: *Cửa sổ Immediate*

Ví dụ: khi bạn thực hiện đoạn lệnh sau.

```
Private Sub Command1_Click()
   Dim intX, intS As Integer

   intS = 0
   For intX = 0 To 10
      intS = intS + intX
      Debug.Print intX
   Next intX
End Sub
```

Cửa sổ Immediate sẽ in lần lượt từng giá trị của biến intX. Nếu bạn sử dụng breakpoint để dừng khi thực hiện vòng lặp For, ngay trong cửa sổ Immediate bạn có thể gõ vào các lệnh và bắt Visual Basic thực hiện tức thì.

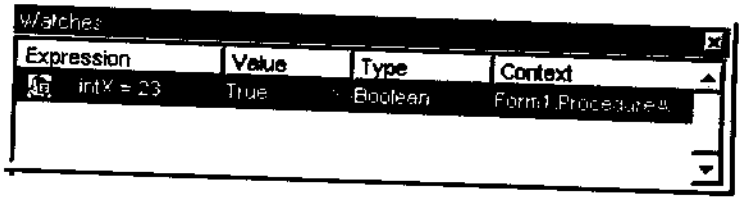
Giả sử, bạn chuyển sang chế độ Debug khi intX có giá trị là 5. Lúc đó, trong cửa sổ Immediate, bạn gõ vào: MsgBox CStr(intX) và nhấn Enter.




**Hình VI. 7**: *Đánh thêm dòng MsgBox CStr(intX) vào cửa sổ Immediate. Khi đó bạn sẽ thấy một Message Box có số 10 hiện lên, trong khi chương trình đang chạy.*

- Cửa sổ *Watch window* hiển thị giá trị của các *watch expression*. *Watch expression* là các biểu thức mà giá trị của chúng được bạn quyết định để điều khiển các công việc khi chương trình chạy. Một *break expression* là một biểu thức làm cho Visual Basic

chuyển sang chế độ break mode khi điều kiện bạn định nghĩa là đúng. Trong cửa sổ Watch, cột Context (ngữ cảnh) xác định ngữ cảnh của các watch expression, đây là các thủ tục hay module. Trong cửa sổ Watch windows, giá trị của một watch expression được hiển thị khi nó đang ở trong ngữ cảnh của mình. Ngược lại, cột Value hiển thị một message để cho biết watch expression không ở trong ngữ cảnh thích hợp. Để hiển thị cửa sổ Watch bạn vào menu View, chọn Watch Window.




**Hình VI. 8**: *Cửa sổ Watch*

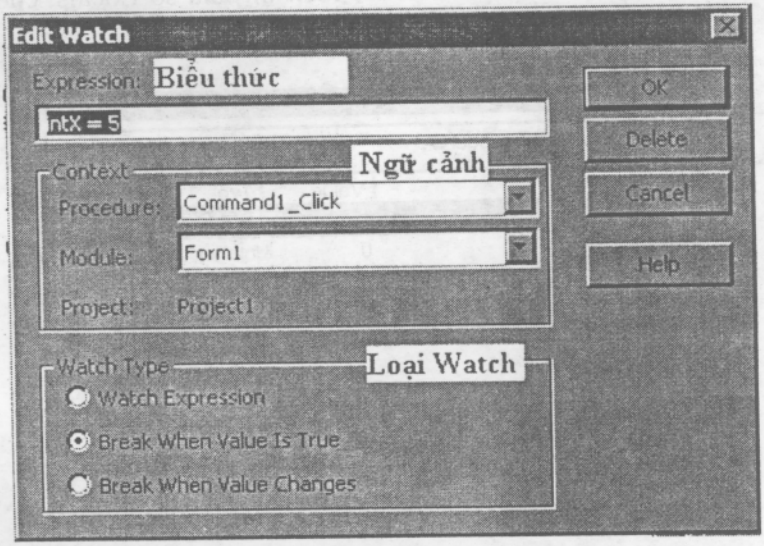
Ví dụ: bạn muốn chương trình của bạn sẽ thoát khỏi chế độ Run Time và chuyển sang chế độ Debug khi biến intX = 5 bạn thực hiện các bước sau:

- Viết lệnh cho thủ tục xử lý sự kiện Click của một nút nào đó như sau.

```
Private Sub Command1_Click()
   Dim intX, intS As Integer

   intS = 0
   For intX = 0 To 10
      intS = intS + intX
   Next intX
End Sub
```

- Right – Click vào cửa sổ Watch, chọn Add Watch.

- Trong dialog Add Watch: Điền các expression (biểu thức) cần theo dõi giá trị trong textbox Expression. Sau đó, bạn chọn context (ngữ cảnh) của expression. Cuối cùng bạn chọn loại của các biểu thức đã chọn. Trong ví dụ này bạn chọn như hình sau:



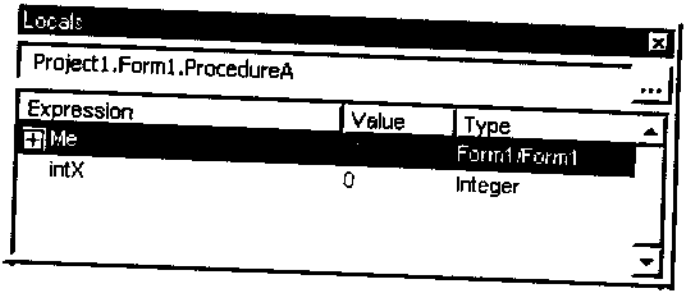

**Hình VI. 9**: *Visual Basic sẽ chuyển sang chế độ debug khi biểu thức intX=5 là đúng (True)*

- Nhấn **OK**.
- Bạn cho chương trình thực hiện bằng cách nhấn F5. Nhấn Command Button. Khi đó Visual Basic sẽ chuyển sang chế độ debug khi intX=5.

Bạn cũng có thể đơn thuần xem giá trị của một biểu thức mà không sử dụng biểu thức này để điều khiển việc chuyển đổi chế độ trong Visual Basic

bằng cách chọn RadioButton **Watch Expression** trong Group **Watch Type.**

- *Cửa sổ Locals* cho biết giá trị của mọi biến trong thủ tục hiện tại. Khi bạn chuyển từ thủ tục này sang thủ tục khác, nội dung của cửa sổ Locals sẽ thay đổi tương ứng với thủ tục mới. Để hiển thị cửa sổ Locals, chọn **Locals Window** ở menu **View**.




**Hình VI. 10**: *Cửa sổ Locals*

Chương VII

# KẾT NỐI CƠ SỞ DỮ LIỆU VỚI ADO

## GIỚI THIỆU SƠ LƯỢC CÁC KỸ THUẬT KẾT NỐI CSDL

**DAO (Data Acess Objects):** DAO là kỹ thuật kết nối cơ sở dữ liệu riêng của Microsoft. Kỹ thuật này chỉ dùng với Jet Database Engine. Lợi điểm của kỹ thuật này là tính dễ dùng, nhanh chóng và tiện lợi, tuy nhiên DAO chỉ có thể liên kết được với hệ quản trị CSDL Microsoft Access.

**ODBC (Open Database Connectivity)**: ODBC được thiết kế để cho chương trình kết nối với nhiều loại CSDL mà chỉ dùng một phương cách duy nhất. Điều này giúp cho lập trình viên chỉ sử dụng một phương thức duy nhất để truy cập vào các hệ quản trị CSDL. Hơn thế nữa, khi chúng ta nâng cấp lên hệ quản trị CSDL cao hơn, ví dụ nâng cấp từ Access lên SQLServer, thì sự sửa đổi trong chương trình sẽ rất ít.

**RDO (Remote Data Object)**: Một trong những lý do chính để RDO được thiết kế là giải quyết khó khăn về sự rắc rối của ODBC. Cách lập trình với RDO đơn giản như DAO, nhưng khả năng của nó thì như ODBC. RDO cho phép nhiều chương trình kết nối với nhiều CSDL. Tuy nhiên, RDO không được sử dụng nhiều.

**ADO (ActiveX Data Objects)**. Sự khác biệt chính giữa ADO và DAO là ADO cho phép ta làm việc với mọi loại nguồn dữ liệu (data sources), không nhất thiết phải là CSDL Micosoft Access hay SQL Server. Data Source có thể là danh sách các địa chỉ email, hay một file text, trong đó mỗi hàng là một

record gồm những fields ngăn cách bởi các dấu phẩy hay dấu tab.

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

## I. GIỚI THIỆU KỸ THUẬT LẬP TRÌNH ADO

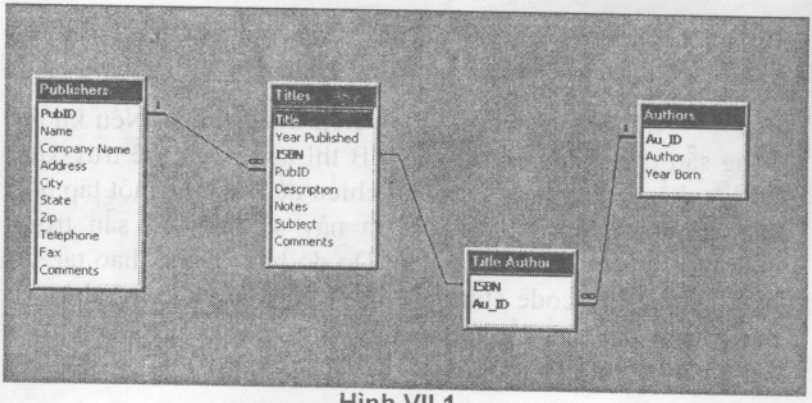
ADO (ActiveX Data Objects) là một kỹ thuật lập trình kết nối cơ sở dữ liệu phát triển gần đây nhất của VB. Không giống như những kỹ thuật truy cập CSDL khác của VB như RDO (Remote Data Objects), DAO (Data Access Objects), ADO có thể cung cấp cho người lập trình nhiều chọn lựa trong việc truy xuất dữ liệu. Mặt khác, ADO không truy cập trực tiếp đến một cơ sở dữ liệu, ADO làm việc với tầng thấp hơn là OLE DB provider và OLE DB provider này có nhiệm vụ truy xuất đến nhiều loại dữ liệu khác nhau, sau đó trình bày CSDL ngược lại đến ADO. Thuận lợi nhất của ADO là chúng ta có thể thay đổi OLE DB provider, ví dụ: nếu dự án làm việc sử dụng CSDL Oracle ta chọn OLE DB provider cho Oracle hoặc CSDL SQL Server ta chọn OLE DB provider cho SQL Server. Kỹ thuật ADO cho phép truy cập dữ liệu thông qua hai cách sau:

• **Data controls (các điều khiển dữ liệu).** Một điều khiển dữ liệu (Data control) được định nghĩa là một điều khiển có chức năng giao tiếp, cập nhật CSDL. Để nhìn thấy dữ liệu ta phải dùng một số control thông thường khác

(textbox, checkbox, label,...) thiết lập một số thuộc tính và "kết buộc dữ liệu" ("**data binding**") đến data control này.

• **Object interface (giao tiếp đối tượng).** Nếu không dùng sẵn các data control của VB thì ta vẫn có thể truy xuất dữ liệu bằng cách tạo một tham chiếu đến ADO - một tập hợp các đối tượng mới được tạo ra này sẽ luôn có sẵn trong chương trình khi cần dùng đến. Do đó bạn có thể thao tác dữ liệu trực tiếp từ code chương trình (dùng các phương thức và thuộc tính do các đối tượng này cung cấp) mà không phải dùng bất kỳ data control nào.

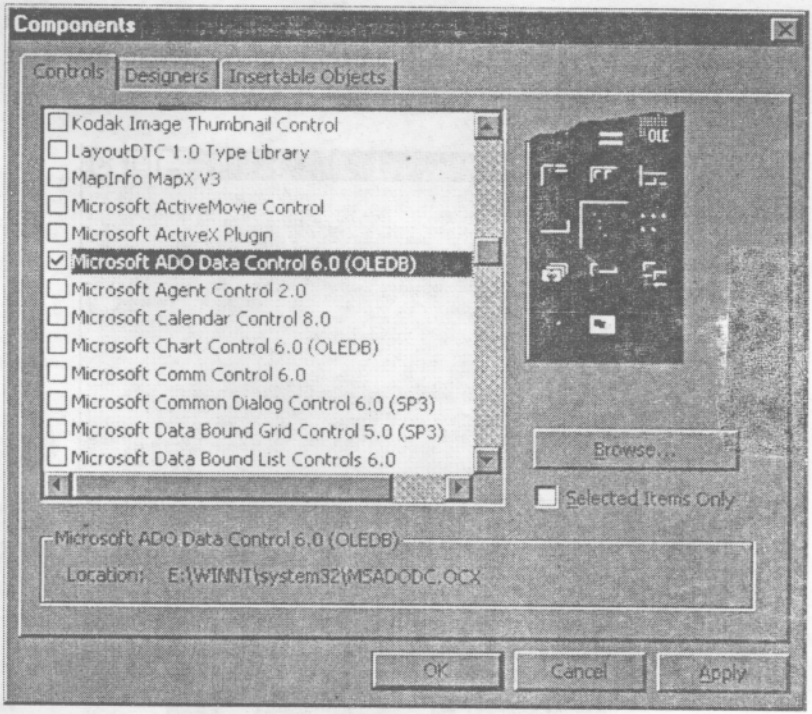
Tuy "Data control" dễ thiết lập hơn nhưng sử dụng "Object interface" thì mạnh hơn và uyển chuyển hơn. Các ví dụ dưới đây sử dụng CSDL là file BIBLIO.mdb, khi cài đặt VB cung cấp sẵn database này đồng thời cung cấp luôn provider cho Access là "Microsoft Jet 4.0 OLE DB Provider" (hoặc "Microsoft Jet 3.51 OLE DB Provider"). Đường dẫn đến database E:\Program Files\Microsoft Visual Studio\VB98\BIBLIO.mdb (nếu bạn cài đặt ổ đĩa nào thì đường dẫn đến database sẽ tương ứng với ổ đĩa đó, nếu cài chương trình VB ở ổ đĩa C thì : C:\ Program Files\Microsoft Visual Studio\VB98\BIBLIO.mdb). Database này gồm bốn table: Publishers (nhà xuất bản), Titles (Đề mục sách), Authors (Tác giả) và Title Authors (Tác giả-đề mục), quan hệ giữa các table thể hiện trong hình sau.




Hình VII.1

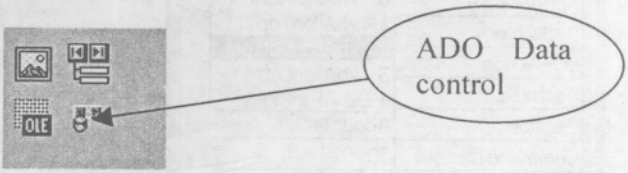
## II. SỬ DỤNG ADO DATA CONTROL

Sau đây bạn sẽ thiết lập một form kết nối CSDL có sử dụng **điều khiển dữ liệu ADO (ADO data control).** Giả sử muốn xem, thêm, xóa và sửa thông tin sách trong database BIBLIO, bạn sẽ thiết kế một form thực hiện các chức năng này, chọn nguồn dữ liệu để thao tác là table Titles (Đề mục sách). ADO Data Control không có sẵn trên hộp thanh công cụ (Toolbox) như một số control thông dụng khác (textbox, combobox, label,...) do đó bạn phải đưa vào bằng cách: trên menu của VB chọn Project →Components (hoặc click chuột phải trên Toolbox chọn Components), trong tab Control bạn check vào ô "Microsoft ADO Data Control 6.0 (OLEDB)" rồi nhấn nút Apply:



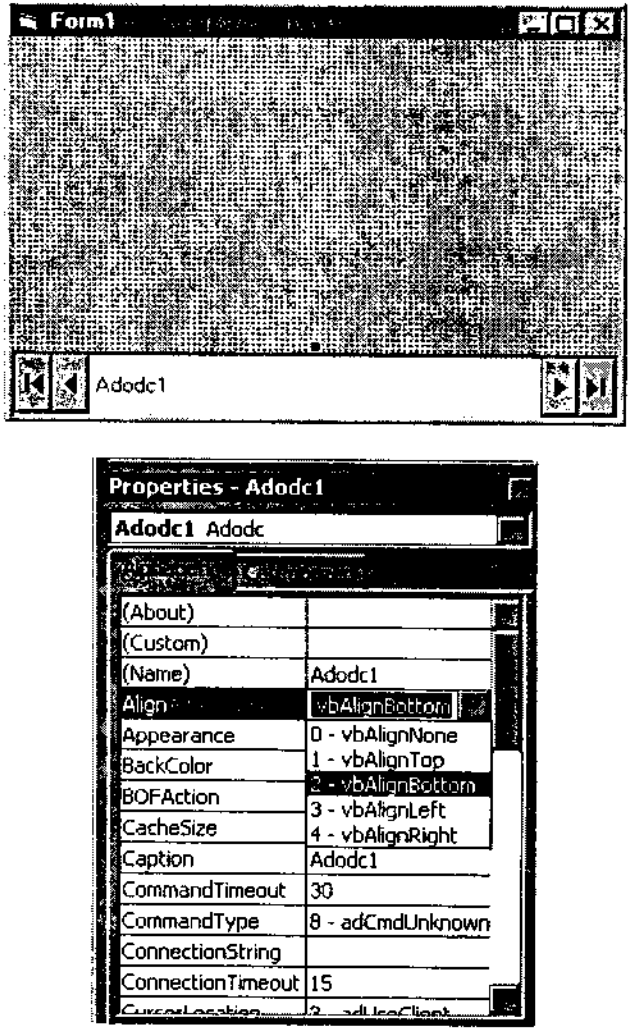

**Hình VII.2**

Lúc này trên Toolbox xuất hiện biểu tượng như hình dưới đây:




Bây giờ, bạn tạo mới **một form**, trong cửa sổ properties đặt thuộc tính name của form là **'frmTitles'**, caption là **'Ten sach'**, StartUpPosition là **'Center Screen'** (để form này xuất hiện giữa màn hình khi chạy chương trình). Kế tiếp bạn vẽ control này vào form, tên mặc định của control là Adodc1

(bạn có thể đổi tên khác), để cho dễ nhìn bạn chỉnh thuộc tính Align là '2-vbAlignBottom' (control nằm ở dưới form) trong cửa sổ Properties như sau:




**Hình VII.3**

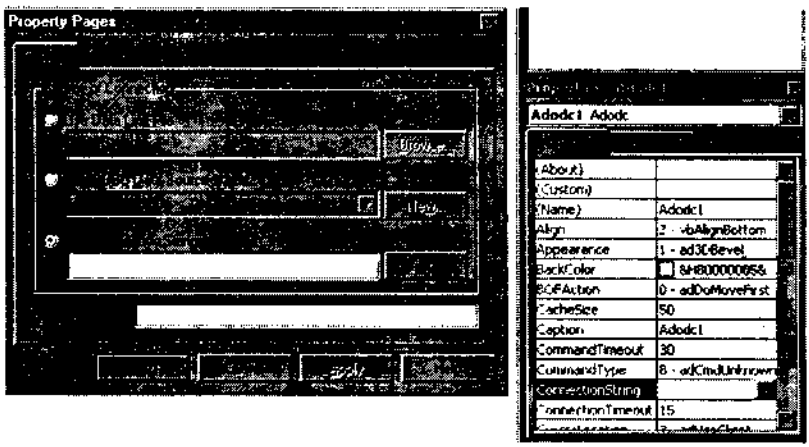
Điều quan trọng tiếp theo: kết nối ADO Data Control với nguồn dữ liệu (Data source) từ CSDL, bạn thực hiện hai bước sau:

**Bước 1**: Tạo một Connection đến CSDL: click chuột chọn Adodc1, trong cửa sổ properties chọn thuộc tính ConnectionString khi đó xuất hiện button ... bạn click vào button này sẽ hiện ra cửa sổ Property Page, có ba chọn lựa:

- Use Data Link File: cho phép bạn load những thông số kết nối đã được lưu trước trong MDL file (Microsoft Data Link).

- Use ODBC Data Source Name: bạn có thể chọn một DSN từ những DSN trong ODBC của máy bạn (bạn phải thiết lập DSN này trong ODBC trước), hoặc bạn có thể tạo mới DSN ngay lúc này bằng cách nhấp chọn 'New' và cấu hình cho DSN này.

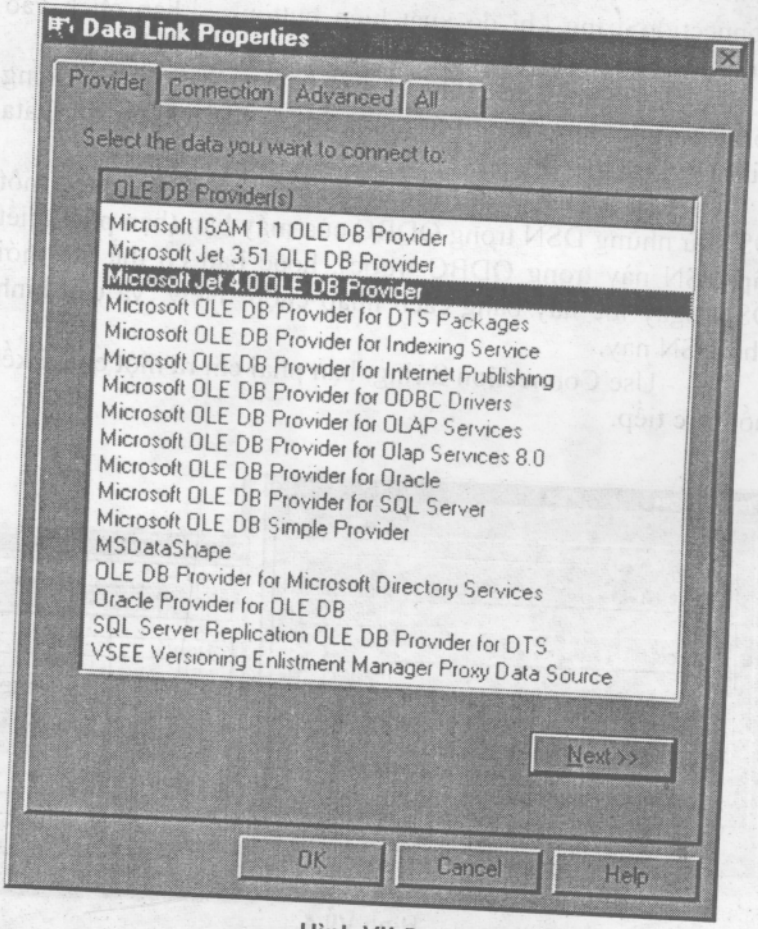
- Use Connection String: bạn phải chỉ ra một chuỗi kết nối trực tiếp.



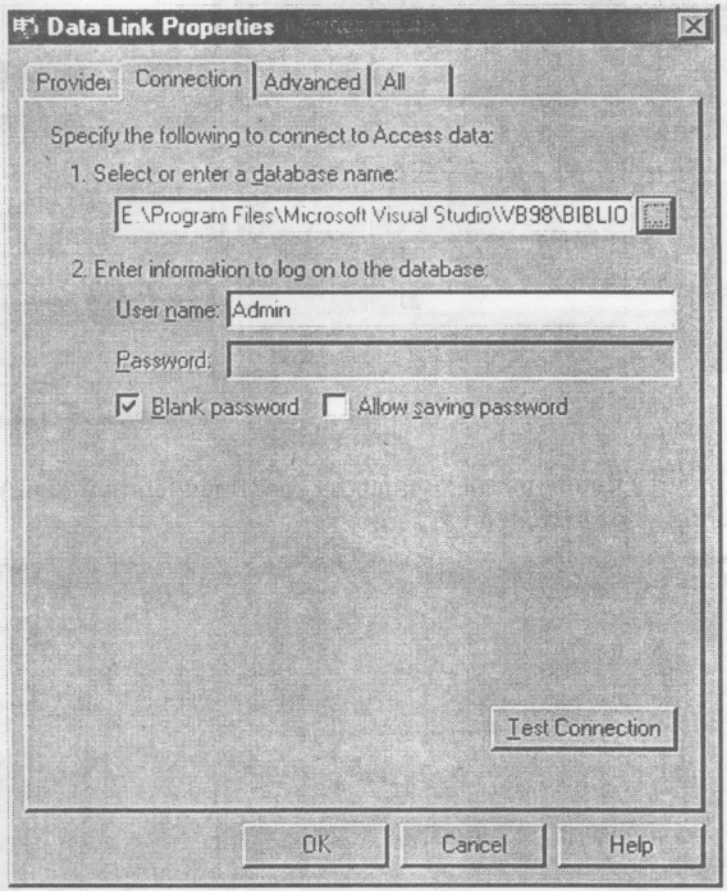

**Hình VII.4**

Chọn option thứ ba là 'Use Connection String', click 'Build', chọn Provider là "Microsoft Jet 4.0 OLE DB Provider" (nếu CSDL SQL Server bạn chọn Provider: Microsoft OLE DB

Provider for SQL Server, tương tự cho Oracle,...), click 'Next' để chỉ đường dẫn đến database BIBLIO.mdb (nếu CSDL là SQL Server bạn cần chỉ ra tên server, username, password và tên database cần thao tác). Kế tiếp click 'Test Connection' thấy thành công →OK → Apply →OK.

