

Giáo trình tin học:

Lập trình với Microsoft Visual Basic 6.0

Tác giả: Nguyễn Sơn Hải (nshai@moet.edu.vn)

Cục Công nghệ thông tin – Bộ Giáo dục và Đào tạo

Hà Nội, tháng 6 - 2006

GIỚI THIỆU

Lập trình ứng dụng là môn học rất quan trọng đối với sinh viên ngành công nghệ thông tin (CNTT). Hiện nay có rất nhiều các ngôn ngữ lập trình chuyên nghiệp có thể sử dụng tốt để lập trình các ứng dụng như: Microsoft .NET, C++ Builder, Delphi, v.v. Nhưng Visual Basic 6.0 (VB6), từ đây gọi tắt là VB có thể coi là một trong những công cụ dễ học nhất.

Giáo trình này viết ra phục vụ nhu cầu học môn Lập trình ứng dụng hoặc Lập trình Visual Basic cho các sinh viên, học sinh trường THPT hệ chuyên CNTT. Sau những nội dung về lập trình VB căn bản, giáo trình sẽ hướng học sinh đến các kỹ thuật lập trình cơ sở dữ liệu (CSDL) trên VB. kết quả cuối cùng là tạo ra các ứng dụng quản lý sử dụng VB như là một công cụ phát triển trên hệ CSDL Access.

Thời lượng thiết kế 75 tiết. Trong đó 24 tiết lý thuyết, 46 tiết thực hành và 5 tiết dành cho kiểm tra. Tùy theo mức độ ứng dụng của từng trường, từng ngành vào môn học này mà mỗi đơn vị sẽ chọn cho mình một quỹ thời gian cũng như các nội dung giảng dạy phù hợp từ giáo trình này.

Nội dung giáo trình gồm một bài mở đầu và 4 chương:

Bài mở đầu

Bài mở đầu sẽ là những lời giới thiệu về ngôn ngữ Visual Basic 6.0, về môi trường làm việc và lịch sử của ngôn ngữ này. Kết thúc bài học, học viên sẽ hiểu được môi trường làm việc, tạo và làm việc trên project đơn giản.

Chương 1: Lập trình VB căn bản

Chương này cung cấp những khái niệm, những cách thức căn bản nhất khi làm việc với bất kỳ một ngôn ngữ lập trình nào, ở đây là VB. Đó là biến, hằng, các cấu trúc lệnh và kỹ thuật chương trình con trong VB.

Chương 2: Làm việc với các điều khiển

Điều khiển là một thành phần rất quan trọng trong những ngôn ngữ lập trình trực quan, hướng đối tượng (như VB). Chương này tập trung vào việc giới thiệu những khái niệm, cách thức làm việc và hướng dẫn sử dụng, lập trình trên hệ thống các điều khiển từ căn bản đến nâng cao của VB. Kỹ thuật bắt lỗi và xử lý lỗi cũng được đề cập đến rất chi tiết. Kết thúc nội dung chương này, học viên có thể thiết kế được một số các ứng dụng chạy trên windows bằng cách sử dụng tốt các kiến thức của chương 1 và chương 2.

Chương 3: Lập trình cơ sở dữ liệu

Hầu hết lập trình ứng dụng đều liên quan đến các kỹ thuật xử lý CSDL. Chương này tập trung giới thiệu hai công nghệ lập trình CSDL phổ biến là DAO (Data Access Objects) và ADO (ActiveX Data Objects) thông qua các bài toán cụ thể như: nhập dữ liệu, tìm kiếm và đặt lọc, in báo cáo.

Chương 4: Hoàn thiện dự án

Chương này tập trung hướng dẫn các qui trình, công cụ và kỹ thuật để thiết kế giao diện chính ứng dụng, cũng như hoàn thiện, đóng gói một dự án hoàn chỉnh trên môi trường VB. Đó là sản phẩm phần mềm cuối cùng gửi tới khách hàng. Kỹ thuật MDI Form, Menu, ToolBar cũng như qui trình đóng gói thông qua tiện ích Package and Deployment Wizard được hướng dẫn sử dụng chi tiết.

Phần cuối giáo trình có đưa ra danh mục các tài liệu để bạn đọc tham khảo; danh mục các từ viết tắt được sử dụng trong sách và giải nghĩa một số thuật ngữ tiếng Anh thường gặp khi lập trình trên VB.

Cuối cùng, chúng tôi xin chân thành cảm ơn sự quan tâm và tạo điều kiện của Sở GD&ĐT Hà Nội; các đồng nghiệp đã tận tình giúp đỡ, đóng góp ý kiến xây dựng để giáo trình được hoàn thiện như bây giờ. Tuy nhiên rất khó tránh khỏi những thiếu sót từ khách quan đến chủ quan. Chúng tôi rất mong nhận được sự góp ý, phê bình để giáo trình ngày càng hoàn thiện hơn, phục vụ tốt cho việc dạy-học trong nhà trường và các bạn đọc.

Hà nội, ngày 20 tháng 10 năm 2005

BÀI MỞ ĐẦU

Nội dung bài học này cung cấp cái nhìn tổng quan về ngôn ngữ Visual Basic: tính năng, môi trường và cách thức làm việc. Tuy không nặng về kiến thức lập trình Visual Basic, nhưng nội dung bài học này là rất quan trọng giúp học sinh tiếp cận ngôn ngữ này một cách thuận lợi. Đặc biệt, nếu đây là lần đầu tiên học kỹ thuật lập trình trên windows.

1. Giới thiệu Visual Basic 6.0

Visual Basic 6.0 (VB) là một ngôn ngữ lập trình hướng đối tượng, trực quan trên môi trường Windows. VB cung cấp một bộ công cụ hoàn chỉnh để đơn giản hóa việc triển khai lập trình ứng dụng, có thể nói đây là cách nhanh và tốt nhất để học và lập trình ứng dụng trên Microsoft Windows.

Phần "Visual- Trực quan" đề cập đến phương pháp được sử dụng để tạo giao diện đồ họa người dùng (GUI - Graphical User Interface). VB có sẵn rất nhiều những bộ phận trực quan gọi là các điều khiển (Controls) mà người lập trình có thể sắp đặt vị trí và quyết định các đặc tính của chúng trên một khung giao diện màn hình, gọi là form. Việc thiết kế các giao diện người dùng ứng dụng trên VB có thể hình dung đơn giản như việc vẽ giao diện trên Word hoặc trên Paint Prush của Windows.

Phần "Basic" đề cập đến ngôn ngữ BASIC (Beginners All-Purpose Symbolic Instruction Code), một ngôn ngữ lập trình đơn giản, dễ học, được viết ra cho các khoa học gia- những người không có thì giờ để học lập trình điện toán sử dụng.

Tuy nhiên, ngôn ngữ Basic trong VB đã được cải thiện rất nhiều để phù hợp với phong cách lập trình hiện đại.

Visual Basic còn có hai dạng khác là Visual Basic for Application (VBA) - một ngôn ngữ nằm phía sau các chương trình Word, Excel, VB, Project, .v.v.. còn gọi là Macros. Dùng VB trong Microsoft Office, ta có thể làm tăng chức năng các ứng dụng bằng cách tự động hóa các chương trình. Và VBScript được dùng lập trình phục vụ các tương tác trên giao diện web.

Visual Basic đã có rất nhiều phiên bản, 2 phiên bản tốt nhất có thể nói đến là Visual Basic 6.0 (VB6) và Visual Basic .NET (VB7 hay VB.NET). Về mặt kiến trúc, hai phiên bản này gần khác nhau hoàn toàn. VB6 phát triển ứng dụng dựa trên công nghệ COM (Common Object Model)- một công nghệ rất phát triển ít nhất cho đến năm 2000. Còn VB.NET dựa trên nền tảng công nghệ .NET Framework - một công nghệ hiện đại hơn và đang rất được ưa chuộng. Giáo trình này chỉ đề cập đến việc sử dụng và phát triển phần mềm ứng dụng trên phiên bản VB6 (gọi tắt là VB). Bởi lẽ phiên bản này rất dễ học và phát triển. Việc tìm hiểu ngôn ngữ VB.NET là rất khuyến khích cho những ai đang muốn tìm cho mình một bộ công cụ phát triển chuyên nghiệp trên đa môi trường hoạt động. Tuy nhiên, khi nắm chắc những nội dung VB6 từ giáo trình này, bạn đọc đã có thể sẵn sàng tiếp cận VB.NET với tư thế rất thuận lợi.

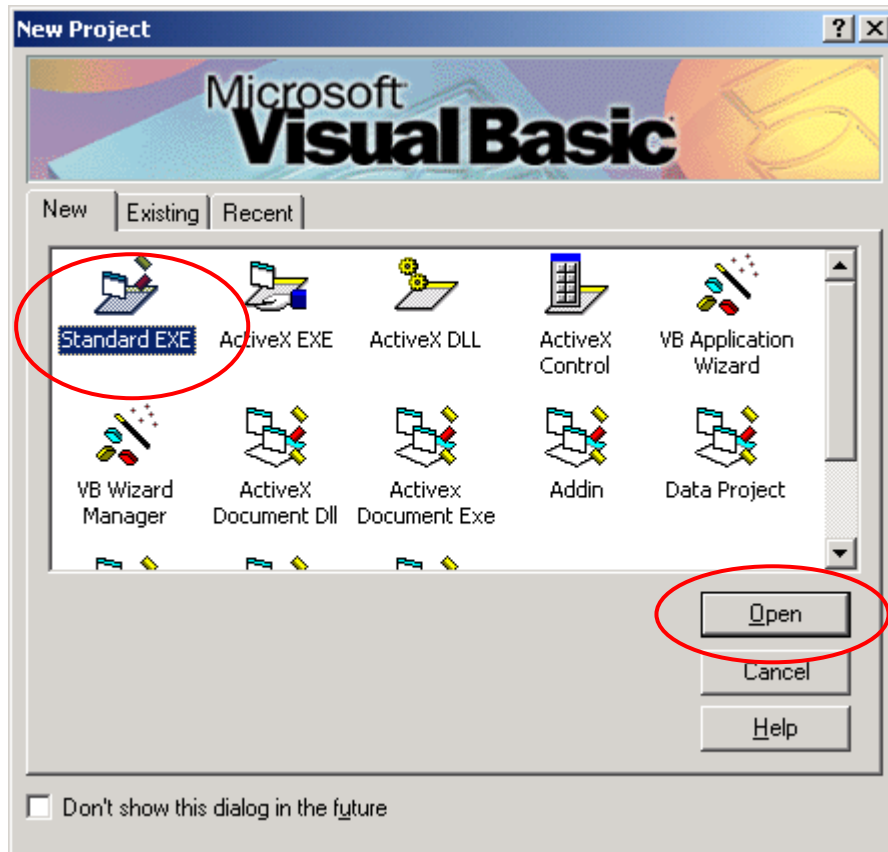
Để cài đặt VB, yêu cầu tối thiểu phải có đĩa 1 - bộ Microsoft Visual Studio 6.0 (bộ này đầy đủ gồm 4 đĩa). Tuy nhiên cũng nên gài cả đĩa 2 và 3 để có được đầy đủ các tài nguyên đi cùng giúp việc nghiên cứu và tìm hiểu VB được thuận lợi.

2. Khởi động

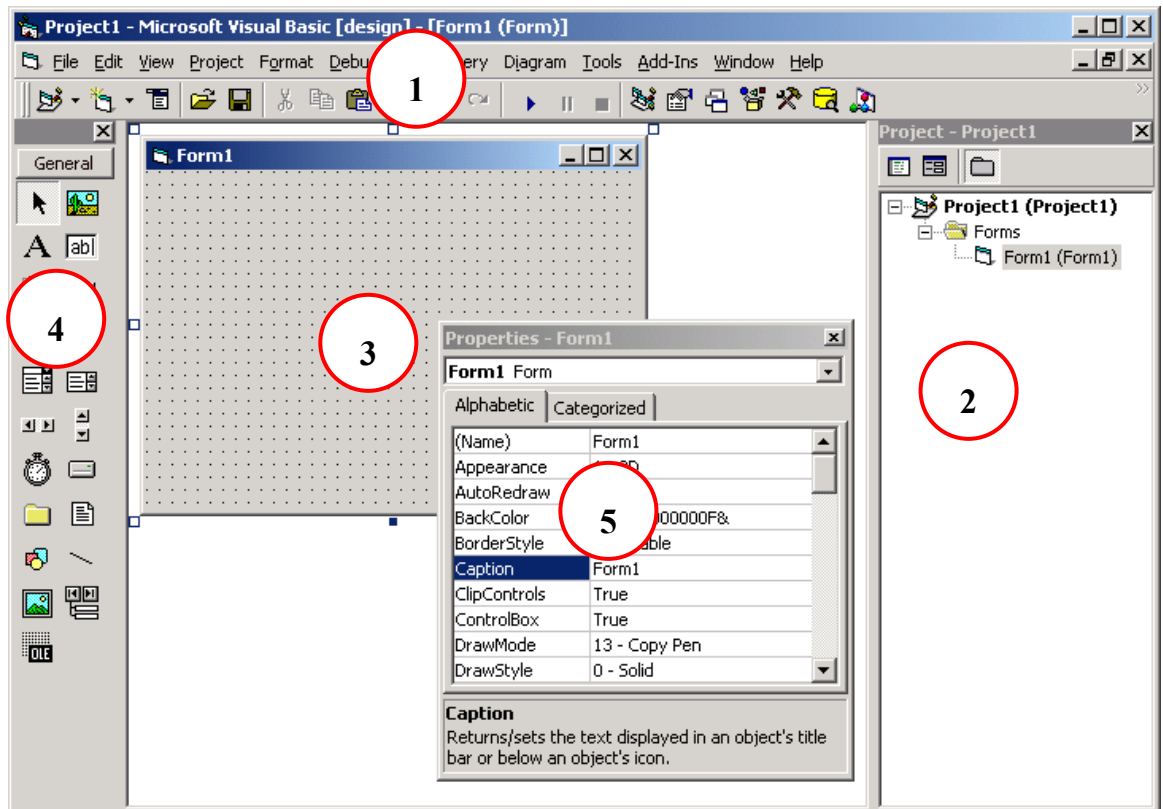
Sau khi cài đặt VB, có thể khởi động từ thanh tác vụ của Windows như sau:

Start | Programs | Microsoft Visual Studio 6.0 | Microsoft Visual Basic 6.0

Hộp thoại đầu tiên của phần mềm xuất hiện:



Để bắt đầu một ứng dụng mới, từ thẻ **New**, chọn **Standard EXE**, nhấn **Open**.
Môi trường làm việc VB xuất hiện:



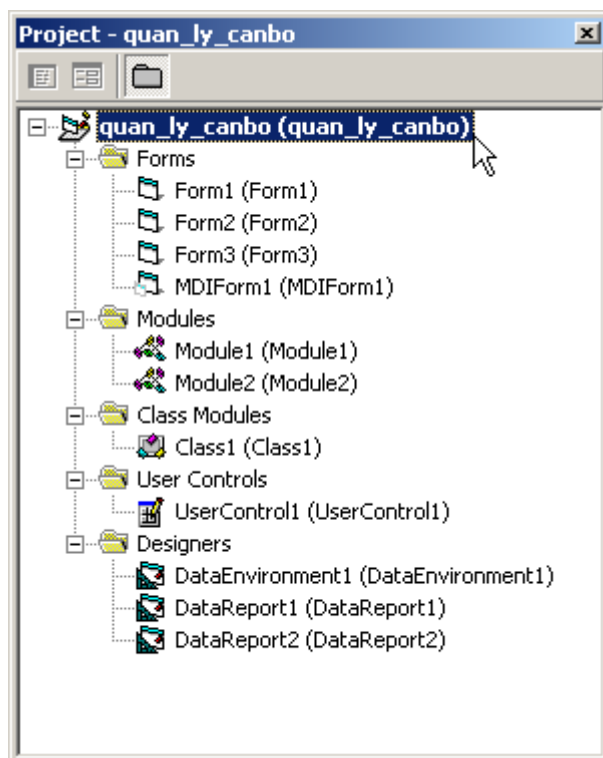
Có rất nhiều các thành phần trong môi trường làm việc của VB. Ở mức đơn giản nhất có 5 thành phần được khoanh tròn trong hình trên đó là:

- (1). Thanh thực đơn và thanh công cụ chuẩn của VB;
- (2). Cửa sổ Project Explorer – nơi quản lý toàn bộ các thành phần mà người lập trình đã làm được trên dự án của VB hiện thời. Làm việc trên VB là làm việc trên các dự án (Projects). Mỗi dự án cần phải tạo ra nhiều thành phần để cấu thành như: giao diện, biểu mẫu báo cáo, thư viện,... tất cả những thành phần này sẽ được quản lý trên cửa sổ Project Explorer;
- (3). Biểu mẫu Form – nơi thường dùng để thiết kế các hộp thoại, cửa sổ - giao diện của người sử dụng với ứng dụng phần mềm;
- (4). Thanh công cụ Toolbox- nơi chứa các điều khiển (Control) giúp người lập trình dễ dàng tạo ra những giao diện thân thiện và lập trình trên chúng một cách thuận lợi, đa năng;
- (5). Cửa sổ Properties – nơi để thiết lập các thuộc tính cho những đối tượng, những điều khiển trong quá trình làm việc trên VB.

3. Lưu trữ

Làm việc trên VB là làm việc trên các dự án (Project). Tại một thời điểm có thể chỉ làm việc với một dự án nào đó, cũng có khi làm việc trên một tập hợp các dự án (Project Group). Tuy nhiên khuôn khổ giáo trình này chỉ nói về làm việc trên một dự án đơn lẻ.

Dự án là công cụ quản lý tất cả những gì cần phải tạo ra cho một dự án phần mềm viết trên VB. Hình dưới mô tả các nội dung có thể được quản lý bởi một dự án mang tên **quan_ly_canbo**



Trong dự án trên có các thành phần:

Forms – để tạo ra các giao diện người sử dụng phần mềm như là các hộp thoại, biểu nhập dữ liệu, cửa sổ giao diện. Có 4 form được tạo ra trong dự án trên.


Modules – là nơi chứa những thư viện khai báo phục vụ việc phát triển phần mềm. Trong mỗi Module có thể chứa các chương trình con, các khai báo biến, hằng, môi trường làm việc mà các thành phần thư viện này có thể dùng riêng hoặc chia sẻ dùng chung trong toàn bộ dự án. Có 2 module được tạo ra trong dự án trên là Module1 và Module2.

Class Modules – nơi tạo ra các lớp đối tượng do người lập trình tự định nghĩa phục vụ các nhu cầu phát triển riêng. Dự án trên có một tệp lớp là Class1.

User Controls – nơi cho phép người lập trình tự định nghĩa ra các điều khiển phục vụ mục đích công việc riêng để phát triển trong dự án. Dự án trên có một đối tượng điều khiển tự định nghĩa là UserControl1.

Designers – nơi tạo ra các môi trường dữ liệu (data environment) và các báo biểu (Data report) phục vụ nhu cầu xử lý, truy xuất và in ấn dữ liệu trong dự án.

Không chỉ dừng lại ở đây, ứng với mỗi dự án trên VB có thể cần tạo ra những đối tượng riêng. Và chúng có thể được quản lý trên cửa sổ Project Explorer.

Để ghi lại một dự án, nhấn thực đơn **File | Save** hoặc nút **Save**  trên thanh công cụ hoặc nhấn tổ hợp phím nóng **Ctrl + S**. VB sẽ lần lượt yêu cầu nhập vào tên tệp tin của các đối tượng đã tạo được trên dự án (việc đặt tên này chỉ xuất hiện ở lần ghi đầu tiên). Tệp tin chính của dự án có phần mở rộng là **.vbp** và biểu tượng như sau:




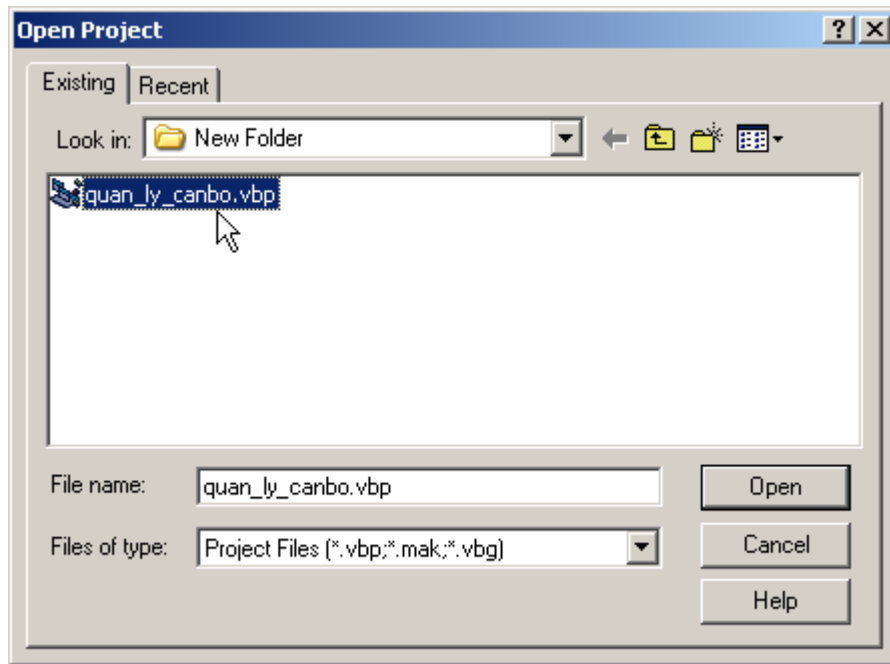
 **Chú ý:**

Do mỗi dự án VB sẽ bao gồm rất nhiều tệp tin, chúng luôn phải được đi cùng nhau. Do vậy ứng với mỗi dự án mới hãy tạo một thư mục và ghi lại toàn bộ những gì làm được và liên quan lên thư mục này. Điều này rất thuận lợi khi phải di chuyển dự án đến nhiều máy, nhiều vị trí khác nhau để làm việc.

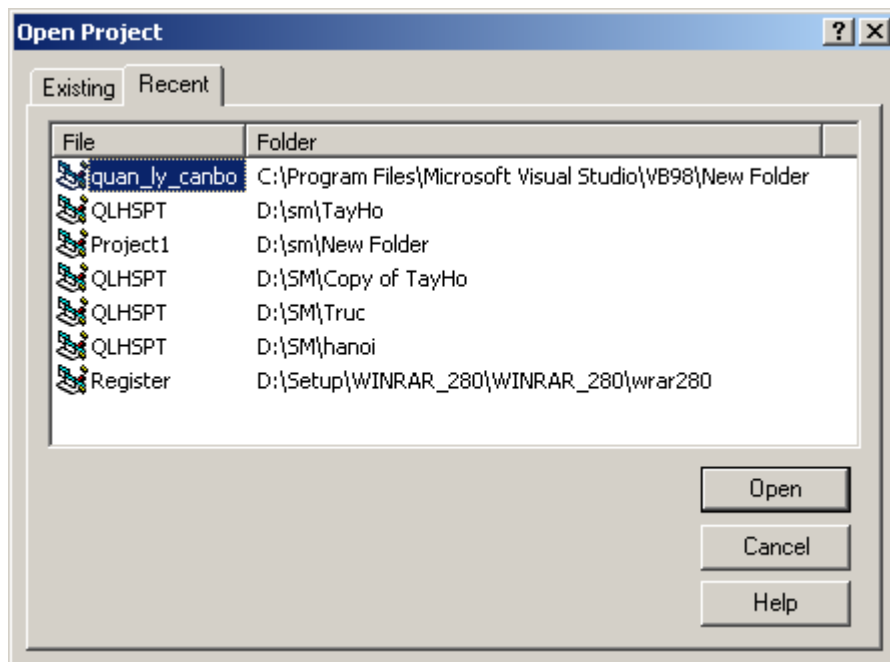
4. Mở dự án đã có

Để mở một dự án ra làm việc trên VB thao tác như sau:

Nhấn nút **Open**  trên thanh công cụ, hoặc nhấn tổ hợp phím tắt **Ctrl + O**, một hộp thoại xuất hiện để chọn dự án cần mở:



Ở thẻ **Existing** có thể tìm đến dự án cần mở và nhấn nút **Open**
Hoặc



Ở thẻ **Recent** có thể chọn dự án từ danh sách làm việc gần nhất để mở.

Sau thao tác này, dự án cần làm việc sẽ được mở ra trên môi trường làm việc của VB.

CHƯƠNG 1

LẬP TRÌNH VB CĂN BẢN

Chương này tập trung giới thiệu chi tiết các thành phần liên quan đến lập trình căn bản trên VB. Đó là những điều kiện tối thiểu, cần thiết nhất để học bất kỳ một ngôn ngữ lập trình nào. Cụ thể học viên sẽ nắm được:

- Các kiểu dữ liệu trong VB;
- Sử dụng biến và hằng;
- Các cấu trúc lập trình căn bản;
- Kỹ thuật chương trình con;
- Cách thức soạn thảo chương trình;
- Kỹ thuật bắt lỗi và xử lý lỗi trên VB.

1. Kiểu dữ liệu - biến và hằng

1.1 Kiểu dữ liệu

Cũng như các ngôn ngữ lập trình khác, VB đều hỗ trợ các kiểu dữ liệu cơ bản. Dưới đây giới thiệu chi tiết về từng kiểu.

Boolean

Kiểu lô gíc, tương tự kiểu Boolean trên Pascal. Kiểu này chiếm 2 byte bộ nhớ; chỉ nhận một trong 2 giá trị là: Yes – No hoặc True – False hoặc đôi khi thể hiện dưới dạng số 0 tương đương với False, True tương ứng với bất kỳ số nào khác 0. Khi lập trình CSDL, kiểu Boolean tương ứng với kiểu Yes/No trong bảng dữ liệu.

Byte

Kiểu số nguyên dương trong phạm vi từ 0..255. Kiểu này chiếm 1 byte bộ nhớ.

Integer

Kiểu nguyên, có giá trị trong khoảng -32768...32767. Kiểu này chiếm 2 bytes bộ nhớ.

Long

Kiểu số nguyên dài, có giá trị trong khoảng 2,147,483,648 .. 2,147,483,647. Kiểu này chiếm 4 bytes bộ nhớ.

Single

Kiểu số thực, có giá trị trong khoảng 1.401298E-45 to 3.402823E38. Chiếm 4 bytes bộ nhớ.

Double

Kiểu số thực có độ lớn hơn kiểu Single, có giá trị trong khoảng 4.94065645841247E-324 to 1.79769313486232E308. Chiếm 8 bytes bộ nhớ.

Currency

Kiểu tiền tệ. Bản chất là kiểu số, độ lớn 8 bytes, có giá trị trong khoảng -922,337,203,685,477.5808 to 922,337,203,685,477.5807. Đặc biệt, kiểu này luôn có ký hiệu tiền tệ đi kèm.

String

Kiểu chuỗi ký tự. Kiểu này tương ứng với kiểu String trong Pascal, tương ứng với kiểu Text trong VB. Độ lớn tối đa 255 bytes tương đương với khả năng xử lý chuỗi dài 255 ký tự.

Variant

Variant là kiểu dữ liệu không tường minh. Biến kiểu này có thể nhận bất kỳ một giá trị nào có thể. Ví dụ :

```
Dim a As Variant  
a = 123  
a = "Nguyễn Văn Ngô"
```

Hoàn toàn không có lỗi.

Người ta thường khai báo biến kiểu Variant trong những trường hợp phải xử lý biến đó mềm dẻo. Khi thì biến nhận giá trị kiểu này, khi thì nhận giá trị và xử lý theo kiểu dữ liệu khác.

Object

Object là một loại biến kiểu Variant, chiếm dung lượng nhớ 4 bytes, dùng để tham chiếu tới một loại đối tượng (Object) nào đó trong khi lập trình. Tất nhiên muốn khai báo biến Object kiểu nào, phải chắc chắn đối tượng đó đã được đăng ký vào thư viện tham chiếu VB bởi tính năng **Project | Reference**.

1.2 Biến

a. Biến – khai báo biến

Biến (Variable) là thành phần của một ngôn ngữ lập trình, giúp xử lý dữ liệu một cách linh hoạt và mềm dẻo.

Thông thường trong các ngôn ngữ lập trình, mỗi biến khi tồn tại phải được định kiểu, tức là phải nhận một kiểu dữ liệu xác định. Tuy nhiên trong VB thì không, mỗi biến có thể định kiểu (được khai báo trước khi sử dụng) hoặc không định kiểu (không khai báo vẫn sử dụng được). Trong trường hợp này biến đó sẽ tự nhận kiểu giá trị Variant.

Biến có thể được khai báo bất kỳ ở đâu trong phần viết lệnh của VB. Tất nhiên, biến có hiệu lực như khai báo chỉ bắt đầu từ sau lời khai báo và đảm bảo phạm vi hoạt động như đã qui định. Vì biến trong VB hoạt động rất mềm dẻo, nên có nhiều cách khai báo biến như:

Ví dụ 1: Khai báo biến i kiểu Integer

```
Dim i As Integer
```

Ví dụ 2: Khai báo 2 biến i, j kiểu Integer

```
Dim i, j As Integer
```

Ví dụ 3: Khai báo biến i kiểu Integer, st kiểu String độ dài 15 ký tự

```
Dim i As Integer, st As String*15
```

Ví dụ 4: Khai báo biến i kiểu Variant

```
Dim i As Variant  
'hoặc  
Dim i
```

Ví dụ 5: Khai báo biến txt kiểu Textbox

```
Dim txt As TextBox
```

Ví dụ 6: Khai báo mảng kiểu String*30 gồm 46 phần tử

```
Dim Hoten(45) As String * 45
```

Ví dụ 7: Khai báo biến mảng 2 chiều A(i, j) trong đó: i = 0..3 và j = 0..4

```
Dim A(3, 4) As Integer
```

Ví dụ 8: Khai báo mảng 3 chiều A(i, j, k) trong đó: i = 1..5; j = 4..9 và k = 3..5

```
Dim A(1 To 5, 4 To 9, 3 To 5) As Double
```

Ví dụ 9: Khai báo một mảng động kiểu Variant. Mảng động là mảng không cố định chiều dài.

```
Dim MyArray()
```

b. Phạm vi biến

Như chúng ta đã biết, mỗi biến sau khi được khai báo nó sẽ nhận một kiểu dữ liệu và có một phạm vi hoạt động, tức là lời khai báo biến chỉ có tác dụng trong những vùng đã được chỉ định; ngoài vùng chỉ định đó biến sẽ không có tác dụng, nếu có tác dụng sẽ theo nghĩa khác (biến cục bộ kiểu Variant chẳng hạn).

Biến cục bộ:

Biến cục bộ được khai báo sau từ khoá **Dim**, nó chỉ có tác dụng trong một chương trình con, cục bộ trong một form hoặc một module nào đó. Dưới đây sẽ chỉ ra 3 trường hợp biến cục bộ này:

- Trong một chương trình con, nếu nó được khai báo trong chương trình con đó;
- Trong cả một Form, nếu nó được khai báo trong phần Decralations của Form đó;
- Trong cả một Reports, nếu nó được khai báo trong phần Decralations của Report đó;
- Trong cả một Modules, nếu nó được khai báo trong phần Decralations của Modules đó;

** Biến chỉ có tác dụng sau lệnh khai báo Dim*

Biến toàn cục:

Biến toàn cục được khai báo sau cụm từ khoá **Public**, nó có tác dụng trong toàn bộ chương trình (ở bất kỳ chỗ nào có thể viết lệnh). Loại biến này luôn phải được khai báo tại vùng *Decralations* của một Module nào đó.

Ví dụ:

```
Public Hoten(45) As String * 45
```

Trên một dự án VB không được phép khai báo trùng tên biến toàn cục. Tuy nhiên tên biến cục bộ vẫn có thể trùng tên biến toàn cục, trong trường hợp đó VB sẽ ưu tiên sử dụng biến cục bộ trong phạm vi của nó.

1.3 Hằng

a. Khai báo hằng

Hằng (Constan) là đại lượng có giá trị xác định và không bị thay đổi trong bất kỳ hoàn cảnh nào. Tương ứng với từng kiểu dữ liệu, sẽ có những hằng tương ứng.

Khai báo hằng số bởi từ khoá **Const**. Sau đây là các ví dụ về khai báo các loại hằng:

Ví dụ 1: Hằng a =5 (hằng số)

```
Const a = 5
```

Ví dụ 2: Hằng ngày = 24/12/2004 kiểu Date (bao bởi cặp dấu thăng #..#)

```
Const ngay = #24/12/2004#
```

Ví dụ 3: Hằng xâu ký tự (bao bởi cặp dấu nháy kép “..”)

```
Const phongban = "Tài vụ"
```

Ví dụ 4: Hằng kiểu Logic xác định bởi *True* hoặc *False*

```
Const ok = True
```

b. Phạm vi hằng

Tương tự như biến, hằng cũng có những phạm vi hoạt động của nó. Hằng được khai báo trong thủ tục nào, hoặc cục bộ trong form, report hoặc module nào sẽ chỉ có tác dụng trong phạm vi đó.

Muốn hằng có phạm vi toàn cục, phải được khai báo sau từ khoá **Public Const**, tại vùng *Declarations* của một module nào đó như sau:

```
Public Const a = 12
```

2. Các cấu trúc lệnh VB

Các cấu trúc lệnh là thành phần cơ bản của mỗi ngôn ngữ lập trình. Thông thường các ngôn ngữ lập trình đều có các cấu trúc lệnh như nhau: lệnh xử lý điều kiện, lệnh lặp biết trước số vòng lặp, lệnh lặp không biết trước số vòng lặp,.. Tuy nhiên cách thể hiện (cú pháp) mỗi cấu trúc lệnh có thể khác nhau tùy thuộc vào mỗi ngôn ngữ lập trình. Hơn nữa, mỗi ngôn ngữ cũng có thể có một số điểm khác biệt, đặc trưng trong mỗi cấu trúc lệnh.

Cũng giống như nhiều ngôn ngữ lập trình hiện đại khác, các cấu trúc lệnh trong VB đều tuân thủ các nguyên tắc:

- Có cấu trúc: mỗi cấu trúc lệnh đều có từ khoá bắt đầu và một từ khóa báo hiệu kết thúc;
- Thực hiện tuần tự (loại trừ trường hợp đặc biệt thủ tục Goto <Label>);
- Có khả năng lồng nhau;

2.1 Cấu trúc IF... END IF

Cấu trúc này thường gọi là *lệnh lựa chọn*. Tức là nếu một điều kiện nào đó xảy ra sẽ là gì, hoặc trái lại có thể làm gì. Trong VB cú pháp lệnh này như sau:

```
If <điều kiện> Then
    <thủ tục 1>
[ Else
    <thủ tục 2> ]
End If
```

Ý nghĩa lệnh trên là: **nếu** <điều kiện> = True **thì** thực hiện các lệnh trong <thủ tục1>, trái lại thực hiện các lệnh trong <thủ tục 2>.

Phần trong cặp dấu ngoặc vuông [...] có thể có hoặc không có trong câu lệnh, tùy thuộc vào mục đích xử lý.

Ví dụ 1: Kiểm tra và trả lời một số là *chẵn* hay *lẻ*?

```
If so Mod 2 = 0 Then
    MsgBox "Là số chẵn !"
Else
    MsgBox "Là số lẻ !"
```

```
End If
```

Cho biết *tháng* (số nguyên) rơi vào đầu năm (1..4), giữa năm (5..8) hay cuối năm (9...12)?

```
If tháng >=9 Then
    MsgBox "Cuối năm "
Else
    If tháng >=5 Then
        MsgBox "Giữa năm "
    Else
        MsgBox "Đầu năm "
    End If
End If
```

2.2 Cấu trúc **SELECT CASE .. END SELECT**

Đây là một loại của cấu trúc lựa chọn. Thông thường hoàn toàn có thể sử dụng `If .. End If` để thực hiện các xử lý liên quan đến kiểu cấu trúc này, nhưng trong những trường hợp đặc biệt, cấu trúc **Select Case .. End Select** thể hiện được sự tiện dụng vượt trội. Trong VB cú pháp lệnh này như sau:

```
Select Case <biểu thức>
    Case <giá trị 1>
        <thủ tục 1>
    Case <giá trị 2>
        <thủ tục 2>
    .....
    Case <giá trị n>
        <thủ tục n>
    [Case Else
        <thủ tục n+1>]
End Select
```

Trong đó: <Biểu thức> luôn trả về giá trị kiểu vô hướng đếm được như: số nguyên, xâu ký tự, kiểu lô gíc,..

Với cấu trúc này, VB hoạt động như sau:

(1) Tính giá trị của biểu thức

(2) Kiểm tra $\langle \text{biểu thức} \rangle = \langle \text{giá trị } 1 \rangle$?

- Nếu đúng thực hiện $\langle \text{thủ tục } 1 \rangle$ và kết thúc lệnh, thực hiện lệnh tiếp theo sau từ khoá End Select.

- Nếu sai, thực hiện tiếp việc so sánh $\langle \text{biểu thức} \rangle = \langle \text{giá trị } i \rangle$ tiếp theo và xử lý tương tự qui trình nêu trên.

(3) Trong trường hợp $\langle \text{biểu thức} \rangle \langle \text{ giá trị } i \rangle$, $i=1..n$ khi đó có 2 khả năng:

- Nếu có tùy chọn *Case Else* thì VB sẽ thực hiện $\langle \text{thủ tục } n+1 \rangle$;

- Nếu không có tùy chọn *Case Else*, VB sẽ không thực hiện bất kỳ thủ tục nào đã liệt kê trong vùng Select .. End Select cả mà chuyển tới thực hiện lệnh tiếp theo sau từ khoá End Select.

Xét ví dụ sau: Kiểm tra một số nguyên (so) và trả về tên tiếng Anh tháng tương ứng với số nguyên đó (biển thang) , ví dụ:

1 - January

2 - February

...

12 - December

>12 - Không xác định

Nếu dùng lệnh If hoàn toàn có thể đáp ứng được bài toán này, thay vào đó sẽ là một tập hợp 12 lệnh If .. Else .. End If như sau:

```

If so = 1 Then
    thang = "January"
Else
    If so = 2 Then
        thang = "February"
    Else
        If so = 3 Then
            thang = "March"
        Else
            If so = 4 Then
                thang = "April"
            Else
                If so = 5 Then
                    thang = "May"
                
```

```

Else
  If so = 6 Then
    thang = "June"
  Else
    If so = 7 Then
      thang = "July"
    Else
      If so = 8 Then
        thang = "August"
      Else
        If so = 9 Then
          thang = "September"
        Else
          If so = 10 Then
            thang = "October"
          Else
            If so = 11 Then
              thang = "November"
            Else
              If so = 12 Then
                thang = "December"
              Else
                thang = "Không xác định"
              End If
            End If
          End If
        End If
      End If
    End If
  End If
End If

```

Tuy nhiên khi sử dụng **Select Case .. End Select**, cấu trúc sẽ gọn gàng và sáng sủa hơn nhiều. Cụ thể như sau:

```

Select Case so
  Case 1
    thang = "January"
  Case 2
    thang = "February"
  Case 3
    thang = "March"
  Case 4
    thang = "April"
  Case 5
    thang = "May"
  Case 6
    thang = "June"

```

```

Case 7
    thang = "July"
Case 8
    thang = "August"
Case 9
    thang = "September"
Case 10
    thang = "October"
Case 11
    thang = "November"
Case 12
    thang = "December"
Case Else
    thang = "Không xác định"
End Select
    
```

2.3 Cấu trúc FOR ... NEXT

For... Next là một cấu trúc lặp biết trước số lần lặp trong VB, tuy nhiên trong những tình huống đặc biệt, vẫn có thể sử dụng cấu trúc này như cấu trúc không biết trước được số lần lặp.

Cú pháp cấu trúc For...Next như sau:

```

For <biến chạy> = <giá trị 1> To <giá trị 2> [Step <n>]
    <thủ tục>
    [Exit For]
Next
    
```

Trong đó:

- <biến chạy> là biến kiểu vô hướng đếm được, hay dùng nhất là biến kiểu nguyên;
- <giá trị 1>, <giá trị 2> là các giá trị mà biến chạy sẽ nhận và thực hiện dịch chuyển sau mỗi lần lặp. Có thể dịch chuyển đi 1 đơn vị, có thể dịch chuyển đi nhiều đơn vị một lần, có thể dịch chuyển tiến, cũng có thể dịch chuyển lùi- tất cả điều này tùy thuộc vào việc có hay không có tùy chọn [**Step** <n>];

- Nếu có tùy chọn [**Step** <n>] biến chạy sẽ dịch n đơn vị sau mỗi lần lặp. Khi đó, nếu n>0 sẽ dịch tiến, ngược lại sẽ dịch lùi;
- Mỗi lần lặp, VB sẽ thực hiện <thủ tục> một lần;
- Trong trường hợp đặc biệt nếu gặp phải lệnh **Exit For** trong vòng lặp, ngay lập tức thoát khỏi lệnh lặp và thực hiện lệnh tiếp ngay sau từ khoá Next. Chính **Exit For** đã làm mất đi tính lặp biết trước được số lần lặp của loại lệnh này.

Tiếp theo là các ví dụ:

Ví dụ 1: Tính tổng các số từ 1 đến 50, giá trị được lưu vào biến *tong*.

```
Dim i As Byte
Dim tong As Integer

tong = 0
For i = 1 To 50
    tong = tong + i
Next
Msgbox tong
```

Ví dụ 2: Tính tổng các số chia hết cho 3 từ 1 đến 50, giá trị được lưu vào biến *tong*.

```
Dim i As Byte
Dim tong As Integer

tong = 0
For i = 3 To 50 Step 3
    tong = tong + i
Next
Msgbox tong
```

Lệnh For trong ví dụ này chỉ khác lệnh For ở ví dụ 1 ở chỗ **Step 3**. Vì <giá trị *i*> = 3 là số chia hết cho 3, nên tất cả các giá trị *i* còn lại sẽ chia hết cho 3 (vì $i = i + 3$).

Ví dụ 3: Kiểm tra một số nguyên (>2) có phải là nguyên tố hay không?

```

Dim so As Integer
Dim uoc As Integer
Dim nguyento As Boolean

nguyento = True
For uoc = 2 To Int(so / 2)
    If so Mod uoc = 0 Then
        nguyento = False
        Exit For
    End If
Next

If nguyento Then
    MsgBox "là nguyên tố"
Else
    MsgBox "không là nguyên tố !"
End If

```

Giải thuật đơn giản để xác định một số có phải nguyên tố hay không là: xác định xem tất cả các số (*uoc*) có thể trở thành ước của số (*so*) cần kiểm tra. Nếu tìm thấy một ước thực sự đầu tiên, kết luận ngay không phải số nguyên tố bởi lệnh *nguyento = False* và thoát khỏi vòng lặp bằng lệnh **Exit For**; trong trường hợp xét toàn bộ các ước có thể mà không tìm được một số nào là ước thực sự, kết luận đây là số nguyên tố (biến *nguyento = True* như giá trị ban đầu)

2.4 Cấu trúc WHILE ... WEND

While ... Wend là một cấu trúc lặp không biết trước số lần lặp trong VB. Cách pháp cấu trúc While...Wend như sau (Wend - viết tắt của cụm từ **While End**):

```

While <điều kiện>
    <thủ tục>
Wend

```

Trong đó:

- **While, Wend** là các từ khoá của lệnh lặp;
- Nếu <điều kiện> = *True*, các lệnh trong <thủ tục> sẽ được thực hiện. Thực hiện xong lại quay lên dòng lệnh **While** để kiểm tra tiếp <điều kiện>;

- Nếu <điều kiện> = *False*, sẽ thoát khỏi vòng lặp và thực hiện lệnh tiếp theo từ khoá **Wend**.

Chú ý: Luôn phải chứng minh được rằng, sau một số hữu hạn lần thực hiện <thủ tục>, giá trị của <biểu thức> phải là *False* để thoát khỏi vòng lặp. Trong trường hợp không thể thoát khỏi vòng lặp, có nghĩa người lập trình đã mắc phải *lỗi lặp vô hạn*. Có thể dẫn đến chương trình bị treo.

Các ví dụ:

Ví dụ 1: Tính tổng các số chia hết cho 3 trong khoảng từ 1 đến 50

```
Dim i As Byte
Dim tong As Integer

tong = 0
i = 3
While i <= 50
    tong = tong + i
    i = i + 3
Wend
Msgbox tong
```

Ví dụ 2: Ví dụ này thể hiện vòng lặp vô hạn. Lý do có thể là chủ quan, rất đơn giản vì gõ nhầm! Hãy chỉ ra dòng lệnh gõ nhầm và thực hiện sửa cho đúng.

```
Dim i As Byte
Dim tong As Integer

tong = 0
i = 1
While i <= 50
    If i Mod 3 = 0 Then
        tong = tong + i
    End If
    j = i + 1
Wend
Msgbox tong
```

3. Chương trình con

Chương trình con (CTC) là một đơn vị mã lệnh VB, nó có thể chứa tập hợp các câu lệnh nhằm thao tác, tính toán hoặc điều khiển mục đích hoặc dữ liệu nào đó. Trong VB có 2 loại CTC:

- CTC dạng thủ tục, được khai báo bởi từ khoá **Sub**;
- CTC dạng hàm, được khai báo bởi từ khoá **Function**.

Về bản chất, 2 loại CTC trên đều như nhau: khai báo, tham số và truyền tham số. Tuy nhiên, điểm khác nhau cơ bản là:

- Function luôn trả về một giá trị kiểu vô hướng chuẩn, ví dụ: hàm *Date()* - trả về giá trị ngày hiện tại kiểu Date. Trong VB đã sẵn có rất nhiều các hàm tính toán, chúng được gọi là các build-in function. Hơn nữa, người dùng hoàn toàn có thể tạo ra các hàm để sử dụng cho các mục đích riêng loại hàm này gọi là user-define function;
- Còn Sub thì không, nó chỉ thực hiện một số các công việc. Tất nhiên những công việc này hoàn toàn có thể làm thay đổi dữ liệu theo mong muốn trong chương trình. Cũng như Function, VB sẵn có một thư viện các thủ tục; hơn nữa người dùng cũng có thể tự tạo thêm những thủ tục mới phục vụ việc xử lý dữ liệu theo mục đích riêng. Đặc biệt, VB còn định nghĩa thủ tục đáp ứng sự kiện. Thủ tục này sẽ được tự động gọi ra khi sự kiện đáp ứng bị ảnh hưởng. Chúng ta sẽ trở lại nội dung này qua các ví dụ lập trình VB.