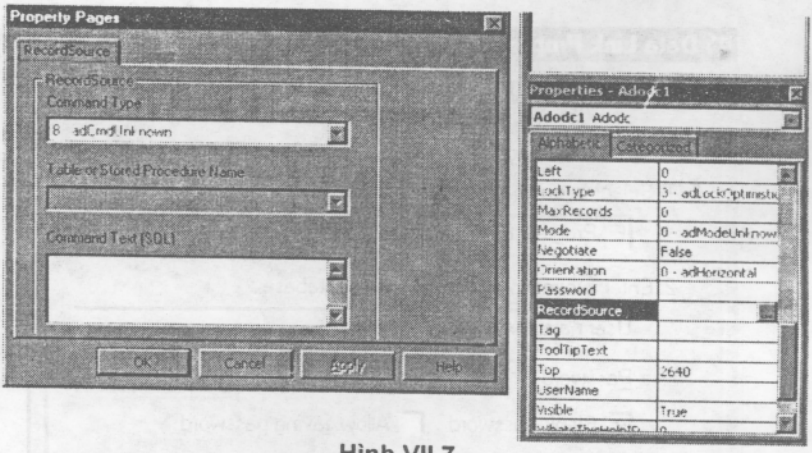



**Hình VII.5**



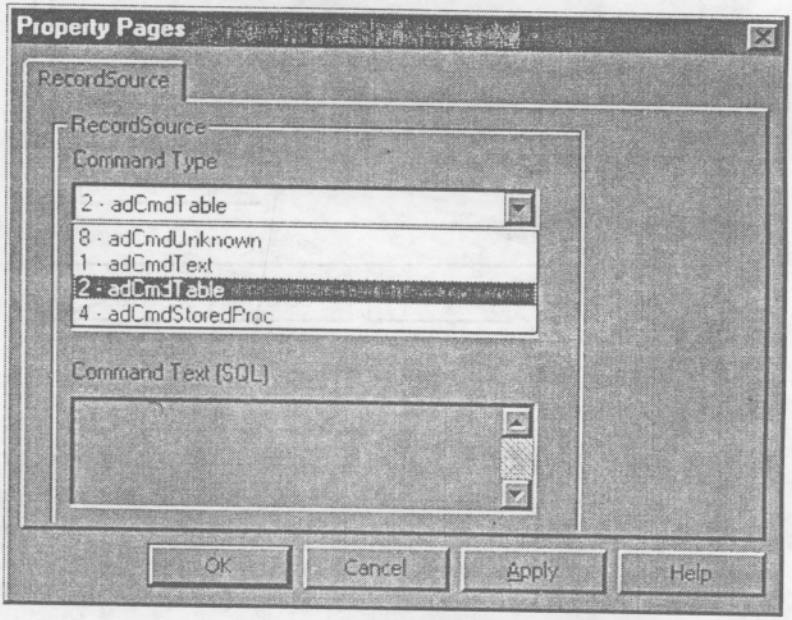

**Hình VII.6**

**Bước 2**: Thiết lập Record Source từ CSDL: bây giờ bạn cần chỉ ra tập hợp mẩu tin nào (thuộc tính Record Source) sẽ được lấy ra từ nguồn dữ liệu trên. Trong cửa sổ properties chọn thuộc tính Record Source khi đó xuất hiện button ..., click vào button này hiện ra cửa sổ Property Page:




**Hình VII.7**

Khi chọn combobox 'Command Type' (kiểu lệnh thi hành) sổ xuống sẽ có các chọn lựa sau:




**Hình VII.8**

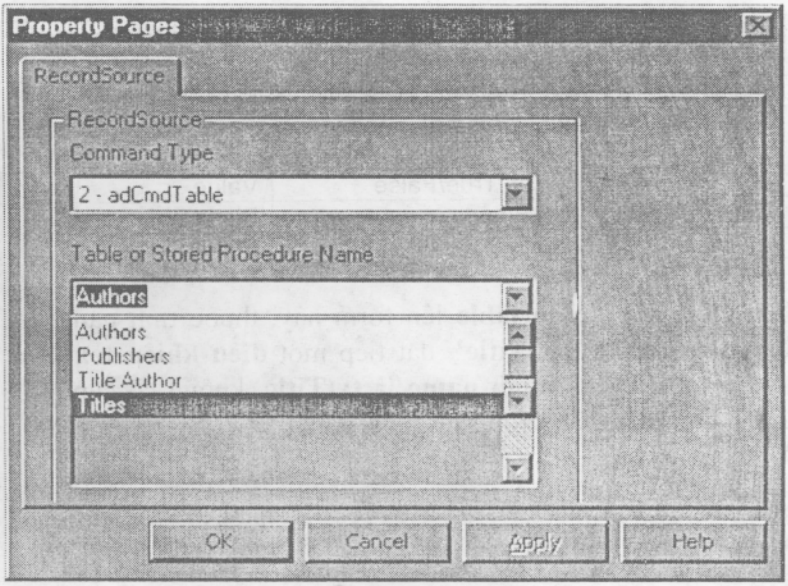
- 8–adCmdUnknown: kiểu lệnh không xác định.

- 1-adCmdText: kiểu lệnh thi hành một câu truy vấn (câu SELECT) trên nguồn dữ liệu, câu SELECT này được ghi bên dưới combobox Commnad Text (SQL), kết quả trả về tập hợp các mẩu tin thỏa câu Select trên.

- 2-adCmdTable: tên của một table cụ thể (kết quả trả về toàn bộ các mẩu tin của table).

- 4-adCmdStoredProc: kiểu lệnh thi hành một Store Procedure trên database, tên của Store procedure này (có thể có tham số đi kèm) ghi bên dưới Commnad Text (SQL).

Tùy thuộc bạn chọn kiểu lệnh nào thì textbox 'Command Text' bên dưới có thể Enable hoặc không. Để xem toàn bộ đề mục sách hiện có trong database → chọn table Titles, chọn '2-adCmdTable', kế đến chọn 'Titles' trong combobox thứ hai và nhấn OK.

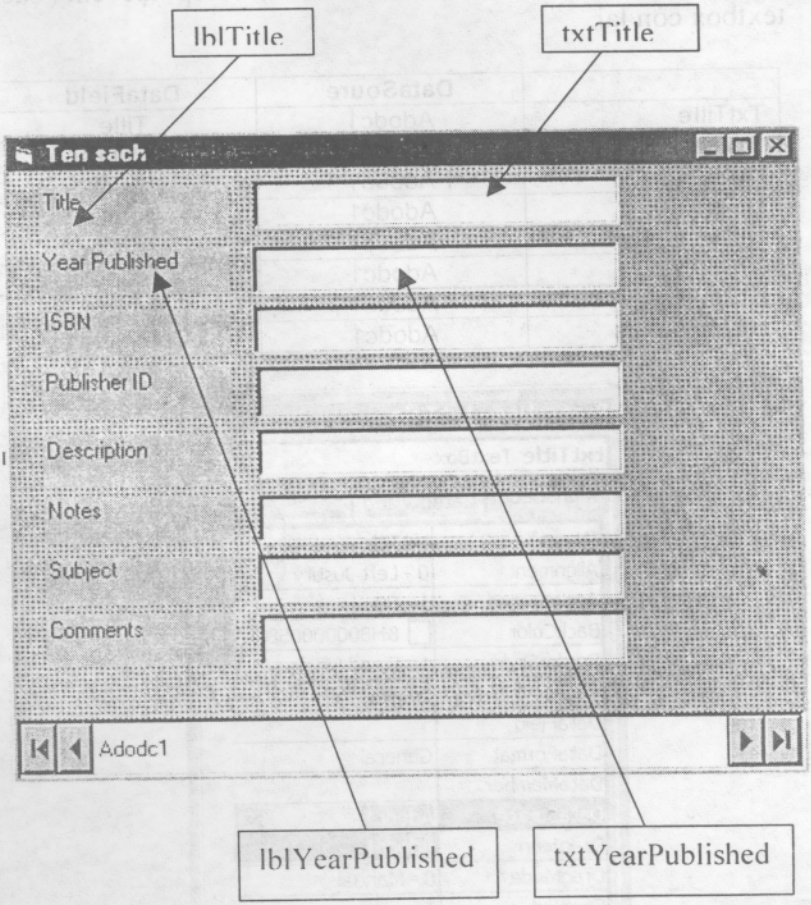



**Hình VII.9**

Sau khi thực hiện xong việc kết nối và chọn nguồn dữ liệu, để hiển thị dữ liệu từ ADO Data Control bạn cần phải có các control thông thường được **'kết buộc dữ liệu' (data binding)** với ADO Data Control này. Do table Titles có 8 field: Title (tên sách, kiểu Text), Year Published (năm xuất bản, kiểu Number), ISBN (mã sách, kiểu Text), PubID (mã nhà xuất bản, kiểu Number), Description (mục lục, kiểu Text), Notes (ghi chú, kiểu Text), Subjects (chủ đề, kiểu Text), Comments (mô tả sách, kiểu memo) nên khi thiết kế form bạn dùng 8 textbox và đặt tên các textbox: txtTitle, txtYearPublished, txtISBN, txtPubID, txtDescription, txtNotes, txtSubjects, txtComments. Tương ứng bạn cho 8 label đặt trước 8 textbox với các tên: lblTitle, lblYearPublished, lblISBN, lblPubID, lblDescription, lblNotes, lblSubjects, lblComments. Dưới đây là bảng tham khảo dùng control nào phù hợp với kiểu dữ liệu có trong database.

| Tên control | Kiểu dữ liệu trong database | Thuộc tính control được sử dụng |
|---|---|---|
| Label | Text, Numeric, Date | Caption |
| Textbox | Text, Memo, Numeric, Date | Text |
| Checkbox | Logical, True/False | Value |
| PictureBox | Long Binary | Picture |
| Image | Long Binary | Picture |

Đặt một **điều khiển lable** lên form này, thuộc tính name là **lblTitle**, caption là **'Title'**, đặt tiếp một **điều khiển textbox** lên form với thuộc tính **name** là **txtTitle**, không có caption. Thực hiện thao tác tương tự cho 7 label và 7 textbox còn lại. Xem hình sau:

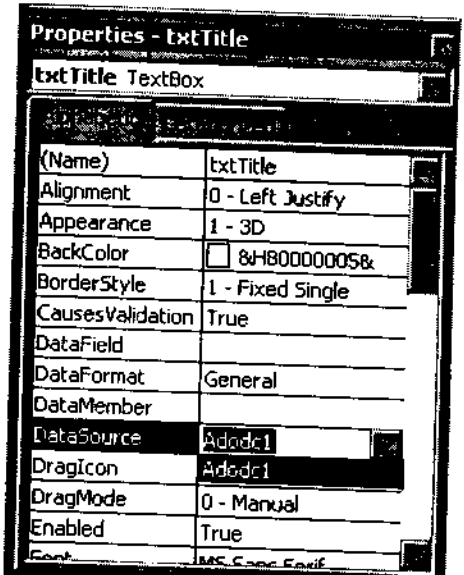


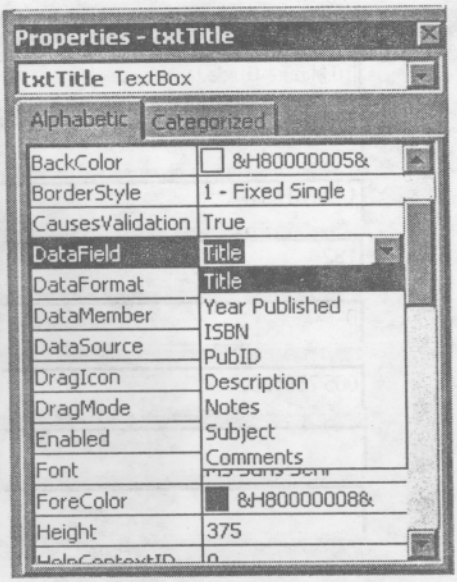

**Hình VII.10**

Để hiển thị dữ liệu lên các textbox này, bạn phải kết buộc mỗi textbox với từng field cụ thể của table 'Titles' dựa vào hai thuộc tính quan trọng của textbox là: DataSource và DataField. Ở đây, DataSource của 8 textboxt đều là Adodc1. Thao tác cho từng DataField như sau: chọn textbox có name 'txtTitle' bạn set thuộc tính DataSource là Adodc1, DataField là Title (combobox DataField hiển thị tất cả field trong table

Title : Title, Year Published, ISBN,...). Tiếp tục cho các textbox còn lại.

| | DataSoure | DataField |
|---|---|---|
| TxtTitle | Adodc1 | Title |
| txtYearPublished | Adodc1 | Year Published |
| txtISBN | Adodc1 | ISBN |
| txtPubID | Adodc1 | PubID |
| txtDescription | Adodc1 | Description |
| txtNotes | Adodc1 | Notes |
| txtSubject | Adodc1 | Subject |
| txtComments | Adodc1 | Comments |

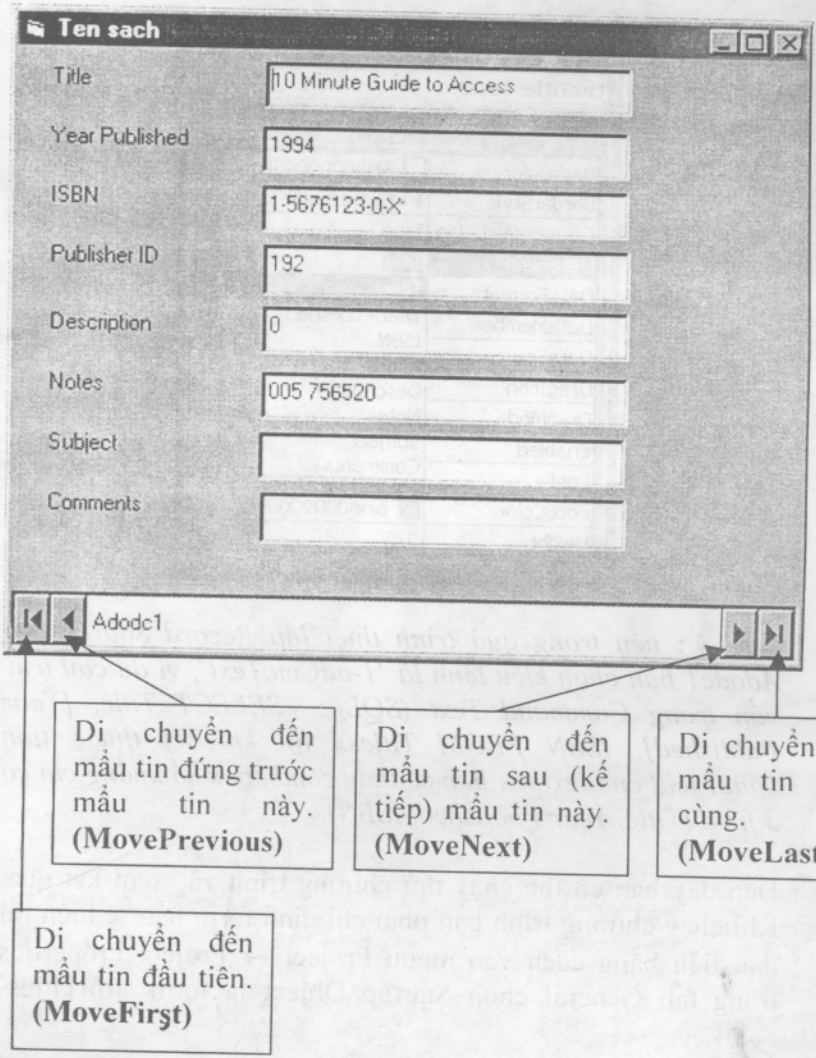





**Lưu ý :** *nếu trong quá trình thiết lập Record Source cho Adodc1 bạn chọn kiểu lệnh là '1-adCmdText', ví dụ câu truy vấn trong Command Text (SQL) : «SELECT Title, [Year Published], ISBN FROM Titles» thì khi set thuộc tính DataField của textbox bạn sẽ thấy combobox sổ xuống chỉ có 3 field (Title, Year Published, ISBN).*

Đến đây bạn có thể chạy thử chương trình và xem kết quả. Khi chạy chương trình bạn phải chỉ định form nào sẽ hiển thị đầu tiên bằng cách vào menu Project → Project Properties trong tab General chọn Startup Object là form 'frmTitles' →OK.

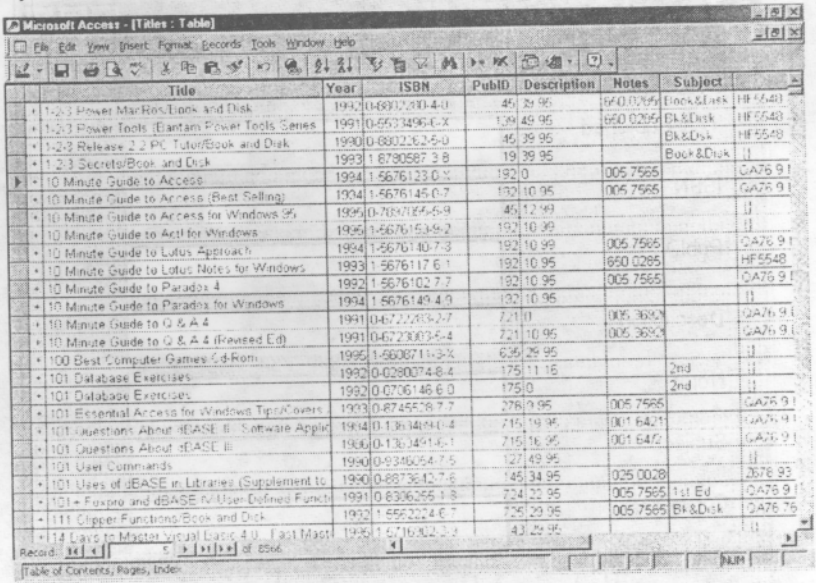
**Màn hình kết quả (Hình VII.11)**




**Hình VII.11**

Ở mỗi thời điểm con trỏ chỉ trỏ đến một mẩu tin (còn gọi là mẩu tin hiện hành).

**Một ví dụ minh họa vị trí các mẩu tin**: khởi động chương trình MS Access bạn open database BIBLIO.mdb, kế tiếp open table Titles thấy hình sau:



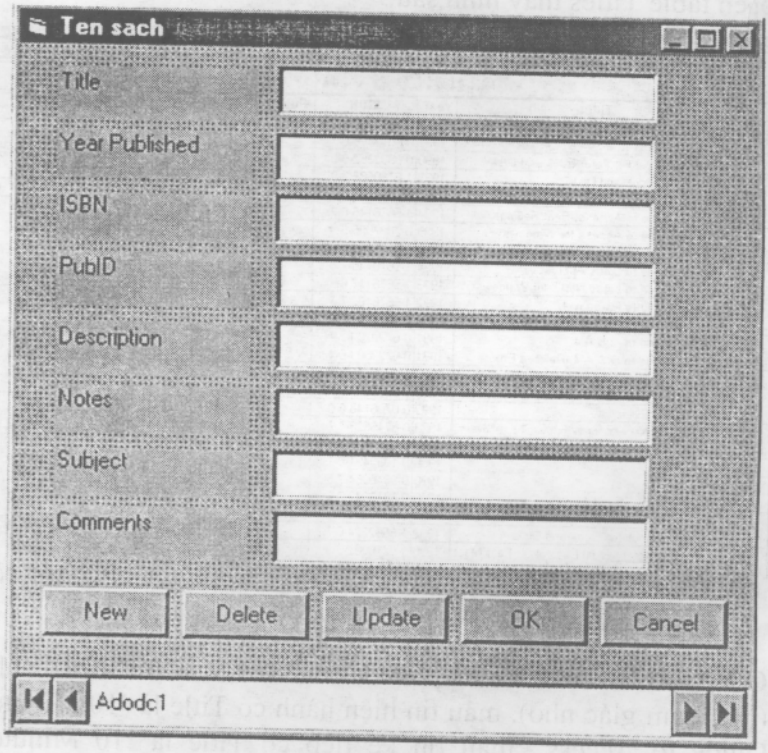

| Title | Year | ISBN | PubID | Description | Notes | Subject | |
|---|---|---|---|---|---|---|---|
| 1-2-3 Power MacRos/Book and Disk | 1992 | 0-8802280-4-0 | 45 | 39.95 | 650.0285 | Book&Disk | HF5548 |
| 1-2-3 Power Tools (Bantam Power Tools Series | 1991 | 0-5533496-6-X | 139 | 49.95 | 650.0285 | Bk&Disk | HF5548 |
| 1-2-3 Release 2.2 PC Tutor/Book and Disk | 1990 | 0-8802262-5-0 | 45 | 39.95 | | Bk&Disk | HF5548 |
| 1-2-3 Secrets/Book and Disk | 1993 | 1-8780587-3-8 | 19 | 39.95 | | Book&Disk | [] |
| 10 Minute Guide to Access | 1994 | 1-5676123-0-X | 192 | 0 | 005.7565 | | QA76.9 |
| 10 Minute Guide to Access (Best Selling) | 1994 | 1-5676145-0-7 | 192 | 10.95 | 005.7565 | | QA76.9 |
| 10 Minute Guide to Access for Windows 95 | 1995 | 0-7897055-5-9 | 45 | 12.99 | | | [] |
| 10 Minute Guide to Act! for Windows | 1995 | 1-5676153-9-2 | 192 | 10.99 | | | [] |
| 10 Minute Guide to Lotus Approach | 1994 | 1-5676140-7-3 | 192 | 10.99 | 005.7565 | | QA76.9 |
| 10 Minute Guide to Lotus Notes for Windows | 1993 | 1-5676117-6-1 | 192 | 10.95 | 650.0285 | | HF5548 |
| 10 Minute Guide to Paradox 4 | 1992 | 1-5676102-7-7 | 192 | 10.95 | 005.7565 | | QA76.9 |
| 10 Minute Guide to Paradox for Windows | 1994 | 1-5676149-4-9 | 192 | 10.95 | | | [] |
| 10 Minute Guide to Q & A 4 | 1991 | 0-6722283-2-7 | 721 | 0 | 005.3692 | | QA76.9 |
| 10 Minute Guide to Q & A 4 (Revised Ed) | 1991 | 0-6723003-5-4 | 721 | 10.95 | 005.3692 | | QA76.9 |
| 100 Best Computer Games Cd-Rom | 1995 | 1-5608711-3-X | 635 | 29.95 | | | [] |
| 101 Database Exercises | 1992 | 0-0280074-8-4 | 175 | 11.16 | | 2nd | [] |
| 101 Database Exercises | 1992 | 0-0706146-6-0 | 175 | 0 | | 2nd | [] |
| 101 Essential Access for Windows Tips/Covers | 1993 | 0-8745528-7-7 | 278 | 9.95 | 005.7565 | | QA76.9 |
| 101 Questions About dBASE II: Software Applic | 1984 | 0-1363489-0-4 | 715 | 19.95 | 001.6421 | | QA76.9 |
| 101 Questions About dBASE III | 1986 | 0-1363491-6-1 | 715 | 16.95 | 001.6472 | | QA76.9 |
| 101 User Commands | 1990 | 0-9346054-7-5 | 127 | 49.95 | | | [] |
| 101 Uses of dBASE in Libraries (Supplement to | 1990 | 0-8873642-7-6 | 145 | 34.95 | 025.0028 | | 2678.93 |
| 101+ Foxpro and dBASE IV User-Defined Functi | 1991 | 0-8306255-1-8 | 724 | 22.95 | 005.7565 | 1st Ed | QA76.9 |
| 111 Clipper Functions/Book and Disk | 1992 | 1-5562224-6-7 | 725 | 29.95 | 005.7565 | Bk&Disk | QA76.76 |
| 14 Days to Master Visual Basic 4.0: Fast Mast | 1995 | 1-5716302-2-3 | 43 | 29.95 | | | [] |

**Hình VII.12**

Con trỏ mẩu tin (record pointer) trỏ đến mẩu tin hiện hành (hình tam giác nhỏ), mẩu tin hiện hành có Title là '10 Minute Guide to Access', mẩu tin kế tiếp có Title là '10 Minute Guide to Access (Best Selling)', mẩu tin đứng trước mẩu tin hiện hành có Title là '1-2-3 Secrets/Book and Disk'. Mẩu tin đầu tiên có Title là '1-2-3 Power MacRos/Book and Disk'. Mẩu tin cuối cùng có Title là 'Zen of Code Optimization/Book and Disk' (trường hợp này bạn phải kéo thanh cuộn đến hàng cuối cùng mới nhìn thấy được).

Để thực hiện thêm, xóa, sửa thông tin sách trong Database cần phải thiết kế bổ sung vào form một số nút có properties Name lần lượt là: cmdNew (caption: New), cmdDelete (caption:

Delete), "cmdUpdate" (caption: Update), cmdOK (caption: OK), cmdCancel (caption: Cancel) như hình sau.




**Hình VII.13**

Mô tả

*Thêm mới một đề mục sách*: người dùng nhấn nút New, gọi phương thức AddNew của recordset trong Adodc1 (lúc này các textbox txtTitle, txtYearPublished, txtISBN,...có **'kết buộc dữ liệu' (data binding)** với ADO Data Control này tự động được làm trống cho người dùng nhập dữ liệu vào), cho hiện lên các nút OK (đồng ý), Cancel (bỏ qua), làm mờ các nút Delete, Update đồng thời cho người dùng edit trên các textbox.

*Cập nhật đề mục sách hiện hành (mẫu tin hiện hành)*: người dùng nhấn nút Update, cho hiện lên các nút OK, Cancel, làm mờ các nút New, Delete đồng thời cho người dùng edit trên các textbox.

*Xóa đề mục sách hiện hành*: người dùng nhấn nút Delete, đưa ra thông báo xác nhận lần nữa có xóa hay không (nếu đồng ý thì xóa đề mục sách này và di chuyển đến đề mục sách kế), trạng thái các nút không thay đổi.

Đối với các textbox txtTitle, txtYearPublished, txtISBN,...có **'kết buộc dữ liệu' (data binding)** với ADO Data Control khi chúng ta vô tình edit các Textbox này sẽ làm thay đổi dữ liệu bên dưới (mà điều này ta không muốn, ta chỉ muốn mọi thao tác thay đổi dữ liệu đều phải được kiểm soát). Các Textbox có hỗ trợ thuộc tính Lock. Nếu Textbox.Lock = True cho phép người dùng vẫn nhìn thấy dữ liệu nhưng không edit được (không làm thay đổi giá trị trên textbox), ngược lại Textbox.Lock = False cho phép người dùng edit trên nó (có thể thay đổi giá trị trên textbox).

Viết code cho các sự kiện click nút New, Delete, Update, OK, Cancel và load form, khóa (lock) các control textbox như sau:

```
Private Sub Form_Load()
   SetButtons (False)  'làm mờ hai nút OK và Cancel
   LockedControls (True) 'không cho người dùng edit các controls
End Sub

Private Sub SetButtons(bVal As Boolean)
   cmdOK.Enabled = bVal
   cmdCancel.Enabled = bVal
   cmdNew.Enabled = Not bVal
   cmdUpdate.Enabled = Not bVal
   cmdDelete.Enabled = Not bVal
End Sub
```

```
Private Sub LockedControls(bVal As Boolean)
' không cho người dùng edit các controls khi bVal = True và ngược
lại
    txtTitle.Locked = bVal
    txtYearPublished.Locked = bVal
    txtISBN.Locked = bVal
    txtPubID.Locked = bVal
    txtDescription.Locked = bVal
    txtNotes.Locked = bVal
    txtSubject.Locked = bVal
    txtComments.Locked = bVal
End Sub

Private Sub cmdNew_Click()
    Adodc1.Recordset.AddNew
    SetButtons (True)   'cho hai nút OK và Cancel hiện lên
    LockedControls (False) 'cho người dùng edit các controls
End Sub

Private Sub cmdOK_Click()
'kiểm tra các ràng buộc toàn vẹn ở đây, không thỏa=>Exit Sub
'bắt đầu xử lí
    Adodc1.Recordset.Update
    SetButtons (False)  'làm mờ 2 button OK và Cancel
    LockedControls (True) 'không cho người dùng edit các controls
End Sub

Private Sub cmdCancel_Click()
    Adodc1.Recordset.CancelUpdate
    setcontrols (False) 'làm mờ hai nút OK và Cancel
    LockedControls (True) 'không cho người dùng edit các controls
End Sub

Private Sub cmdDelete_Click()
    On Error GoTo DeleteErr
    If (Adodc1.Recordset.BOF = False And Adodc1.Recordset.EOF =
False) Then   'có mẫu tin cần xóa
        returnvalue = MsgBox("Do you want to delete this record?",
vbOKCancel + vbQuestion, "Notice")
```

```
        If returnvalue = vbOK Then
            Adodc1.Recordset.Delete  'xóa mẩu tin hiện hành
            Adodc1.Recordset.MoveNext  ' di chuyển đến mẩu tin kế
            If Adodc1.Recordset.EOF Then Adodc1.Recordset.MoveLast
'kiểm tra EOF
        End If
        Exit Sub
    End If
DeleteErr:
    MsgBox Err.Description
End Sub

Private Sub cmdUpdate_Click()
    SetButtons (True)
    LockedControls (False) 'cho người dùng edit các controls
End Sub
```

Chạy chương trình và thực hiện thao tác thêm, xóa, sửa.
(**Lưu ý:** *khi thêm mới một mẩu tin bạn thêm giá trị PubID phải là giá trị có sẵn trong Database vì field PubID trong table Titles là khóa ngoại).*

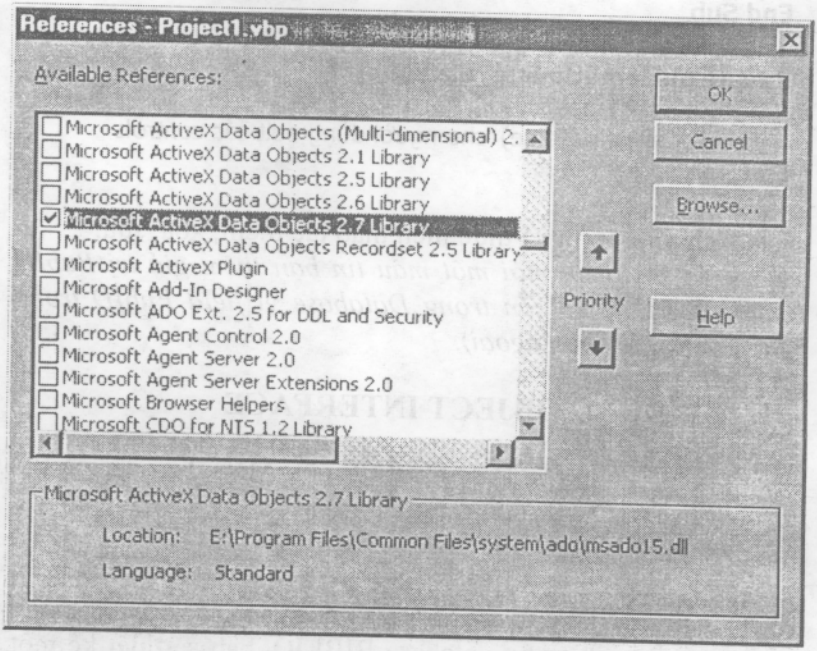
## III. SỬ DỤNG OBJECT INTERFACE

Nếu không dùng các Data Controls của ADO, bạn vẫn có thể giao tiếp với CSDL bên dưới thông qua các đối tượng có sẵn trong thư viện ADO.