Tùy từng tính huống cụ thể sẽ lựa chọn sử dụng Function hoặc Sub.

3.1 Chương trình con dạng hàm

Cú pháp

```
Function <tên hàm>([<danh sách các tham số>]) As <kiểu DL hàm>
    <thủ tục>
End Function
```

Trong đó:

- **Function, End Function** là các từ khoá bắt buộc khai báo cấu trúc một CTC dạng hàm;
- *<tên hàm>* là tên gọi hàm định khai báo. Tên không được chứa dấu cách (space) và các ký tự đặc biệt;
- *<danh sách các tham số>* - danh sách các tham số cần thiết cho hàm. Có hay không có danh sách này tùy thuộc vào hàm cần định nghĩa;
- *<kiểu DL hàm>* - kiểu dữ liệu mà hàm sẽ trả lại. Phần này bắt buộc phải được khai báo với mỗi hàm;
- *<thủ tục>* - thân CTC. Trong đó câu lệnh *<tên hàm> = <biểu thức>* phải xuất hiện ít nhất một lần trong thủ tục. Câu lệnh này có tác dụng gán giá trị cho hàm.

Nếu không có từ khoá Public trước Function, hàm đó chỉ có tác dụng cục bộ: trong một module, trong một report hoặc trong một form. Khi có từ khoá Public trước Function, hàm sẽ có tác dụng toàn cục. Tức là có thể sử dụng bất kỳ nơi nào trên tệp VB đó. Tất nhiên, tất cả những gì khai báo là Public phải được khai báo trong phần **Decralations** của một Module nào đó.

Các ví dụ:

Ví dụ 1: hàm tính tổng 2 số

```
Function Tong2So(a, b As Double) As Double
    Tong2So = a + b
End Function
```

Ví dụ 2: hàm kiểm tra một số có phải là nguyên tố hay không?

```
Function laNguyenTo(so As Integer) As Boolean
    Dim uoc As Integer

    laNguyenTo = True
    If so > 2 Then
        For uoc = 2 To Int(Sqr(so))
            If so Mod uoc = 0 Then
                laNguyenTo = False
                Exit For
            End If
        Next uoc
    End If
End Function
```

```

        End If
    Next
End If
End Function

```

Ví dụ trên có sử dụng đến:

- hàm *Int(number)* – hàm lấy phần nguyên của một số;
- hàm *Sqr(number)* – hàm lấy căn bậc hai một số

Ví dụ 3: hàm tách tên trong xâu họ và tên.

Đây là một bài toán gặp phải rất nhiều trong thực tế. Cụ thể bài toán giải quyết vấn đề sau:

Nếu biết họ tên là *Nguyễn Sơn Hải*, hàm sẽ tách ra được tên là *Hải*. Toàn bộ mã lệnh hàm như sau:

```

Function GetTen(hoten As String) As String
    Dim pos As Integer

    pos = 1
    If InStr(pos, Trim(hoten), " ") = 0 Then
        GetTen = hoten
        Exit Function
    End If
    While InStr(pos + 1, Trim(hoten), " ") > 0
        pos = InStr(pos + 1, Trim(hoten), " ")
    Wend

    GetTen = Mid(hoten, pos)
End Function

```

Ví dụ 4: Hàm dùng so sánh 2 xâu kiểu chữ TCVN3 chúng tôi đưa ra dưới đây là một tham khảo rất tốt. Trong Word, VB cũng như các bảng dữ liệu tiếng Việt có dấu trên máy tính, việc sắp xếp xâu ký tự là một bài toán mà người Việt phải giải quyết. Ví dụ, dưới đây là một danh sách trên Word:

STT	Tên
1	Quang

2	Đức
3	Đoàn
4	Băng
5	Bang
6	An
7	Ân

Sau khi sử dụng tính năng sắp xếp (Sort) của Word theo cột Tên theo thứ tự tăng dần, được danh sách kết quả như sau:

STT	Tên
7	Ân
3	Đoàn
2	Đức
6	An
4	Băng
5	Bang
1	Quang

Mà danh sách sắp xếp đúng phải là:

STT	Tên
6	An
7	Ân
5	Bang


```

End Function

Function Mahoa(Ckt As String) As String
    Dim vt1 As Integer
    Dim kq, Ctam As String
    Ckt = Ckt & " "
    kq = ""
    vt1 = InStr(Ckt, " ")

    Do While vt1 <> 0
        Ctam = Trim(Left(Ckt, vt1 - 1))
        Ckt = Right(Ckt, Len(Ckt) - vt1)
        kq = MahoaTCVN3(Ctam) & " " & kq
        vt1 = InStr(Ckt, " ")
    Loop
    Mahoa = kq
End Function

```

3.2 Chương trình con dạng thủ tục

Cú pháp

```

[Public] [Private] Sub <tên CTC>([<danh sách các tham số>])
    <thủ tục>
End Sub

```

Trong đó:

- **Sub, End Sub** là các từ khoá bắt buộc khai báo cấu trúc một CTC dạng thủ tục;
- <tên CTC> là tên gọi thủ tục định khai báo. Tên không được chứa dấu cách (space) và các ký tự đặc biệt;
- <danh sách các tham số> - danh sách các tham số cần thiết cho thủ tục. Có hay không có danh sách này tùy thuộc vào thủ tục cần tạo
- <thủ tục> - thân CTC.

Nếu không có từ khoá Public trước Sub, thủ tục đó chỉ có tác dụng cục bộ: trong một module, trong một report hoặc trong một form. Khi có từ khoá Public

trước Sub, thủ tục sẽ có tác dụng toàn cục. Tức là có thể sử dụng bất kỳ nơi nào trên tệp VB đó. Tất nhiên, tất cả những gì khai báo là Public phải được khai báo trong phần Declarations của một Module nào đó.

Các ví dụ:

Ví dụ 1: Thủ tục tính tổng hai số

```
Sub tong2so(a, b As Double)
    tong = a + b

    'chú ý: tong- là biến được khai báo toàn cục
End Sub
```

Ví dụ 2: Cũng là tính tổng, nhưng thủ tục sau đây không có ý nghĩa gì!

```
Sub tong2so(a, b As Double)
    Dim tong As Double

    tong = a + b

    'chú ý: tong- là biến được khai báo toàn cục
End Sub
```

Vì sao? Vì biến *tong* được khai báo cục bộ trong CTC *tong2so*, nên khi CTC này kết thúc, biến *tong* cũng bị giải thoát khỏi bộ nhớ luôn. Không gây ảnh hưởng gì đến dữ liệu cũng như thể hiện của chương trình.

Ví dụ 3: CTC giải phương trình bậc 2 với 3 tham số đầu vào là a, b và c ($a > 0$)

```
Sub GPTB2(a, b, c As Double)
    Dim delta As Double
    Dim x1, x2 As Double

    delta = b * b - 4 * a * c
    If delta < 0 Then
        MsgBox "Phương trình vô nghiệm !"
    Else
        If delta = 0 Then
            x1 = -b / (2 * a)
            MsgBox "Nghiem kép x1 = x2 = " + Trim(Str(x1))
        Else
            x1 = (-b + Sqr(delta)) / (2 * a)
            x2 = (-b - Sqr(delta)) / (2 * a)
        End If
    End If
End Sub
```



```

        MsgBox "2 nhienem x1=" + Trim(Str(x1)) + ", " _
        + "x2=" + Trim(Str(x2))
    End If
End If
End Sub

```

Ví dụ 4: CTC hiển thị dòng chữ chào mừng (một CTC không có tham số)

```

Sub Hello()
    MsgBox "Chào các bạn !"
End Sub

```

3.3 Sử dụng chương trình con

a. Sử dụng thủ tục

Lệnh Call để gọi một thủ tục ra làm việc. Cú pháp như sau:

```

Call <Tên CTC> (<Danh sách các tham trị>)

```

Trong đó:

- **Call** là từ khoá - lệnh để gọi và thi hành một CTC ;
- <**Tên CTC**> là tên của CTC cần gọi ra thi hành;
- <**Danh sách các tham trị**> - là danh sách các giá trị (nếu cần) để truyền vào các tham số tương ứng của CTC. Chú ý: kiểu dữ liệu của các tham trị và các tham số phải tương ứng nhau. Nếu tham trị được truyền gián tiếp qua tên một biến thì biến đó phải được khai báo trước đó và có định kiểu hết như kiểu của tham số cần truyền vào.

Ví dụ 1: Gọi CTC giải phương trình bậc 2 (với 3 tham số là A, B, C kiểu thực) đã lập trình từ trước.

```

Call GPTB2(3, 4, 1)

```

Hoặc truyền tham số gián tiếp qua tên các biến

```
Dim a, b, c As Double
a = 3
b = 4
c = 1

Call GPTB2(a, b, c)
```

Ví dụ 2: Gọi CTC tính tổng 2 số với 2 số cho trước là 7 và 8

```
Call tong2so(7, 8)
```

Hoặc gọi gián tiếp qua các tên biến

```
Dim a, b As Double
a = 7
b = 8
Call tong2so(a, b)
```

Đặc biệt, với những CTC dạng thủ tục không có tham số đầu vào có thể bỏ qua lệnh Call mà gõ vào trực tiếp tên CTC khi sử dụng.

Ví dụ 3: Gọi CTC Hello ở trên

```
'Gọi theo kiểu chính tắc
Call Hello

'Bỏ qua từ khoá Call vì CTC này không có tham số
Hello
```

b. Sử dụng hàm

Sau khi đã tạo được các hàm, việc sử dụng chúng như sử dụng các hàm có sẵn của VB. Chú ý: khác với CTC dạng thủ tục, tên CTC hàm phải được gọi thông qua một biểu thức (vì hàm luôn trả về một giá trị được xác định kiểu cho trước).

Ví dụ 1: Để gán kết quả tổng của 2 số vào một ô Textbox có tên Text1, sử dụng hàm *tong2so* ở trên. Khi đó mã lệnh sẽ viết như sau:

```
Text1.Text = tong2so(5, 7)
```

Hoặc cách gọi sử dụng tên biến làm tham trị cho CTC

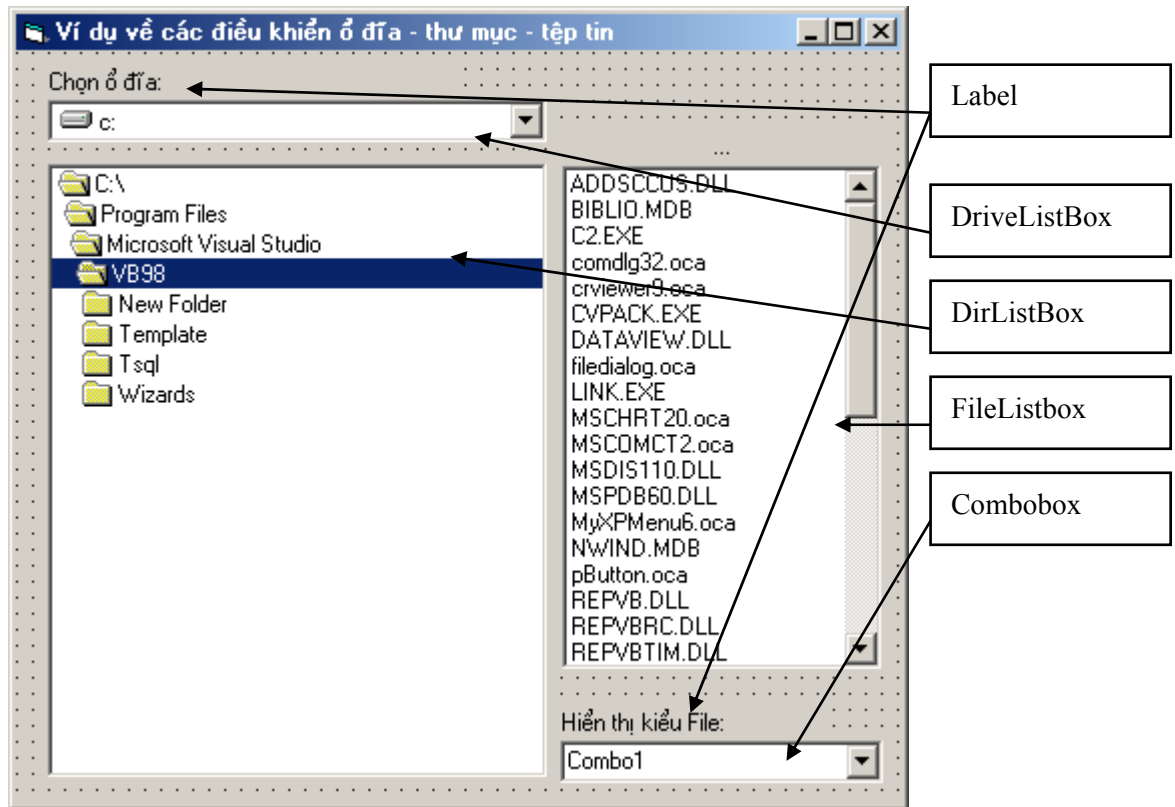
```
Dim a, b As Double  
a = 5  
b = 7  
Text1.Text = tong2so(a, b)
```

4. Soạn thảo chương trình và xử lý lỗi

4.1 Soạn thảo chương trình

Soạn thảo chương trình bao gồm tập hợp các công việc về thiết kế, cài đặt các chức năng của ứng dụng. Trong qui trình xây dựng phần mềm, công việc này thường của lập trình viên (Coder). Có 2 phần việc chính trong soạn thảo chương trình là thiết kế giao diện và cài đặt thủ tục.

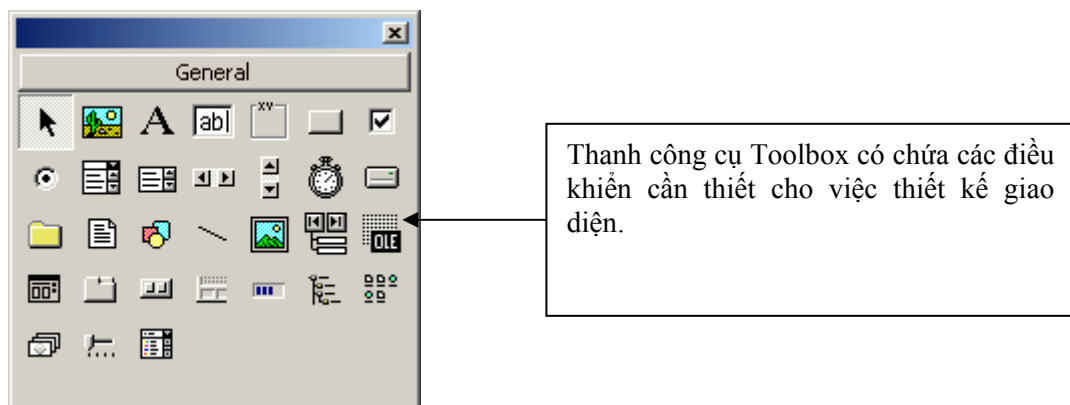
Thiết kế giao diện: là việc sử dụng các điều khiển (Controls), các đối tượng (Objects) cũng các kỹ năng, kỹ thuật cần thiết để tạo ra các giao diện sử dụng phần mềm. VB là ngôn ngữ lập trình trực quan, nên việc tạo ra các giao diện sử dụng cũng được thực hiện một cách trực quan, dễ dàng hầu hết bằng cách sử dụng con chuột. Ví dụ: để thiết kế giao diện như sau:



Cách làm như sau:

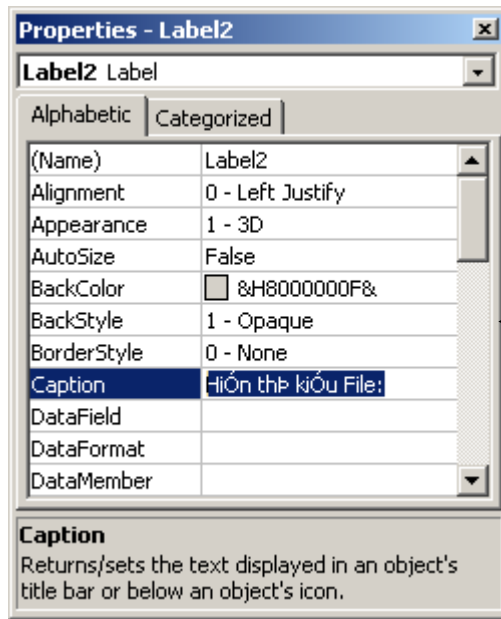
Bước 1: Xác định đúng các điều khiển cần sử dụng cho các đối tượng cần thiết kế trên giao diện (xem minh họa các loại đối tượng ở hình trên);

Bước 2: Dùng chuột đưa các điều khiển từ thanh công cụ Toolbox lên Form cần thiết kế. Việc này thực hiện đơn giản như vẽ một khối hình.



Nếu không thấy xuất hiện thanh công cụ, có thể gọi chúng ra bằng lệnh **View | Toolbox**

Bước 3: Thiết lập các thuộc tính phù hợp cho các đối tượng trên giao diện thông qua cửa sổ Properties:



Cửa sổ Properties – nơi có thể thiết lập trực tiếp các thuộc tính cho điều khiển. Cột bên trái là tên các thuộc tính; cột bên phải là giá trị các thuộc tính tương ứng mà có thể thay đổi được.

Nếu không thấy xuất hiện cửa sổ Properties, có thể gọi chúng ra bằng lệnh **View | Properties Windows** hoặc nhấn phím chức năng **F4**.

Gài đặt thủ tục: khác với các ngôn ngữ lập trình trên DOS như Pascal hoặc C. Việc lập trình (viết mã lệnh) trên VB đa dạng hơn, đặc biệt với các điều khiển được thiết kế trên form, việc lập trình đa phần là viết các thủ tục đáp ứng các sự kiện (chúng ta còn quay lại phần này ở những phần tiếp theo).

Mỗi câu lệnh VB được viết trên một dòng (kết thúc bởi ký tự Enter). Trong trường hợp các câu lệnh quá dài, có thể viết trên nhiều dòng văn bản nhưng phải được nối với nhau bởi ký hiệu ghép dòng lệnh. Dưới đây là một dòng lệnh:

```
Msgbox "Chào các bạn, tôi tên là Nam, mong được làm quen!"
```

Vẫn lệnh đó, có thể viết trên nhiều dòng, phải sử dụng kỹ tự ghép dòng lệnh như sau:

```
Msgbox "Chào các bạn, tôi tên là Nam, "
      & "mong được làm quen!"
```

Ký tự để ghép tiếp dòng lệnh tiếp theo (một dấu cách, tiếp theo là dấu gạch dưới).

Để thi hành một dự án VB, nhấn phím **F5** hoặc nhấn nút **Start** trên thanh ToolBar.

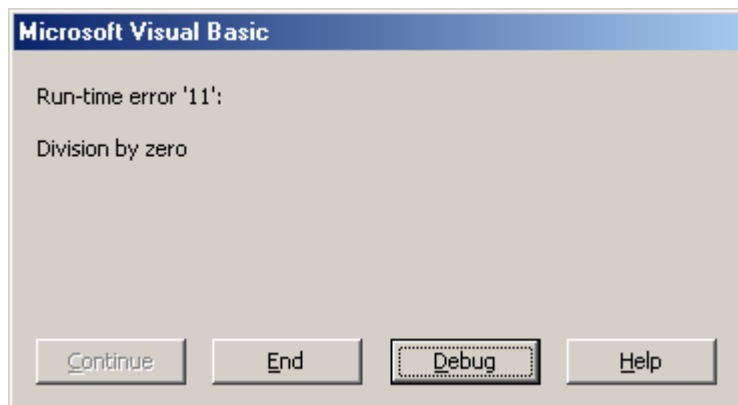
Để ngừng thi hành dự án VB, nhấn nút **End**.

4.2 Lỗi và xử lý lỗi

Lỗi là hiện tượng rất bình thường trong lập trình. Khi lập trình viên đưa ra những câu lệnh không hợp lý về cú pháp, về xử lý dữ liệu rất có thể gây ra lỗi. Có rất nhiều nguyên nhân gây ra lỗi; các nguyên nhân này có thể được lường trước hoặc không được lường trước. Tuy nhiên, khi chương trình được đóng gói và chuyển giao tới khách hàng sử dụng thì không được phép tồn tại lỗi phần mềm. Kỹ thuật xử lý lỗi bao gồm các kỹ năng phát hiện và xử lý các tình huống khi chương trình gây lỗi để làm sao đảm bảo chương trình giảm tới đa lỗi.

a. Xử lý lỗi

Là việc xử lý khi đang lập trình gặp phải lỗi. Thông thường khi chạy thử chương trình trong lúc đang xây dựng phần mềm nếu gặp phải lỗi, sẽ xuất hiện hộp thoại thông báo lỗi có dạng:



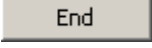
Thông thường một hộp thoại thông báo lỗi gồm 2 thành phần:

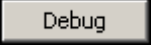
- Thành phần báo lỗi bao gồm:

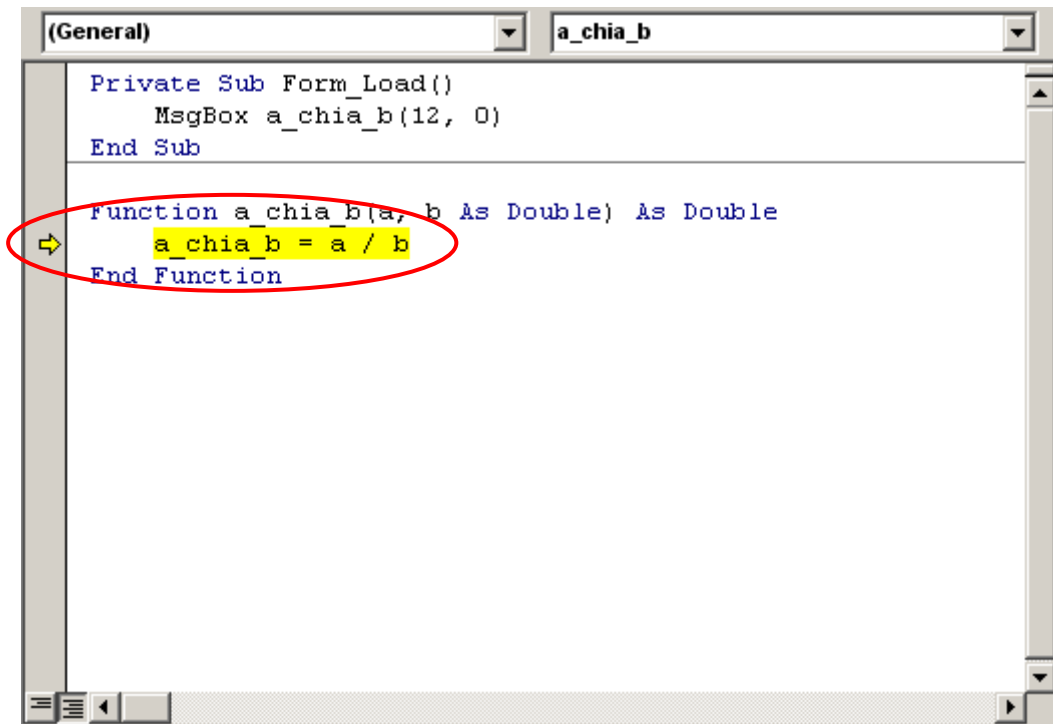
+ **Mã số lỗi** - Mỗi lỗi mà VB có thể kiểm tra được đều có một mã số, được hiển thị ở dòng thông báo: *Run-time error 'mã số lỗi'*: Ví dụ trên là : *Run-time error '11'*:

+ **Tên lỗi**. Ở ví dụ trên tên lỗi là: *Division by zero* - lỗi sai kiểu dữ liệu.