Với yêu cầu tương tự như trên: xem, thêm, xóa và sửa thông tin đề mục sách trong database BIBLIO, bạn sẽ thiết kế một form thực hiện các chức năng này, chọn nguồn dữ liệu để thao tác là table Titles (Đề mục sách). Đầu tiên bạn phải tạo một kết nối đến CSDL, sau đó chọn ra nguồn dữ liệu cần thao tác. ADO cung cấp đối tượng 'Connection' để kết nối CSDL, đối tượng 'Recordset' để chứa tập các mẩu tin mà người dùng cần khai thác đồng thời cung cấp một số phương pháp cho người dùng truy cập và cập nhật dữ liệu, ngoài ra còn có đối tượng

'Command' để thực thi một câu lệnh SQL hay gọi một thủ tục trong database.

Khai báo thư viện ADO để sử dụng các đối tượng trên: vào menu Project → References → check vào ô 'Microsoft Active Data Objects 2.7 Library' (có nhiều version, chọn version mới nhất). Nếu thấy thư viện này đã được check trước đó (với version cũ hơn) bạn có thể uncheck rồi check version mới nhất.




Hình VII.14

## III.1. Đối tượng Connection

### III.1.1. Lệnh khai báo và khởi tạo đối tượng Connection

**Dim** tên-connection **as** ADODB.Connection
**Set** tên-connection = **new** ADODB.Connection

hoặc có thể khai báo chung một dòng:

**Dim** tên-connection **as new** ADODB.Connection

Ví dụ: Dim cnn as new ADODB.Connection

### III.1.2. Chỉ ra chuỗi kết nối CSDL (thuộc tính ConnectionString)

**Tên-biến.ConnectionString** =
"Provider=Microsoft.Jet.OLEDB.4.0; Data Source = E:\Program Files\Microsoft Visual Studio\VB98\BIBLIO.MDB"

Nếu không muốn dùng đường dẫn trực tiếp trong ConnectionString, bạn dùng đường dẫn tương đối qua đối tượng **App** do VB cung cấp để chỉ ra vị trí lưu trữ database. Giả sử chương trình ứng dụng đặt ở thư mục E:\MyVBProg (tạo project ở thư mục E:\MyVBProg) và database BIBLIO.mdb cũng nằm trong thư mục này. Để lấy đường dẫn đến thư mục hiện hành, bạn dùng thuộc tính App.Path (cho kết quả là "E:\MyVBProg")

Chỉ ra *đường dẫn tương đối* đến database như sau:

```
Dim strTemp as string
strTemp = App.Path  'đường dẫn đến thư mục hiện hành
strTemp = strTemp & "\BIBLIO.mdb" 'đường dẫn database
cnn.ConnectionString = Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=" & strTemp
```

**Ghi chú**

■ Khi CSDL là SQL Server thì ConnectionString có dạng sau:

cnn.ConnectionString = "provider=MSDASQL; driver={SQL Server}; server=tên-server; uid=sa; pwq=; database=pubs"

- Hoặc bạn có thể kết nối qua ODBC bằng cách tạo một DSN trong ODBC đặt tên là PUB, chuỗi kết nối:

cnn.ConnectionString = "DSN=PUB; uid=sa; pwq="

■ Khi CSDL là Oracle thì ConnectionString có dạng sau:

cnn.ConnectionString="Provider = MSDAORA; Password= chuyenvien; User ID=chuyenvien; Data Source=CITD;"
(giả sử bạn đang làm việc trong Oracle với user "chuyenvien", password là "chuyenvien", SID là 'CITD')

### III.1.3. Mở kết nối

Dùng phương thức Open: **Tên-connection.Open**
Để kiểm tra việc kết nối có thành công hay không → dùng thuộc tính State. Nếu kết nối thành công thì cnn.State = adStateOpen (hay = 1), ngược lại cnn.State = adStateClose (hay =0).

Ví dụ

```
Dim cnn as new ADODB.Connection
strTemp=App.Path & "\BIBLIO.mdb" 'đường dẫn database
cnn.ConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" &
strTemp
cnn.Open
If (cnn.State = 1) then
        Msgbox "Success!"
Else
        Msgbox "Connecting database fail."
End If
```

### III.1.4. Đóng kết nối

Dùng phương thức Close: **Tên-connection.Close**
Ví dụ: cnn.Close

## III.2. Đối tượng Recordset

Recordset được gọi là tập hợp các mẩu tin.

### III.2.1. Khai báo và khởi tạo đối tượng Recordset

**Dim** tên-recordset **as** ADODB.Recordset

**Set** tên- recordset = **new** ADODB.Recordset

hoặc có thể khai báo chung một dòng:

**Dim** tên- recordset **as new** ADODB.Recordset

Ví dụ: Dim rsTitle as new ADODB.Recordset

### III.2.2. Dùng phương thức Open chọn nguồn dữ liệu

• **Cách 1**

**Tên-recordset.Open Source, ActiveConnection, CursorType, LockType, Options**

Trong đó:

**Source**: là một câu lệnh SQL hoặc tên một table (đặt trong dấu ngoặc kép)

**ActiveConnetion**: Tên-connection kết nối CSDL.

**CursorType**: chỉ định loại con trỏ sử dụng trong recordset, chọn một giá trị trong các giá trị sau.

| Trị hằng | Mô tả |
|---|---|
| AdOpenForwardOnly | Mặc định, chỉ cho phép con trỏ mẩu tin di chuyển tới các mẫu tin tiếp sau (MoveNext), không có MovePrevious. |
| AdOpenDynamic | Chương trình của bạn sẽ thấy những thay đổi thêm, xóa, sửa mà người dùng khác thực hiện trên tập mẫu tin này. |
| AdOpenKeySet | Giống như con trỏ Dynamic nhưng bạn sẽ không thấy những mẩu tin được thêm vào bởi người dùng khác, không truy cập được những mẫu tin đã bị xóa bởi người dùng khác nhưng dữ liệu thay đổi bởi người dùng khác vẫn được nhìn thấy. |
| AdOpenStatic | Cung cấp thể hiện tĩnh của CSDL, bạn không thể thấy những thay đổi mà người dùng khác thực hiện. |

**LockType**: kiểu khóa mẩu tin. Quan tâm đến thuộc tính này khi bạn sử dụng một CSDL có nhiều người dùng, việc khóa mẩu tin sẽ ngăn ngừa những người dùng khác cập nhật trên cùng một record ở cùng một thời điểm bạn đang thao tác.

| Trị hằng | Mô tả |
|---|---|
| AdLockReadOnly | Dữ liệu trong recordset chỉ cho phép đọc. |
| AdLockPessimistic | Cung cấp cơ chế khóa cứng nhắc, nghĩa là mẩu tin sẽ được khóa trong khi bạn cập nhật dữ liệu lên nó. |
| AdLockOptimisitc | Cung cấp cơ chế khóa tốt hơn, nghĩa là mẩu tin sẽ được khóa chỉ khi nào bạn gọi phương thức Update. |
| adLockBatchOptimisitc | Cập nhật nhiều mẩu tin tại một thời điểm, tương ứng với phương thức UpdateBatch |

**Options**: tùy chọn (tương ứng với thuộc tính Source)

| Trị hằng | Mô tả |
|---|---|
| AdCmdTable | Nếu Source là tên một table |
| AdCmdText | Nếu Source là câu lệnh SQL |
| AdCmdStoreProc | Nếu Source là tên của một Store Procedure trong SQL. |

Ví dụ 1: rsTitles.Open "Select * from TITLES", cnn, adOpenStatic, adLockReadOnly
Ví dụ 2: rsTitles.Open "TITLES", cnn, AdOpenKeySet, adLockOptimistic, adCmdTables

• **Cách 2**: có thể gán trực tiếp các giá trị trên cho các thuộc tính của recordset.
**Tên-recordset.ActiveConnection = <Tên-connection>**
**Tên-recordset.Source = <Tên bảng hay câu SQL>**
**Tên-recordset.CursorType = <CursorType>**
**Tên-recordset.LockType = <LockType>**

Cuối cùng gọi phương thức Open : **Tên-recordset.Open**
Ví dụ:

```
rsTitles.Source = " Select * from TITLES"
rsTitles.ActiveConnection = cnn
rsTitles.CursorType = adOpenStatic
rsTitles.LockType = adLockReadOnly
rsTitles.Open
```

### III.2.3. Lấy giá trị của field

Truy xuất dữ liệu trong Recordset → chọn field cần lấy giá trị.

• Cách 1: Tên-recordset.Fields("tên-field")

Ví dụ: lấy thông tin năm xuất bản của mẩu tin hiện hành: rstTitles.Fields("Year Published")

• Cách 2: Tên-recordset.Fields(index)

Ví dụ: giả sử field "Year Published" là field thứ 4 trong table Title thì index sẽ là 3 (index bắt đầu từ 0), lấy thông tin năm xuất bản của mẩu tin hiện hành: rstTitles.Fields(3)

• Cách 3: Tên-recordset!tên-field

Ví dụ: lấy thông tin tên sách, năm xuất bản của mẩu tin hiện hành, cú pháp: rstTitles!Title, rstTitles![Year Published]

**Lưu ý**: *khi tên field có khoảng trắng ở giữa, bạn phải dùng cặp ngoặc [].*

### III.2.4. Các phương thức di chuyển

Vị trí các mẩu tin:

| BOF |
|---|
| mẩu tin đầu tiên |
| mẩu tin thứ 2 |
| ..... |
| mẩu tin cuối cùng |
| EOF |

MoveFirst: di chuyển con trỏ mẩu tin (record pointer) đến mẩu tin đầu tiên. Cú pháp: tên-recordset.MoveFirst

MoveLast: di chuyển con trỏ mẩu tin đến mẩu tin cuối cùng. Cú pháp: tên-recordset.MoveLast

MovePrevious: di chuyển con trỏ mẩu tin từ mẩu tin hiện hành đến mẩu tin trước. Sau khi di chuyển, mẩu tin này được xem là mẩu tin hiện hành. Cú pháp: tên-recordset.MovePrevious.

**Lưu ý:** *khi đang ở mẩu tin đầu tiên, bạn MovePrevious thì cờ hiệu (BOF) bật thành True.*

MoveNext: di chuyển con trỏ mẩu tin từ mẩu tin hiện hành đến mẩu tin kế tiếp. Sau khi di chuyển, mẩu tin này được xem là mẩu tin hiện hành. Cú pháp: tên-recordset.MoveNext.

**Lưu ý:** *khi đang ở mẩu tin cuối cùng, bạn MoveNext thì cờ hiệu (EOF) bật thành True.*

Move n: từ mẩu tin hiện hành di chuyển đến n mẩu tin kế tiếp.
Cú pháp: Tên-recordset.Move n
Ví dụ: rsTitles.Move 3

Nếu recordset không có mẩu tin nào thì cờ hiệu BOF và EOF được bật thành True.

### III.2.5. Các thao tác trên mẩu tin

a) Thêm một mẩu tin (phương thức AddNew)

Để thêm mới một mẩu tin bạn sử dụng phương thức AddNew của đối tượng recordset.

Cú pháp:

**Tên-recordset.AddNew**
Tên-recordset.Fields("tên-field_1")=<giá-trị-thêm>

```
Tên-recordset.Fields("tên-field_2")=<giá-trị-thêm>
Tên-recordset.Fields("tên-field_3")=<giá-trị-thêm>
.....
Tên-recordset.Update
```

b) Sửa một mẩu tin (phương thức Update)

Để sửa một số thuộc tính của mẩu tin hiện hành (cập nhật mẩu tin hành) bạn sử dụng phương thức Update của đối tượng recordset.

Cú pháp:

```
Tên-recordset.Fields("tên-field_1")=<giá-trị-sửa>
Tên-recordset.Fields("tên-field_2")=<giá-trị-sửa>
Tên-recordset.Fields("tên-field_3")=<giá-trị-sửa>
.....
Tên-recordset.Update
```

Nếu bạn không muốn lưu việc cập nhật này, bạn dùng lệnh sau:

```
Tên-recordset.CancelUpdate
```

**Lưu ý:** *Trong kỹ thuật DAO bạn gọi phương thức Tên-recordset.Edit trước khi thay đổi giá trị của một field nhưng trong kỹ thuật ADO việc thay đổi giá trị của field tự động chuyển sang mode Edit do đó không có phương thức Tên-recordset.Edit. Tuy nhiên ADO cung cấp cho bạn thuộc tính EditMode để biết được tình trạng mẩu tin hiện hành.*

| Trị hằng | Mô tả |
|---|---|
| AdEditNone | Tình trạng bình thường |
| AdEditAdd | Đang thêm mới mẩu tin, chưa cập nhật. |
| adEditInProgress | Đang sửa đổi mẩu tin, chưa cập nhật |
| AdEditDelete | Đang được đánh dấu xóa, chưa xóa hẳn. |

c) Xóa một mẩu tin (phương thức Delete)

Để xóa mẩu tin hiện hành, sử dụng phương thức Delete của đối tượng recordset.

Cú pháp:

Tên-recordset.Delete

d) Tìm kiếm mẩu tin (phương thức Find)

Cú pháp:

Tên-recordset.Find Criteria, SkipRecord, SearchDirection,Start

Các thông số

**Criteria**: chuỗi điều kiện tìm. Chuỗi này bao gồm cột tìm kiếm (field), toán tử so sánh (>, <, =, <>, ≥, ≤, like), và giá trị tìm (nếu là chuỗi ký tự bạn đặt trong cặp dấu nháy đơn, nếu giá trị là ngày bạn đặt trong dấu #). Ví dụ: "Title = 'dBASE III Plus'"

**SkipRecord**: đây là giá trị tùy chọn, mặc định là 0, việc tìm kiếm sẽ bắt đầu từ mẩu tin hiện hành. Nếu bạn chỉ định giá trị n (kiểu long) thì việc tìm kiếm sẽ bỏ qua n mẩu tin (tính từ mẩu tin hiện hành) và bắt đầu tìm kiếm ở mẩu tin tiếp theo.

**SearchDirection**: giá trị tùy chọn, cho biết hướng tìm kiếm. Có hai giá trị: **adSearchForward** (tìm từ mẩu tin hiện hành hướng tới mẩu tin cuối, nếu không tìm thấy con trỏ mẩu tin sẽ nằm ở vị trí EOF), **adSearchBackward** (tìm từ mẩu tin hiện hành lùi về mẩu tin đầu tiên, nếu không tìm thấy con trỏ mẩu tin sẽ nằm ở vị trí BOF). Trị mặc định: adSearchForward.

**Start**: giá trị tùy chọn, cho biết vị trí bắt đầu tìm. Mặc định vị trí bắt đầu tìm là mẩu tin hiện hành. Nếu bạn muốn vị trí bắt đầu tìm là mẩu tin đầu tiên bạn dùng trị hằng **adBookmarkFirst** (lúc này bạn phải search theo hướng adSearchForward), ngược lại vị trí bắt đầu tìm là mẩu tin cuối cùng thì dùng trị hằng adBookmarkLast (bạn phải search theo hướng adSearchBackward)

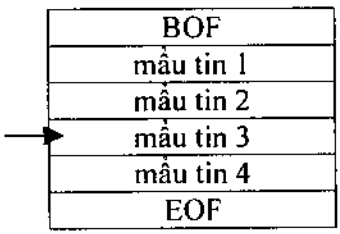
Ví dụ: rsTitles.Find "Title = 'dBASE III Plus'", , , adbookmarkFirst

### III.2.6. Thuộc tính Bookmark, AbsolutePosition của đối tượng Recordset

Thuộc tính Bookmark (**Tên-recordset.Bookmark**) cho biết vị trí được xác định là duy nhất của mẩu tin hiện hành trong Recorsdet, Bookmart được hệ thống nhận diện duy nhất cho mỗi mẩu tin (khi thêm, xóa, sửa mẩu tin trên recordset thì vị trí bookmart của các mẩu tin khác vẫn không thay đổi).

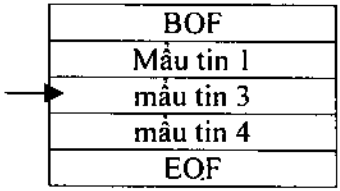
Thuộc tính AbsolutePosition (**Tên-recordset.AbsolutePosition**) lưu trữ thông tin vị trí của mẩu tin hiện hành trong recordset (so sánh với các mẩu tin khác **hiện có** trong recordset, khi xóa mẩu tin trong recordset thì vị trí của mẩu tin hiện hành có thể thay đổi).

Ví dụ:



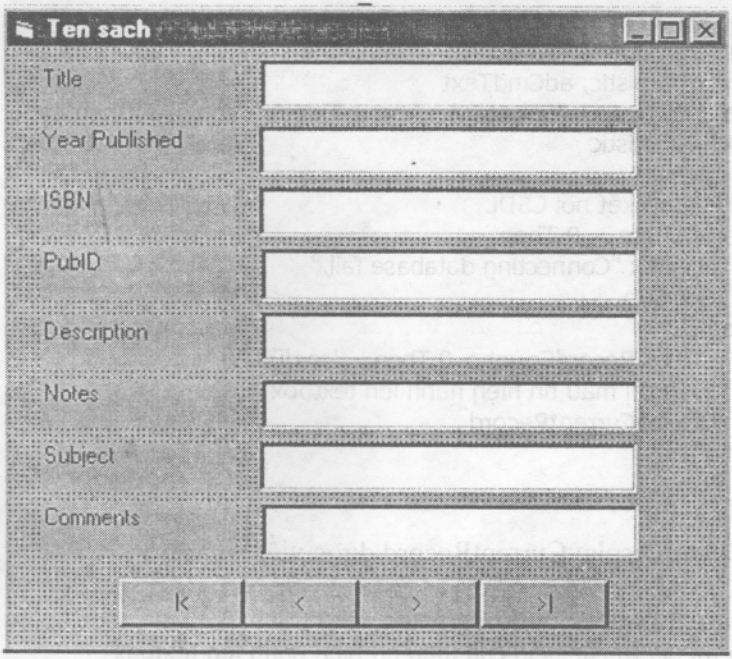

Xem mẩu tin 3 là mẩu tin hiện hành trong recordset, giả sử Bookmart của mẩu tin hiện hành là 514 (recorset.Bookmart=514, hệ thống tự gán), khi bạn di chuyển về mẩu tin 1 chắc chắn Bookmart sẽ là một giá trị khác (ví dụ 512), tương tự mẩu tin 2 (ví dụ 513). Khi xóa mẩu tin 2 Bookmart của mẩu tin 3 vẫn là 514.

Vị trí AbsolutePosition của mẩu tin 1 là 1, của mẩu tin 2 là 2, của mẩu tin 3 là 3, của mẩu tin 4 là 4. Khi xóa mẩu tin 2 (lúc này recordset chỉ gồm 3 mẩu tin) ta có AbsolutePosition của mẩu tin 1 là 1, AbsolutePosition của mẩu tin 3 là 2, AbsolutePosition của mẩu tin 4 là 3.




### III.3. Tạo một form ứng dụng

Bạn tạo mới một form, trong cửa sổ properties đặt thuộc tính name của form là **'frmTitles_1'**, caption là **'Ten sach'**, StartUpPosition là **'Center Screen'** (để form này xuất hiện giữa màn hình khi chạy chương trình). Dùng 8 textbox: txtTitle, txtYearPublished, txtISBN, txtPubID, txtDescription, txtNotes, txtSubjects, txtComments. Tương ứng bạn cho 8 label đặt trước 8 textbox với tên: lblTitle, lblYearPublished, lblISBN, lblPubID, lblDescription, lblNotes, lblSubjects, lblComments (***Lưu ý: bạn không set thuộc tính DataSource và DataField cho từng textbox***). Để giả lập các nút di chuyển đến mẩu tin đầu, mẩu tin trước, mẩu tin kế và mẩu tin cuối bạn dùng bốn nút: cmdFirst (có caption '|<'), cmdNext (có caption '<'), cmdPrevious (có caption '<'), cmdLast (có caption '>|'). Xem hình VII.15




**Hình VII.15**

Chuyển sang viết code (View Code), khai báo General như sau:

```
Dim cnn As New ADODB.Connection
Dim rsTitles As New ADODB.Recordset
```

Đoạn code sự kiện Sub Form_Load() dưới đây:

```
Private Sub Form_Load()
   Dim DatabasePath As String
   DatabasePath = App.Path & "\BIBLIO.mdb"
   cnn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=" & DatabasePath
   cnn.Open
   rsTitles.Open "TITLES", cnn, adOpenStatic, adLockOptimistic,
adCmdTable
```

```
    'hoặc bạn dùng cách khác
    'rsTitles.Open "Select * from Titles", cnn, adOpenStatic,
adLockOptimistic, adCmdText
    'rsTitles.Open "Select * from Titles", cnn, adOpenStatic,
adLockOptimistic

    ' kiểm tra kết nối CSDL
    If cnn.State = 0 Then
        MsgBox "Connecting database fail."
        Exit Sub
    End If
    If rsTitles.RecordCount > 0 Then    'có dữ liệu
        'hiển thị mẩu tin hiện hành lên textbox
        DisplayCurrentRecord
    End If
End Sub
```

Với hàm DisplayCurrentRecord được viết như sau:

```
Private Sub DisplayCurrentRecord()
'hien thi gia tri cac field cua mau tin hien hanh lên textbox
'kiem tra gia tri cua field co Null hay khong? Neu khac Null → gan
textbox.Text = gia tri cua field nay

    If IsNull(rsTitles.Fields("Title")) = True Then
        txtTitle.Text = ""
    Else
        txtTitle.Text = rsTitles.Fields("Title")
    End If
    If IsNull(rsTitles.Fields("Year Published")) = True Then
        txtYearPublished.Text = ""
    Else
        txtYearPublished.Text = rsTitles.Fields("Year Published")
    End If
    txtISBN.Text = rsTitles.Fields("isbn") ' vì ISBN là khóa chính→
không được null

    If IsNull(rsTitles.Fields("PubID")) = True Then
        txtPubID.Text = ""
```

```
    Else
        txtPubID.Text = rsTitles.Fields("PubID")
    End If
    If IsNull(rsTitles.Fields("Description")) = True Then
        txtDescription.Text = ""
    Else
        txtDescription.Text = rsTitles.Fields("Description")
    End If
    If IsNull(rsTitles.Fields("Notes")) = True Then
        txtNotes.Text = ""
    Else
        txtNotes.Text = rsTitles.Fields("Notes")
    End If
    If IsNull(rsTitles.Fields("Subject")) = True Then
        txtSubject.Text = ""
    Else
        txtSubject.Text = rsTitles.Fields("Subject")
    End If
    If IsNull(rsTitles.Fields("Comments")) = True Then
        txtComments.Text = ""
    Else
        txtComments.Text = rsTitles.Fields("Comments")
    End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
    rsTitles.Close
    cnn.Close
End Sub
```

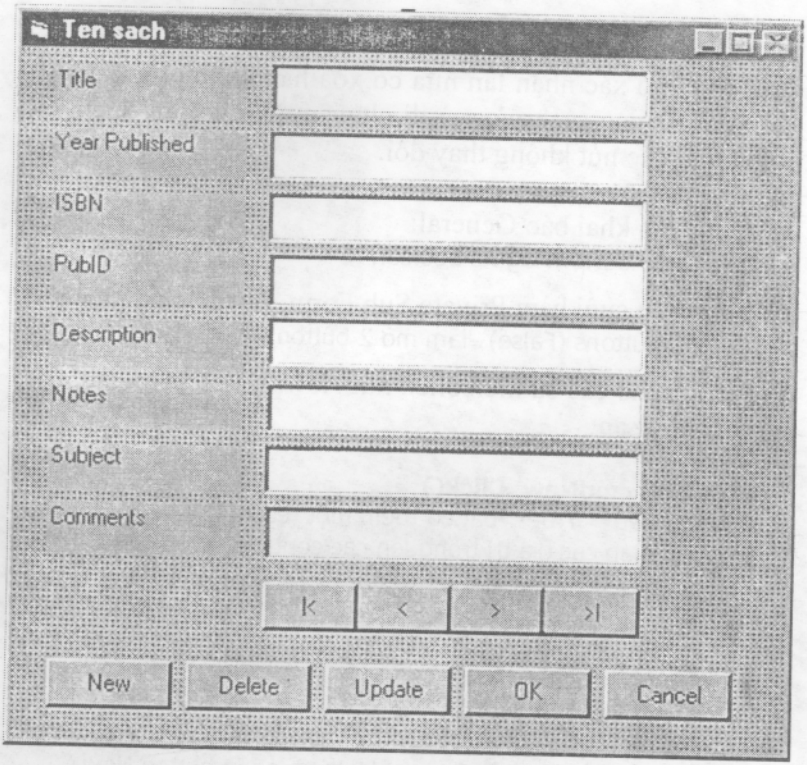
Đến đây bạn chạy chương trình chỉ thấy được một mẩu tin hiện hành. Muốn di chuyển đến mẩu tin trước nó (  ), mẩu tin sau nó (  ),...**bạn phải viết code để thực hiện thao tác di chuyển này** (sử dụng các method **MovePrevious, MoveNext, MoveFirst, Move Last** của đối tượng recordset).