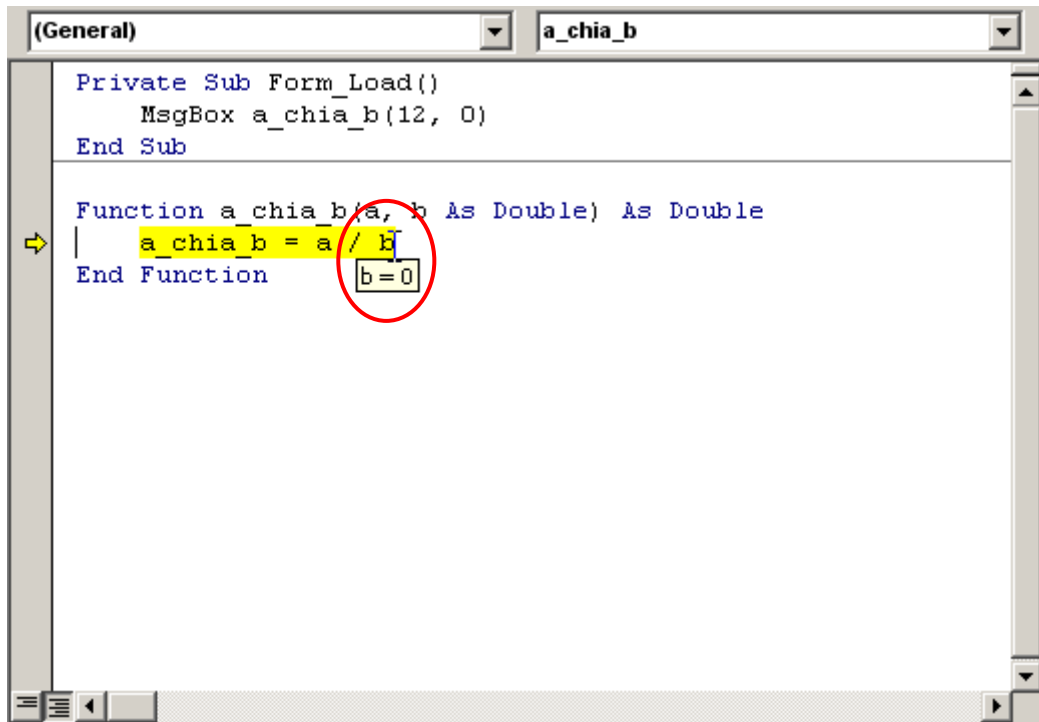
- Thành phần xử lý lỗi gồm 2 nút lệnh:

+ Nút  - để dừng ngay chương trình, chuyển về chế độ **Design** - thiết kế bình thường;

+ Nút  - để dừng chương trình chuyển về chế độ **Break** - sửa lỗi trực tiếp. Khi đó câu lệnh lỗi sẽ được tô bởi màu nền vàng cho phép người lập trình có thể sửa được mã chương trình:

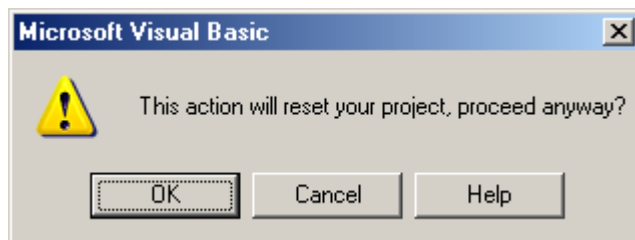


Khi dịch chuột lên một biến nào đó, giá trị biến sẽ được hiển thị dưới dạng **Tool tip**.




Hình trên khi di chuột lên biến b, giá trị biến b xuất hiện dưới dạng Tool tip (giá trị b = 0).


Sau khi chọn nút **Debug**, lập trình viên hoàn toàn có thể thực hiện sửa mã lệnh trong chương trình. Tuy nhiên, trong một số trường hợp khi sửa mã lệnh VB sẽ hỏi:

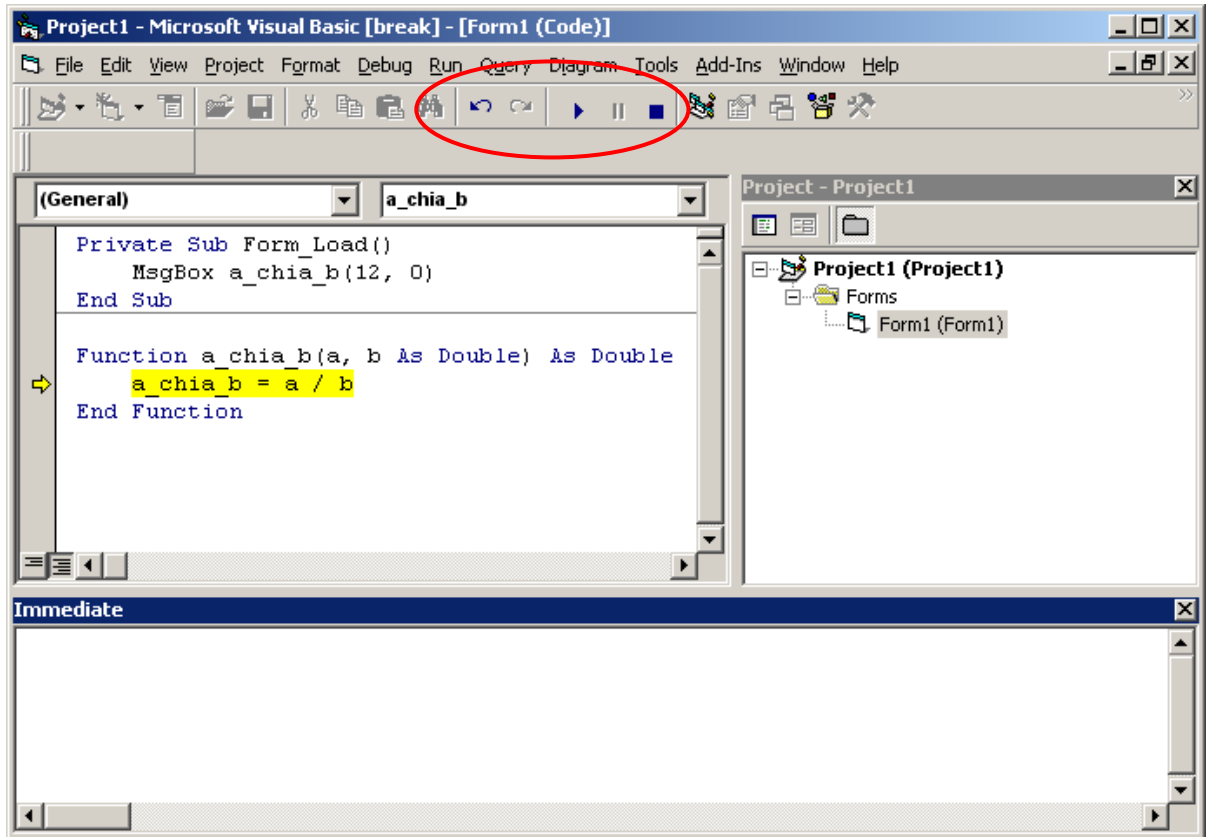


Điều này có nghĩa: việc sửa đổi mã lệnh vừa rồi, VB yêu cầu phải trở về chế độ thiết kế bình thường nếu nhấn **Ok**; trái lại nhấn **Cancel**- việc thay đổi mã lệnh sẽ không được chấp nhận.

Sau khi thực hiện sửa mã lệnh, có thể yêu cầu VB thực thi tiếp chương trình.

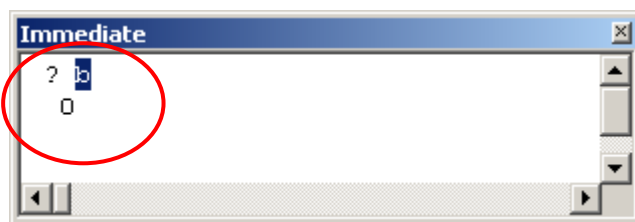
Việc thực thi sẽ được tiến hành tiếp tục tại vị trí vệt sáng đang trở. Hoặc có thể dùng chuột để dịch chuyển vệt sáng về lệnh cần thực thi (chỉ trong cùng một CTC). Để thực thi tiếp nhấn phím F5 hoặc nút **Continue**  trên thanh công cụ;

hoặc nhấn nút **Stop**  nếu muốn dừng việc sửa mã lệnh trong chế độ *Break*, chuyển về chế độ *Design*.



Cửa sổ Immediate

Là công cụ hữu hiệu hỗ trợ việc dò lỗi bởi: hộp thoại này cho phép thực thi từng câu lệnh trên chế độ hội thoại.



Giả sử ví dụ trên sau khi gõ lệnh:

? b

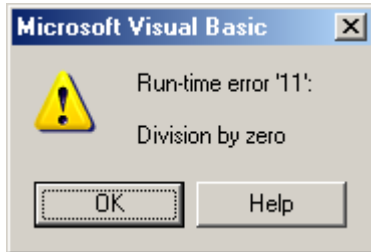
Xem giá trị của biến b. Sau khi nhấn Enter sẽ nhận được kết quả

0

Hoặc nếu gõ:

? a / b

Sẽ nhận được thông báo lỗi:



Vì $b = 0$.

b. Bẫy lỗi

Mục 4.2.1 đã trình bày những kỹ năng để xử lý lỗi khi đang soạn thảo chương trình. Các thao tác đó chỉ được thực hiện trong lúc đang xây dựng phần mềm (VB IDE), do người lập trình xử lý. Khi phần mềm đã được đóng gói để chuyển đến người dùng nếu gặp lỗi, nó sẽ hiển thị một hộp thoại thông báo lỗi (*Error Dialog*) cho biết lý do vắn tắt về lỗi. Sau khi nhấn OK, chương trình sẽ ngừng hoạt động, bị thoát.

Để xử lý lỗi trong tình huống này, có 2 phương pháp bẫy lỗi mà chúng tôi đưa ra dưới đây để tham khảo; hy vọng lập trình viên sẽ chọn lựa được tình huống phù hợp để sử dụng một trong các phương pháp này đảm bảo chương trình viết ra chạy được đúng theo mục đích.

Sử dụng lệnh **On Error Resume Next**

Khi đó từ chỗ đó trở đi, nếu chương trình gặp lỗi, nó sẽ bỏ qua (ignore) hoàn toàn. Điểm này tiện ở chỗ giúp chương trình EXE của ta tránh gặp lỗi thoát khỏi đột ngột như phân tích ở trên. Nhưng nó cũng bất lợi là khi khách hàng cho hay họ gặp những trường hợp lạ, không giải thích được (vì lỗi đã bị bỏ qua mà không ai để ý), thì ta cũng bí luôn, có thể không biết bắt đầu từ đâu để gỡ lỗi. Do đó, trong lúc gỡ lỗi ta không nên dùng nó, nhưng trước khi giao cho khách hàng nên cân nhắc kỹ có nên sử dụng trong các đoạn mã lệnh hay không.

Ví dụ sử dụng **On Error Resume Next** để bỏ qua lỗi:

```
Function A_chia_B(a, b As Double) As Double
    On Error Resume Next

    A_chia_B = Null

    A_chia_B = a / b

End Function
```

Trong CTC trên, nếu $b = 0$, lệnh $A_chia_B = a / b$ sẽ gặp phải lỗi. Do có lời khai báo **On Error Resume Next** nên lệnh lỗi này được bỏ qua (không thực hiện). Tức là giá trị hàm là Null.

Sử dụng câu lệnh **On Error Goto <nhãn>**

Khi một thủ tục được đặt câu lệnh này, nếu gặp phải một lỗi nào đó, VB sẽ chuyển thẳng việc thực hiện đến <nhãn> đã chỉ định. Thông thường các lệnh tiếp theo của <nhãn> là xử lý các tình huống lỗi.

Sau đây là ví dụ sử dụng phương pháp **On Error Goto <nhãn>** để bắt lỗi:

```
Function A_chia_B(a, b As Double) As Double
    On Error GoTo Loi

    A_chia_B = a / b
    MsgBox " Ok! "

Loi:

    If Err.Number = 11 Then
        MsgBox "Lỗi chia cho 0 !"
    End If

End Function
```

Trong CTC trên, trong trường hợp $b = 0$ câu lệnh $A_chia_B = a / b$ sẽ gây ra lỗi. Theo như khai báo **On Error Goto Loi** ban đầu, VB sẽ bỏ qua tất cả các lệnh sau lệnh lỗi và chuyển thẳng tới các lệnh sau nhãn **Loi**: Ở đây là lệnh kiểm tra lỗi. Nếu $Mã\ lỗi = 11 \rightarrow$ kết luận ngay một thông báo lỗi tiếng Việt. *Lỗi chia cho 0!*

Phương pháp này cũng được dùng phổ biến cả trong quá trình xây dựng để phát hiện lỗi, cũng như trong phần mềm đã đóng gói gửi đến khách hàng. Mỗi

khi gặp lỗi sẽ được thông báo nguyên nhân gây ra lỗi bằng tiếng Việt (chẳng hạn) mà vẫn không ảnh hưởng đến hoạt động khác của phần mềm.

Trong phương pháp này, người lập trình nên khai thác tối đa đối tượng **Err** - đối tượng mang những thông tin về lỗi đang xảy ra, cụ thể:

Hành động	Kết quả
Err.Description	Mô tả tên lỗi
Err.Number	Đưa ra mã lỗi
Err.Clear	Xoá bỏ các giá trị của đối tượng Err

Bài tập chương 1

Bài tập 1:

Xây dựng thủ tục tìm ước số chung lớn nhất của 2 số nguyên a và b.

Bài tập 2:

Xây dựng thủ tục tìm bội số chung nhỏ nhất của của 2 số nguyên a và b.

Bài tập 3:

Xây dựng thủ tục thống kê tổng số từng loại ký tự có trong một chuỗi. Ví dụ có chuỗi ban đầu “cong hoa xa hoi chu nghia viet nam”. Kết quả thủ tục này cho biết:

Tổng số ký tự	a: 3
	c: 1
	e: 1
	g: 2
	h: 4
	i: 1
	m: 1
	n: 3
	o: 3
	v: 1

Bài tập 4:

Xây dựng hàm tìm số cực đại trong bốn số nguyên a, b, c và d.

Bài tập 5:

Xây dựng hàm đảo vị trí ký tự của một chuỗi. Ví dụ: chuỗi đầu vào có dạng “abcd”, kết quả hàm này sẽ trả về chuỗi ngược lại là “dcba”.

Bài tập 6:

Xây dựng thủ tục giải phương trình $Ax^2 + Bx + C = 0$. Trong đó A, B, C là các tham số đầu vào, x là nghiệm cần tìm.

Bài tập 7:

Xây dựng hàm tính tổng các số nguyên trong một khoảng số nguyên [a, b] cho trước.

Bài tập 8:

Xây dựng hàm tách từ đầu tiên của một chuỗi ký tự. Ví dụ có một chuỗi như sau: “Việt Nam”, hàm này sẽ tách ra được chuỗi “Việt” – là từ đầu tiên trong chuỗi đó.

CHƯƠNG 2

SỬ DỤNG ĐIỀU KHIỂN

Một thế mạnh của ngôn ngữ VB là lập trình trực quan thông qua các điều khiển (Controls), nhiều khi người ta còn gọi là các đối tượng (Objects). Đây là sự khác biệt căn bản giữa lập trình trực quan (trên windows) với cách thức lập trình trên DOS (Pascal, C, C++) trước đây. Điều khiển là gì? Làm việc trên các lớp điều khiển như thế nào? Là các vấn đề sẽ được trình bày kỹ trong chương này.

1. Sơ lược về điều khiển

Controls là những đối tượng được dùng để thiết kế giao diện trong những ngôn ngữ lập trình trực quan. Trong VB, các đối tượng Controls được đặt trên thanh công cụ Toolbox mà lập trình viên có thể lấy từ đây ra để thiết kế lên các mẫu giao diện (Forms) hoặc báo cáo (Report).

Mỗi Control được xem như là một đối tượng (Object), có 3 thành phần cơ bản sau:

1.1. Tập thuộc tính

Thuộc tính (Properties) là các thành phần mô tả tính chất một đối tượng. Ví dụ: nếu coi một chiếc máy vi tính là một đối tượng thì tập thuộc tính của đối tượng này có thể là:

Tên thuộc tính	Giá trị
Loại bộ vi xử lý:	<i>Intel</i>
Tốc độ CPU:	<i>3 Gh</i>
Bộ nhớ trong:	<i>512 Mb</i>
Bộ nhớ ngoài:	<i>40 Gb</i>
Kích cỡ màn hình:	<i>15"</i>
Màu màn hình:	<i>Black</i>
...	..

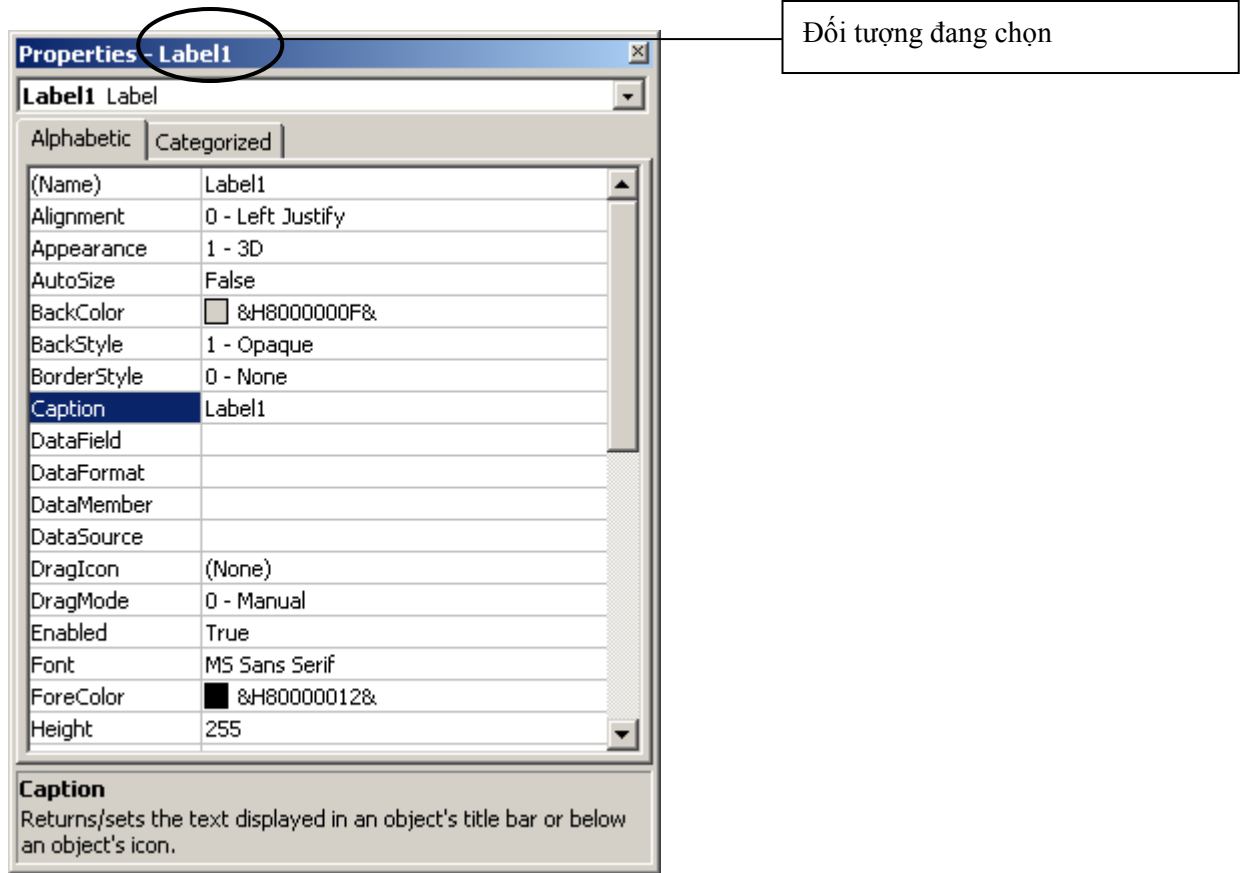
Mỗi thuộc tính luôn có một giá trị xác định. Ví dụ trên thì: thuộc tính Loại bộ vi xử lý có giá trị là Intel. Đối với một máy tính khác, giá trị thuộc tính này có thể là ADM; hoặc thuộc tính Bộ nhớ trong là 521 Mb nhưng có thể một máy tính khác giá trị thuộc tính này là 256 Mb.

Có 2 cách để thiết lập giá trị thuộc tính cho một đối tượng:

Thiết lập trực tiếp qua cửa sổ Properties

Cách này chỉ thực hiện ở chế độ thiết kế giao diện (Design view). Muốn thiết lập thuộc tính cho đối tượng nào, hãy chọn đối tượng đó bằng chuột rồi kích hoạt

cửa sổ Properties (bằng cách nhấn **F4** hoặc thực đơn **View | Properties Windows**)



Cửa sổ trên hiển thị và cho phép thiết lập các thuộc tính của đối tượng Label1, đối tượng này thuộc điều khiển Label. Danh sách bên trái cửa sổ hiển thị tên các thuộc tính, danh sách bên phải cửa sổ hiển thị và cho phép thiết lập giá trị các thuộc tính tương ứng bên trái. Thông thường, mỗi khi thiết lập xong giá trị một thuộc tính trên hộp thoại Properties, có thể nhìn thấy ngay kết quả trên màn hình thiết kế.

Thuộc tính Name

Thuộc tính **Name** cho biết tên gọi của đối tượng. Giá trị thuộc tính này không được chứa dấu cách. Trên một cửa sổ thiết kế giao diện giá trị thuộc tính Name của mỗi đối tượng là duy nhất (không được đặt trùng nhau).

Việc thiết lập thuộc tính Name trong khi lập trình là rất cần thiết và quan trọng vì mỗi khi lập trình điều khiển đối tượng nào đó phải sử dụng thuộc tính Name để tham chiếu.