Đoạn code cho sự kiện click button CmdMoveFirst, CmdMovePrevious, CmdMoveNext, CmdMoveLast như sau:

```
Private Sub cmdFirst_Click()
'kiểm tra recordset có mẩu tin =>di chuyển đến mẩu tin đầu tiên
   If (rsTitles.BOF = False And rsTitles.EOF = False) Then
      rsTitles.MoveFirst
      DisplayCurrentRecord  'hiển thị thông tin record
   End If
End Sub

Private Sub cmdLast_Click()
'kiểm tra recordset có mẩu tin =>di chuyển đến mẩu tin cuối cùng
   If (rsTitles.BOF = False And rsTitles.EOF = False) Then
      rsTitles.MoveLast
      DisplayCurrentRecord  'hien thi thong tin record
   End If
End Sub

Private Sub cmdNext_Click()
'kiểm tra recordset có mẩu tin =>di chuyển đến mẩu tin kế
   If (rsTitles.BOF = False And rsTitles.EOF = False) Then
      rsTitles.MoveNext
      If rsTitles.EOF = True Then 'nếu EOF => di chuyển đến mẩu
tin cuối cùng
         rsTitles.MoveLast
      End If
      DisplayCurrentRecord  'hiển thị thông tin record
   End If
End Sub

Private Sub cmdPrevious_Click()
'kiểm tra recordset có mẩu tin =>di chuyển đến mẩu tin trước
   If (rsTitles.BOF = False And rsTitles.EOF = False) Then
      rsTitles.MovePrevious
      If rsTitles.BOF = True Then  'neu BOF => di chuyển đến mẩu
tin đầu tiên
         rsTitles.MoveFirst
      End If
      DisplayCurrentRecord  'hiển thị thông tin record
   End If
End Sub
```

Đến đây chạy chương trình và xem kết quả khi nhấn các nút di chuyển.

Để thực hiện thêm, xóa, sửa thông tin đề mục sách trong Database cần phải thiết kế bổ sung vào form một số button có properties Name lần lượt là: cmdNew (caption: New), cmdDelete (caption: Delete), "cmdUpdate" (caption: Update), cmdOK (caption: OK), cmdCancel (caption: Cancel) như hình VII.16




**Hình VII.16**

Mô tả:

Dùng một cờ hiệu AddNewFlag cho biết người dùng chọn chức năng thêm mới (=True) hay cập nhật (=False).

*Thêm mới một đề mục sách*: người dùng nhấn nút New, cờ hiệu AggNewFlag bật thành True, làm trống 8 textbox Title, Year Published,...., làm hiện lên các nút OK (đồng ý), Cancel (bỏ qua), làm mờ các nút Delete, Update.

*Cập nhật đề mục sách hiện hành (mẫu tin hiện hành)*: người dùng nhấn nút Update, cờ hiệu AggNewFlag bật thành False, làm hiện lên các nút OK, Cancel, làm mờ các nút New, Delete.

*Xóa đề mục sách hiện hành*: người dùng nhấn nút Delete, đưa ra thông báo xác nhận lần nữa có xóa hay không (nếu đồng ý thì xóa đề mục sách này và di chuyển đến đề mục sách kế), trạng thái các nút không thay đổi.

Bổ sung vào khai báo General:

```
Dim AddNewFlag As Boolean
```

Bổ sung vào cuối hàm Private Sub Form_Load() dòng lệnh

```
SetButtons (False)  'làm mờ 2 button OK và Cancel
```

Viết code cho các sự kiện click nút New, Delete, Update, OK, Cancel như sau:

```
Private Sub cmdNew_Click()
   AddNewFlag = True   'bật cờ thêm một record
   ClearAll  'gán các giá trị trống lên các control.
   SetButtons (True)      'cho 2 button OK và Cancel hiện lên
End Sub

Private Sub cmdOK_Click()
'kiểm tra các Ràng buộc toàn vẹn ở đây, nếu không thỏa => Exit
Sub
  If AddNewFlag = True Then ' người dùng nhấn button New
     rsTitles.AddNew
     rsTitles.Fields("ISBN") = txtISBN.Text 'không được null
     rsTitles.Fields("Title") = txtTitle.Text 'không được null
     rsTitles.Fields("Year Published") = Cint(txtYearPublished.Text)
```

```
        rsTitles.Fields("PubID") = CIng(txtPubID.Text) 'không được
null
        If txtDescription.Text <> "" Then rsTitles.Fields("Description")
= txtDescription.Text
        If txtNotes.Text <> "" Then rsTitles.Fields("Notes") =
txtNotes.Text
        If txtSubject.Text <> "" Then rsTitles.Fields("Subject") =
txtSubject.Text
        If txtComments.Text <> "" Then rsTitles.Fields("Comments")
= txtComments.Text
        rsTitles.Update
    Else 'TH nhan nút Update, không cập nhật khóa chính ISBN
    If (rsTitles.BOF = False And rsTitles.EOF = False) Then
'recordset có mẫu tin
        If rsTitles.Fields("Title").OriginalValue <> txtTitle.Text Then
'nếu khác giá trị ban đầu của filed=>cập nhật field
            rsTitles.Fields("Title") = txtTitle.Text
        End If
        If rsTitles.Fields("Year Published").OriginalValue <>
CInt(txtYearPublished.Text) Then
            'nếu khác giá trị ban đầu của filed=> cập nhật field
            rsTitles.Fields("Year Published") =
CInt(txtYearPublished.Text)
        End If
        If rsTitles.Fields("PubID").OriginalValue <>
CLng(txtPubID.Text) Then
            rsTitles.Fields("PubID") = CLng(txtPubID.Text)
        End If
        If rsTitles.Fields("Description").OriginalValue <>
txtDescription.Text Then
            rsTitles.Fields("Description") = txtDescription.Text
        End If
        If rsTitles.Fields("Notes").OriginalValue <> txtNotes.Text
Then
            rsTitles.Fields("Notes") = txtNotes.Text
        End If
        If rsTitles.Fields("Subject").OriginalValue <> txtSubject.Text
Then
            rsTitles.Fields("Subject") = txtSubject.Text
        End If
```

```
        If          rsTitles.Fields("Comments").OriginalValue          <>
txtComments.Text Then
            rsTitles.Fields("Comments") = txtComments.Text
        End If
        rsTitles.Update
    End If    SetButtons (False)  'làm mờ hai nút OK và Cancel
End Sub

Private Sub cmdCancel_Click()
    SetButtons (False)  ' làm mờ hai nút OK và Cancel
    DisplayCurrentRecord  'hiển thị mẫu tin hiện hành
End Sub

Private Sub cmdDelete_Click()
    On Error GoTo DeleteErr
    If (rsTitles.BOF = False And rsTitles.EOF = False) Then
'recordset có mẫu tin cần xóa
        returnvalue = MsgBox("Do you want to delete this record?",
vbOKCancel + vbQuestion, "Notice")
        If  returnvalue = vbOK Then
          rsTitles.Delete 'xóa mẫu tin hiện hành
          rsTitles.MoveNext ' di chuyển đến mẫu tin kế
          If rsTitles.EOF = True Then rsTitles.MoveLast 'kiểm tra EOF
          DisplayCurrentRecord
        End If
        Exit Sub
    End If
DeleteErr:
    MsgBox Err.Description
End Sub

Private Sub cmdUpdate_Click()
    AddNewFlag = False 'trạng thái cập nhật mẫu tin
    SetButtons (True)
End Sub

Private Sub SetButtons(bVal As Boolean)
    cmdOK.Enabled = bVal
    cmdCancel.Enabled = bVal
    cmdNew.Enabled = Not bVal
```

```
    cmdUpdate.Enabled = Not bVal
    cmdDelete.Enabled = Not bVal
End Sub

Private Sub ClearAll()
    txtTitle.Text = ""
    txtYearPublished.Text = ""
    txtISBN.Text = ""
    txtPubID.Text = ""
    txtDescription.Text = ""
    txtNotes.Text = ""
    txtSubject.Text = ""
    txtComments.Text = ""
End Sub
```

Chạy chương trình và thực hiện thao tác thêm, xóa, sửa.

**Lưu ý:** *khi thêm mới một mẩu tin bạn thêm giá trị PubID phải là giá trị có sẵn trong Database vì field PubID trong table Titles là khóa ngoại.*

Chương VIII

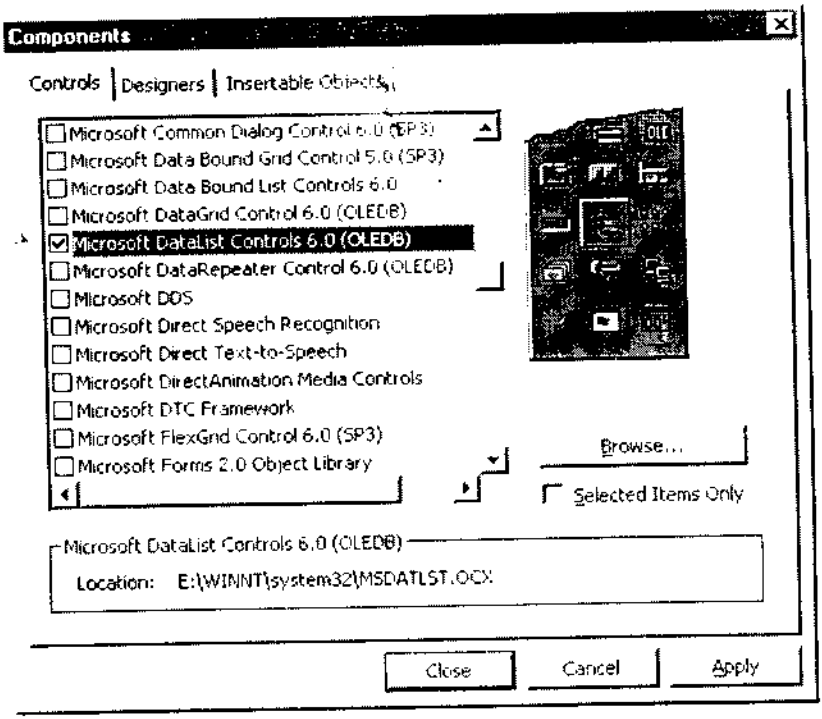
# MỘT SỐ ĐIỀU KHIỂN KẾT BUỘC DỮ LIỆU

Các Data-Bound controls (điều khiển kết buộc dữ liệu) được dùng để hiển thị dữ liệu từ ADO Data Control (điều khiển dữ liệu) hoặc từ đối tượng Recordset (tập các mẩu tin) như: TextBox, ComboBox,...bên cạnh đó ADO hỗ trợ thêm một số điều khiển kết buộc dữ liệu khác như: DataCombo, DataList, DataGrid.

## I. DATACOMBO VÀ DATALIST

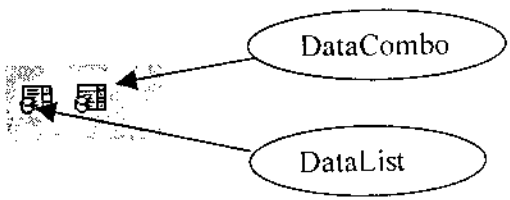
### I.1. Giới thiệu DataCombo và DataList

DataCombo và DataList không có sẵn trên Toolbox như một số control thông dụng khác (textbox, combobox, label,...) do đó bạn phải đưa vào: trên menu của VB chọn Project → Components (hoặc click chuột phải trên Toolbox chọn Components), trong tab Control bạn check vào ô "Microsoft DataList Control 6.0 (OLEDB)" rồi nhấn nút Apply:




**Hình VIII.1**

Lúc này trên Toolbox xuất hiện biểu tượng như hình dưới đây:




Sau đây minh họa cách sử dụng DataCombo, thao tác tương tự cho DataList. Bảng liệt kê một số thuộc tính tiêu biểu của DataCombo control.

| Thuộc tính | Mô tả |
|---|---|
| DataSource | Liên kết với nguồn dữ liệu thứ nhất dùng cho việc cập nhật. Nguồn dữ liệu có thể là một ADO Data control thứ nhất hoặc một recordset thứ nhất. |
| DataField | Filed cập nhật dữ liệu trong nguồn dữ liệu trên |
| RowSource | Liên kết với nguồn dữ liệu thứ hai để xem sự liên hệ giữa nguồn dữ liệu thứ nhất. Nguồn dữ liệu thứ hai có thể là một ADO Data control thứ hai hoặc một recordset thứ hai. |
| ListField | Giá trị Field này sẽ được hiển thị lên Combo, có thể trùng với DataField hoặc khác với DataField |
| BoundColumn | Nếu ListField trùng DataField thì mặc định khi thay đổi giá trị trên combo sẽ là thay đổi giá trị trong field của nguồn dữ liệu . Nếu nhất khác nhau bạn phải chỉ ra field nào trong nguồn dữ liệu hai được cập nhật cho field trong nguồn dữ liệu nhất khi người dùng thay đổi giá trị trong combo. |

Ngoài ra DataCombo cung cấp thuộc tính SelectedItem cho biết vị trí Bookmark của Item được chọn (quyết định bởi DataField của mẩu tin hiện hành trong ADO Data Control thứ 1 (hoặc recordset 1) trong số các Item (ListField) của ADO Data Control thứ 2 (hoặc recordset thứ 2). Để gán Bookmart này thành **Bookmart hiện hành** của ADO Data Control 2 (hoặc recordset 2) bạn phải gán Recordset2.Bookmart = DataCombo.SelectedItem (nghĩa là Bookmart của mẩu tin hiện hành trong recordset 2 trùng với Bookmart của Item được chọn).

Ví dụ:

Thêm mới **một form**, trong cửa sổ Properties đặt thuộc tính **Name** của form là '**frmTitles_2**', caption là '**Ten sach**', **StartUpPosition** là '**Center Screen**'. Kế tiếp bạn đặt control

DataCombo vào form, đặt tên "dcPubID", thuộc tính Text rỗng, tiếp tục đặt hai ADO Data Control lên form với tên "Adodc1", "Adodc2". Kết nối CSDL cho Adodc1 bạn cần chỉ ra chuỗi ConnectString, RecordSource là table Titles, Adodc2 có RecordSource là table Publishers.

Đoạn code sự kiện Form_Load (có field cập nhật dữ liệu là mã nhà xuất bản "PubID", field hiển thị trên DataCombo là mã nhà xuất bản, và ràng buộc cập nhật dữ liệu (BoundColumn) là mã nhà xuất bản mỗi khi người dùng thay đổi mã nhà xuất bản trên DataCombo).

```
Private Sub Form_Load()
   Set dcPubID.DataSource = Adodc1
   dcPubID.DataField = "PubID"
   Set dcPubID.RowSource = Adodc2
   dcPubID.ListField = "PubID"
   dcPubID.BoundColumn = "PubID"
End Sub
```

Chạy chương trình, di chuyển trước, sau điều khiển Adodc1 thì giá trị được chọn trên datacombo có thể thay đổi nhưng khi di chuyển trước, sau điều khiển Adodc2 thì giá trị được chọn trên datacombo không thay đổi theo.

Áp dụng tính chất SelectedItem của DataCombo bạn sẽ thiết kế bổ sung một textbox vào form để hiển thị tên nhà xuất bản (Name) ứng với mã nhà xuất bản (PubID) thay đổi trong datacombo mỗi khi người dùng di chuyển tới lui điều khiển Adodc1. Sự kiện này gọi là DataCombo_change. Đặt textbox tên txtName, thuộc tính text là rỗng. Bổ sung đoạn code:

```
Private Sub dcPubID_Change()
   If IsNull(Adodc1.Recordset.Fields("PubID")) = True Then
      txtName.Text = ""
   Else
```

```
    If dcPubID.SelectedItem > 0 Then
      Adodc2.Recordset.Bookmark = dcPubID.SelectedItem
      txtName.Text = Adodc2.Recordset.Fields("Name")
    End If
  End If
End Sub
```

Chạy chương trình, di chuyển tới lui điều khiển Adodc1 và xem kết quả.

Nếu không dùng các điều khiển dữ liệu (ADO Data Control) bạn vẫn có thể dùng hai đối tượng recordset (ví dụ rsTitles, rsPublishers) để liên kết với DataCombo này. Trở lại màn hình thiết kế form, bạn **delete** hai ADO Data Control "Adodc1" và "Adodc2", thêm bốn nút cmdFirst, cmdPrevious, cmdNext, cmdLast sau đó cập nhật code như sau:

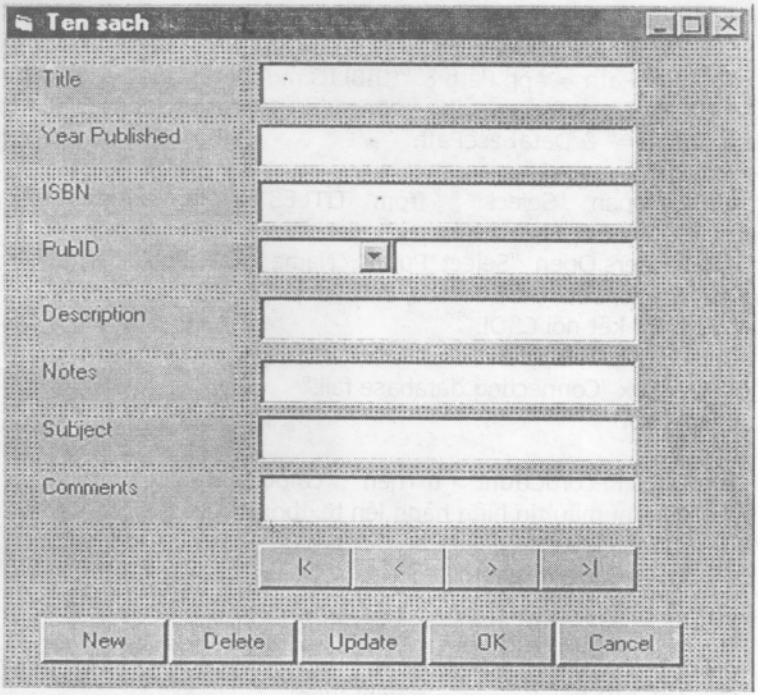
```
Dim cnn As New ADODB.Connection
Dim rsTitles As New ADODB.Recordset
Dim rsPublishers As New ADODB.Recordset

Private Sub Form_Load()
  Dim DatabasePath As String
  DatabasePath = App.Path & "\BIBLIO.mdb"
  cnn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=" & DatabasePath
  cnn.Open
  rsTitles.Open "Select * from TITLES", cnn, adOpenStatic,
adLockOptimistic
  rsPublishers.Open "Select PubID, Name from Publishers", cnn,
adOpenStatic, adLockOptimistic, cmdText
  ' kiểm tra kết nối CSDL
  If cnn.State = 0 Then
    MsgBox "Connecting database fail."
    Exit Sub
  End If
  Set dcPubID.DataSource = rsTitles
```

```
    dcPubID.DataField = "PubID"
    Set dcPubID.RowSource = rsPublishers
    dcPubID.ListField = "PubID"
End Sub

Private Sub dcPubID_Change()
    If IsNull(rsTitles.Fields("PubID")) = True Then
        txtName.Text = ""
    Else
      If dcPubID.SelectedItem > 0 Then
        rsPublishers.Bookmark = dcPubID.SelectedItem
        txtName.Text = rsPublishers.Fields("Name")
      End If
    End If
End Sub

Private Sub cmdFirst_Click()
'kiểm tra có m ẩu tin => di chuyển đến mẩu tin đầu tiên
    If (rsTitles.BOF = False And rsTitles.EOF = False) Then
        rsTitles.MoveFirst
    End If
End Sub

Private Sub cmdLast_Click()
'kiểm tra có mẩu tin => di chuyển đến mẩu tin cuối cùng
    If (rsTitles.BOF = False And rsTitles.EOF = False) Then
        rsTitles.MoveLast
    End If
End Sub

Private Sub cmdNext_Click()
'kiểm tra có mẩu tin => di chuyển đến mẩu tin kế
    If (rsTitles.BOF = False And rsTitles.EOF = False) Then
        rsTitles.MoveNext
        If rsTitles.EOF = True Then 'nếu EOF, đến mẩu tin cuối cùng
            rsTitles.MoveLast
        End If
    End If
End Sub
```

```
Private Sub cmdPrevious_Click()
'kiểm tra có mẩu tin => di chuyển đến mẩu tin trước
   If (rsTitles.BOF = False And rsTitles.EOF = False) Then
      rsTitles.MovePrevious
      If rsTitles.BOF = True Then ' nếu BOF, đến mẩu tin đầu tiên
         rsTitles.MoveFirst
      End If
   End If
End Sub
```

## I.2. Chương trình ứng dụng

Thiết kế form mới (name "**Titles_3**") dùng 8 textbox: txtTitle, txtYearPublished, txtISBN, txtName, txtDescription, txtNotes, txtSubjects, txtComments, 1 datacombo (name là dcPubID), 8 label đặt trước các textbox với tên: lblTitle, lblYearPublished, lblISBN, lblPubID, lblDescription, lblNotes, lblSubjects, lblComments, 4 nút cmdFirst, cmdPrevious, cmdNext, cmdLast, 5 nút cmdNew, cmdDelete, cmdUpdate, cmdOK, cmdCancel.




**Hình VIII.2**

Sửa đổi code các hàm **Private Sub Form_Load,Private Sub SetButtons, Private Sub DisplayCurrentRecord, Private Sub ClearAll, Private Sub OK_Click, Private Sub Form_Unload** (có thay đổi) và bổ sung thêm hàm **Private Sub dcPubID_Change**. Code các hàm khác vẫn giữ nguyên như trong form **frmTitles_1** (tuy nhiên vẫn liệt kê ở đây để các bạn tham khảo).

```
Dim cnn As New ADODB.Connection
Dim rsTitles As New ADODB.Recordset
Dim rsPublishers As New ADODB.Recordset
Dim AddNewFlag As Boolean 'thêm một record mới(=True), cập nhật một record(=false)
```

```
Private Sub Form_Load()
   Dim DatabasePath As String
   DatabasePath = App.Path & "\BIBLIO.mdb"
   cnn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=" & DatabasePath
   cnn.Open
   rsTitles.Open "Select * from TITLES", cnn, adOpenStatic,
adLockOptimistic
   rsPublishers.Open "Select PubID, Name from Publishers", cnn,
adOpenStatic, adLockOptimistic, cmdText
   ' kiểm tra kết nối CSDL
   If cnn.State = 0 Then
      MsgBox "Connecting database fail."
      Exit Sub
   End If
   If rsTitles.RecordCount > 0 Then   'có dữ liệu
      'Hiển thị mẩu tin hiện hành lên textbox
      rsTitles.MoveFirst
      DisplayCurrentRecord
   End If
   Set dcPubID.DataSource = rsTitles
   dcPubID.DataField = "PubID"
   Set dcPubID.RowSource = rsPublishers
   dcPubID.BoundColumn = "PubID"
   dcPubID.ListField = "PubID"

   SetButtons (False)
```

'Lần đầu load form Item trong datacombo chưa được **selected** do đó khi vào sự kiện PubID_Change() sẽ không lấy được Name tương ứng với PubID đó, bạn phải dùng tạm một recordset "rsTemp" để lấy Name của PubID trong datacombo (Bạn có thể bỏ đoạn code này để kiểm tra sự khác biệt).

```
   Dim rsTemp As New ADODB.Recordset
   rsTemp.Open "Select PubID,Name from Publishers where
PubID=" & CLng(dcPubID.Text), cnn, adOpenStatic,
adLockReadOnly, cmdText
   If Not rsTemp.EOF Then
      txtName.Text = rsTemp.Fields("Name")
   End If
```

```
    rsTemp.Close
End Sub

Private Sub dcPubID_Change()
    If IsNull(rsTitles.Fields("PubID")) = True Then
        txtName.Text = ""
    Else
      If dcPubID.SelectedItem > 0 Then
        rsPublishers.Bookmark = dcPubID.SelectedItem
        txtName.Text = rsPublishers.Fields("Name")
      End If
    End If
End Sub

Private Sub SetButtons(bVal As Boolean)
    cmdOK.Enabled = bVal
    cmdcancel.Enabled = bVal
    cmdnew.Enabled = Not bVal
    cmdUpdate.Enabled = Not bVal
    cmddelete.Enabled = Not bVal

    '(cho/khong cho) edit DataCombo dcPubID
    dcPubID.Locked = Not bVal
End Sub

Private Sub DisplayCurrentRecord()
'hiển thị giá trị các field của mẩu tin hiện hành lên textbox
'kiểm tra giá trị field có null? Khác null → gán textbox.Text = giá trị
của field
    If IsNull(rsTitles.Fields("Title")) = True Then
        txtTitle.Text = ""
    Else
        txtTitle.Text = rsTitles.Fields("Title")
    End If
    If IsNull(rsTitles.Fields("Year Published")) = True Then
        txtYearPublished.Text = ""
    Else
        txtYearPublished.Text = rsTitles.Fields("Year Published")
    End If
```

```
    txtISBN.Text = rsTitles.Fields("isbn") ' vi ISBN la khoa chinh →
khong duoc null
    If IsNull(rsTitles.Fields("Description")) = True Then
        txtDescription.Text = ""
    Else
        txtDescription.Text = rsTitles.Fields("Description")
    End If
    If IsNull(rsTitles.Fields("Notes")) = True Then
        txtNotes.Text = ""
    Else
        txtNotes.Text = rsTitles.Fields("Notes")
    End If
    If IsNull(rsTitles.Fields("Subject")) = True Then
        txtSubject.Text = ""
    Else
        txtSubject.Text = rsTitles.Fields("Subject")
    End If
    If IsNull(rsTitles.Fields("Comments")) = True Then
        txtComments.Text = ""
    Else
        txtComments.Text = rsTitles.Fields("Comments")
    End If
End Sub

Private Sub cmdFirst_Click()
'kiểm tra có mẩu tin => di chuyển đến mẩu tin đầu tiên
    If (rsTitles.BOF = False And rsTitles.EOF = False) Then
        rsTitles.MoveFirst
        DisplayCurrentRecord
    End If
End Sub

Private Sub cmdLast_Click()
'kiểm tra có mẩu tin => di chuyển đến mẩu tin cuối cùng
    If (rsTitles.BOF = False And rsTitles.EOF = False) Then
        rsTitles.MoveLast
        DisplayCurrentRecord
    End If
End Sub
```

```
Private Sub cmdNext_Click()
'kiểm tra có mẩu tin => di chuyển đến mẩu tin kế
   If (rsTitles.BOF = False And rsTitles.EOF = False) Then
      rsTitles.MoveNext
      If rsTitles.EOF = True Then 'nếu EOF, đến mẩu tin cuối cùng
         rsTitles.MoveLast
      End If
   DisplayCurrentRecord
   End If
End Sub

Private Sub cmdPrevious_Click()
'kiểm tra có mẩu tin => di chuyển đến mẩu tin trước
   If (rsTitles.BOF = False And rsTitles.EOF = False) Then
      rsTitles.MovePrevious
      If rsTitles.BOF = True Then 'nếu BOF, đến mẩu tin đầu tiên
         rsTitles.MoveFirst
      End If
      DisplayCurrentRecord
   End If
End Sub

Private Sub cmdCancel_Click()
   SetButtons (False)
   DisplayCurrentRecord 'hiển thị mẩu tin hiện hành
End Sub

Private Sub cmdDelete_Click()
   On Error GoTo DeleteErr
   If (rsTitles.BOF = False And rsTitles.EOF = False) Then
'recordset có mẩu tin cần xóa
      returnvalue = MsgBox("Do you want to delete this record?",
vbOKCancel + vbQuestion, "Notice")
      If returnvalue = vbOK Then
        rsTitles.Delete 'xóa mẩu tin hiện hành
        rsTitles.MoveNext  'di chuyển đến mẩu mẫu tin kế
        If rsTitles.EOF = True Then rsTitles.MoveLast 'kiểm tra EOF
          DisplayCurrentRecord
      End If
      Exit Sub
```