Ví dụ: Kiểm tra ô tuổi (txtTuoi) xem người đó là già, trẻ hay là trung niên.

```

If Val(txtTuoi.Text) < 30 Then
    MsgBox " Thanh - thiếu niên !"
Else
    If Val(txtTuoi.Text) < 60 Then
        MsgBox " Trung niên !"
    Else
        MsgBox " Cao niên !"
    End If
End If

```

Ở ví dụ trên, muốn tham chiếu đến giá trị của ô txtTuoi phải dùng đến thuộc tính Text. Khi đó phải viết biểu thức thông qua thuộc tính *Name* của điều khiển *txtTuoi* như sau: **txtTuoi.Text**



Mẹo : Cách đặt tên cho các đối tượng

Theo kinh nghiệm lập trình về cách đặt tên các đối tượng, thường đặt theo qui tắc sau:

2 hoặc 3 ký tự đầu tiên của tên đối tượng viết chữ in thường, giá trị là từ viết tắt của kiểu đối tượng. Ví dụ:

Loại đối tượng	Từ viết tắt
Textbox	<i>txt</i>
Label	<i>lbl</i>
Command	<i>cmd</i>
Check box	<i>chk</i>
Form	<i>frm</i>
Picture	<i>pct</i>
Optional	<i>opt</i>
Frame	<i>fme</i>

Loại đối tượng	Từ viết tắt
Listbox	<i>lst</i>
Combobox	<i>cbo</i>
Line	<i>ln</i>
TreeView	<i>tv</i>
Listview	<i>lv</i>

Các ký tự còn lại của tên đối tượng đặt theo từ gợi nhớ, có thể bằng tiếng Việt hoặc bằng tiếng Anh tùy theo sở thích của người lập trình cho dễ nhớ. Dưới đây là một số ví dụ về đặt tên thuộc tính để bạn đọc tham khảo:

Mô tả thuộc tính	Tên thuộc tính	Thuộc lớp Control
Họ tên	txtHoten	Textbox

Ngày sinh	txtNgaysinh	Textbox
Giới tính	chkGioitinh	Checkbox
Địa chỉ	txtDiachi	Textbox
Ảnh	pctChandung	Picture

Thiết lập thuộc tính bằng mã lệnh

Phương pháp này dùng thiết lập thuộc tính cho các đối tượng khi chương trình đang chạy (Running time), cách thiết lập như sau:

<tên đối tượng>.<tên thuộc tính> = <giá trị>

Ví dụ:

```
'thiết lập giá trị cho Text1
Text1.Text = "Nguyễn Trọng Cường"

'thiết lập phông chữ
Text1.Font = "Arial"

'thiết lập màu chữ màu xanh
Text1.ForeColor = &HC0000

'thiết lập màu nền màu vàng
Text1.BackColor = &H80FFFF
```

Một số thuộc tính hay gặp

Dưới đây là một số thuộc tính hay sử dụng khi làm việc với các đối tượng. Mỗi thuộc tính dưới đây không phải bất kỳ thuộc tính nào cũng có, chỉ đưa ra đây để tham khảo và làm việc được khi gặp phải.

Thuộc tính Caption

Để hiển thị văn bản (Text) trên đối tượng. Hầu hết các đối tượng có văn bản đi kèm đều có thuộc tính này như: Label, Checkbox, Frame, Command,.. Đặc biệt Caption của đối tượng Form là dòng chữ làm tiêu đề cho cửa sổ. Đối tượng Textbox tuy có văn bản đi kèm nhưng không có thuộc tính Caption, thay vào đó là thuộc tính Text.

Thuộc tính Font

Để hiển thị phông chữ trên đối tượng. Thuộc tính này thường xuất hiện với những đối tượng có chữ (Text) đi kèm. Một trong những phông chữ hệ thống tiếng Việt sử dụng quen thuộc là Ms Sans serife hoặc Microsoft Sans serife .

Thuộc tính Alignment

Canh lề văn bản của đối tượng. Thường xuất hiện ở những đối tượng có văn bản đi kèm. Có 3 giá trị có thể thiết lập là:

Giá trị	Tác dụng
0 – Left Justify	Canh lề trái
1 – Right Justify	Canh lề phải
2 – Center	Canh lề vào giữa

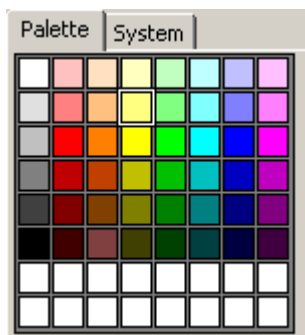
Thuộc tính Appearance

Chọn cách thức hiển thị đối tượng theo kiểu nào? Có 2 cách thức để chọn lựa:

Giá trị	Tác dụng
0 – Flat	Kiểu mảnh, phẳng
1 – 3D	Kiểu 3 chiều, có gờ nổi

Thuộc tính Backcolor

Thiết lập màu nền hiển thị trên đối tượng. Có đa màu sắc để chọn lựa trên bảng màu khi nhấn chuột lên thuộc tính này:



Thuộc tính ForeColor

Thiết lập màu chữ hiển thị trên đối tượng. Cách làm việc như với thuộc tính Backcolor.

Thuộc tính **Enable**

Thuộc tính này để cho phép làm việc hay không được phép làm việc (cấm) trên đối tượng nào đó. Điều này thể hiện rất rõ ràng khi sử dụng các phần mềm máy tính, có những lúc một nút nào đó mờ đi không thể nháy chuột lên được – lúc đó là khi đối tượng đang bị thiết lập thuộc tính **Enable = False**. Ngược lại những lúc làm việc được bình thường trên một đối tượng, tức là thuộc tính Enable của đối tượng đó đang là True.

Thuộc tính **Visible**

Thuộc tính này cho phép hiển thị (**Visible = True**) hoặc ẩn (**Visible = False**) một đối tượng nào đó khi chương trình đang chạy. Khi đối tượng bị Visible = False, thực tế đối tượng vẫn tồn tại trên form, vẫn hoạt động, chỉ khác một điều là nó hiển thị dưới dạng ẩn, người dùng không nhìn thấy được.

Thuộc tính **Left**

Thiết lập tọa độ trái của đối tượng trên Form (trục ox nếu coi form như một tọa độ cực). Giá trị là một số nguyên, cho biết tọa độ trái của đối tượng (0 là mép trái của Form, lớn hơn 0 hiển thị tăng dần về bên phải form, nhỏ hơn 0 hiển thị tăng dần về bên trái form)

Thuộc tính **Top**

Thiết lập tọa độ trên của đối tượng trên Form (trục oy nếu coi form như một tọa độ cực). Giá trị là một số nguyên, cho biết tọa độ trên của đối tượng (0 là mép trên cùng của Form, lớn hơn 0 hiển thị tăng dần về phía dưới form, nhỏ hơn 0 hiển thị tăng dần về bên trên form)

Thuộc tính **Height**

Hiển thị và cho phép thiết lập chiều cao của đối tượng

Thuộc tính **Width**

Cho phép và hiển thị chiều dài của đối tượng

1.2. Tập phương thức

Phương thức (Methods) là những khả năng mà đối tượng có thể làm được. Ví dụ với đối tượng một bộ máy tính, các phương thức có thể là:

Tên phương thức	Tác dụng
Turn On	Bật nguồn điện máy tính. Khi đó máy tính sẽ ở trạng thái sẵn sàng khởi động.
Restart	Khởi động lại máy tính. Khi đó máy tính sẽ thực hiện tắt rồi bật lại và thực hiện các thủ tục cần thiết để khởi động máy.
Running	Để chạy một chương trình nào đó, giả sử hệ điều hành yêu cầu mở chương trình soạn thảo văn bản WindWord chẳng hạn

Mỗi đối tượng có thể có những tập phương thức khác nhau tùy thuộc vào đặc trưng của từng loại đối tượng.

Khi gọi một phương thức ra làm việc, tức là đã yêu cầu đối tượng thực hiện một số công việc đã được lập trình trước. Việc gọi một phương thức giống như việc gọi một chương trình con dạng thủ tục xử lý một số vấn đề liên quan đến đối tượng đang làm việc. Dưới đây là một số ví dụ:

Ví dụ 1: Phương thức Clear của một hộp Textbox có tên Text1. Khi ra lệnh sau:

```
Text1.Clear
```

Máy tính sẽ thực hiện xóa giá trị có trên ô Text1, thiết lập ô này ở trạng thái rỗng (Null)

Ví dụ 2: Phương thức AddItem của hộp Combobox dùng để đưa một giá trị vào danh sách các mục chọn (Items) của hộp thả. Giả sử làm việc trên Combo có tên Combo1. Nếu ra lệnh sau:

```
Combo1.AddItem "Voi"
```

Combo1.AddItem "Sự tử"

Máy tính sẽ thực hiện thêm 2 mục *Voi* và *Sự tử* vào hộp thả Combo để có thể chọn lựa.

1.3. Tập sự kiện

Sự kiện (Events) là tập hợp các tác động có thể xảy ra đối với một đối tượng. Ví dụ: một nút lệnh Command có một tập sự kiện sau:

Tên sự kiện	Xảy ra khi
Click	Khi nhấn chuột trái lên nút lệnh
LostFocus	Khi điểm tab dời khỏi nút lệnh
KeyDown	Khi điểm tab đang nằm trên nút lệnh và bấm một phím
MouseMove	Khi di chuột qua nút lệnh

Thủ tục đáp ứng sự kiện của một đối tượng là một chương trình con được định nghĩa theo một cú pháp cho trước để xử lý theo mục đích của nhà lập trình khi sự kiện đó xuất hiện. Tên của chương trình con này được tạo bởi cấu trúc

<Tên đối tượng>_<tên sự kiện> (<các tham số cần thiết>)

Ví dụ: thủ tục đáp ứng sự kiện **Click** của nút lệnh **Command1** được khai báo như sau:

```
Private Sub Command1_Click()
    'Nội dung thủ tục viết ở đây
End Sub
```

Thủ tục trên có ý nghĩa như sau: khi chương trình đang chạy, nếu nhấn chuột lên nút lệnh có tên Command1, tức là xuất hiện sự kiện Click của nút lệnh này, khi đó thủ tục **Command1_Click** sẽ được thực hiện nếu nó tồn tại. Trong trường hợp nếu không khai báo thủ tục đáp ứng sự kiện, nếu sự kiện đó xảy ra máy tính sẽ không thực hiện gì cả. Một điều được rút ra là: nếu tạo một nút lệnh

nhưng không viết (lập trình) thủ tục đáp ứng sự kiện Click cho nút lệnh đó thì khi nhấn chuột lên nút này máy tính sẽ không làm gì cả.

Một số sự kiện hay sử dụng

Sự kiện Click

Xảy ra khi nhấn chuột trái lên đối tượng. Sự kiện này hay được sử dụng nhất ở đối tượng nút lệnh Command. Ngoài ra đối tượng Listbox và Combobox cũng hay được sử dụng

Sự kiện DbClick

Xảy ra khi nhấn đúp chuột lên đối tượng. Chú ý rằng, trước khi sự kiện DbClick xuất hiện thì sự kiện Click sẽ xuất hiện. Vì muốn nhấn kép chuột tức là người dùng phải nhấn đơn chuột ít nhất một lần.

Sự kiện Validate

Sự kiện này dùng để kiểm tra tính đúng đắn của dữ liệu. Đi kèm với sự kiện sẽ có tham số Cancel. Điều này rất thuận lợi khi người lập trình muốn kiểm tra việc nhập dữ liệu cho một ô hoặc một đối tượng nào đó. Nếu đồng ý dữ liệu đã nhập đúng hãy thiết lập tham số Cancel = False (ngầm định Cancel = False); trong trường hợp phát hiện dữ liệu nhập vào chưa hợp lệ, hãy thiết lập tham số Cancel = true, khi đó máy tính sẽ yêu cầu nhập lại dữ liệu đến khi nào thỏa mãn thì thôi. Chúng ta sẽ trở lại với sự kiện này trong phần Kỹ thuật kiểm tra tính đúng đắn dữ liệu sẽ trình bày ở chương tiếp theo.

Sự kiện LostFocus

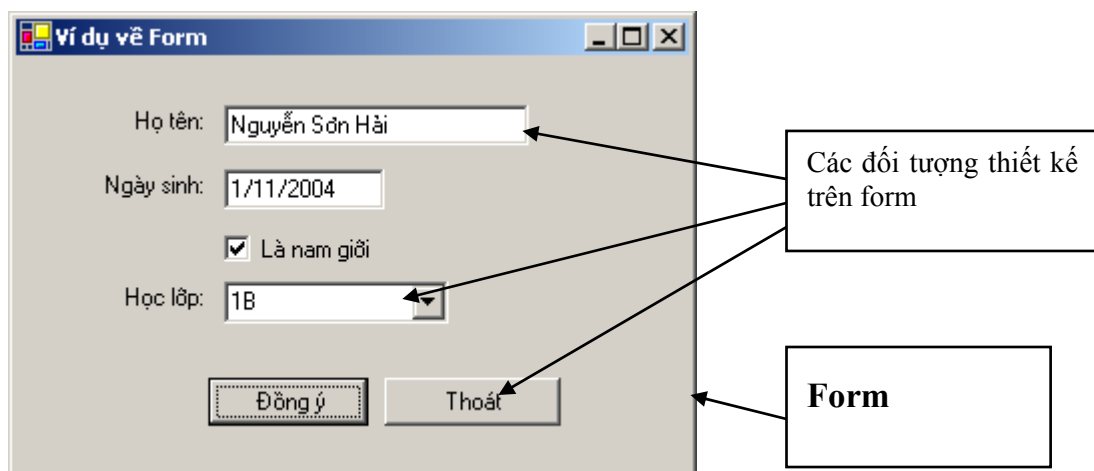
Xuất hiện khi dời điểm Tab ra khỏi đối tượng. Việc dời điểm Tab sẽ xuất hiện khi người dùng nhấn phím Tab hoặc đôi khi là phím Enter. Sự kiện này thường dùng để kiểm tra hoặc xử lý một vấn đề nào đó sau khi kết thúc nhập dữ liệu của một ô nào đó. Một ví dụ: giả sử ngay sau khi nhập xong dữ liệu ô Textbox Họ tên, người lập trình muốn tách được Họ, Đệm và Tên riêng ra để xử lý; sự kiện LostFocus sẽ rất thích hợp để làm việc này.

2. Một số thành phần điều khiển cơ bản

Phần này sẽ giới thiệu một số các điều khiển rất cơ bản và hay được sử dụng nhiều nhất mỗi khi lập trình trên VB.

2.1 Điều khiển Form

Form là đối tượng chính tạo giao diện sử dụng trong các ngôn ngữ lập trình trực quan. Nơi nhà lập trình đặt các đối tượng khác như: ô nhập dữ liệu, nút lệnh, viết các xử lý,.. để hoàn thành một giao diện người sử dụng. Dưới đây là ví dụ một form đã được xây dựng và đang sử dụng làm giao diện người dùng:



Form hoàn toàn chỉ là cái nền để thiết kế các đối tượng khác lên để tạo thành một giao diện người sử dụng (như một cửa sổ).

Một số thuộc tính của form:

Caption – tiêu đề form, được thể hiện trên thanh tiêu đề của cửa sổ. Với form trên giá trị thuộc tính Caption là **Ví dụ về Form**;

ControlBox – cho phép hiển thị (giá trị là **True**) hoặc ẩn (giá trị là **False**) các nút điều khiển form (nút Minimize, nút Restore và nút Close của cửa sổ). Với form trên, ControlBox = True;

WindowState - để chọn lựa kích cỡ của form khi bắt đầu hiển thị. Có 3 lựa chọn là: **0 – Normal** (chế độ bình thường, tức là thiết kế như thế nào khi hiển thị kích cỡ form đúng bằng như vậy); **1 – Minimized** (chế độ thu gọn về thanh tác

vụ của Windows, tức là khi hiển thị form sẽ tự động thu về (Minimize) thanh tác vụ của Windows); **2- Maximized** (chế độ phóng to toàn màn hình, tức là khi bắt đầu kích hoạt, form sẽ phóng to toàn màn hình như khi nhấn nút Maximize trên cửa sổ). Với form trên WindowState = 0 – Normal (đây cũng là chế độ ngầm định của form);

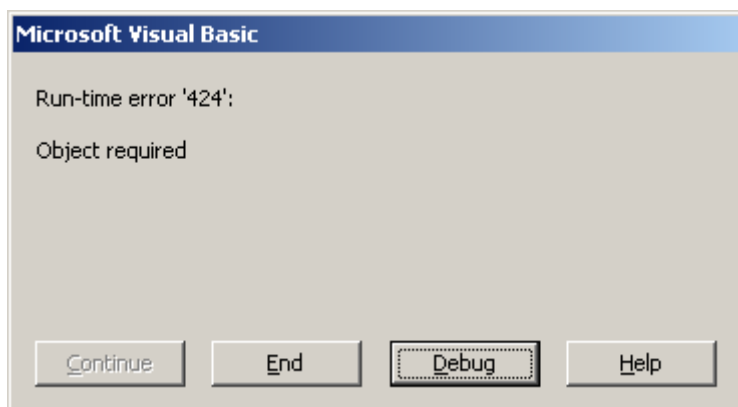
Icon - để chọn biểu tượng (icon) cho form. Muốn vậy trên máy tính phải có tệp làm biểu tượng cho form dạng *.ico.

Một số sự kiện của form

Form có rất nhiều sự kiện, ở đây sẽ giới thiệu 4 sự kiện mà hay được sử dụng nhất mỗi khi lập trình trên form:

Sự kiện Form Load

Xảy ra khi form bắt đầu kích hoạt. Là một trong những sự kiện xuất hiện đầu tiên (chỉ sau sự kiện Form_Initialize) mỗi khi kích hoạt một form. Sự kiện này thường dùng để thiết lập môi trường làm việc cho form. Tuy nhiên khi thực hiện thủ tục đáp ứng sự kiện này, một số đối tượng thiết kế trên form chưa được nạp đầy đủ, nên cũng có thể gặp phải lỗi như sau:



Sự kiện Form Activate

Xảy ra sau sự kiện Form_Load. Cũng tương tự như form_load, sự kiện này thường dùng để thiết lập môi trường làm việc cho form. Điều khác biệt là nó xảy ra sau sự kiện Form_Load và các đối tượng trên form đã được nạp và nhận dạng đầy đủ;

Sự kiện **Form Resize**

Xảy ra khi người dùng làm thay đổi kích cỡ form. Sự kiện này hay được dùng để canh lại vị trí các đối tượng trên form mỗi khi kích cỡ của form được thay đổi;

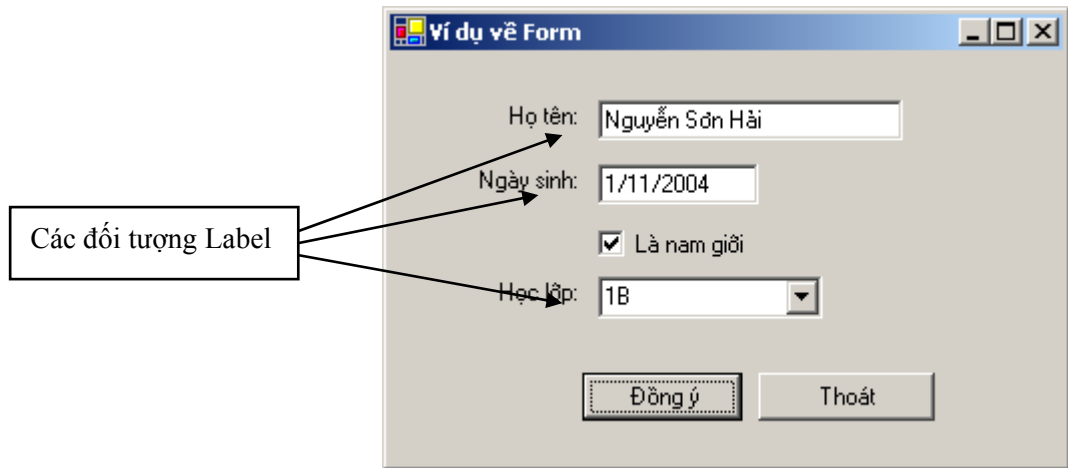
Sự kiện **Form Unload**

Xảy ra khi ngừng kích hoạt form, giải phóng form ra khỏi bộ nhớ. Nó tương đương với việc thực hiện lệnh Unload Me hoặc khi dùng chuột nhấn nút Close trên form. Đi kèm với thủ tục đáp ứng sự kiện Form_load có tham số **Cancel** kiểu Boolean (ngầm định là False). Khi Cancel = True, lệnh Unload form sẽ không được thực hiện, khi đó form sẽ vẫn ở trạng thái hoạt động. Sự kiện này thường sử dụng để kiểm tra các điều kiện cần thiết trước khi quyết định đóng một form lại, giải phóng bộ nhớ. Trong trường hợp chưa đủ điều kiện để đóng form, hãy thiết lập tham số **Cancel = True**.

2.2 Điều khiển Label



Label là một trong những điều khiển đơn giản nhất. Nó có tác dụng hiển thị một giá trị (nhãn) lên form dưới dạng văn bản (Text). Điểm đặc biệt của Label là chỉ cho phép hiển thị, không cho phép người dùng có thể sửa và xóa giá trị bằng bàn phím (khác với Textbox sau này cho phép nhập, sửa và xóa giá trị). Hình dưới minh họa việc sử dụng điều khiển Label trong một Form:



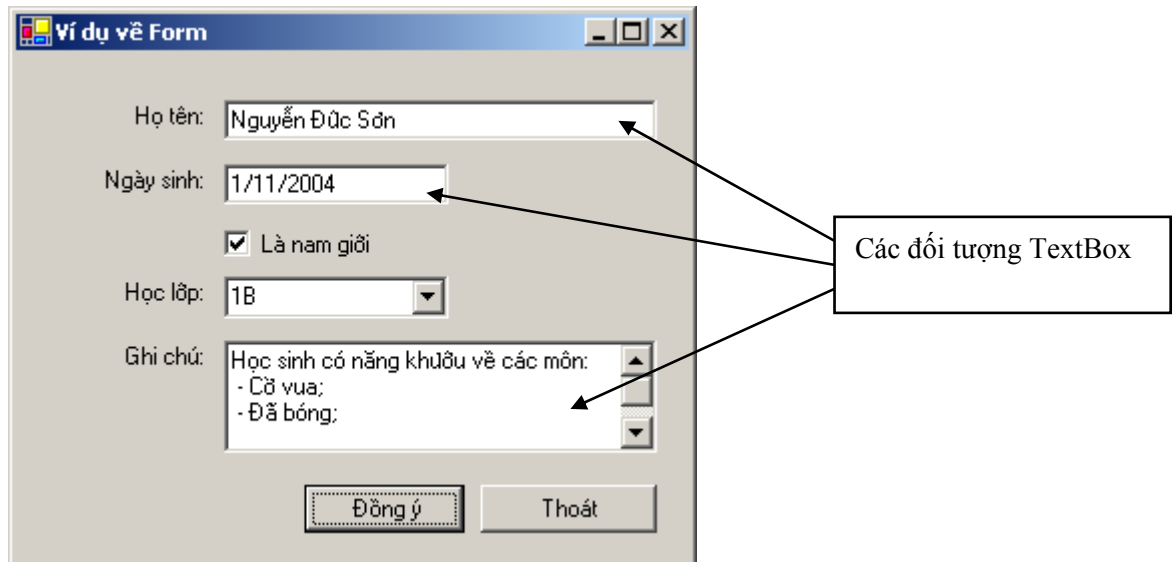
Có 4 thuộc tính thường quan tâm đến khi sử dụng Label là:

- **Caption** để nhập, thiết lập giá trị hiển thị lên Label;
- **BackColor** - thiết lập màu nền cho Label;
- **ForeColor** - thiết lập màu chữ cho Label;
- **Align** – canh lề chữ cho Label.