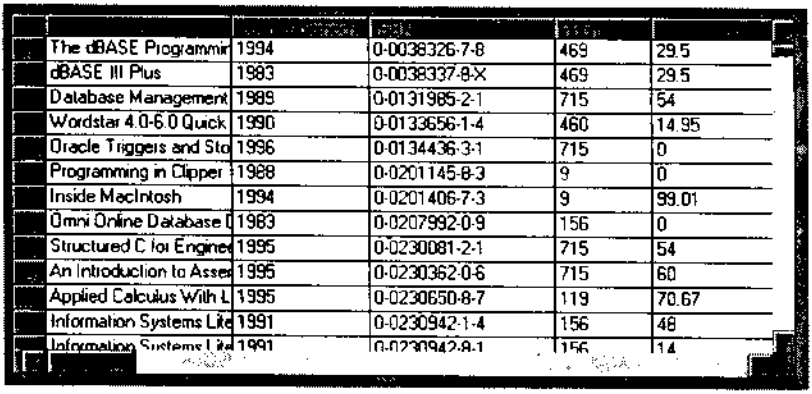
```
    End If
DeleteErr:
    MsgBox Err.Description
End Sub

Private Sub cmdUpdate_Click()
    AddNewFlag = False
    SetButtons (True)
End Sub

Private Sub cmdNew_Click()
    AddNewFlag = True   'bật cờ thêm một record = true
    ClearAll  'gán giá trị trên các control là rỗng.
    SetButtons (True)
End Sub

Private Sub ClearAll()
    txtTitle.Text = ""
    txtYearPublished.Text = ""
    txtISBN.Text = ""
    txtDescription.Text = ""
    txtNotes.Text = ""
    txtSubject.Text = ""
    txtComments.Text = ""
End Sub

Private Sub cmdOK_Click()
'bổ sung các kiểm tra RBTV, nếu không thỏa Exit Sub
    If AddNewFlag = True Then ' người dùng nhấn button New
    'giữ lại giá trị dcPubID người dùng đã chọn vào biến PubID vì:
khi gọi method AddNew giá trị trong datacombo dcPubID sẽ rỗng
        Dim PubID As Long
        PubID = CLng(dcPubID.Text)
        rsTitles.AddNew
        rsTitles.Fields("ISBN") = txtISBN.Text 'không được null
        rsTitles.Fields("Title") = txtTitle.Text 'không được null
        rsTitles.Fields("Year                    Published")                =
CInt(txtYearPublished.Text)
        rsTitles.Fields("PubID") = PubID
```

```
    If txtDescription.Text <> "" Then rsTitles.Fields("Description")
= txtDescription.Text
    If txtNotes.Text <> "" Then rsTitles.Fields("Notes") =
txtNotes.Text
    If txtSubject.Text <> "" Then rsTitles.Fields("Subject") =
txtSubject.Text
    If txtComments.Text <> "" Then rsTitles.Fields("Comments")
= txtComments.Text
    rsTitles.Update
    dcPubID.Text = PubID 'khi goi method AddNew gia tri trong
datacombo dcPubID se la ro^~ng do do sau khi goi method
Update ba.n phai tra? lai gia tri trong datacombo dcPubID ma nguoi
dung da chon
  Else  ' nhấn button Update, không cập nhật khóa chính ISBN
  If (rsTitles.BOF = False And rsTitles.EOF = False) Then
'recordset có mẫu tin
    If rsTitles.Fields("Title").OriginalValue <> txtTitle.Text Then
'nếu khác giá trị ban đầu của filed=>cập nhật field
      rsTitles.Fields("Title") = txtTitle.Text
    End If
    If rsTitles.Fields("Year Published").OriginalValue <>
txtYearPublished.Text Then ' nếu khác giá trị ban đầu của
filed=>cập nhật field
      rsTitles.Fields("Year Published") = txtYearPublished.Text
    End If

'không có lệnh sau, bởi vì kết buộc dữ liệu DataField trên
datacombo dcPubID nên giá trị PubID sẽ tự động được cập nhật
vào recordset mỗi khi có sự thay đổi
  'If (txtPubID.Text<>"") and
rsTitles.Fields("PubID").OriginalValue<>CLng(txtPubID.Text) Then
    '   rsTitles.Fields("PubID") = CLng(txtPubID.Text)
    ' End If

    If rsTitles.Fields("Description").OriginalValue <>
txtDescription.Text Then
      rsTitles.Fields("Description") = txtDescription.Text
    End If
    If rsTitles.Fields("Notes").OriginalValue <> txtNotes.Text
Then
```

```
            rsTitles.Fields("Notes") = txtNotes.Text
        End If
        If rsTitles.Fields("Subject").OriginalValue <> txtSubject.Text
Then
            rsTitles.Fields("Subject") = txtSubject.Text
        End If
        If          rsTitles.Fields("Comments").OriginalValue          <>
txtComments.Text Then
            rsTitles.Fields("Comments") = txtComments.Text
        End If
        rsTitles.Update
      End If
    End If
    SetButtons (False)  'làm mờ hai nút OK và Cancel
End Sub

Private Sub Form_Unload(Cancel As Integer)
    rsTitles.Close
    rsPublishers.Close
    cnn.Close
End Sub
```

## II. DATAGRID

DataGrid là một control cho phép bạn hiển thị dữ liệu theo dạng lưới gồm các dòng và cột, mỗi cột tương ứng với một field trong CSDL, mỗi dòng sẽ hiển thị thông tin của một mẩu tin tùy thuộc vào số field hiển thị. Hình VIII .3 minh họa một DataGrid.



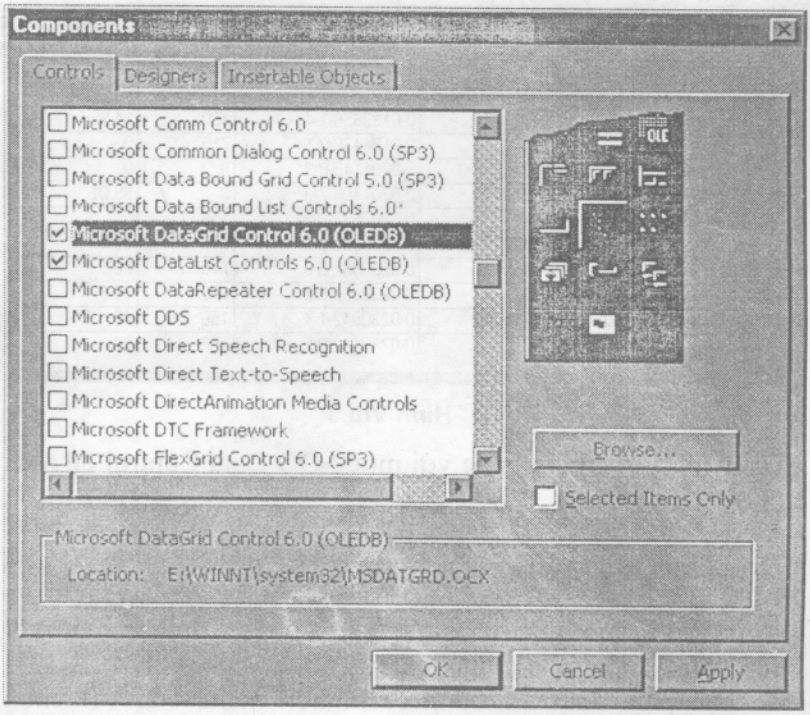
| | | | | |
|---|---|---|---|---|
| The dBASE Programmir | 1994 | 0-0038326-7-8 | 469 | 29.5 |
| dBASE III Plus | 1983 | 0-0038337-8-X | 469 | 29.5 |
| Database Management | 1989 | 0-0131985-2-1 | 715 | 54 |
| Wordstar 4.0-6.0 Quick | 1990 | 0-0133656-1-4 | 460 | 14.95 |
| Oracle Triggers and Sto | 1996 | 0-0134436-3-1 | 715 | 0 |
| Programming in Clipper | 1988 | 0-0201145-8-3 | 9 | 0 |
| Inside MacIntosh | 1994 | 0-0201406-7-3 | 9 | 99.01 |
| Omni Online Database C | 1983 | 0-0207992-0-9 | 156 | 0 |
| Structured C for Engine | 1995 | 0-0230081-2-1 | 715 | 54 |
| An Introduction to Asse | 1995 | 0-0230362-0-6 | 715 | 60 |
| Applied Calculus With L | 1995 | 0-0230650-8-7 | 119 | 70.67 |
| Information Systems Life | 1991 | 0-0230942-1-4 | 156 | 48 |
| Information Systems Life | 1991 | 0-0230942-9-1 | 156 | 14 |

**Hình VIII.3**

DataGrid được ràng buộc với một ADO Data Control để hiển thị dữ liệu.

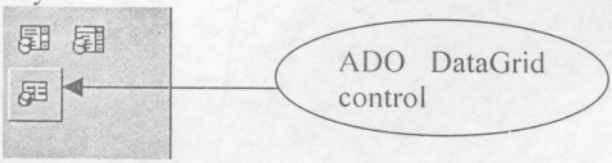
## II.1. Thiết lập DataGrid trên form

DataGrid là một control không có sẵn trên Toolbox như một số control thông dụng khác (textbox, combobox, label,...) do đó bạn phải đưa vào bằng cách: trên menu của VB chọn Project → Components (hoặc click chuột phải trên Toolbox chọn Components), trong tab Control bạn check vào ô "Microsoft ADO DataGrid Control 6.0 (OLEDB)" rồi nhấn nút Apply:




**Hình VIII.4**

Lúc này trên Toolbox xuất hiện biểu tượng như hình dưới đây:

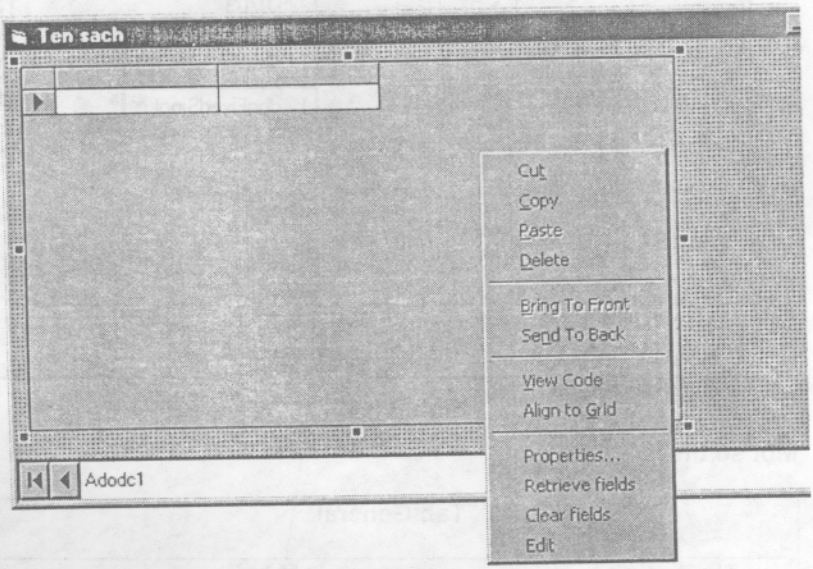



Bạn tạo mới một form, đặt thuộc tính **Name** của form là '**frmTitles_4**'. Kế tiếp bạn vẽ control này vào form, tên mặc định của control là DataGrid1 (có thể đổi tên khác). Để hiển thị dữ liệu trên DataGrid cần phải có một DataSource, bạn có thể dùng ADO Data Control. Giả sử dùng một ADO Data Control làm DataSource cho DataGrid, bạn vẽ một ADO Data

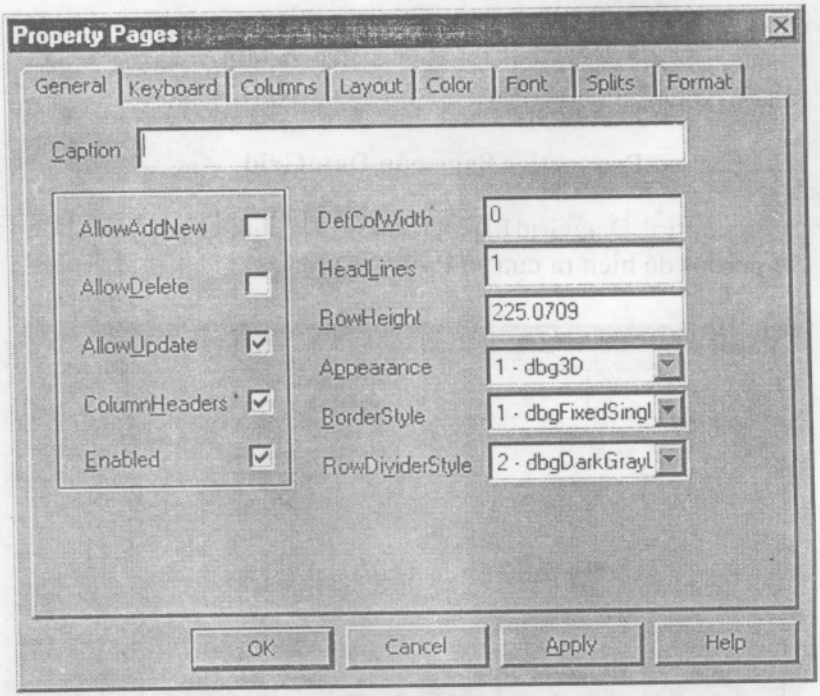
Control lên form, đặt name "Adodc1", chỉ ra chuỗi ConnectString đến database BIBLIO.mdb, thuộc tính RecordSource là table Titles.

## II.2. Cửa sổ Properties Page của DataGrid

Chọn control DataGrid1 trên form, click phải chuột bạn chọn Properties để hiện ra cửa sổ Properties Page




**Hình VIII.5**




**Hình VIII.6**

**Một số thuộc tính chung**

| Tab General | |
|---|---|
| **Thuộc tính** | **Mô tả** |
| Caption | Tiêu đề của DataGrid, ví dụ : Danh mục sách |
| AllowAddNew | Cho phép nhập một mâu tin mới trên DataGrid |
| AllowDelete | Cho phép xóa trực tiếp một mẩu tin trên DataGrid |
| AllowUpdate | Cho phép chỉnh sửa mẩu tin trên DataGrid |
| ColumnHeaders | Tiêu đề của các cột, nếu check → sẽ hiển thị tiêu đề của các cột (mặc định là tên của các field) |
| Enabled | Cho phép kích hoạt DataGrid hay không |

| Tab Column | |
|---|---|
| **Thuộc tính** | **Mô tả** |
| Column | Cột sẽ hiển thị trên DataGrid (mặc định là hai cột, bạn có thể thêm số cột), mỗi cột ứng với một field trong RecordSource |
| Caption | Tên của cột hiển thị |
| DataField | Chỉ ra field (cột) nào trong nguồn dữ liệu sẽ hiển thị |

Ngoài ra còn một số thuộc tính về định dạng font chữ, màu nền, định dạng kiểu dữ liệu cho từng cột,....

Bạn có thể hiệu chỉnh số cột cho DataGrid theo ý muốn bằng cách click phải chuột DataGrid → chọn Edit → click phải chuột lần nữa → chọn Insert (thêm cột mới), Delete (xóa bớt cột), Append (chèn cột mới ở vị trí cuối).

### II.3. Kết nối DataGrid với nguồn dữ liệu

Bạn thao tác trực tiếp trên form hoặc thiết lập bằng lệnh:
**Set DataGrid.DataSource = ADODataControl.Recordset**

Tạo một form mới, đặt name "frmTitles_4", đặt một điều khiển ADO DataControl (name là Adodc1), đặt một điều khiển DataGrid control lên form (name là DataGrid1).

Kết nối DataGrid với nguồn dữ liệu:

**Cách 1**: Chỉ ra chuỗi ConnectString, RecordSource (table Titles) cho ADO Data Control như đã thực hiện trong các phần trước. Chọn DataGrid, vào cửa sổ Propoerties chọn thuộc tính DataSource là Adodc1 (click chuột vào ComboBox xổ xuống bạn sẽ nhìn thấy Adodc1).

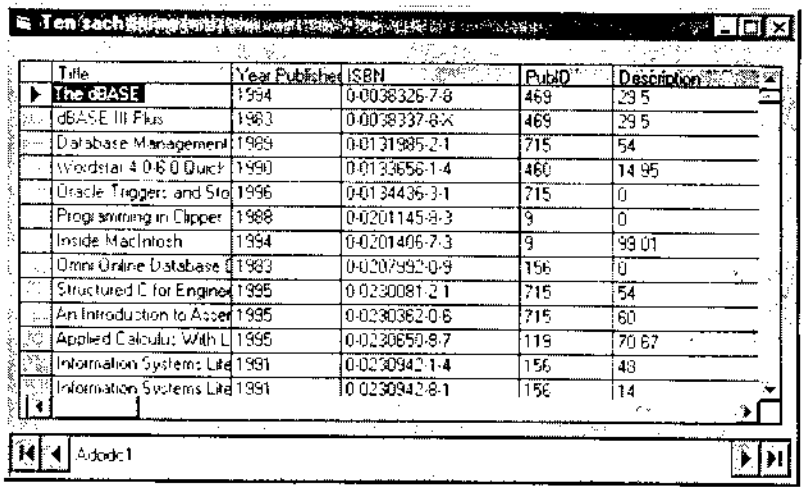
**Lưu ý:** *Nếu chưa chỉ định chuỗi ConnectString và RecordSource trong Adodc1 bạn không được chỉ định thuộc*

*tính DataSource=Adodc1 cho DataGrid nếu không khi chạy chương trình sẽ báo lỗi vì Adodc1 chưa được khởi tạo.*

**Cách 2:** Trường hợp bạn muốn chỉ định kết nối CSDL mỗi khi chạy chương trình, bạn viết code trong sự kiện Form_Load() như sau:

```
Private Sub Form_Load()
   Dim DatabasePath As String
   DatabasePath = App.Path & "\BIBLIO.mdb"
   Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=" & DatabasePath
   Adodc1.RecordSource = "Select * from TITLES"
   Adodc1.Refresh 'thi hành kết nối CSDL
   Set DataGrid1.DataSource = Adodc1.Recordset 'đưa dữ liệu vào
datagrid
End Sub
```

Chạy chương trình với cả hai cách, xem màn hình kết quả dưới đây (bạn di chuyển đến các mẩu tin đầu, cuối, kế tiếp, trước đó).




**Hình VIII.7**

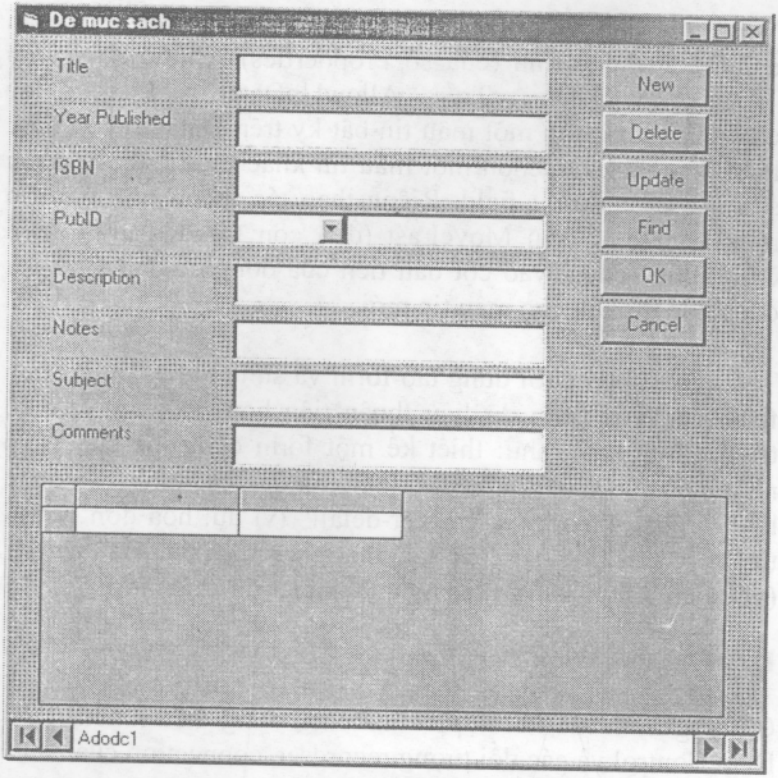
Nếu muốn thao tác trực tiếp đến các mẩu tin trên lưới, bạn chỉnh các thuộc tính (cửa sổ Propoerties) AllowAddNew = True, AllowDelete =True, AllowUpDate = True. Chạy chương trình, chọn một mẩu tin bất kỳ trên DataGrid và thực hiện thao tác xóa, chọn một mẩu tin khác thực hiện thao tác sửa giá trị của một field. Riêng thao tác thêm một mẩu tin mới, bạn thực hiện MoveLast (đưa con trỏ đến dòng cuối cùng) click chuột vào cột đầu tiên của dòng có dấu * và bắt đầu nhập dữ liệu cho mẩu tin mới.

Thông thường người dùng tạo form và sử dụng DataGrid trên form cho việc xem dữ liệu thuận tiện hơn. Đối với một số trường hợp khác như: thiết kế một form cùng lúc cho phép nhập liệu vào cả hai table thì thông thường các table này quan hệ với nhau theo kiểu "master-detail" (ví dụ: hóa đơn và chi tiết hóa đơn), lúc này thao tác nhập liệu hay xóa, sửa "detail" (ví dụ chi tiết hóa đơn) ta cho thực hiện trên DataGrid.

### II.4. Chương trình ứng dụng

Chượng trình sau kết hợp cả hai cách dùng điều khiển ADO Data Control và các đối tượng recordset, connection của ADO vào chung một form.

Bạn tạo một form mới, đặt name "frmTitles_5", dùng 8 textbox: txtTitle, txtYearPublished, txtISBN, txtName, txtDescription, txtNotes, txtSubjects, txtComments, 8 label đặt trước 8 textbox với thuộc tính name lần lượt là: lblTitle, lblYearPublished, lblISBN, lblPubID, lblDescription, lblNotes, lblSubjects, lblComments, đặt một điều khiển DataCombo (name là dcPubID), đặt một điều khiển ADO DataControl (Adodc1), đặt tiếp một điều khiển DataGrid control lên form (DataGrid1), đặt 6 nút New (cmdNew), Delete (cmdDelete), Edit (cmdEdit), OK (cmdOK), Cancel (cmdCancel), Find (cmdFind) như hình sau.




**Hình VIII.8**

Form này không sử dụng các nút di chuyển đến mẩu tin đầu, cuối, sau, trước (cmdFirst, cmdLast...) mà chỉ di chuyển bằng điều khiển ADO Data Control. Mặt khác, ta không kết buộc dữ liệu "**Data Binding**" các textbox với ADO Data Control này (để thao tác dễ dàng hơn cho người dùng) do đó các textbox này không tự động hiển thị dữ liệu lấy từ ADO Data Control này, bạn phải viết hàm hiển thị dữ liệu từ ADO Data Control lên textbox mỗi khi di chuyển mẩu tin. Sự kiện mỗi khi di chuyển mẩu tin hoàn tất trong ADO Data Control gọi là "MoveComplete" (Hàm Adodc1_MoveComplete()). Bạn chỉ muốn cho xem dữ liệu trên DataGrid, các thao tác thêm xóa, sửa thông qua các nút New, Delete, Update do đó bạn phải set

thuộc tính của DataGrid trong cửa sổ Properties như sau: AllowAddNew = False, AllowDelete = False, AllowUpdate = False.

Khai báo General:

```
Dim cnn As New ADODB.Connection
Dim rsPublishers As New ADODB.Recordset
Dim AddNewFlag As Boolean 'them mot record moi(=True), cap nhat mot record(=false)
```

Đoạn code sự kiện Form_Load, ta dùng "Adodc1.Recordset" thay cho đối tượng recordset "rsTitles" đã dùng trong form frmTitles_3, do đó đoạn code trong các hàm DisplayCurrentRecord và code xử lý các chức năng thêm mới, xóa, cập nhật cũng thay đổi theo.

```
Private Sub Form_Load()
    Dim DatabasePath As String
    DatabasePath = App.Path & "\BIBLIO.mdb"
    cnn.ConnectionString =
    "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" &
    DatabasePath
    cnn.Open
    'kiểm tra kết nối CSDL
    If cnn.State = 0 Then
        MsgBox "Connecting database fail."
        Exit Sub
    End If

    Adodc1.ConnectionString = cnn.ConnectionString
    Adodc1.RecordSource = "Select * from TITLES"
    Adodc1.Refresh 'thi hành kết nối CSDL
    rsPublishers.Open "Select PubID, Name from Publishers",
    cnn, adOpenStatic, adLockOptimistic
    If Adodc1.Recordset.RecordCount > 0 Then 'có mẩu tin
    'Chuyển đến mẩu tin đầu tiên, sau khi di chuyển sẽ thực
    hiện hàm Adodc1_MoveComplete
        Adodc1.Recordset.MoveFirst
```

```
        End If
        Set DataGrid1.DataSource = Adodc1.Recordset
        Set dcPubID.DataSource = Adodc1.Recordset
        dcPubID.DataField = "PubID"
        Set dcPubID.RowSource = rsPublishers
        dcPubID.ListField = "PubID"

        SetButtons (False)

        'Khi Form_Load Item trong datacombo chưa được selected
        do đó vào sự kiện PubID_Change() sẽ không lấy được
        Name ứng với PubID đó, bạn phải dùng tạm một recordset
        "rsTemp" để lấy Name của PubID trong datacombo.
        '(Bạn có thể bỏ đoạn code này để kiểm tra sự khác biệt).
        Dim rsTemp As New ADODB.Recordset
        rsTemp.Open "Select PubID,Name from Publishers where
        PubID=" & CLng(dcPubID.Text), cnn, adOpenStatic,
        adLockReadOnly, cmdText
        If Not rsTemp.EOF Then
            txtName.Text = rsTemp.Fields("Name")
        End If
        rsTemp.Close
End Sub
```

Sự kiện khi thay đổi giá trị PubID trong DataCombo dcPubID

```
Private Sub dcPubID_Change()
  If IsNull(Adodc1.Recordset.Fields("PubID")) = True Then
    txtName.Text = ""
  Else
   If dcPubID.SelectedItem > 0 Then
    rsPublishers.Bookmark = dcPubID.SelectedItem
    txtName.Text = rsPublishers.Fields("Name")
   End If
  End If
End Sub

Private Sub DisplayCurrentRecord()
'hiển thị giá trị các field của mẩu tin hiện hành lên textbox
```

```
'kiểm tra giá trị null? Nếu khác null gán textbox.Text=giá trị field
   If IsNull(Adodc1.Recordset.Fields("Title")) = True Then
      txtTitle.Text = ""
   Else
      txtTitle.Text = Adodc1.Recordset.Fields("Title")
   End If
   If IsNull(Adodc1.Recordset.Fields("Year Published")) = True
Then
      txtYearPublished.Text = ""
   Else
      txtYearPublished.Text = Adodc1.Recordset.Fields("Year
Published")
   End If
   txtISBN.Text = Adodc1.Recordset.Fields("isbn") 'ISBN là khóa
chính nên khác null
   If IsNull(Adodc1.Recordset.Fields("Description")) = True Then
      txtDescription.Text = ""
   Else
      txtDescription.Text = Adodc1.Recordset.Fields("Description")
   End If
   If IsNull(Adodc1.Recordset.Fields("Notes")) = True Then
      txtNotes.Text = ""
   Else
      txtNotes.Text = Adodc1.Recordset.Fields("Notes")
   End If
   If IsNull(Adodc1.Recordset.Fields("Subject")) = True Then
      txtSubject.Text = ""
   Else
      txtSubject.Text = Adodc1.Recordset.Fields("Subject")
   End If
   If IsNull(Adodc1.Recordset.Fields("Comments")) = True Then
      txtComments.Text = ""
   Else
      txtComments.Text = Adodc1.Recordset.Fields("Comments")
   End If
End Sub

Private Sub Adodc1_MoveComplete(ByVal adReason As
ADODB.EventReasonEnum, ByVal pError As ADODB.Error,
```

```
adStatus As ADODB.EventStatusEnum, ByVal pRecordset As
ADODB.Recordset)
   If (Adodc1.Recordset.BOF = False And Adodc1.Recordset.EOF =
False) Then
      If Adodc1.Recordset.EditMode = adEditAdd Then Exit Sub '
nếu recordset đang ở tình trạng thêm mới => exit sub
      DisplayCurrentRecord
      Adodc1.Caption = "Mau tin thu " &
CStr(Adodc1.Recordset.AbsolutePosition) & " tren " &
CStr(Adodc1.Recordset.RecordCount) 'hiển thị thứ tự mẩu tin trên
tiêu đề của điều khiển Adodc1.
   End If
End Sub
```

Đoạn code cho năm nút New, Edit, Delete, OK, Cancel (có sửa đổi).

```
Private Sub cmdCancel_Click()
   SetButtons (False)
   If (Adodc1.Recordset.BOF = False And Adodc1.Recordset.EOF =
False) Then
      DisplayCurrentRecord
   End If
End Sub

Private Sub cmdNew_Click()
   AddNewFlag = True   'bật cờ thêm một record = true
   ClearAll  'gán giá trị rỗng lên các control.
   SetButtons (True)
End Sub

Private Sub cmdDelete_Click()
   On Error GoTo DeleteErr
   If (Adodc1.Recordset.BOF = False And Adodc1.Recordset.EOF =
False) Then 'recordset có mẩu tin cần xóa
      returnvalue = MsgBox("Do you want to delete this record?",
vbOKCancel + vbQuestion, "Notice")
      If returnvalue = vbOK Then
         Adodc1.Recordset.Delete 'xóa mẩu tin hiện hành
```

```
        Adodc1.Recordset.MoveNext 'chuyển đến mẩu tin kế
        If Adodc1.Recordset.EOF = True Then
Adodc1.Recordset.MoveLast 'kiểm tra EOF
      End If
      Exit Sub
   End If
DeleteErr:
   MsgBox Err.Description
End Sub

Private Sub cmdUpdate_Click()
   AddNewFlag = False
   Me.Caption = "Cập nhật đề mục sách"  'đổi tiêu đề form cho
người dùng biết đang cập nhật mẩu tin
   SetButtons (True)
End Sub

Private Sub cmdNew_Click()
   AddNewFlag = True  'bật cờ thêm một record = true
   ClearAll  'gán giá trị rỗng lên các control.
   Me.Caption = "Thêm mới đề mục sách"  'đổi tiêu đề form cho
người dùng biết đang thêm mẩu tin mới
   SetButtons (True)
End Sub

Private Sub cmdOK_Click()
'kiểm tra các Ràng buộc toàn vẹn ở đây, nếu không thỏa Exit Sub

   If AddNewFlag = True Then ' người dùng nhấn nút New
   'giữ lại giá trị dcPubID mà người dùng đã chọn vào biến PubID
vì: khi gọi method AddNew giá trị datacombo dcPubID sẽ rỗng
      Dim PubID As Long
      PubID = CLng(dcPubID.Text)
      Adodc1.Recordset.AddNew
      Adodc1.Recordset.Fields("ISBN") = txtISBN.Text 'không được
null
      Adodc1.Recordset.Fields("Title") = txtTitle.Text 'không được
null
      Adodc1.Recordset.Fields("Year Published") =
CInt(txtYearPublished.Text)  'không được null
```

```
        Adodc1.Recordset.Fields("PubID") = PubID
        If txtDescription.Text <> "" Then
Adodc1.Recordset.Fields("Description") = txtDescription.Text
        If txtNotes.Text <> "" Then Adodc1.Recordset.Fields("Notes")
= txtNotes.Text
        If txtSubject.Text <> "" Then
Adodc1.Recordset.Fields("Subject") = txtSubject.Text
        If txtComments.Text <> "" Then
Adodc1.Recordset.Fields("Comments") = txtComments.Text
        Adodc1.Recordset.Update
        dcPubID.Text = PubID 'khi gọi method AddNew giá trị trong
datacombo dcPubID sẽ rỗng do đó sau khi gọi method Update ta
phải trả lại giá trị trong dcPubID mà người dùng đã chọn.
    Else 'nhấn nút Update, không cập nhật khóa chính ISBN
    If (Adodc1.Recordset.BOF = False And Adodc1.Recordset.EOF =
False) Then 'recordset có mẩu tin
        If Adodc1.Recordset.Fields("Title").OriginalValue <>
txtTitle.Text Then 'nếu khác giá trị ban đầu của filed=>cập nhật
            Adodc1.Recordset.Fields("Title") = txtTitle.Text
        End If
        If Adodc1.Recordset.Fields("Year Published").OriginalValue
<> txtYearPublished.Text Then ' nếu khác giá trị ban đầu của
filed=>cập nhật field
            Adodc1.Recordset.Fields("Year Published") =
txtYearPublished.Text
        End If
        If Adodc1.Recordset.Fields("Description").OriginalValue <>
txtDescription.Text Then
            Adodc1.Recordset.Fields("Description") =
txtDescription.Text
        End If
        If Adodc1.Recordset.Fields("Notes").OriginalValue <>
txtNotes.Text Then
            Adodc1.Recordset.Fields("Notes") = txtNotes.Text
        End If
        If Adodc1.Recordset.Fields("Subject").OriginalValue <>
txtSubject.Text Then
            Adodc1.Recordset.Fields("Subject") = txtSubject.Text
        End If
```

```
      If  Adodc1.Recordset.Fields("Comments").OriginalValue  <>
txtComments.Text Then
         Adodc1.Recordset.Fields("Comments")                    =
txtComments.Text
      End If
      Adodc1.Recordset.Update
      DataGrid1.Refresh
    End If  'recordset có mẩu tin
  End If
  SetButtons (False)  'làm mờ hai nút OK và Cancel
End Sub
```

Hàm tìm kiếm các mẩu tin, điều kiện tìm là tên sách (field "Titles" trong table Titles). Đoạn code hàm cmdFind_Click() như dưới đây.

```
Private Sub cmdFind_Click()
  If Adodc1.Recordset.RecordCount <> 0 Then
    Dim flag As Boolean
    Dim currentrecord_primarykey As String
    flag = True
    return_value = "0"
    While flag = True And return_value <> ""
    'vòng lặp dừng khi tìm thấy tên sách (field Title) hoặc người
dùng nhấn nút Cancel ( lúc này return_value="")
      return_value = InputBox("Input Title:", "Look up Title",
"0")   'gán giá trị mặc định là chuỗi "0"
      If (return_value <> "") Then
        'lưu lại khóa chính của mẩu tin hiện hành để khi tìm
không được ta sẽ trở lại mẩu tin này
        currentrecord_primarykey                              =
Adodc1.Recordset.Fields("ISBN")
        Adodc1.Recordset.Find "Title='" & return_value & "'", , ,
adBookmarkFirst
        If Not Adodc1.Recordset.EOF Then
          flag = False 'tìm thấy → dừng
        Else
          MsgBox "This title not found!", vbOKOnly +
vbInformation, "Infomation"
```

```
                Adodc1.Recordset.Find          "ISBN='"          &
currentrecord_primarykey & "'", , , adBookmarkFirst
                If Adodc1.Recordset.EOF Then
                   Adodc1.Recordset.MoveFirst
                End If
             End If
          End If
       Wend
    End If
End Sub
```

Với các hàm SetButtons và ClearAll() như sau:

```
Private Sub SetButtons(bVal As Boolean)
    cmdOK.Enabled = bVal
    cmdcancel.Enabled = bVal
    cmdnew.Enabled = Not bVal
    cmdUpdate.Enabled = Not bVal
    cmddelete.Enabled = Not bVal

    '(cho/khong cho) edit data combo dcPubID
    dcPubID.Locked = Not bVal

  'trả lại tiêu đề đầu tiên của form khi người dùng nhấn OK/Cancel
    If Not bVal Then Me.Caption = "De muc sach"
End Sub

Private Sub ClearAll()
    txtTitle.Text = ""
    txtYearPublished.Text = ""
    txtISBN.Text = ""
    txtDescription.Text = ""
    txtNotes.Text = ""
    txtSubject.Text = ""
    txtComments.Text = ""
End Sub

Private Sub Form_Unload(Cancel As Integer)
    rsPublishers.Close
    cnn.Close
```

```
End Sub
```

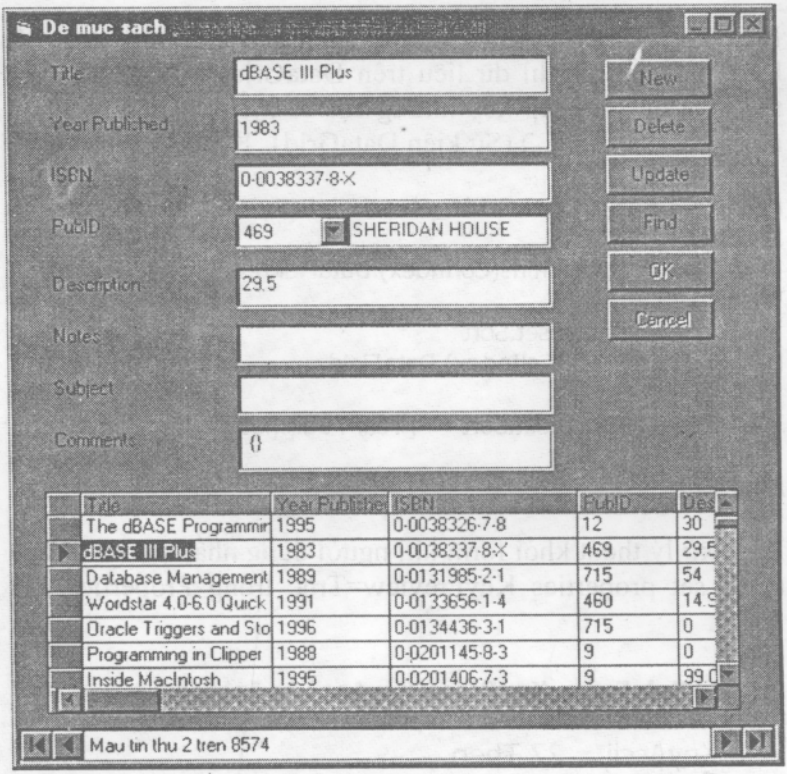
Hàm xử lý: mỗi khi người dùng click chuột trên một cột bất kỳ của DataGrid thì dữ liệu trên DataGrid sẽ được sắp xếp theo cột đó (chỉ sắp xếp những cột có kiểu dữ liệu phù hợp như text, number,...) (Sự kiện DataGrid1_HeadClick).

```
Private Sub DataGrid1_HeadClick(ByVal ColIndex As
Integer)
   If DataGrid1.Columns(ColIndex).DataField <> "Year Published"
Then
      Adodc1.Recordset.Sort                                    =
DataGrid1.Columns(ColIndex).DataField
   Else
      Adodc1.Recordset.Sort = "[Year Published]"
   End If
End Sub
```

Hàm xử lý thoát khỏi form khi người dùng nhấn phím Escape (phải set properties KeyPreview=True trong Properties của form).

```
Private Sub Form_KeyPress(KeyAscii As Integer)
'Trong Properties của form bạn phải set KeyPreview=true
   If KeyAscii = 27 Then
      Set rec = Nothing
      Unload Me
   End If
End Sub
```

## Màn hình kết quả




**Hình VIII.9**

# Chương IX

# REPORT

## I. MỞ ĐẦU

### I.1. Giới thiệu

Ở các chương trước chúng ta đã tìm hiểu về form. Form được dùng để đưa thông tin dữ liệu từ bên ngoài vào chương trình. Tuy nhiên, việc khai thác chương trình không chỉ ở chỗ nhập vào các thông tin mà còn ở chỗ hiển thị các thông tin trong chương trình ra theo một thứ tự nào đó. Có thể các thông tin đó phải được tổng hợp, sắp xếp lại. Sau các bước trên, các thông tin được xuất ra trên màn hình và nó có thể in ra giấy được. Visual Basic hỗ trợ cho chúng ta một công cụ để thực hiện công việc trên, công cụ đó là DataReport.

DataReport có hạn chế lớn nhất là nó không hỗ trợ cho lập trình viên các thao tác về đồ thị. Do đó khi sử dụng các báo biểu (report) về đồ thị người ta hay sử dụng Cystal Report (công cụ này tuy có nhiều chức năng nhưng khó sử dụng khi đóng gói chương trình). Tuy có hạn chế trên nhưng DataReport là một công cụ đơn giản để cho lập trình viên sử dụng để kết xuất các báo biểu cho chương trình.

### I.2. Các đối tượng của Data Report

Data Report bao gồm các đối tượng sau:

1. **Đối tượng DataReport** — Tương tự như một form của Visual Basic, DataReport có hai phần: phần thiết kế giao diện trực quan và phần viết lệnh. Trong phần viết lệnh chúng ta có thể lập trình cho các control và section trong report.

2. **Đối tượng Section**— Section là một vùng hiển thị trong Data Report. Mỗi section được biểu thị bởi một cái đầu đề (header).

3. **Data Report Controls**—Chỉ có một vài control đặc biệt có thể đặt trên Data Report. Bạn không thể dùng các control trên form hay Active X control để gắn vào Data Report. Những control này được tìm thấy trên Toolbox phần tab DataReport.

### I.3. Phân loại section trong Data Report

Data Report mặc định chứa các section sau:

- Report Header—chứa các dòng chữ hay hình ảnh xuất hiện ở đầu của mỗi report ví dụ như : Tiêu ngữ ("Cộng hòa Xã hội Chủ nghĩa Việt Nam"), tiêu đề của report... Nếu bạn muốn Report Header chiếm nguyên một trang đầu thì bạn có thể thiết lập thuộc tính ForcePageBreak của nó thành **rptPageBreakAfter**.

- Page Header—chứa các thông tin mà nó hiện diện ở đầu mỗi trang, ví dụ như tựa báo cáo, tên CSDL...

- Group Header/Footer—chứa các section lặp lại của data report. Mỗi group header được kết thúc với một group footer. Cặp header và footer này được liên kết với một đối tượng Command đơn trong Data Environment.

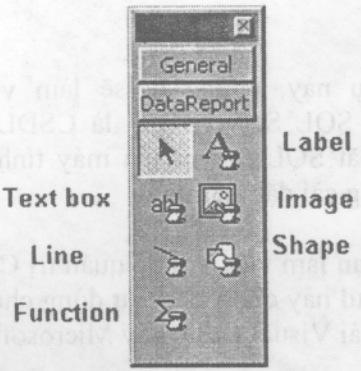
- Details—chứa phần (record) lặp lại tận cùng nhất của report. Details section được liên kết với đối tượng Command có cấp thấp nhất trong sơ đồ phân cấp Data Environment.

- Page Footer—chứa các thông tin nằm ở cuối mỗi trang như số trang...

- Report Footer—chứa các thông tin xuất hiện ở cuối mỗi report, như các thông tin về tổng kết, số lượng mẫu tin trong báo cáo, địa chỉ .... Report Footer xuất hiện ở giữa Page Header và Page Footer cuối cùng.

## I.4. Các control của Data Report

Khi một Data Report mới được thêm vào project, các control sau sẽ xuất hiện trong tab Toolbox của DataReport:




**Hình IX. 2**: *Các control của Report*

- TextBox Control (RptTextBox)— cho phép bạn liên kết (bound) với một field trong CSDL. Control này giống TextBox trong form.

- Label Control (RptLabel)— cho phép bạn ghi một dòng chữ lên DataReport. Control này giống control Label trong form.

- Image Control (RptImage)—cho phép bạn chèn một hình lên report.

- Line Control (RptLine)—Cho phép bạn vẽ các đường thẳng lên report.

- Shape Control (RptShape)—cho phép bạn vẽ các hình chữ nhật, hình tam giác, hình tròn, hình oval lên report.

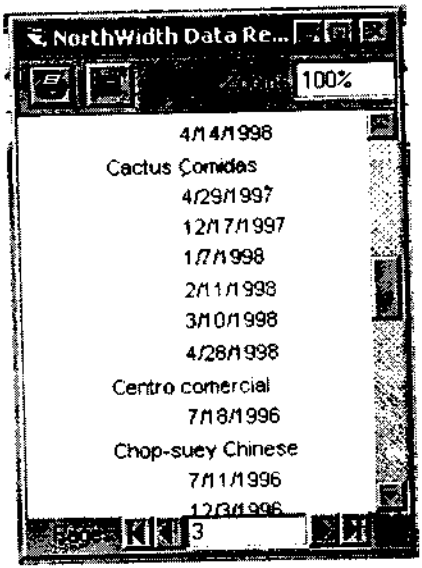
- Function Control (RptFunction)—đây là một textbox đặc biệt, nó có thể tính toán giá trị khi report được tạo ra.

## II. VÍ DỤ 1

Trong các ví dụ này, chúng ta sẽ làm việc trên CSDL **NorthWind** của SQL Server. Đây là CSDL mẫu của SQL Server, khi cài đặt SQL Server lên máy tính thì CSDL này cũng được tự động cài đặt thêm vào.

Tuy nhiên, nếu bạn làm việc với hệ quản trị CSDL Access thì CSDL **NorthWind** này cũng được tự động chép vào máy tính của bạn khi bạn cài Visual Basic hay Microsoft Access.

Chúng ta sẽ bắt đầu tạo một data report đơn giản bằng cách dùng Data Environment như là một nguồn dữ liệu. Ở đây chúng ta sẽ tạo ra một data report bằng cách dùng dữ liệu của hai bảng Customers và Orders kết nối thông qua trường CustomerID. Cuối cùng ta thu được datareport như sau




**Hình IX. 3**: *Minh họa một Report*

## II.1. Tạo một con trỏ phân cấp đơn giản trong Data Environment designer

1. Tạo một project mới theo kiểu **Standard Exe**. Đặt tên cho project là prjNwind. Đặt lại tên form là frmShowReport.
2. Trên menu **Project/Add Data Environment** để thêm một chức năng vào project của bạn. Nếu chức năng này không được liệt kê trong menu **Project**, bạn click vào menu **Project/Components**. Trong **Designers Tab**, chọn Data Environment để thêm chức năng này vào menu.
3. Click vào icon **Data Environment1**, vào cửa sổ Properties đổi tên của **Data Environment1** thành deNwind.

4. Click phải chuột vào icon **Connection1**, chọn Properties. Cửa sổ **Data Link Properties** sẽ hiện ra. Trên cửa sổ này, chọn **Microsoft OLE DB Provider for SQL Server**. Nếu bạn sử dụng hệ quản trị CSDL Access thì bạn chọn **Microsoft Jet 3.51 OLE DB Provider** hay **Microsoft Jet 4.0 OLE DB Provider.**
5. Chọn nút **Next** để qua tab **Connection**.
6. Trong **Connection Tab** lần lượt làm các thao tác sau:

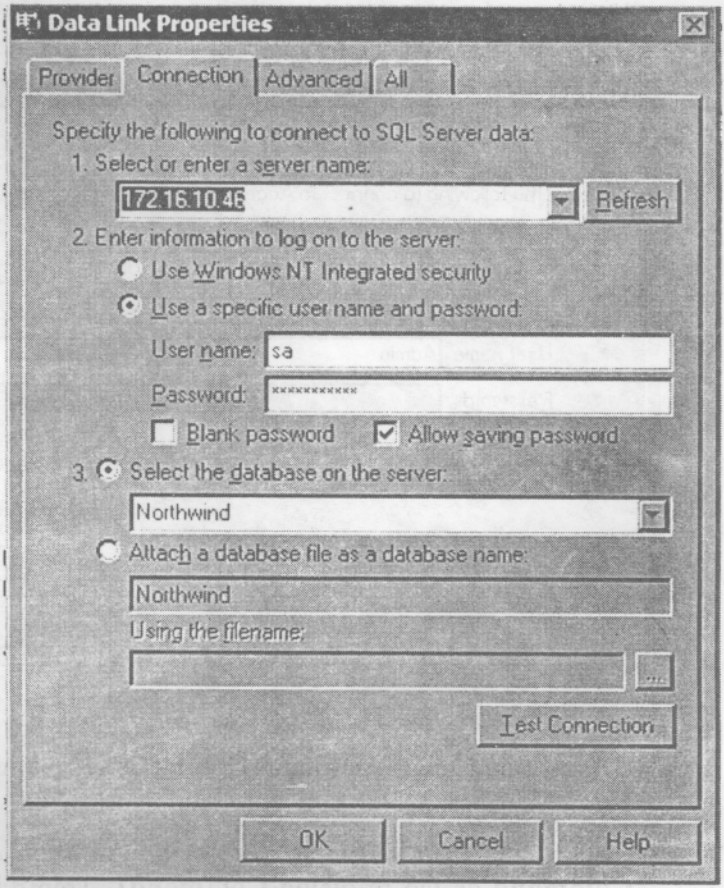
**Nếu bạn sử dụng SQL Server**

a. Ở ComboBox tên server bạn chọn tên của server chứa CSDL.

b. Điền username và password để thêm các thông tin để có thể truy cập vào server có chứa CSDL.

c. Chọn thêm tên của CSDL bạn muốn thao tác.

d. Cuối cùng nhấn **OK** để đóng dialog này



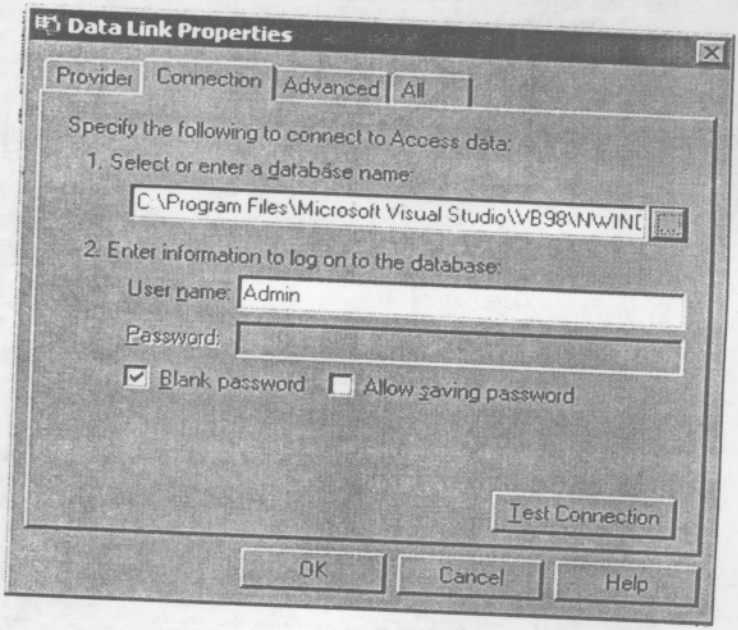

**Hình IX. 4**: *Thiết lập kết nối vào CSDL SQL Server*

**Nếu bạn sử dụng Microsoft Access**

a. Chọn nút cạnh ô TextBox đầu tiên.

b. Dialog **Select Access Database** sẽ hiện ra. Khi đó bạn chỉ định file **nwind.mdb**, file này sẽ được tự động chép vào thư mục Program Files\Microsoft Visual Studio\Vb98 khi bạn cài đặt VB.

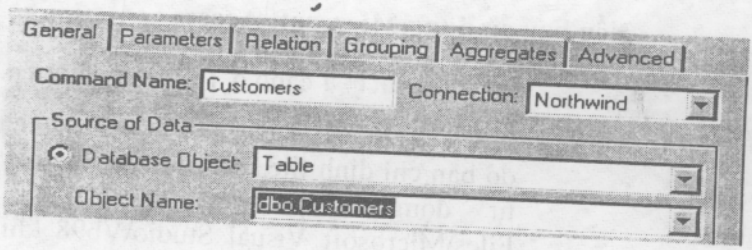
c. Cuối cùng nhấn **OK** để đóng dialog này




**Hình IX. 5**: *Thiết lập kết nối vào CSDL Access*

7. Click chuột phải vào icon **Connection1**, và chọn **Rename**. Thay đổi tên của icon thành Northwind.

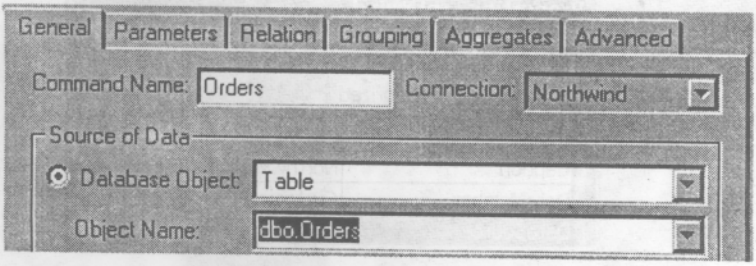
8. Click chuột phải vào icon **Northwind**, và chọn **Add Command** để tạo ra dialog **Command1**. Trên dialog này thiết lập thuộc tính như hình IX. 6




**Hình IX. 6**: *Thiết lập các thông số cho Command*

9. Click **OK** để đóng dialog.

10. Click phải chuột vào command **Customers,** và click **Add Child Command** để tạo dialog **Command2**. Trong dialog này, thiết lập các tham số như hình sau




**Hình IX. 7**: *Thiết lập thông số cho Child Command*

11. Click vào tab **Relation**. Check vào checkbox **Relate to a Parent Command Object**. Ở ô **Parent** chọn **Customers**; Chọn **CustomerID** ở hai ComboBox **Parent Fields** và **Child Fields/Parameters.**

12. Click vào nút **Add**. Sau đó click **OK** để đóng dialog.

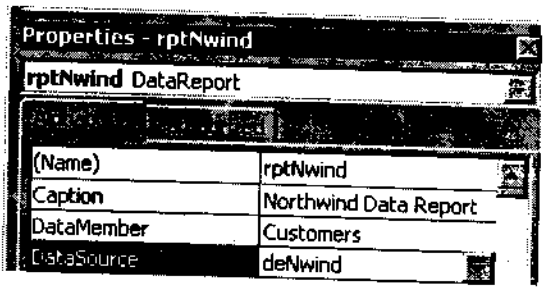
13. Lưu project.