2.3 Điều khiển Textbox



Textbox là một trong những điều khiển được dùng khá rộng rãi. Nó thường được sử dụng vào việc hiển thị, nhập và sửa dữ liệu dạng văn bản trên các ô dữ liệu đặt trên form. Hình dưới đây minh họa việc sử dụng Textbox trên một form:



Đặc biệt, đối tượng textbox có thể hiển thị trên được nhiều dòng và như hộp *Ghi chú* form trên. Để làm được như vậy cần thiết lập thuộc tính **Multiline** = **True** và thuộc tính **ScrollBar** (để tùy chọn thanh cuộn cho Textbox) một cách phù hợp.

Giá trị trên ô TextBox được thể hiện qua thuộc tính **Text**. Thông thường, VB sẽ ngầm hiểu giá trị các ô textbox là kiểu chuỗi ký tự (Text) nên lập trình viên phải hết sức chú ý tới việc sử dụng phép tính cộng “+” kiểu số, vì máy tính sẽ hiểu là phép ghép chuỗi (cộng hai chuỗi). Khi đó có thể sử dụng hàm *Val* (hàm chuyển đổi chuỗi → số) như sau:

```
Dim num As Double

num = Val(Text1.Text) + Val(Text2.Text)

'Viết như thế này sẽ bị sai kết quả
num = Text1.Text + Text2.Text
```

Tuy nhiên trong VB nếu chỉ viết nguyên tên đối tượng TextBox (ví dụ Text1 chẳng hạn), máy tính cũng hiểu là lấy giá trị của thuộc tính Text của đối tượng này. Tức là:

```
'Lệnh
num = Text1.Text + Text2.Text

'cũng tương đương với lệnh
num = Text1 + Text2
```

**Khuyến cáo:**

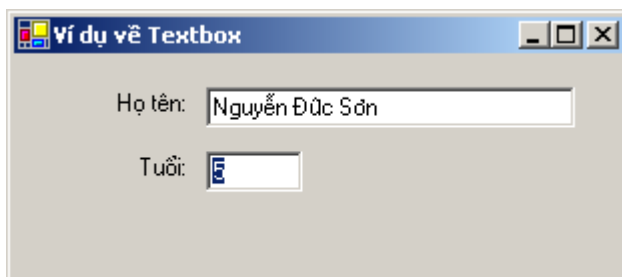
Chúng tôi khuyến cáo không nên viết tắt như vậy, sẽ tạo thành một thói quen không tốt khi sử dụng những công cụ lập trình chuyên nghiệp sau này.

Sự kiện của Textbox rất nhiều, ở đây sẽ giới thiệu 2 sự kiện hay được sử dụng nhất với Textbox:

Sự kiện Validate

Một sự kiện rất quan trọng trong việc kiểm tra tính đúng đắn dữ liệu khi nhập vào một điều khiển. Ví dụ sau đây minh họa điều đó:

Yêu cầu tạo form cho phép nhập vào Họ tên và Tuổi như sau:



Trong đó ô Họ tên không được để trống, ô Tuổi phải nhập vào một số. Giả sử thuộc tính Name (tên) của ô Họ tên là *txtHoten*, ô Tuổi là *txtTuoi*. Khi đó mã lệnh để kiểm tra tính đúng đắn 2 ô Textbox trên là 2 thủ tục đáp ứng sự kiện **Validate** như sau:

```
'kiểm tra tính đúng đắn ô Họ tên
,
Private Sub txtHoten_Validate(Cancel As Boolean)
    If txtHoten = "" Then
        Cancel = True
        MsgBox "Họ tên không được để trống !"
    End If
End Sub

'kiểm tra tính đúng đắn cho ô nhập Tuổi
,
Private Sub txtTuoi_Validate(Cancel As Boolean)
    If Not IsNumeric(txtTuoi) Then
        Cancel = True
        MsgBox "Tuổi phải là một số !"
    End If
End Sub
```

```
End If
End Sub
```

Sự kiện **LostFocus**

Sự kiện này thường dùng khi dịch điểm Tab qua đối tượng vừa làm việc. Trong ví dụ về Textbox, sẽ trình bày ví dụ về tách lấy Tên ngay sau khi Nhập xong dữ liệu cho ô Họ tên (có Name là txtHoten). Ví dụ này sử dụng đến hàm GetTen, toàn bộ mã lệnh như sau:

```
Private Sub txtHoten_LostFocus ()
    MsgBox "Tên vừa nhập là : " + GetTen(txtHoten)
End Sub

Function GetTen(st As String) As String
    Dim pos As Integer
    pos = 1
    If InStr(pos, Trim(st), " ") = 0 Then
        GetTen = st
        Exit Function
    End If
    While InStr(pos + 1, Trim(st), " ") > 0
        pos = InStr(pos + 1, Trim(st), " ")
    Wend

    GetTen = Trim(Mid(st, pos))
End Function
```

2.4 Điều khiển *CommandButton*

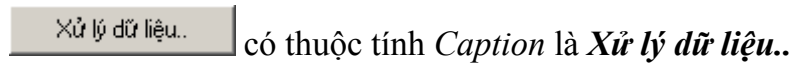


Điều khiển *CommandButton* là đối tượng mà người dùng thường gọi là nút lệnh trên các giao diện sử dụng. Chúng được thiết kế để chấp nhận thi hành một mệnh lệnh nào đó.

Một số thuộc tính thường sử dụng của **CommandButton**:

Thuộc tính **Caption**

Để thiết lập hiển thị chữ cho nút lệnh. Ví dụ nút lệnh sau đây:



Thuộc tính **Appearance**

Để chọn lựa kiểu hiển thị cho nút lệnh, có 2 kiểu lựa chọn: *0 – Flat* (kiểu phẳng) và *1 – 3D* (kiểu ba chiều – thông thường hay dùng kiểu 3D).

Thuộc tính **Cancel**

Nếu thuộc tính *Cancel* của một nút lệnh là *True*, khi form chứa nút lệnh đang kích hoạt, nếu bấm phím Esc (Escape), ngay lập tức sự kiện Click nút lệnh này được thi hành. Thông thường thuộc tính này hay được thiết lập là *True* cho những nút *Thoát*, *Kết thúc*, *Hủy* hay *Cancel*.

Thuộc tính **Default**

Trái lại với thuộc tính *Cancel*, một nút lệnh nếu được thiết lập thuộc tính *Default* là *True*, khi form chứa nút lệnh này kích hoạt, nếu nhấn phím Enter, ngay lập tức sự kiện Click của nút lệnh này được kích hoạt. Thông thường *Default* được thiết lập là *True* cho các nút *Đồng ý* hoặc *Ok*.



Chú ý:

Trên một form, không thể thiết lập đồng thời 2 thuộc tính *Cancel* và *Default* cho một nút lệnh là *True*.

Thuộc tính **Picture**

Để nạp một ảnh làm nền cho nút lệnh. Để thuộc tính này có hiệu lực, phải thiết lập giá trị thuộc tính *Style* của nút lệnh là *Graphic*.

Sự kiện của nút lệnh rất khiêm tốn, hầu hết lập trình chỉ sử dụng sự kiện *Click* tức là được kích hoạt khi nhấn chuột trái lên nút lệnh.

2.5 Điều khiển Picture



Picturebox là điều khiển cho phép hiển thị và xử lý một số thao tác về ảnh lên form như: phóng to, thu nhỏ ảnh (Zooming); cuộn ảnh (Scrolling); lật ảnh (flipping); ...

Một số thuộc tính của PictureBox

Thuộc tính Picture

Để chỉ đến tệp ảnh cần hiển thị lên PictureBox. Có thể thiết lập thuộc tính này trên hộp thoại Properties trong chế độ thiết kế form. Hoặc dùng mã lệnh: giả sử có Picture1 trên form, một tệp ảnh nằm ở c:\windows\setup\setup.bmp. Thủ tục **LoadPicture** sẽ hiển thị ảnh này lên điều khiển picturebox trên form:

```
Picture1.Picture = LoadPicture("c:\windows\setup\setup.bmp")
```

Khi đó lệnh để xoá bỏ dữ liệu trên Picture1 khi không cần thiết như sau:

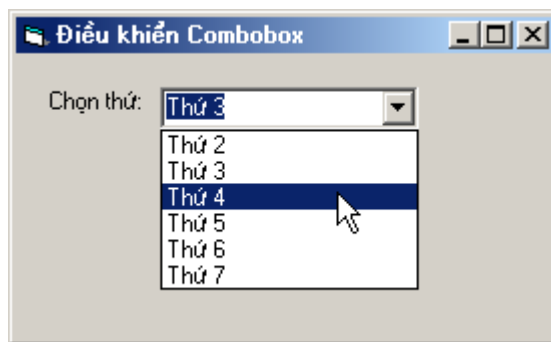
```
Picture1.Picture = LoadPicture("")
`hoặc
Set Picture1.Picture = Nothing
```

3. Nhóm điều khiển trình bày giao diện

3.1 Điều khiển ComboBox



Điều khiển Combobox là dạng điều khiển trình bày giao diện, giúp việc chọn lựa một giá trị trên một danh sách trở nên dễ dàng. Khi sử dụng, điều khiển này có dạng như sau:



Cấu trúc của một Combobox bao gồm một hộp chọn, trong đó có thể có một danh sách các mục để chọn (Items). Làm việc trên Combobox tức là làm việc trên các Items. Bao gồm các công việc như:

- Làm thế nào để thêm các Items vào danh sách mục chọn?
- Làm thế nào để xóa bỏ mục chọn ra khỏi danh sách?
- Làm thế nào để định vị, di chuyển, ... các mục chọn?

Tất cả có thể thực hiện được thông qua tập các thuộc tính và phương thức của điều khiển này.

Tập hợp các phương thức:

Phương thức AddItem

Để thêm một Item lên danh sách. Cú pháp như sau:

```
<tên Combobox>.AddItem <giá trị>
```

Trong đó:

- <tên combobox> là giá trị thuộc tính Name của Combo;
- <giá trị> là nhãn hiển thị của Item mới thêm vào.

Với yêu cầu như Combobox ở trên, phải thực hiện lần lượt 7 câu lệnh AddItem như sau:

```
Combo1.AddItem "Thứ 2"  
Combo1.AddItem "Thứ 3"  
Combo1.AddItem "Thứ 4"  
Combo1.AddItem "Thứ 5"  
Combo1.AddItem "Thứ 6"  
Combo1.AddItem "Thứ 7"  
Combo1.AddItem "Chủ nhật"
```

Hơn nữa, tại cửa sổ Properties cũng có thể khởi tạo được danh sách các Items cho một Combobox bằng cách gõ chúng vào thuộc tính List.

Phương thức RemoveItem

Dùng để gỡ bỏ một Item nào đó ra khỏi danh sách. Muốn vậy, phải xác định được Item có số thứ tự bao nhiêu sẽ được gỡ ra (số thứ tự này được tính từ 0). Cú pháp như sau:

```
<tên Combobox>.RemoveItem <stt>
```

Trong đó:

- <tên combobox> là giá trị thuộc tính Name của Combo cần gỡ Item;
- <stt> là số thứ tự của Item cần gỡ bỏ, STT này được tính từ 0.

Ví dụ: muốn gỡ bỏ Item có giá trị *Thứ 4* ở trên, phải thực hiện câu lệnh gỡ bỏ Item với STT là 3 như sau:

```
Combo1.RemoveItem 3
```

Phương thức Clear

Phương thức Clear dùng để xóa bỏ toàn bộ các Items đang có trên một Combobox. Sau khi gọi phương thức này, Combobox trở nên rỗng trắng.

```
Combo1.Clear
```

Một số các thuộc tính hay sử dụng

Thuộc tính ListCount

Cho biết tổng số các Item đang có trên Combobox

Thuộc tính List

Để truy cập tới một Item nào đó. Vị trí của Item cần truy cập để được thể hiện qua số thứ tự của nó. Ví dụ muốn thiết lập giá trị của Item có số thứ tự 3 từ *Thứ 4* sang *Thứ tư*, có thể sử dụng thuộc tính List thông qua câu lệnh sau:

```
Combo1.List (3) = "Thứ tư"
```

Thuộc tính ListIndex

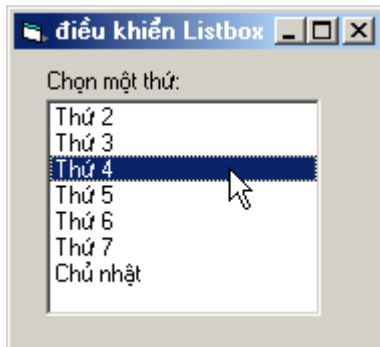
Cho biết số thứ tự của Item đang được chọn trên Combobox. Ví dụ sau cho phép hiển thị số thứ tự của Item đang được chọn trên Combo1:

```
Msgbox "Đang chọn Item thứ: " + Str(Combo1.ListIndex)
```

3.2 Điều khiển ListBox



Điều khiển Listbox là dạng điều khiển trình bày giao diện, giúp việc chọn lựa một giá trị trên một danh sách trở nên dễ dàng. Khi sử dụng, điều khiển này có dạng như sau:



Cấu trúc của một Listbox bao gồm một bảng chọn, trong đó có thể có một danh sách các mục để chọn (Items). Cấu trúc này hết như cấu trúc của Combobox

3.3 Điều khiển CheckBox



Điều khiển Checkbox chỉ có thể hiện được 2 trạng thái: được chọn (Checked) và không được chọn (Unchecked), cho nên điều khiển này thường được dùng để biểu diễn dữ liệu dạng Logical (True hoặc False).

Một số thuộc tính hay dùng

Thuộc tính Caption

Để biểu diễn nhãn hiển thị trên điều khiển

Thuộc tính Value

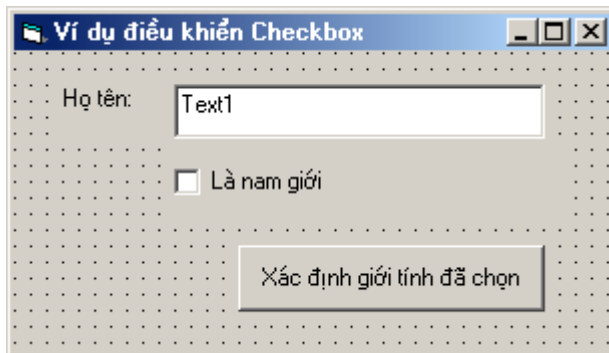
Để biết được trạng thái (dữ liệu) trên điều khiển. Có 3 trạng thái là:

- 0 – *Unchecked* – điều khiển không được chọn;

- 1 – *Checked* - điều khiển đã được chọn (tích);
- 2 – *Grayed* - trạng thái lờng. Trạng thái này thường xuất hiện lúc đầu tiên, khi điều khiển Checkbox chưa được khởi tạo là Checked hay Unchecked. Sau khi đã xác định Checked hay Unchecked, sẽ không tồn tại giá trị Grayed nữa.

Ví dụ dưới đây minh họa cách sử dụng điều khiển Checkbox vào việc biểu diễn dữ liệu giới tính cho nhân sự. Nếu là nam giới, sẽ được Checked, trái lại là nữ giới. Nút lệnh xác định kết quả trả lời giá trị điều khiển checkbox đang chọn.

Bước 1: thiết kế form như sau:



Bước 2: viết mã lệnh cho các sự kiện Click của nút lệnh *Xác định giới tính đã chọn* như sau:

```

Private Sub Command1_Click()
    '-----
    'căn cứ vào thuộc tính Value để xác định giới tính
    '
    If Check1.Value = Checked Then
        MsgBox "Là nam giới !"
    Else
        MsgBox "Là nữ giới !"
    End If
End Sub

```

3.4 Điều khiển *OptionButton*



Điều khiển *OptionButton* thường dùng trong những tình huống chọn lựa lấy một kết quả từ danh sách các lựa chọn có sẵn.

Có 2 thuộc tính quan trọng:

Thuộc tính **Caption**

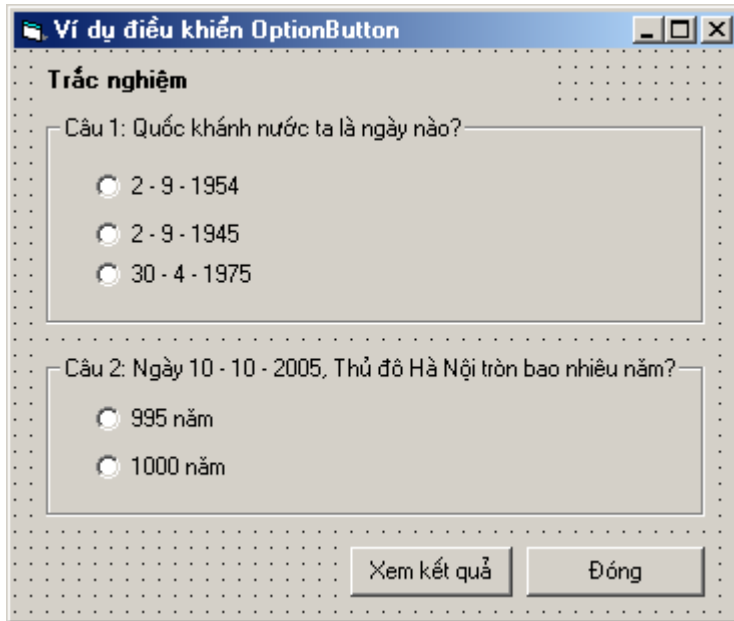
Để thiết lập nhãn hiển thị cho *OptionButton*

Thuộc tính **Value**

Để xác định giá trị của *OptionButton*. Bằng *True* nếu *OptionButton* được chọn, trái lại là *False*.

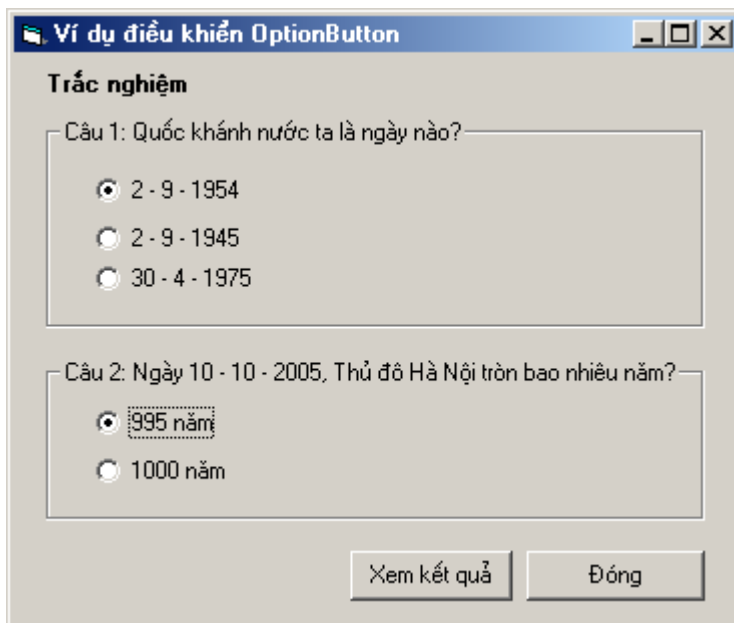
Đặc biệt, một tập hợp các điều khiển *OptionButton* trên một *Frame* chỉ có thể tồn tại nhiều nhất 1 điều khiển *OptionButton* nhận giá trị *True*. Ví dụ sau đây sẽ sử dụng các điều khiển *OptionButton* để thực hiện thiết kế 2 câu hỏi thi trắc nghiệm trên máy tính:

Bước 1: Thiết kế form như sau:



Trong đó: tên của các OptionButton trên *Câu 1* và *Câu 2* lần lượt đặt theo thứ tự từ trên xuống dưới là: *opt11*, *opt12*, *opt13* và *opt21*, *opt22*.

Đáp án đúng của câu 1 nếu Chọn *opt12*; đáp án đúng Câu 2 nếu chọn *opt21* (hình dưới).



Bước 2: viết lệnh cho thủ tục đáp ứng sự kiện Click của nút *Kết quả* là:

```
Private Sub cmdKetqua_Click()
    Dim ketqua As Integer

    '-----
    'khởi tạo biến kết quả, ban đầu chưa trả lời đúng câu nào
    '
```



```

ketqua = 0

'-----
'nếu chọn opt12, đã trả lời đúng 1 câu
'
If opt12.Value = True Then ketqua = ketqua + 1

'-----
'nếu chọn opt21, đã trả lời đúng thêm 1 câu nữa
'
If opt21.Value = True Then ketqua = ketqua + 1

'-----
'in kết quả
'

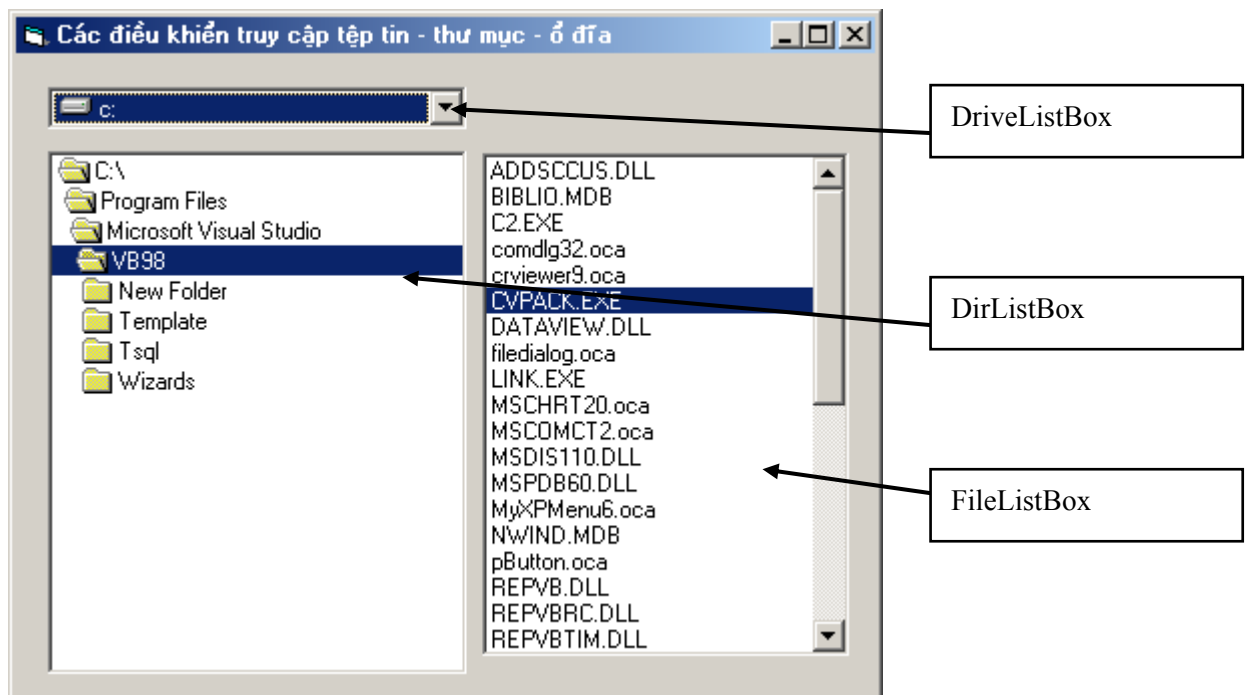
MsgBox "Bạn trả lời đúng " + Str(ketqua) + "/ 2 câu !"

End Sub

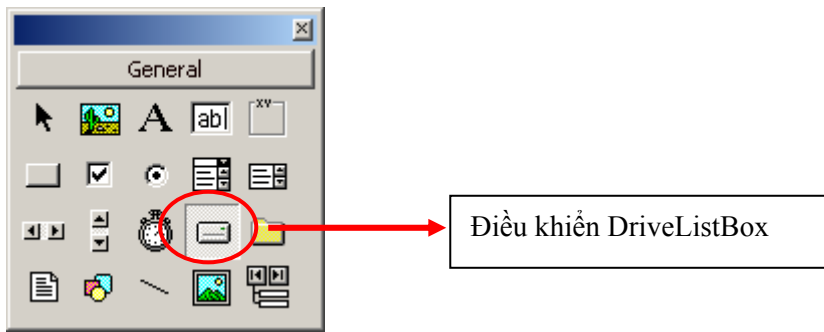
```

4. Nhóm điều khiển làm việc thư mục, tệp tin

Với VB, thông qua các đối tượng điều khiển tập tin, thư mục và ổ đĩa, việc lập trình truy cập tới các thư mục, tệp tin, ổ đĩa trên máy tính là chuyện đơn giản. Phần dưới đây trình bày cách sử dụng hiệu quả các điều khiển này trong dự án VB.