Khi Data Environment designer đã được tạo, bạn bắt đầu tạo một data report. Ta sẽ cho hiển thị tất cả ngày đặt hàng và nhóm theo tên công ty.

## II.2. Tạo mới một data report

1. Trên menu **Project**, click **Add Data Report**, Visual Basic sẽ tạo ra một Data Report trắng cho bạn. Nếu chức năng này không có trong menu **Project**, bạn click vào menu **Components**. Chọn tab **Designers** tab, và chọn **Data Report** để thêm chức năng này vào menu.

2. Thiết lập bốn thuộc tính **Name, Caption, DataMember, DataSource** (thiết lập thuộc tính **DataSource** trước khi thiết lập **DataMember**) cho Data Report này trong cửa sổ properties như sau :




**Hình IX. 8**: *Thiết lập thuộc tính cho Report*

3. Click phải chuột vào Data Report designer, và chọn **Retrieve Structure**.

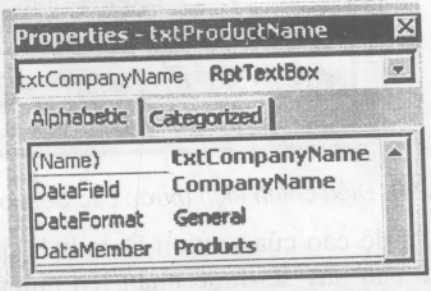
Bạn để ý là có một group section mới được tạo ra. Mỗi group section có một sự tương ứng một- một với một đối tượng Command trong data environment; trong trường hợp này, một Group section mới được tạo ra tương ứng với đối tượng Customers Command. Chú ý thêm là Group Header đi liền với một Group Footer section.

**Chú ý:** *Data Environment cho phép bạn tạo ra một đối tượng Command có phân cấp, tức là một đối tượng Command có một hay nhiều đối tượng con (child Command) ngang hàng. Tuy nhiên, Data Report không linh hoạt, chúng không hiển thị tất cả các đối tượng con của Command vào một lúc. Thay vào đó, khi nhấn Retrieve Structure, Data Report sẽ hiển thị chỉ child command đầu tiên. Do đó, bạn nên tránh tạo*

*Command phân cấp có nhiều children command ngang hàng.*

4. Từ Data Environment, kéo các Field **CompanyName** field (thuộc command **Customers**) vào **Group Header (Customers_Header)** section. Bạn có thể kéo các field khác của command Customers vào **Group Header** này.

   **Chú ý:** *Bạn cũng có thể tạo các textbox như trên bằng cách như sau. Đầu tiên, vẽ một textbox lên section thích hợp (trong trường hợp này là **Group Header (Customers_Header)**). Sau đó, điều chỉnh các thuộc tính của textbox này như sau:*




**Hình IX. 9**: *Thiết lập thuộc tính cho các Control*

5. Xóa control Label (rptLabel) named **Label1**.

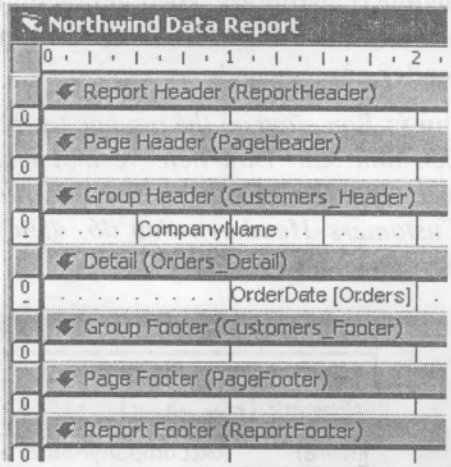
   Nếu bạn không muốn control Label đi kèm với control TextBox, Bạn có thể click phải chuột vào icon Data Environment chọn **Options/** tab **Field Mapping** và uncheck chức năng **Drag and Drop Fields Caption.**

6. Từ Data Environment, kéo field **OrderDate** (của command **Orders**) vào section **Details (Orders_Detail)**. Xóa section Label.

   Section Details biểu thị cho section lặp lại trong cùng, và do đó nó tương ứng với đối tượng Command thấp

nhất trong sơ đồ phân cấp Data Environment: đối tượng Orders Command.

7. Chỉnh lại kích thước độ cao của section thuộc Data Report




**Hình IX. 10**: *Điều chỉnh kích thước các Section trong Report*

Chỉnh lại độ cao của section Details là quan trọng bởi vì chiều cao này sẽ được nhân lên nhiều lần với mỗi giá trị OrderDate của CompanyName.

8. Lưu lại project.

## II.3. Xem Data Report

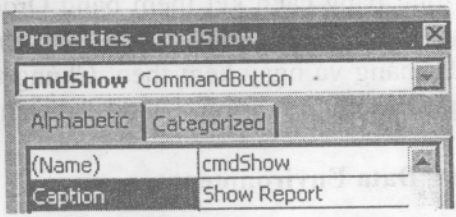
Bạn đã tạo xong đối tượng data environment và data report. Tiếp theo bạn viết lệnh để hiển thị data report.

### II.3.1. Để hiện ra data report lúc chạy chương trình

1. Trên cửa sổ **Project Explorer**, double-click vào icon **frmShowReport** để hiển thị Form designer.
2. Trên **Toolbox**, chọn tab **General**.

3. Click icon **CommandButton** và kéo một **CommandButton** vào form.

Thiết lập thuộc tính cho control **Command1** như sau




**Hình IX. 11**: *Thiết lập thuộc tính cho Command Button*

4. Double-click vào nút vừa tạo và viết lệnh sau

```
Private Sub cmdShow_Click()
        rptNwind.Show
        End Sub
```

5. Lưu lại và chạy project (chú ý khi chạy chương trình form mặc định của project là **frmShowReport**).
6. Click vào nút **Show Report** để hiển thị report ở chế độ print preview.

### II.3.2. Thiết lập Data Report như là một đối tượng Startup

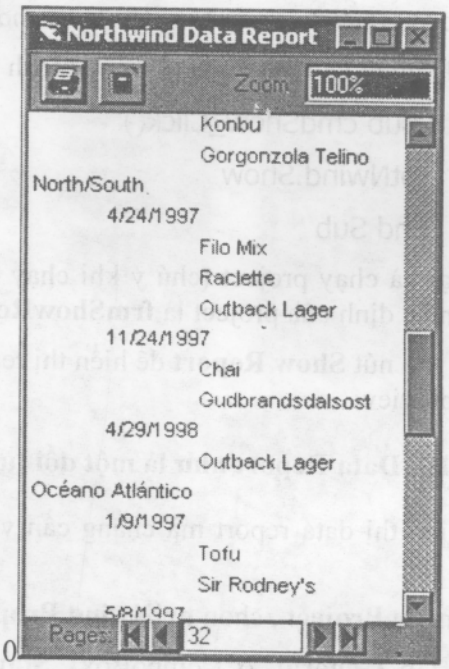
Bạn có thể hiển thị data report mà chẳng cần viết một dòng lệnh nào.

1. Vào menu **Project** , chọn **prjNwind Properties**.
2. Chọn tab **General,** ở ComboBox **Startup Object,** chọn **rptNwind**.
3. Lưu và chạy Project.

## III. VÍ DỤ 2

Ở ví dụ này phức tạp hơn một chút. Data environment ở trên sẽ được mở rộng bằng cách kết thêm bảng Order Details và Products trong CSDL. Report bây giờ có thêm tên của các sản phẩm được đặt hàng và ngày kèm theo. Chúng ta sẽ tiếp tục sử dụng kết quả của chương trình ở ví dụ 1 để làm ví dụ 2.

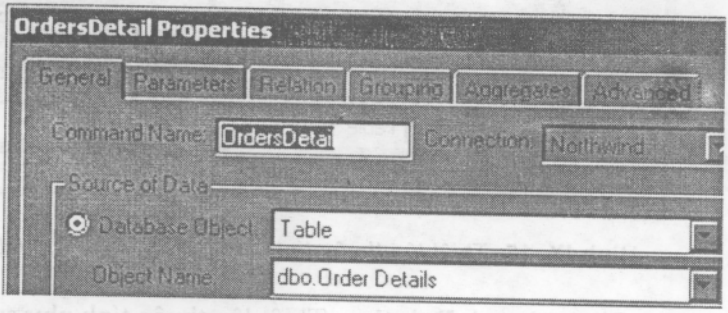
### III.1. Mở rộng Data Environment




**Hình IX. 12**: *Report phân cấp*

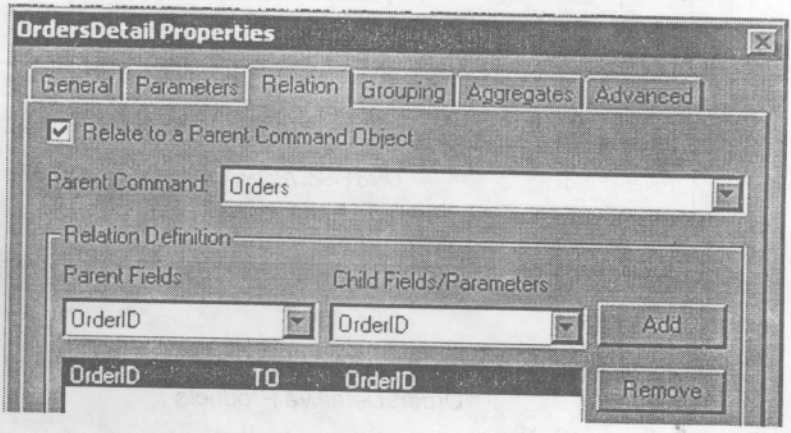
1. Trong Data Environment, right-click đối tượng **Orders** Command. Rồi click **Add Child Command**.

2. Trên dialog **Command1 Properties**, thiết lập các thuộc tính sau



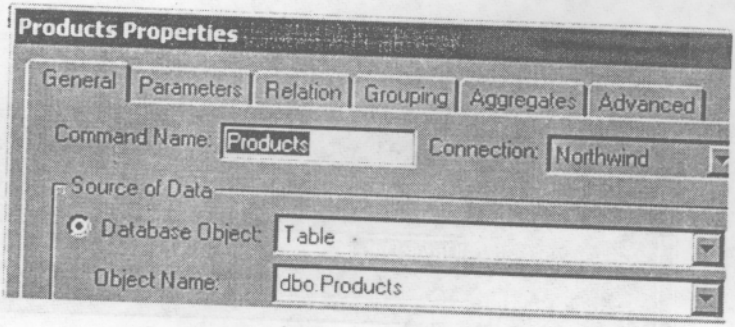

**Hình IX. 13**: *Thiết lập các thuộc tính cho Command Orders*

3. Chọn tab **Relation.** Thiết lập các thuộc tính như sau



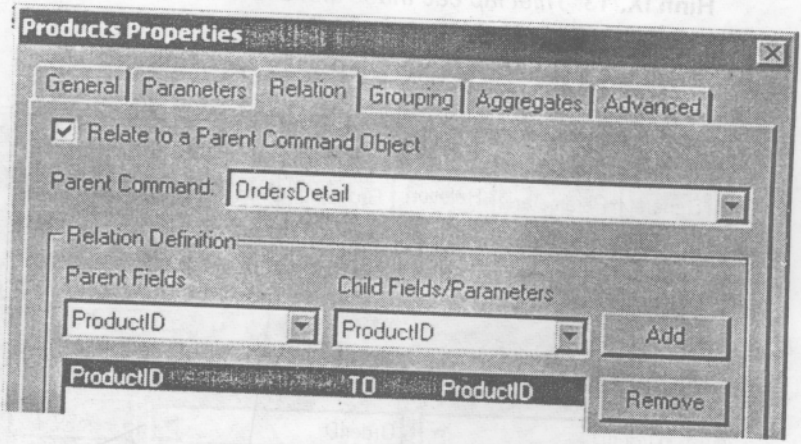

**Hình IX. 14**: *Thiết lập thuộc tính liên hệ giữa Command OrdersDetail và Orders*

4. Right-click vào đối tượng Command **OrderDetails**, và click **Add Child Command**. Thiết lập thuộc tính như sau:




**Hình IX. 15**: *Thiết lập thuộc tính cho Command Products*

5. Click vào tab **Relation**. Thiết lập thuộc tính như sau.




**Hình IX. 16**: *Thiết lập thuộc tính liện hệ giữa Command OrdersDetail và Products*

## III.2. Mở rộng Data Report

1. Right-click vào **Data Report**, không chọn (uncheck) menu **Show Page Header/Footer**.

2. Right-click Data Report, và click vào menu **Insert Group Header/Footer**. Dialog **Insert New Group Header/Footer** sẽ xuất hiện.

   Dialog box này cho phép bạn xác định một cặp header và footer mới sẽ được bao một cặp header/footer khác.

3. Click nút **OK** để chọn nơi đặt cặp header/footer mới theo chế độ mặc định.

4. Chọn group header mới, bạn vào cửa sổ **Properties**, thay đổi tên của nó từ Section1 thành **Orders_Header**. Bạn cũng thay đổi tên của footer tương ứng từ Section4 (có thể lúc này section không mang tên là Section2 hay SectionX (X là số nguyên) ) thành **Orders_Footer**.

5. Lặp lại bước 2 và 3. Tên của group header mới là **OrderDetails_Header**, và tên của group footer mới là **OrderDetails_Footer**.

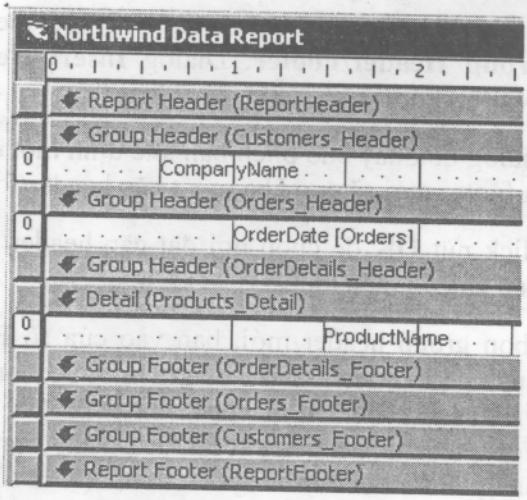
6. Click vào section **Detail (Orders_Detail)** để chọn nó. Trên cửa sổ **Properties**, thay đổi tên của section thành **Products_Detail**.

7. Dùng chuột kéo field **OrderDate** từ section **Detail (Products_Detail)** đến section **Orders_Header**.

8. Từ Data Environment, kéo field **ProductName** (của command **Products**) vào section **Detail (Products_Detail)**.

9. Xóa control **Label** có tên là Label1.

10. Chỉnh lại kích thước các group header, và sắp xếp chúng như hình IX. 17




**Hình IX. 17**: *Điều chỉnh kích thước cho DataReport*

Sau đây là phần giải thích một số các chú ý trên hình trên. Đầu tiên, Group footers được kéo gần nhau để tiết kiệm không gian như đã nói ở ví dụ 1.

Group Header **OrderDetails_Header** cũng bị kéo lên và không một field nào được hiển thị. Điều này là do bảng Order Details trong CSDL Northwind chỉ là một bảng liên kết (join table) – bảng chỉ chứa ID của các bảng ghi (record) từ bảng Orders liên kết với ID của bảng Products. Do đó bảng Order Details không chứa các cột (field) có giá trị ta muốn hiển thị. Trong Data Report, Bảng Order Details chỉ làm chức năng nhóm các bảng ghi "product names" theo ngày đặt hàng.

Cuối cùng, Section **Details** chỉ chứa tên của sản phẩm. Section Details tương trưng cho vòng lặp trong cùng của các vòng lặp bảng ghi lồng nhau.

11. Lưu và chạy project.

## IV. THÊM MỘT FIELD TÍNH TOÁN (CALCULATED FIELD) VÀO DATA REPORT

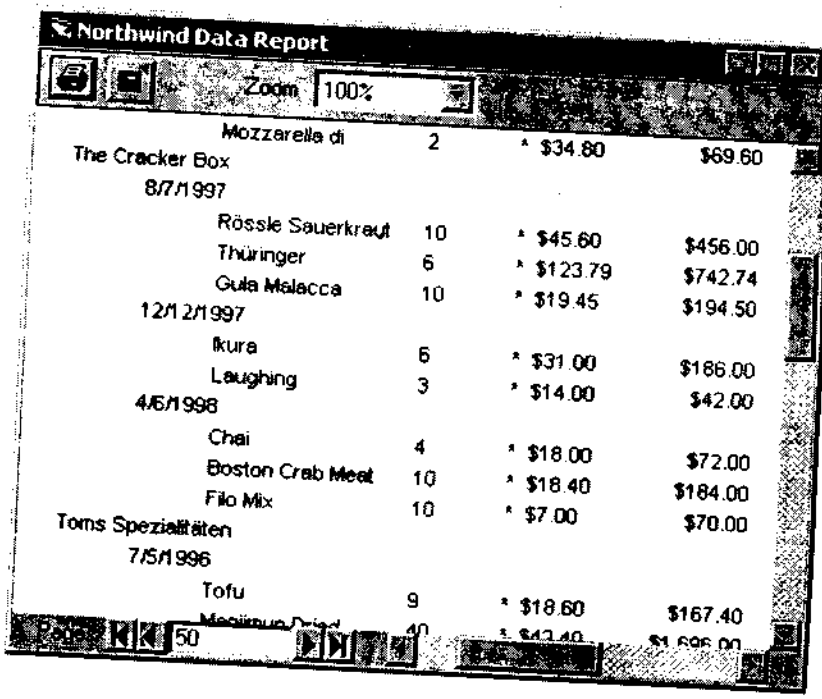
Một calculated field là một field mà giá trị được tính toán khi report được tạo ra. Ví dụ, khi ước lượng số tiền thuế trên các đơn đặt hàng, bạn phải nhân tổng số tiền với tỉ lệ tính thuế. Nhưng do tỉ lệ tính thuế hay thay đổi và do bạn không thể lưu trên CSDL nên giá trị tiền thuế được tạo ra khi report được tạo ra – đây gọi là một calculated field.

Trong ví dụ này, the Data Report trong ví dụ thứ hai sẽ được sửa đổi, nó sẽ có thêm một cột (field) **Total** là kết quả của Quantity * UnitPrice. Để thêm một calculated field chúng ta làm theo bước sau:

1. Dùng câu lệnh SQL trong Command của đối tượng Data Environment để tạo calculated field.

2. Thêm vào data report ba textbox: **Quantity**, **UnitPrice**, và (một calculated field) **Total**.

Khi thực hiện lại, data report có dạng như sau.




**Hình IX. 18**: *Data Report có các Calculated Field*

**Để thêm một calculated field vào data report**

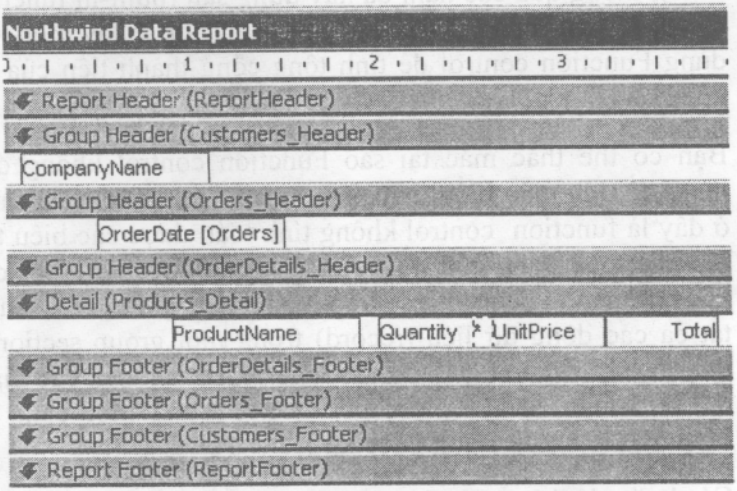
1. Trong Data Environment, right-click vào command **OrderDetails**. Chọn **Properties** để hiển thị dialog **OrderDetails Properties**.
2. Ở tab **General**, chọn radio button **SQL Statement**, và đánh vào dòng lệnh SQL sau:

   ```
   Select OrderID, ProductID, UnitPrice, Quantity,
   (Quantity * UnitPrice) As Total
   From [Order Details]
   ```

3. Nhấn **OK**.

4. Từ Data Environment, kéo các field Quantity, UnitPrice, và Total thuộc command **OrderDetails** vào section **Detail (Products_Detail)** củaData Report.

5. Xóa hai Label controls đi kèm , và thay đổi thuộc tính **Caption** của Label cuối cùng thành *, và sắp xếp các control như sau.




**Hình IX. 19**

6. Trên Data Report, chọn text box **UnitPrice**. Lúc đó vào cửa sổ **Properties**, double-click vào thuộc tính **DataFormat** để hiển thị dialog **Property Pages**.

7. Trên Listbox **Format Type**, chọn **Currency**. Trên combo box **Symbol,** chọn đơn vị tiền tệ tương ứng.

8. Lặp lại bước 6 và 7 để thay đổi thuộc tính **DataFormat** của textbox **Total** thành **Currency**.

9. Chọn text box **Total**. Trên cửa sổ Properties, thay đổi thuộc tính Alignment thành **1 – rptJustifyRight**.

10. Lưu và chạy project.

## V. THÊM MỘT FUNCTION CONTROL VÀO DATA REPORT

Như đã nói ở phần trước Data Report có các control của riêng nó, các control Function. Các control này dùng để hiển thị các thông tin được tính toán trong khi chạy chương trình (run time), thông qua các hàm đã xây dựng sẵn (built-in function), cụ thể là khi report được tạo ra. Chúng tôi sẽ lấy một ví dụ dùng Function control để tính tổng cộng thành tiền của một hóa đơn bất kỳ.

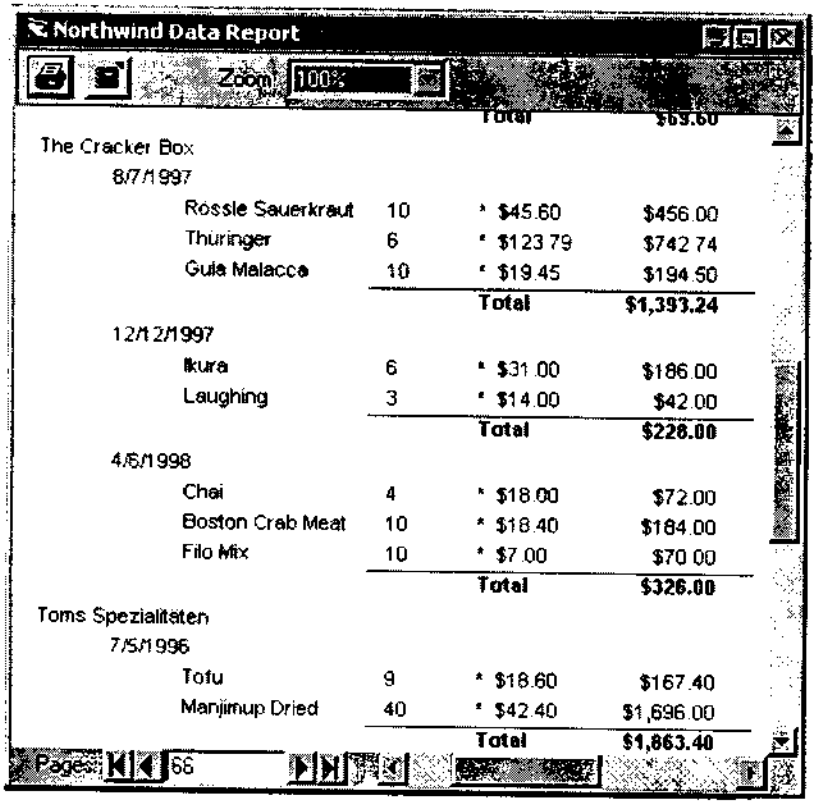
Bạn có thể thắc mắc tại sao Function control không được dùng để tính toán biểu thức Quantity * UnitPrice. Câu trả lời ở đây là function control không tính toán được các biểu thức mà các toán hạng của nó trong cùng một section. Nói cách khác, Function control chỉ dùng để tính các gái trị chit sau khi tất cả các dòng dữ liệu (record) trong một group section đã được xử lý. Ngược lại, các câu lệnh SQL tính toán các giá trị mà các toán hạng là các cột (field) trong một bảng ghi (record) lúc chúng đang được xử lý.

Các bước để thêm một Function control vào Data Report:

1. Kéo một Function control vào một Footer section của Data Report.

2. Thiết lập DataMember và thuộc tính DataField ứng với các giá trị của đối tượng Command.

## V.1. Cách tạo Data Report sử dụng Function Control để hiển thị tổng cộng

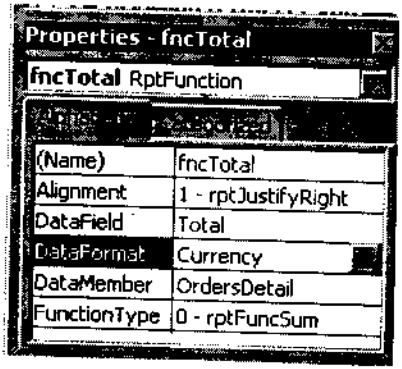



**Hình IX. 20**

**Để thêm một Function control vào data report**

1. Click vào group footer **Customers_Footer** kéo nó xuống để tạo một khoảng trống giữa nó và **Orders_Footer** footer.
2. Trên **Toolbox**, chọn control **rptFunction**.
3. Tạo một control **rptFunction** trên khoảng trống vừa tạo được giữa hai footer.

4. Thiết lập thuộc tính của control **rptFunction** như sau:

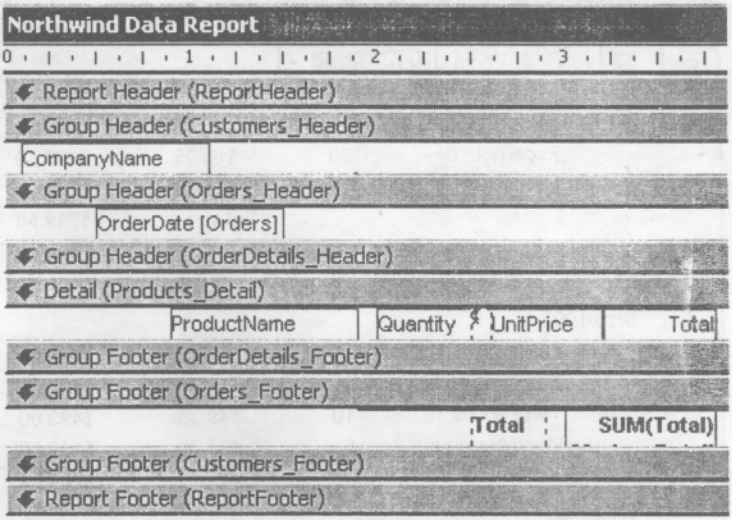



**Hình IX. 21**

Thuộc tính **FunctionType** của control xác định loại toán tử nào sẽ được sử dụng với dữ liệu trong **DataField**. Các toán tử có thể là: Trung bình (Average), Nhỏ nhất (Minimum), Lớn nhất (Maximum), Đếm số lượng(Row Count), Độ lệch tiêu chuẩn (STDDev (Standard Deviation)), and Đếm số lượng các giá trị (Value Count).

5. Vẽ một đường thẳng bằng cách dùng control **Line** ở phía trên text box Quantity, và một control **Label** có thuộc tính **Caption** là **Total** ở bên trái của Label.
6. Click vào control **fncTotal**. Trong cửa sổ Properties, double-click vào **Font** để hiện dialog **Font**. Thay đổi **Font style** thành **Bold**.
7. Lặp lại bước 6 với control Label.

Chỉnh lại kích thước của các control như hình sau:

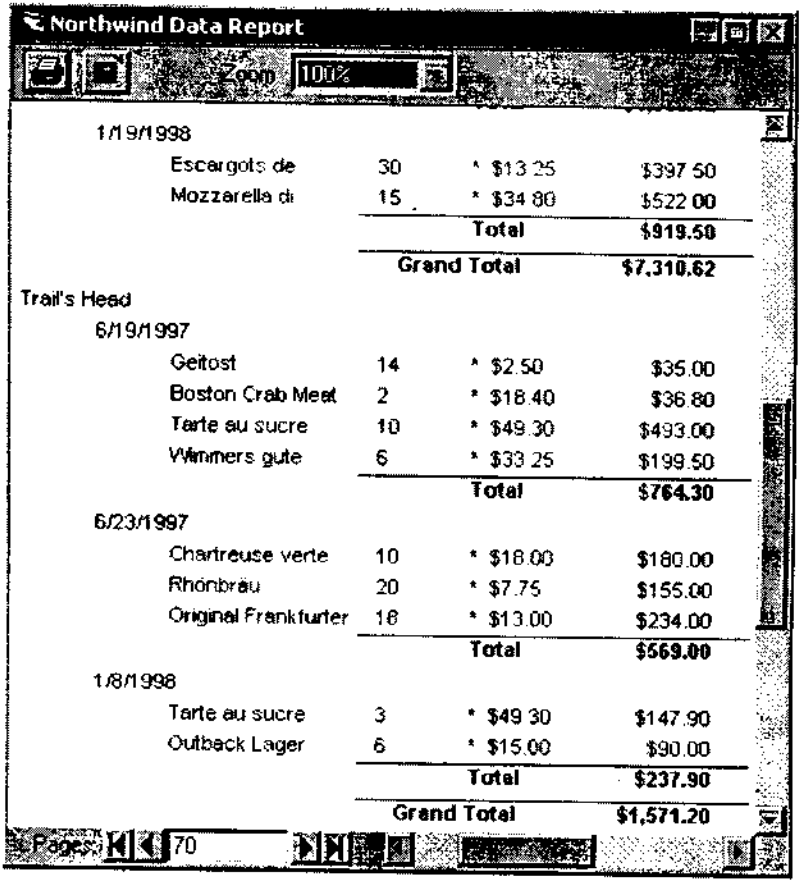



**Hình IX. 21**

8. Lưu và chạy thử project.

## V.2. Sử dụng các Function control phức tạp

Bạn có thể gia tăng sự thuận tiện của function control bằng cách sử dụng các control này trong các group footer thấp hơn. Ví dụ, bạn phải tổng kết số tiền mỗi khách hàng control (customer) cho các đơn đặt hàng.

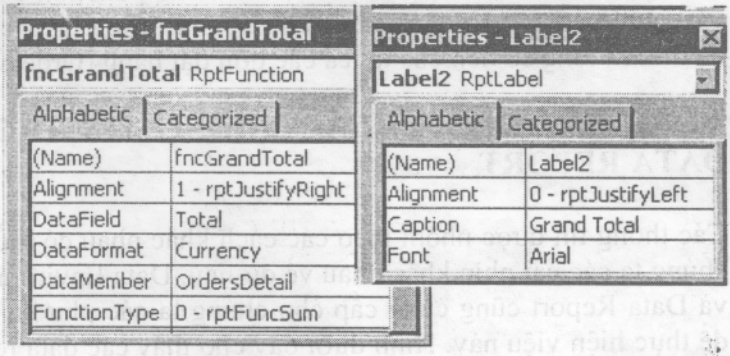



**Hình IX.23**: *Data Report phức tạp*

**Để thêm một "tổng của tổng" ( grand total) cho tất cả các đơn đặt hàng (orders) của một khách hàng**

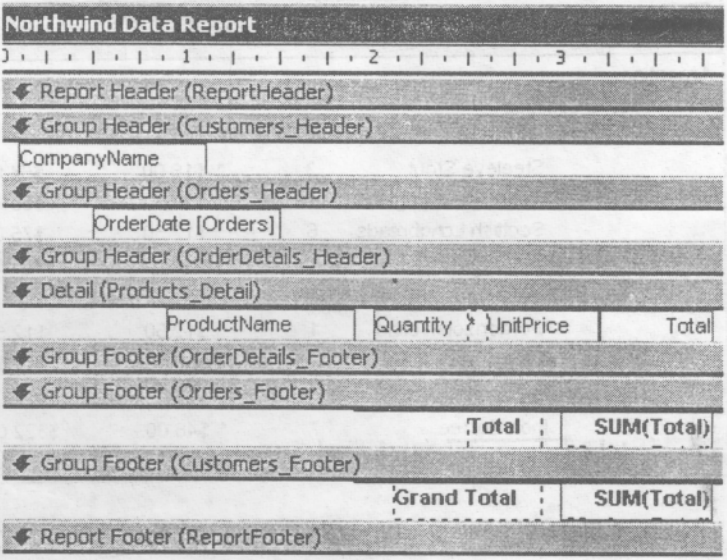
1. Thêm một control **Function** mới vào trong section **Customers_Footer**.
2. Thêm một control **Label** bên trái của **Function** control mới được tạo.

3. Thiết lập của Function control và Label control như sau:




**Hình IX.24**: *Thiết lập các thuộc tính cho Function và Lable Control*

4. Cuối cùng Data Report sẽ có dạng như sau:




**Hình IX.25**: *Chỉnh lại kích thước trong Data Report*

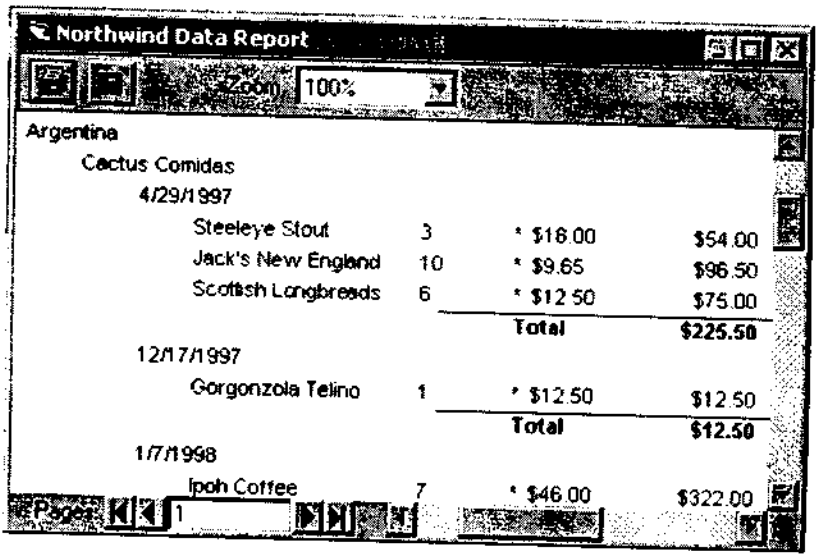
5. Lưu và chạy thử project.

### V.3. Thêm các tổng kết của một Report Total

Tương tự như thế, chúng ta có thể thêm các Function control vào section Report Footer. Bạn có thể làm tương tự như trên để tính ra tổng số tiền của tất cả các đơn đặt hàng (Orders)

## VI. NHÓM (GROUP) CÁC THÔNG TIN TRONG DATA REPORT

Các thông tin được nhóm theo các cách khác nhau có thể cho chúng ta các cái nhìn khác nhau về dữ liệu. Data Environment và Data Report cũng cung cấp cho chúng ta các phương tiện để thực hiện việu này. Hình dưới đây cho thấy các data report được nhóm theo quốc gia (country).




**Hình IX.26**: *Data Report được nhóm theo tên nước ( Country)*

## VI.1. Nhóm bằng Command Field

Việc nhóm các field bằng đối tượng Command của data environment khác với các cách group trong câu lệnh SQL. Lệnh nhóm của Data Environment có thể được sử dụng trong việc tạo thêm các group header và group footer. Thay vì dùng bảng (table) của CSDL như là nhân tố để nhóm, lệnh Grouping của Data Environment cho phép bạn chọn một trường nào đó của một cột trong bảng mà không cần tạo thêm một đối tượng Command.

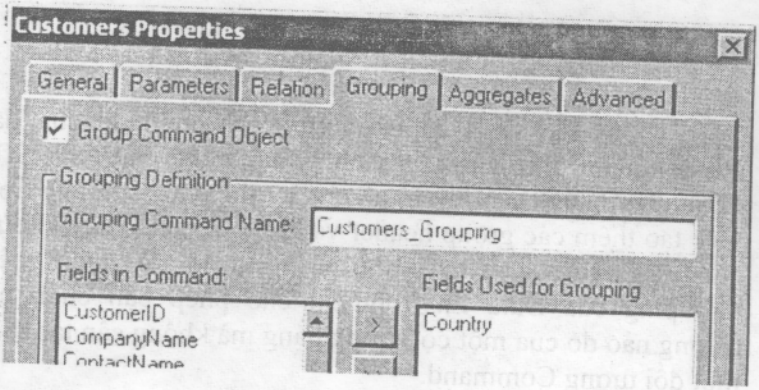
Các bước để tạo grouping field bao gồm:

1. Tạo group field trong Data Environment.
2. Thêm Group Header/Footer vào Data Report để "tương thích" với command mới.
3. Thiết lập lại thuộc tính DataMember của data report bằng đối tượng Grouping Command mới vừa tạo trong data environment.
4. Kéo group field từ data environment vàodata report.

## VI.2. Thêm một group field vào Data Report

1. Trong Data Environment right-click vào command **Customers**. Chọn Properties, dialog **Customers Properties** xuất hiện.

2. Thiết lập các thuộc tính trong tab **Grouping** như hình:




**Hình IX. 27**

Để ý là đối tượng Customers Command sẽ bị đổi tên thành **Customers Grouped using Customers_Grouping**, và có hai folder mới được tạo ra trong Command này. Folder đầu tiên tên là **Summary fields in Customer_Grouping** sẽ chứa các field được nhóm (group). Các field còn lại sẽ được chứa trong thư mục **Detail Fields in Customer_Grouping**.

3. Right-click vào Data Report, và chọn **Insert Group Header/Footer**.

4. Nhấn vào nút ba lần. Thao tác này sẽ đưa group header mới tạo lên trên cùng. Nhấn **OK** để kết thúc.

5. Trên Data Report, chọn **Group Header** mới tạo, thay đổi tên nó thành **Customers_Grouping_Header** (bằng cách vào cửa sổ **Properties**). Tương tự thay đổi tên của **Group Footer** mới bằng **Customers_Grouping_Footer**.

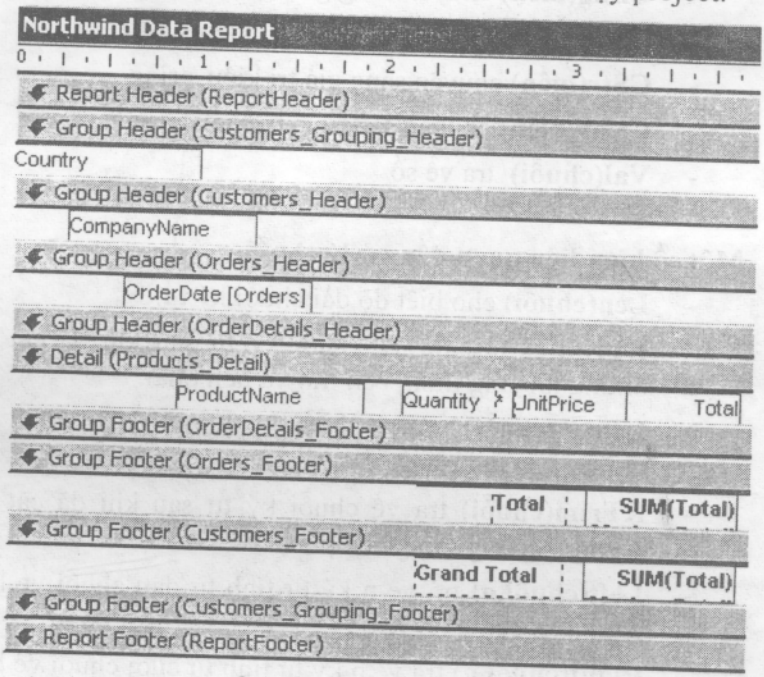
6. Chọn cả Data Report bằng cách click vào title bar của data report. Trong cửa sổ **Properties**, thay đổi thuộc

tính **DataMember** từ Customers thành **Customers_Grouping**.

Khi grouping field đực tạo ra trong data environment, đối tượng Command đối tượng tương ứng mới cũng được thêm vào Data report. Đối tượng ảo đó cũng được tự động thêm vào listbox data members dưới tên **Customers_Grouping**.

7. Trong Data Environment, mở **Summary Fields in Customers_Grouping**. Và kéo field **Country** vào section mới của Data Report.

8. Xóa control **Label** mà nó đi kèm với field Country. Chỉnh lại kích thước các section, lưu và chạy project.




**Hình IX.28:** *Chỉnh lại kích thước của Data Report*

# PHỤ LỤC

**Một số hàm chuyển đổi kiểu**

- **CBool(biến)** chuyển sang giá trị kiểu boolean.
- **Cbyte(biến)** chuyển sang giá trị kiểu byte.
- **CCur(biến)** chuyển sang giá trị kiểu currency.
- **CDate(biến)** chuyển sang giá trị kiểu date.
- **CDbl(biến)** chuyển sang giá trị kiểu double.
- **CDec(biến)** chuyển sang giá trị kiểu decimal.
- **CInt(biến)** chuyển sang giá trị kiểu int.
- **CLng(biến)** chuyển sang giá trị kiểu long.
- **CSng(biến)** chuyển sang giá trị kiểu single.
- **CStr(biến)** chuyển sang giá trị kiểu string.
- **CVar(biến)** chuyển sang giá trị kiểu varian.
- **Val(chuỗi)** trả về số

**Một số hàm liên quan đến xử lý chuỗi**

- **Len(chuỗi)** cho biết độ dài chuỗi ký tự.
- **LTrim(chuỗi)** trả về chuỗi ký tự sau khi đã cắt bỏ những khoảng trắng bên trái và bên phải.
- **LTrim(chuỗi)** trả về chuỗi ký tự sau khi đã cắt bỏ những khoảng trắng bên trái.
- **RTrim(chuỗi)** trả về chuỗi ký tự sau khi đã cắt bỏ những khoảng trắng bên phải.
- **Left(chuỗi,n)** trả về n ký tự tính từ đầu chuỗi về bên phải n ký tự.
- **Right(chuỗi,n)** trả về n ký tự tính từ cuối chuỗi về bên trái n ký tự.

- **Mid(chuỗi,vị-trí-bắt-đầu, n)** trả về n ký tự tính từ vị-trí-bắt-đầu trong chuỗi về bên phải n ký tự.
- **Instr(start, chuỗi1, chuỗi2, tùy-chọn)** hàm kiểm tra chuỗi 2 có phải là con của chuỗi 1 hay không (bắt đầu so sánh chuỗi tại vị trí start trong chuỗi2), tùy-chọn có giá trị =0 (nếu so sánh chính xác từng kí tự một) hoặc tùy-chọn có giá trị = 1 (so sánh không phân biệt chữ hoa hay chữ thường)
- **UCase(chuỗi)** trả về chuỗi chữ in hoa.
- **LCase(chuỗi)** trả về chuỗi chữ thường.
- **Asc(chuỗi)** trả về mã ASCII của ký tự đầu tiên trong chuỗi.
- **Chr(mã ASCII)** trả về ký tự tương ứng với mã ASCII.
- **Space(n)** trả về chuỗi gồm n khoảng trắng.
- **Str(số)** trả về chuỗi ký tự từ số đưa vào.

**Một số hàm xử lý tính toán**

- **Abs(số)** trả về trị tuyệt đối của một số.
- **Sin(số)** trả về sin của một góc (số).
- **Cos(số)** trả về cos của một góc (số).
- **Tan(số)** trả về tang của một góc.
- **Int(số)** trả về giá trị nguyên của một số.
- **Fix(số)** trả về phần nguyên của một số.
- **Sgn(số)** kết quả trả về cho biết dấu của một số nguyên (nếu là số dương: trả về 1, không: 0, âm: -1).
- **Sqr(số)** trả về căn bậc 2 của một số.

# MỤC LỤC


## CHƯƠNG I: TỔNG QUAN VỀ VISUAL BASIC ..... 5

I. KHỞI ĐỘNG CHƯƠNG TRÌNH VISUAL BASIC ..... 5

II. TẠO THƯ MỤC LÀM VIỆC ..... 6

III. GIỚI THIỆU GIAO DIỆN CỬA SỔ CỦA VISUAL BASIC ..... 6
III.1. Thanh tiêu đề (Title Bar) ..... 7
III.2. Thanh thực đơn (Menu Bar) ..... 8
III.3. Thanh công cụ (Tool Bar) ..... 9

IV. TẠO / LƯU PROJECT LÀM VIỆC ..... 10
IV.1. Tạo project ..... 10
IV.2. Lưu project ..... 13

V. LÀM VIỆC VỚI CỬA SỔ PROJECT ..... 14

VI. GIỚI THIỆU TỔNG QUAN VỀ FORM ..... 16
VI.1. Lập trình trực quan ..... 16
VI.2. Giới thiệu form ..... 18
VI.3. Cách tạo form ..... 18
VI.4. MDI Form (Multiple Document Interface) ..... 20
VI.5. Lưu và chạy form ..... 22

## CHƯƠNG II: FORM VÀ CÁC CONTROL TRONG VISUAL BASIC ..... 27

I. GIỚI THIỆU ..... 27

**II. CÁC THAO TÁC CƠ BẢN TRÊN FORM .....30**
II.1. Thêm một control vào form .....30
II.2. Chọn nhiều control.....30
II.3. Thay đổi kích thước, di chuyển, xóa ... control .....30
II.4. Thiết lập lại thuộc tính cho các control.....31
II.5. Thứ tự Tab của các control .....31

**III. CÁC THUỘC TÍNH CHUNG .....31**
III.1. Thuộc tính Name .....31
III.2. Thuộc tính định dạng.....31
III.3. Thuộc tính giá trị (Value) .....32

**IV. GIỚI THIỆU CÁC CONTROL CƠ BẢN TRONG TOOLBOX.....33**
IV.1. Pointer (Đối tượng lựa chọn).....33
IV.2. Label.....33
IV.3. TextBox .....34
IV.4. CommandButton .....38
IV.5. OptionButton .....38
IV.6. CheckBox .....39
IV.7. Frame.....40
IV.8. ListBox .....40
IV.9. ComboBox.....45
IV.10. VScrollBar và HScrollBar.....47
IV.11. Timer .....47
IV.12. DriveListBox .....48
IV.13. FileListBox.....48
IV.14. Line.....48
IV.15. Shape .....48
IV.16. PictureBox.....48
IV.17. Image.....49
IV.18. Data .....50

**CHƯƠNG III: THIẾT KẾ MENU.....51**

I. THIẾT KẾ GIAO DIỆN.....56
I.1. Các bước tiến hành.....56
I.2. Một số hiệu ứng đặc biệt trên menu.....65

II. VIẾT LỆNH CHO CÁC THỦ TỤC ĐÁP ỨNG BIẾN CỐ CHỌN MỤC MENU.....72

CHƯƠNG IV: SỰ KIỆN TRÊN FORM.....76

I. SỰ KIỆN TRÊN FORM.....76

II. SỰ KIỆN TRÊN NHÃN (LABEL).....78

III. SỰ KIỆN TRÊN HỘP VĂN BẢN (TEXTBOX).....79

IV. SỰ KIỆN TRÊN NÚT (BUTTON).....80

V. SỰ KIỆN TRÊN HỘP KIỂM TRA (CHECK BOX)..81

VI. SỰ KIỆN TRÊN NÚT TÙY CHỌN (OPTION BUTTON).....85

VII. SỰ KIỆN TRÊN HỘP LIỆT KÊ (LIST BOX).....87

VIII. SỰ KIỆN TRÊN HỘP LIỆT KÊ THẢ (COMBOBOX).....88

CHƯƠNG V: VIẾT LỆNH.....90

I. GIỚI THIỆU.....90
I.1. Ý nghĩa.....90
I.2. Các thành phần và khái niệm.....91

II. CÁC KIỂU DỮ LIỆU TRONG VISUAL BASIC.....91
II.1. Kiểu dữ liệu Variant.....92

II.2. Giá trị rỗng ..... 94
II.3. Giá trị NULL ..... 94
II.4. Kiểu dữ liệu Chuỗi ..... 94
II.5. Kiểu dữ liệu Ngày ..... 95
II.6. Kiểu dữ liệu do người dùng định nghĩa ..... 95
II.7. Hằng số ..... 97
II.8. Mảng ..... 97

**III. KHAI BÁO BIẾN ..... 98**
III.1. Khai báo biến ..... 98
III.2. Từ khóa Public ..... 100
III.3. Từ khóa Private ..... 100
III.4. Từ khóa Static ..... 100
III.5. Từ khóa Option Explicit ..... 101

**IV. CÁC CẤU TRÚC ĐIỀU KHIỂN TRONG VISUAL BASIC ..... 101**
IV.1. Cấu trúc If ...then ..... 101
IV.2. Cấu trúc Select Case ..... 103
IV.3. Cấu trúc lặp For ..... 104
IV.4. Cấu trúc lặp Do ... Loop ..... 105

**V. HÀM VÀ THỦ TỤC ..... 107**
V.1. Giới thiệu ..... 107
V.2. Khai báo ..... 108
V.3. Tham số của hàm và thủ tục ..... 108
V.4. Gọi một hàm hay thủ tục trong cùng Module ..... 110
V.5. Gọi một hàm hay thủ tục khác Module ..... 111

**CHƯƠNG VI: DEBUG ..... 112**

**I. GIỚI THIỆU ..... 112**
I.1. Ý nghĩa ..... 112
I.2. Các loại lỗi ..... 112

II. DESIGN TIME, RUN TIME, VÀ BREAK MODE ..114
II.1. Xác định trạng thái hiện thời ..... 114
II.2. Dùng Toolbar để thay đổi chế độ làm việc ..... 116

III. CÁCH SỬ DỤNG CÁC CÔNG CỤ DEBUG ..... 118
III.1. Debug Toolbar ..... 118
III.2. Dùng Breakpoint ..... 120
III.3. Cửa sổ Debug ..... 122

CHƯƠNG VII: KẾT NỐI CƠ SỞ DỮ LIỆU VỚI ADO ..... 127

I. GIỚI THIỆU KỸ THUẬT LẬP TRÌNH ADO ..... 128

II. SỬ DỤNG ADO DATA CONTROL ..... 130

III. SỬ DỤNG OBJECT INTERFACE ..... 147
III.1. Đối tượng Connection ..... 148
III.2. Đối tượng Recordset ..... 150
III.3. Tạo một form ứng dụng ..... 158

CHƯƠNG VIII: MỘT SỐ ĐIỀU KHIỂN KẾT BUỘC DỮ LIỆU ..... 168

I. DATACOMBO VÀ DATALIST ..... 168
I.1. Giới thiệu DataCombo và DataList ..... 168
I.2. Chương trình ứng dụng ..... 174

II. DATAGRID ..... 182
II.1. Thiết lập DataGrid trên form ..... 183
II.2. Cửa sổ Properties Page của DataGrid ..... 185
II.3. Kết nối DataGrid với nguồn dữ liệu ..... 187
II.4. Chương trình ứng dụng ..... 189

CHƯƠNG IX: REPORT ..... 201

I. MỞ ĐẦU ..... 201
I.1. Giới thiệu ..... 201
I.2. Các đối tượng của Data Report ..... 201
I.3. Phân loại section trong Data Report ..... 202
I.4. Các Control của Data Report ..... 203

II. VÍ DỤ 1 ..... 204
II.1. Tạo một con trỏ phân cấp đơn giản trong Data Environment designer ..... 205
II.2. Tạo mới một data report ..... 209
II.3. Xem Data Report ..... 212

III. VÍ DỤ 2 ..... 214
III.1. Mở rộng Data Environment ..... 214
III.2. Mở rộng Data Report ..... 216

IV. THÊM MỘT FIELD TÍNH TOÁN (CALCULATED FIELD) VÀO DATA REPORT ..... 219

V. THÊM MỘT FUNCTION CONTROL VÀO DATA REPORT ..... 222
V.1. Cách tạo Data Report sử dụng Function Control để hiển thị tổng cộng ..... 223
V.2. Sử dụng các Function control phức tạp ..... 225
V.3. Thêm các tổng kết của một Report Total ..... 228

VI. NHÓM (GROUP) CÁC THÔNG TIN TRONG DATA REPORT ..... 228
VI.1. Nhóm bằng Command Field ..... 229
VI.2. Thêm một group field vào Data Report ..... 229

Giáo trình

# LẬP TRÌNH CƠ SỞ DỮ LIỆU VỚI VISUAL BASIC

*Nguyễn Văn Toàn – Nguyễn Thị Kim Phụng – Nguyễn Đình Loan Phương*

**NHÀ XUẤT BẢN**
**ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH**
KP 6, P. Linh Trung, Q. Thủ Đức, TPHCM
**ĐT**: 724 2181 + 1421, 1422, 1423, 1425, 1426
**Fax**: 724 2194; **Email**: vnuhp@vnuhcm.edu.vn

☆☆☆

*Chịu trách nhiệm xuất bản*
*PGS TS* NGUYỄN QUANG ĐIỂN

*Biên tập*
NGUYỄN VIỆT HỒNG

*Sửa bản in*
TRẦN VĂN THẮNG

*Trình bày bìa*
XUÂN THẢO

In 500 cuốn, khổ 14.5 x 20,5 cm tại Nhà in ĐHQG TPHCM. Giấy CNĐKKHXB số 140/233/XB-QLXB do Cục Xuất bản cấp ngày 03/3/2005. Giấy trích ngang KHXB số 347/ KHXB của NXB ĐHQG TPHCM ký ngày 24/6/2005. In xong và nộp lưu chiểu tháng 7/2005.




**Sách trợ giá dành cho SV ĐHQG – HCM**
**Giá: 15.000 đ**