4.1 Điều khiển DriveListBox



Điều khiển DriveListBox dùng để tạo giao diện truy cập đến danh sách các ổ đĩa tìm thấy trên máy tính.

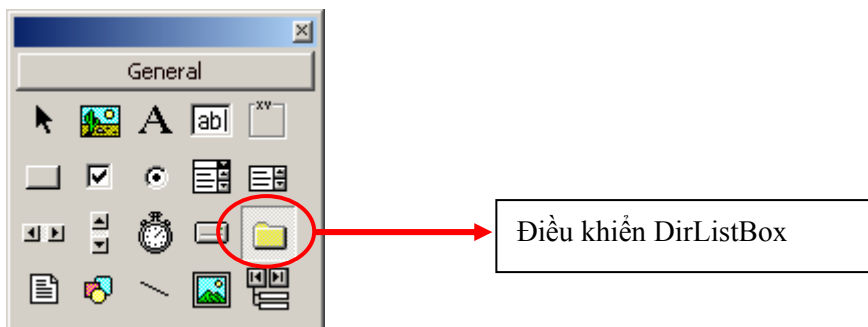
Do là DriveListBox nên hình dáng cũng như một số thuộc tính của điều khiển này giống với điều khiển ComboBox trình bày ở trên (mục 3). Dưới đây là một số thuộc tính hay được sử dụng:

List - để truy cập tới từng ổ đĩa trên DriveListBox. Ví dụ **Drive1.List(1)** – cho biết đường dẫn của ổ đĩa thứ 2 (thứ tự được tính từ 0) trên Drive1.

ListCount- cho biết tổng số ổ đĩa truy cập được trên máy tính.

ListIndex - cho biết số thứ tự của ổ đĩa đang chọn.

4.2 Điều khiển DirListBox



Điều khiển DirListBox dùng để tạo giao diện truy cập đến cây thư mục trên máy tính. Dưới đây là một số thuộc tính hay được sử dụng:

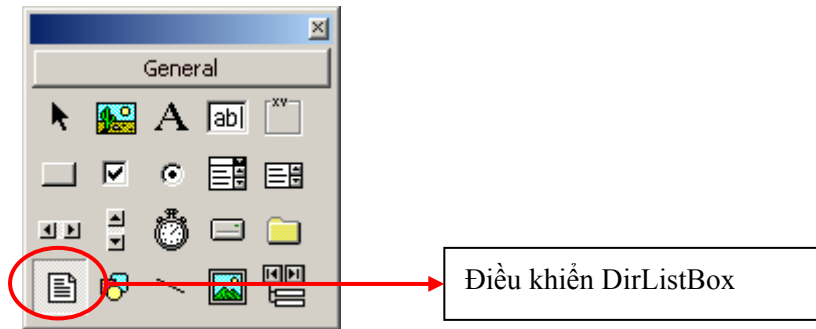
List - để truy cập tới từng thư mục con (Item) trên cây thư mục. Ví dụ **Dir1.List(2)** – cho biết đường dẫn của thư mục thứ 3 (thứ tự được tính từ 0) từ trên xuống trên cây thư mục Dir1.

ListCount- cho biết tổng số thư mục có trên cây thư mục DirListBox.

ListIndex - cho biết số thứ tự của thư mục con đang được chọn trên cây thư mục.

Path - cho biết đường dẫn đầy đủ thư mục mẹ của thư mục con đang chọn trên cây thư mục (thư mục gốc là ổ đĩa đang chọn).

4.3 Điều khiển FileListBox



Điều khiển FileListBox dùng để tạo giao diện truy cập đến danh sách các tệp tin (files) trên một thư mục nào đó. Dưới đây là một số thuộc tính hay được sử dụng:

List - để truy cập tới từng tệp tin trên danh sách. Ví dụ **File1.List(2)** – cho biết tệp tin thứ 3 (thứ tự được tính từ 0) trong danh sách các tệp tin đang hiển thị ở File1..

ListCount- cho biết tổng số tệp tin đang hiển thị trên FileListBox.

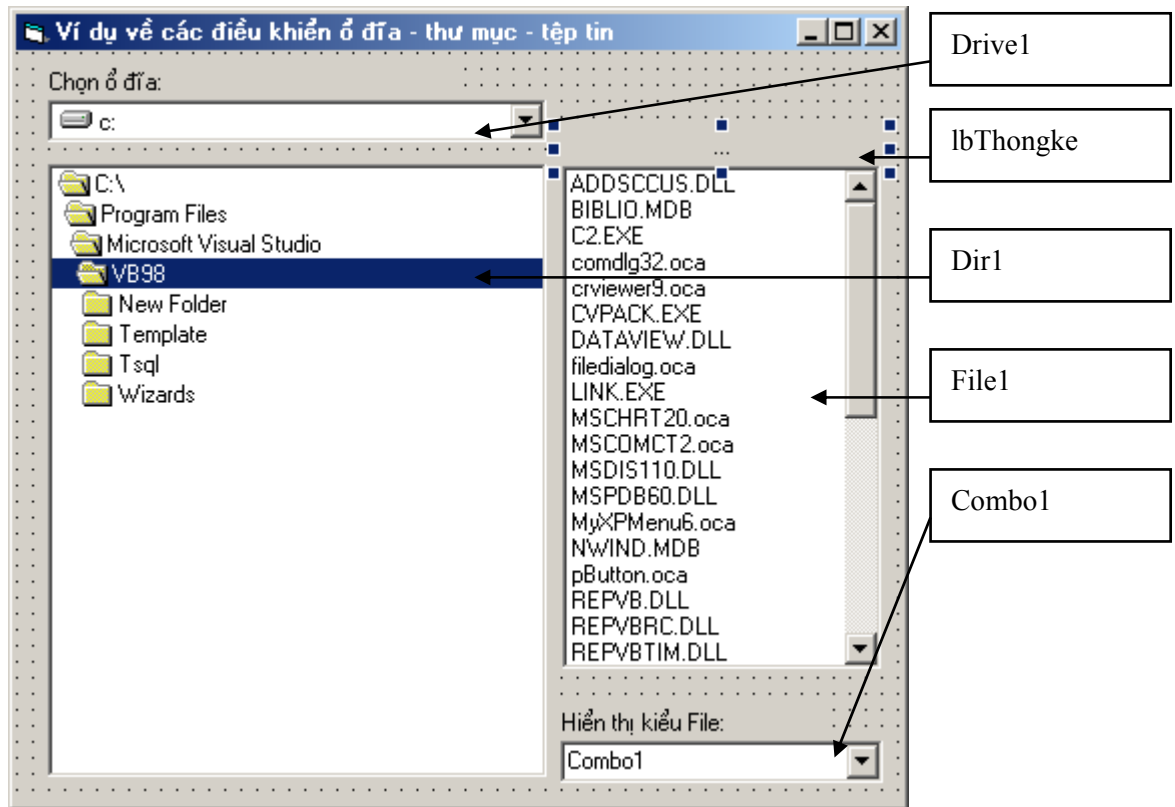
ListIndex - cho biết số thứ tự của tệp tin đang được chọn trên FileListBox.

Path - đường dẫn tới thư mục cho FileListBox làm việc.

Pattern- thiết lập đặt lọc cho các tệp tin hiển thị trên FileListBox.

4.4 Ví dụ tổng hợp

Thiết kế một giao diện như sau:



Khi chọn một ổ đĩa, cây thư mục của ổ đĩa đó xuất hiện trên một DirListBox có tên *Dir1*;

Khi chọn một thư mục trên *Dir1*, danh sách các tệp tin của thư mục đó với phần mở rộng của tệp tin thỏa mãn Combo1 sẽ được hiển thị lên một FileListBox có tên *File1*. Đồng thời máy tính sẽ đếm số tệp tin đang hiển thị trên *File1* ra một Label có tên *lbThongke*.

Toàn bộ mã lệnh cho form trên như sau:

```

Private Sub Form_Load()
    '-----
    'Add các kiểu file cần lọc lên Combo1
    '
    Combo1.AddItem "*.*"
    Combo1.AddItem "*.doc"
    Combo1.AddItem "*.txt"
    Combo1.AddItem "*.exe"

    '-----
    'ngâm định sẽ hiển thị *.*
    '
    Combo1.ListIndex = 0

End Sub

Private Sub Drive1_Change()

```

```

On Error GoTo Thoat

    '-----
    'Thiết lập ổ đĩa cho Dir1 là Drive1
    '
    Dir1.Path = Drive1

    Exit Sub

    '-----
    'Trường hợp ổ đĩa không sẵn sàng, hiển thị một thông báo
lỗi
    '
    Thoat:
        MsgBox "Không truy cập được ổ đĩa này!", vbCritical
End Sub

Private Sub Dir1_Change()
    '-----
    'gọi thủ tục hiển thị tệp tin lên File1
    '
    Call Show_file

End Sub

Private Sub Combo1_Click()
    '-----
    'gọi thủ tục hiển thị tệp tin lên File1
    '
    Call Show_file

End Sub

Sub Show_file()
    '-----
    'Thiết lập đường dẫn cho File1 là thư mục đang chọn ở Dir1
    '
    File1.Path = Dir1

    '-----
    'Thiết lập thuộc tính đặt lọc cho File1 ở Combo1
    '
    File1.Pattern = Combo1

    '-----
    'thống kê tổng số File hiển thị đưa ra lbThongke
    '
    lbThongke.Caption = "có " + Str(File1.ListCount) + "
file(s)"

End Sub

```

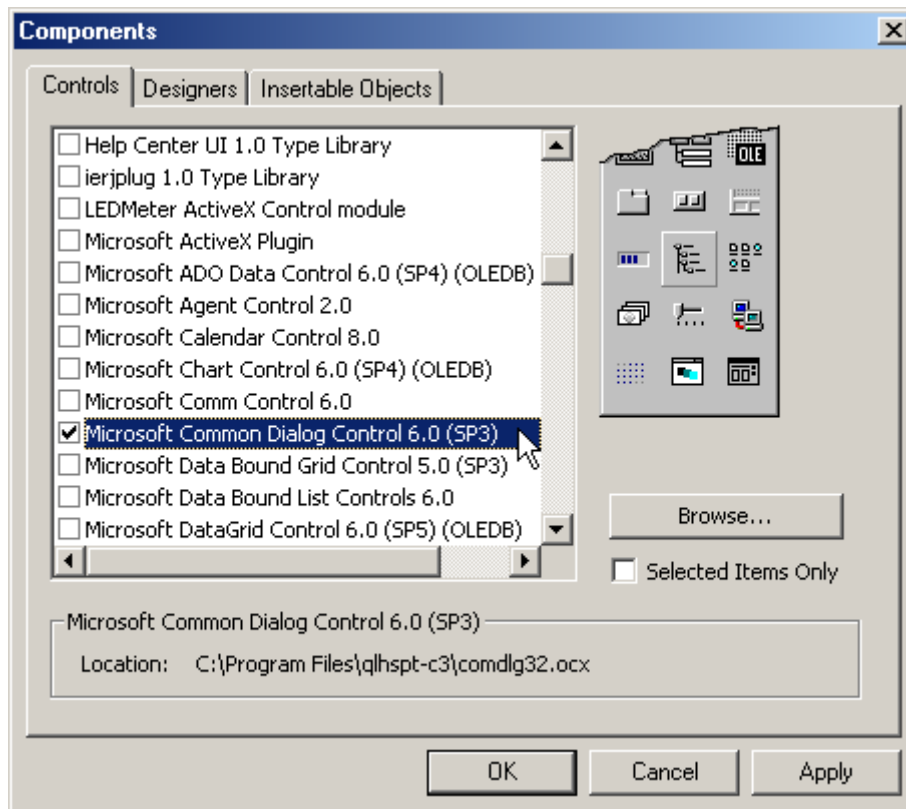
5. Một số điều khiển chung

5.1 Điều khiển MS Common Dialog

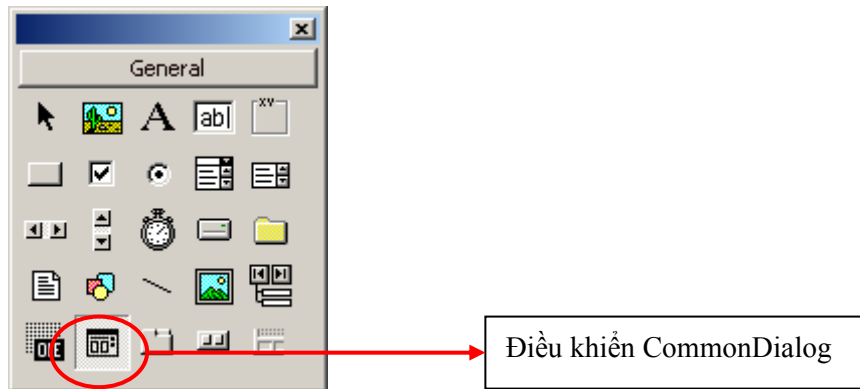
MS Common Dialog là điều khiển giúp gọi ra các hộp thoại: mở tệp tin (Open file), ghi tệp tin (Save file), mở hộp thoại phông chữ (Font), mở hộp thoại chọn màu hệ thống.

Thông thường, điều khiển này chưa có sẵn trên thanh công cụ Toolbox, muốn có làm như sau:

Mở thực đơn **Project | Components**, hoặc nhấn tổ hợp phím nóng **Ctrl + T**, một hộp thoại xuất hiện. Hãy tìm đến mục *Microsoft Common Control 6.0* và chọn như sau:



Đến đây có thể sử dụng điều khiển này trên thanh công cụ:



Một số thuộc tính thường sử dụng:

Thuộc tính CancelError

Nếu thiết lập là False, để bỏ qua lỗi trong quá trình mở hộp thoại hệ thống Common Dialog

Thuộc tính DefaultExt

Để thiết lập kiểu tệp ngầm định sẽ lọc trên hộp thoại.

Thuộc tính DialogTitle

Để thiết lập tiêu đề cửa sổ hộp thoại hệ thống khi mở ra.

Thuộc tính FileName

Để đặt tên tệp tin cần mở hoặc ghi trên hộp thoại Common Dialog. Hơn nữa thuộc tính này dùng để lấy giá trị là tệp tin đã được chọn trên hộp thoại Open File.

Thuộc tính Filter

Để tạo danh sách đặt lọc tệp tin

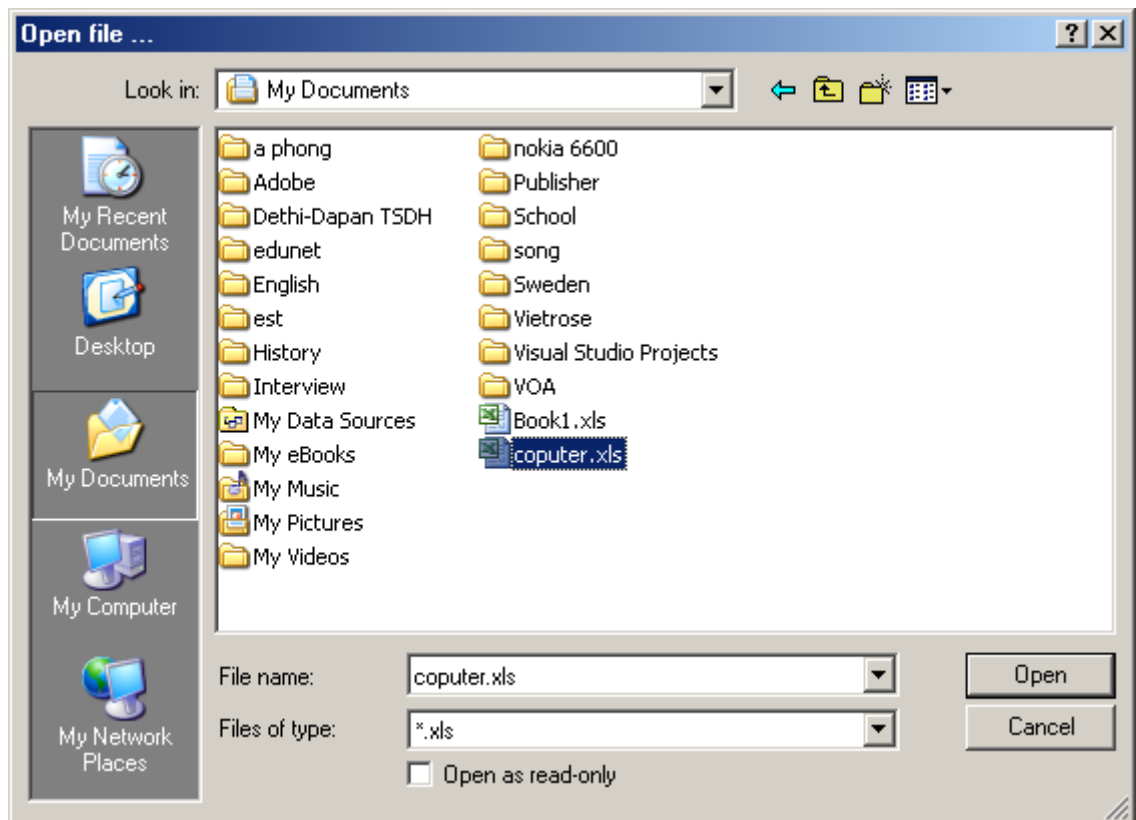
Thuộc tính InitDir

Để thiết lập thư mục ngầm định trở tới khi hộp thoại Open File hoặc Save được mở ra.

Một số phương thức của điều khiển Common Dialog

Phương thức ShowOpen

Để hiển thị Common Dialog dưới dạng hộp thoại mở tệp tin (Open File).
Dạng hộp thoại này như sau:

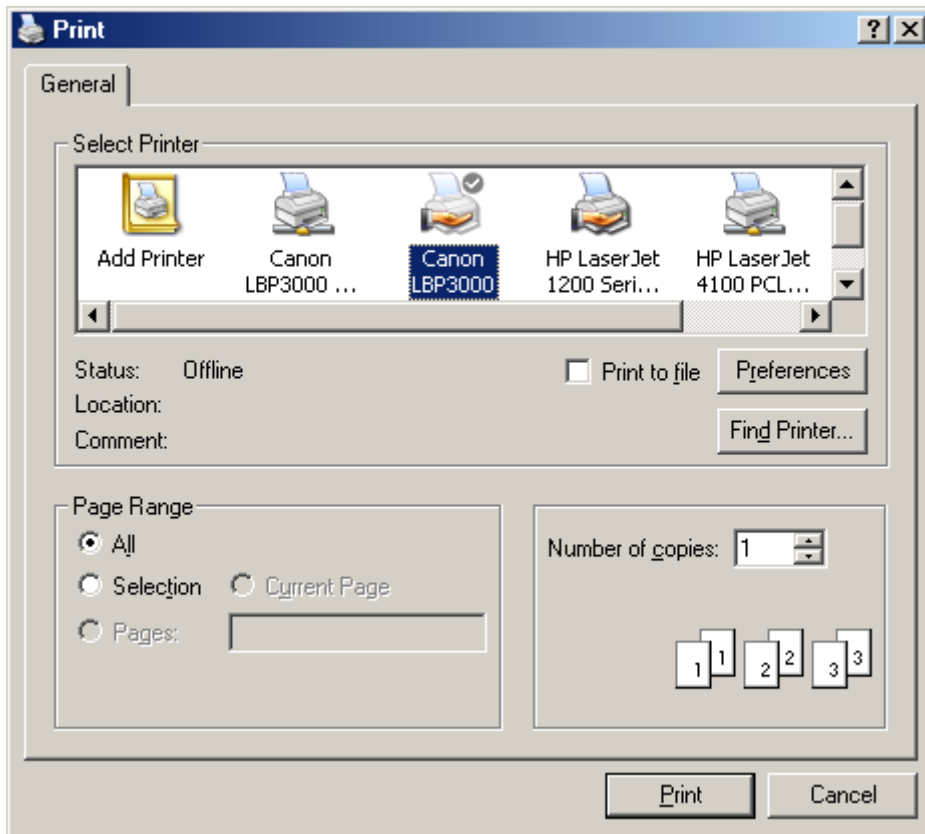


Phương thức ShowSave

Để hiển thị Common Dialog dưới dạng hộp thoại ghi tệp tin (Save As).
Dạng hộp thoại này giống như hộp Open File.

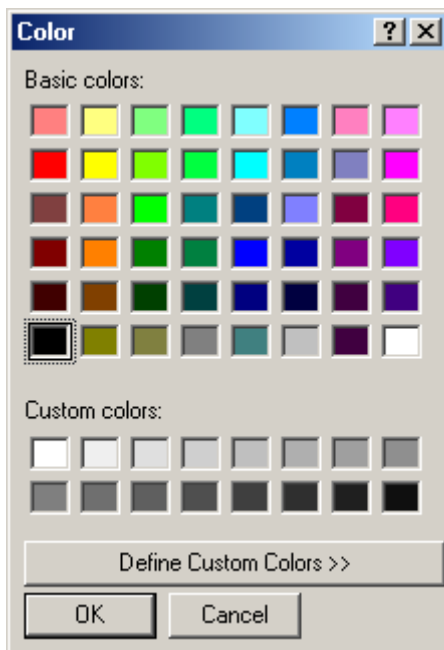
Phương thức ShowPrinter

Để hiển thị Common Dialog dưới dạng hộp thoại tùy chọn máy in hệ thống.
Dạng hộp thoại này như sau:



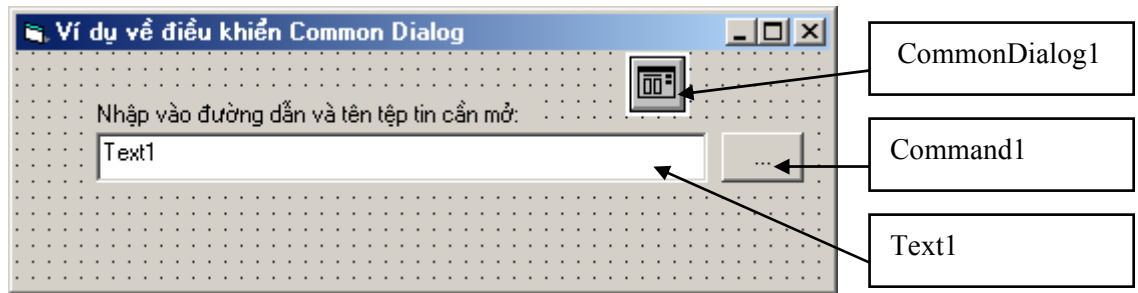
Phương thức ShowColor

Để hiển thị Common Dialog dưới dạng hộp thoại màu sắc hệ thống. Dạng hộp thoại này như sau:



Ví dụ:

Thiết kế form như sau:



Mã lệnh cho sự kiện Click của nút lệnh *Command1* như sau:

```
Private Sub Command1_Click()

    CommonDialog1.CancelError = False
    CommonDialog1.DefaultExt = "*.xls"
    CommonDialog1.DialogTitle = "Open file ..."
    CommonDialog1.Filter = "*.xls|*.xls|*.doc"
    CommonDialog1.InitDir = "c:\setup"

    CommonDialog1.ShowOpen

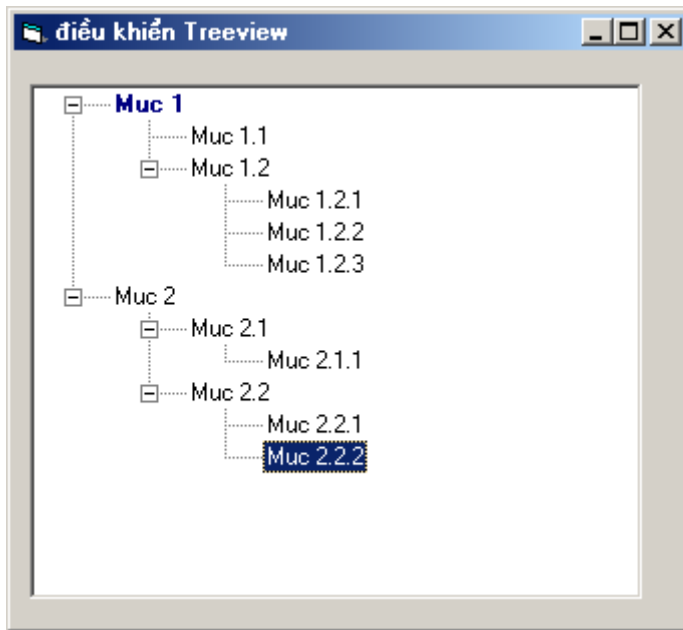
    Text1.Text = CommonDialog1.FileName

End Sub
```

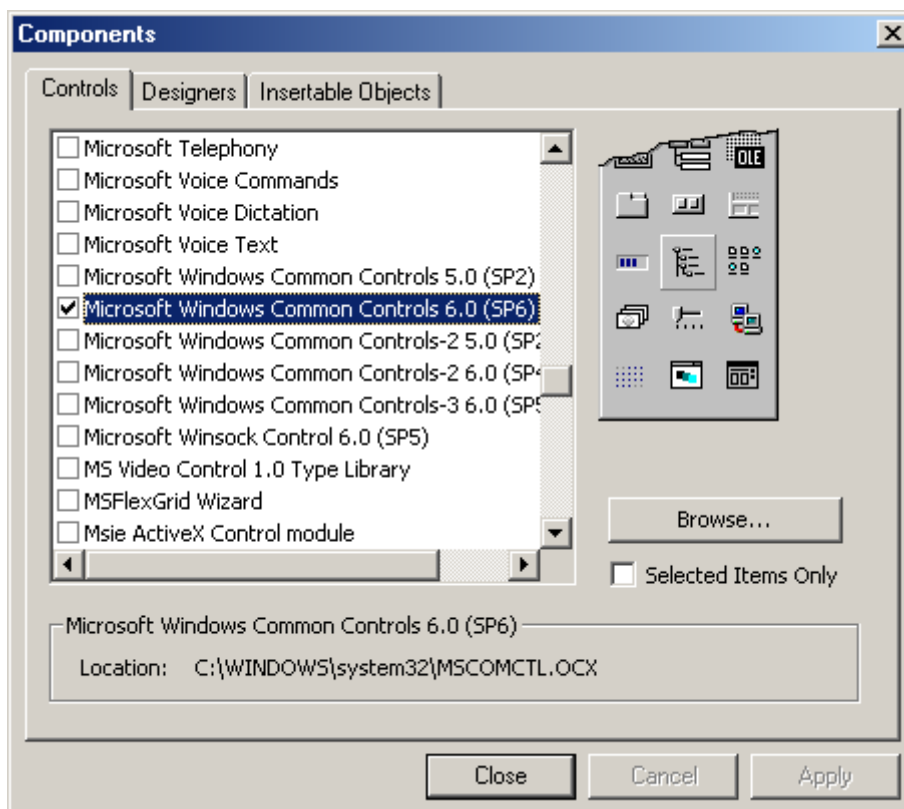
5.2 Điều khiển Treeview



Điều khiển Treeview là đối tượng có thể thể hiện dữ liệu dưới dạng cây (Tree). Điều này rất thuận tiện khi dữ liệu cần trình bày có cấu trúc phân cấp. Hình dưới là một Treeview đang trình bày cây phân cấp các mục:



Để sử dụng được Treeview, cần gài đặt thư viện nhóm điều khiển dạng **Microsoft Windows Common Controls 6.0** trên cửa sổ Components như hình dưới:



Trước hết tìm hiểu cấu trúc một treeview:

Cấu trúc một Treeview bao gồm tập hợp các nút (Nodes). Có 2 loại nút là: nút gốc (Root) và nút con (Sub-Node);

Mỗi nút sẽ bao gồm một tập hợp các thuộc tính như sau:

- Thuộc tính **Text** - để thiết lập nhãn hiển thị cho nút đó. Ví dụ trên, các giá trị Mục 1, Mục 1.1 hoặc Mục 1.2.1, ... chúng là giá trị của thuộc tính Text của các nút tương ứng;
- Thuộc tính **Key** - để thiết lập giá trị khóa cho nút . Mỗi nút có thể có một giá trị khóa, giá trị này là duy nhất (không được phép tồn tại 2 nút có giá trị khóa như nhau). Ký tự đầu tiên của một nút phải là một chữ cái. Thuộc tính Key thường dùng để xác định được dấu hiệu cho biết người dùng đang chọn nút nào trên treeview. Hơn nữa, Key bắt buộc phải có nếu nút đó sẽ tồn tại các nút con của nó;
- Thuộc tính **ForeColor** - thiết lập màu sắc cho Text của nút;
- Thuộc tính **Bold** - thiết lập kiểu chữ đậm cho nút;
- Thuộc tính **Tag** – thông thường để gán một giá trị nào đó cho từng nút. Khác với Key, giá trị của Tag có thể trùng nhau và có thể gán đa dạng về kiểu dữ liệu.
- Thuộc tính **Image** - để chọn biểu tượng ảnh cho nút. Đó là số thứ tự của ảnh trên một điều khiển ListImage đã thiết lập cho treeview này trước đó;
- Thuộc tính **Parent** - để truy cập đến nút cha của nút nào đó.

Một số thuộc tính của Treeview:

- Thuộc tính **Checkboxes** - để thiết lập dấu Checkbox trên mỗi nút;
- Thuộc tính **ImageList** - để thiết lập ImageList chứa các biểu tượng ảnh để làm biểu tượng cho các nút trên treeview;
- Thuộc tính **SelectedItem** để xác định nút đang được chọn trên treeview. Nút này sẽ được gán giá trị vào một biến kiểu Node;

Cách làm việc với Treeview:

Để làm việc với Treeview, cần khai báo ít nhất một biến kiểu Node như sau:

```
Dim nd As Node
```

Thêm một nút gốc

Nút gốc (Root) là nút không có nút cha của nó.

```
Set nd = <tên treeview>.Nodes.Add()
```

Trong đó: *nd* là biến kiểu Node đã được khai báo trước đó; *<tên treeview>* là giá trị thuộc tính Name của Treeview đang làm việc.

Tiếp theo, thông qua biến **nd** đã được gán cho nút mới để thiết lập các thuộc tính cần thiết cho nút này. Ví dụ:

```
Set nd = TreeView1.Nodes.Add()  
nd.Text = "Mục 1"  
nd.key = "m1"  
nd.Tag = 1  
nd.ForeColor = &H800000  
nd.Bold = True
```

Thêm một nút con

Nút con (Sub-Node) là nút mà luôn tồn tại một nút cha của nó. Nút cha này có thể là một nút dạng nút gốc, nhưng cũng có thể là một nút dạng nút con.

```
Set nd = <tên treeview>.Nodes.Add(<key nút cha>, 4)
```

Trong đó: *nd* là biến kiểu Node đã được khai báo trước đó; *<tên treeview>* là giá trị thuộc tính Name của Treeview đang làm việc; *<key nút cha>* là giá trị Key của nút cha.

Tiếp theo, thông qua biến **nd** đã được gán cho nút mới để thiết lập các thuộc tính cần thiết cho nút này. Ví dụ dưới đây sẽ tạo một nút con của nút **Mục 1** đã tạo ở trên:

```
Set nd = TreeView1.Nodes.Add("m1", 4)  
nd.Text = "Mục 1.1"  
nd.key = "m1.1"
```

Với hình minh họa trên, mã lệnh cho form đó như sau:

```

Private Sub Form_Load()
    Dim nd As Node

    '-----
    'Tạo nút gốc có nhãn Mục 1
    '
    Set nd = TreeView1.Nodes.Add()
    nd.Text = "Mục 1"
    nd.key = "m1"
    nd.ForeColor = &H800000
    nd.Bold = True

    '-----
    'Tạo các nút con của nút Mục 1
    '
    Set nd = TreeView1.Nodes.Add("m1", 4)
    nd.Text = "Muc 1.1"
    nd.key = "m1.1"

    Set nd = TreeView1.Nodes.Add("m1", 4)
    nd.Text = "Muc 1.2"
    nd.key = "m1.2"

    '-----
    'Tạo tiếp lớp các nút con nữa
    '
    Set nd = TreeView1.Nodes.Add("m1.2", 4)
    nd.Text = "Muc 1.2.1"

    Set nd = TreeView1.Nodes.Add("m1.2", 4)
    nd.Text = "Muc 1.2.2"

    Set nd = TreeView1.Nodes.Add("m1.2", 4)
    nd.Text = "Muc 1.2.3"

    '-----
    'Tạo nút gốc có nhãn Mục 2
    '
    Set nd = TreeView1.Nodes.Add()
    nd.Text = "Muc 2"
    nd.key = "m2"

    '-----
    'Tạo các nút con của nút Mục 2
    '
    Set nd = TreeView1.Nodes.Add("m2", 4)
    nd.Text = "Muc 2.1"
    nd.key = "m2.1"

    Set nd = TreeView1.Nodes.Add("m2", 4)

```

```
nd.Text = "Muc 2.2"  
nd.key = "m2.2"  
  
'-----  
'Tạo tiếp lớp các nút con nữa  
'  
Set nd = TreeView1.Nodes.Add("m2.2", 4)  
nd.Text = "Muc 2.2.1"  
  
Set nd = TreeView1.Nodes.Add("m2.2", 4)  
nd.Text = "Muc 2.2.2"  
  
Set nd = TreeView1.Nodes.Add("m2.1", 4)  
nd.Text = "Muc 2.1.1"  
  
End Sub  
  
Private Sub TreeView1_DblClick()  
  
    MsgBox "Đang chọn nút có Key là:" +  
TreeView1.SelectedItem.key _  
    & " Có nhãn là: " + TreeView1.SelectedItem.Text  
  
End Sub
```

Sự kiện **TreeView1_DblClick()** để hiển thị giá trị Key và Text của nút đang chọn khi nhấn kép chuột lên nút đó.

Bài tập chương 2

Bài tập 1:

Tạo form tìm ước số chung lớn nhất, bội số chung nhỏ nhất của 2 số nguyên A và B như sau:

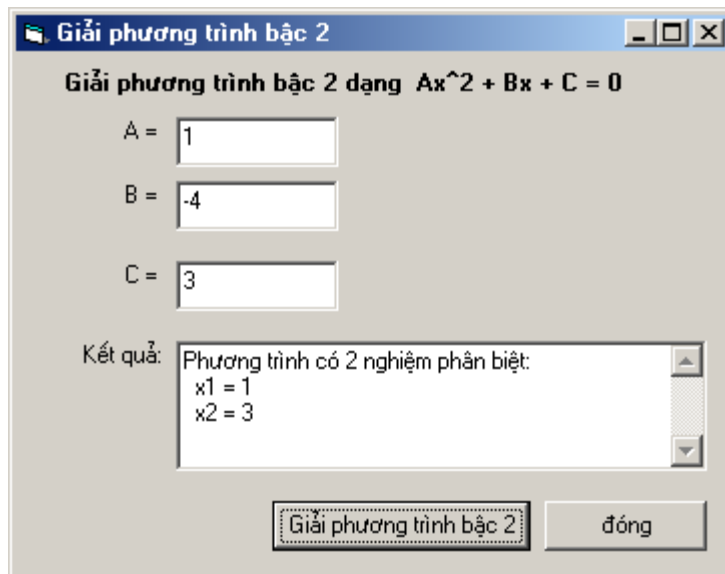
Bài tập 2:

Tạo form liệt kê các số nguyên tố trong khoảng từ 1 đến n, n là số được nhập trên form ở ô Nhập số:

Sau khi nhấn nút Hiển thị số nguyên tố, danh sách các số nguyên tố tìm được sẽ hiển thị ra ô Kết quả như hình trên.

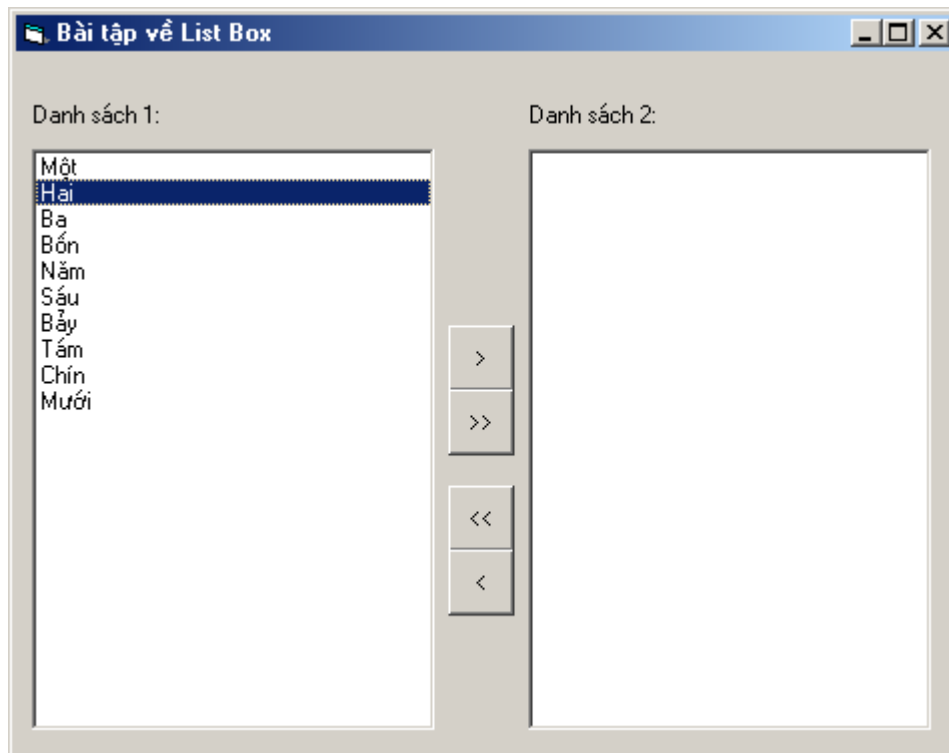
Bài tập 3:

Tạo form cho phép giải phương trình $Ax^2 + Bx + C = 0$ trong đó A, B, C là 3 số thực được nhập vào, x – là nghiệm cần tìm. Mọi kết quả sẽ được hiển thị lên ô Kết quả.



Bài tập 4:

Xây dựng form gồm 2 List box (hình dưới), các nút >, >>, <<, < có tác dụng chuyển một hoặc tất cả các mục (Items) từ danh sách 1 sang danh sách 2 và ngược lại.



Hơn nữa, thiết kế sao cho khi nhấn kép chuột lên một mục ở một danh sách, mục đó sẽ được chuyển sang danh sách bên kia.

Bài tập 5:

Sử dụng Treeview, ImageList để xây dựng cây danh mục Tỉnh (Thành) \ Huyện (Quận) \ Xã (Phường) như hình sau:



CHƯƠNG 3

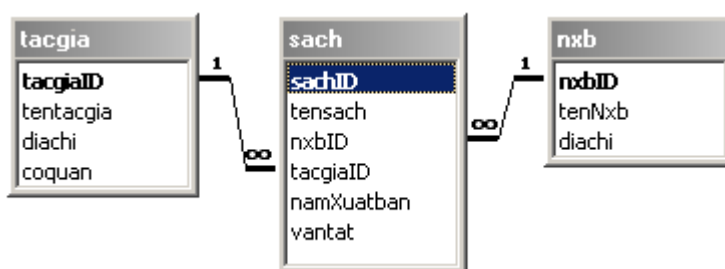
LẬP TRÌNH CƠ SỞ DỮ LIỆU

Trong chương này sẽ trình bày các cách dùng VB để lập trình điều khiển dữ liệu trên một hệ CSDL nào đó. Hiện nay đang tồn tại khá nhiều các kỹ thuật lập trình CSDL, tuy nhiên khuôn khổ cuốn sách này chỉ trình bày hai kỹ thuật căn bản là DAO (Data Access Objects) và ADO (ActiveX Data Objects) thông qua các bài toán lập trình CSDL cụ thể như:

- Bài toán cập nhập dữ liệu;
- Các phương pháp trình bày dữ liệu;
- Bài toán tìm kiếm, đặt lọc dữ liệu;
- Bài toán thiết kế in dữ liệu

Trong toàn bộ chương này, phần lý thuyết và các ví dụ sẽ được trình bày thông qua 2 hệ CSDL mẫu trên Access như sau:

Cơ sở dữ liệu Quản lý thư viện



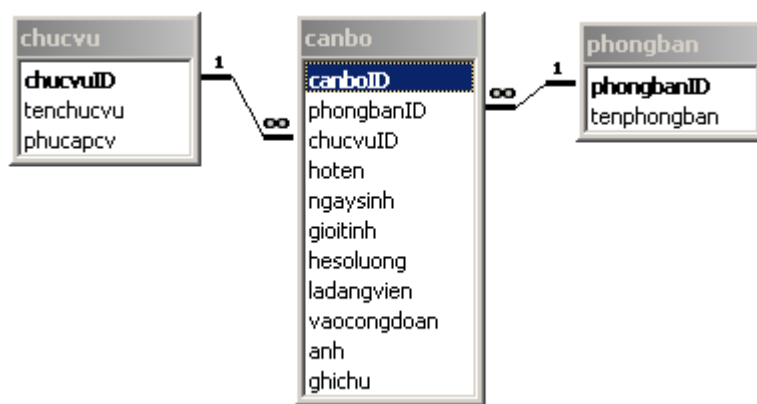
Trong đó:

Bảng NXB lưu trữ danh sách các nhà xuất bản;

Bảng TACGIA lưu trữ danh sách các tác giả viết sách

Bảng SACH lưu trữ các thông tin về sách trong thư viện. Mỗi sách sẽ có những thông tin sau: sachID – mã sách, tên sách, nxbID - xuất bản ở đâu? Năm xuất bản, tacgiaID - của tác giả nào? vắn tắt - những thông tin giới thiệu vắn tắt về cuốn sách này.

Cơ sở dữ liệu Quản lý lương cán bộ:



Trong đó:

Bảng CHUCVU – lưu trữ danh mục các chức vụ trong cơ quan, phụ cấp chức vụ (phucapcv) cũng được xác định cho mỗi loại chức vụ tương ứng trong bảng này;

Bảng PHONGBAN – lưu trữ danh mục các phòng ban có trong đơn vị quản lý;

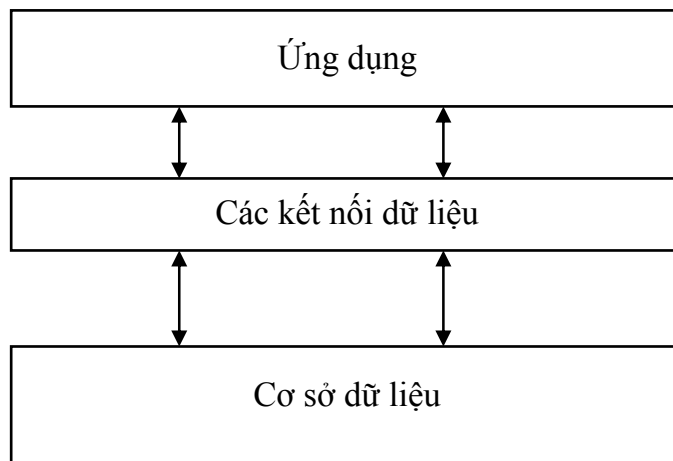
Bảng CANBO – lưu trữ toàn bộ thông tin về cán bộ trong cơ quan. Bảng này kết nối sang bảng PHONGBAN và CHUCVU để biết được các thông tin về chức vụ và phòng ban của cán bộ.

1. Kỹ thuật DAO

DAO (Data Access Objects – Các đối tượng truy xuất dữ liệu) là tập hợp bao gồm lớp các đối tượng có thể dùng để lập trình truy cập và xử lý dữ liệu trong các hệ CSDL. Ở đây là CSDL Access, ngôn ngữ lập trình VB.

DAO được phát triển khá sớm, gần đây nhất là phiên bản DAO 3.5 và 3.51- nó có thể thực hiện tốt được trên các phiên bản Access từ 97 trở về trước. Với Access 2000, XP phải dùng phiên bản DAO 3.6. Với phiên bản mới này, DAO 3.6 sử dụng nền Microsoft Jet 4.0. Vì vậy, có thể làm việc được trên nền Unicode dễ dàng.

Cách thức lập trình trên một hệ CSDL được mô tả tổng quát như sau:



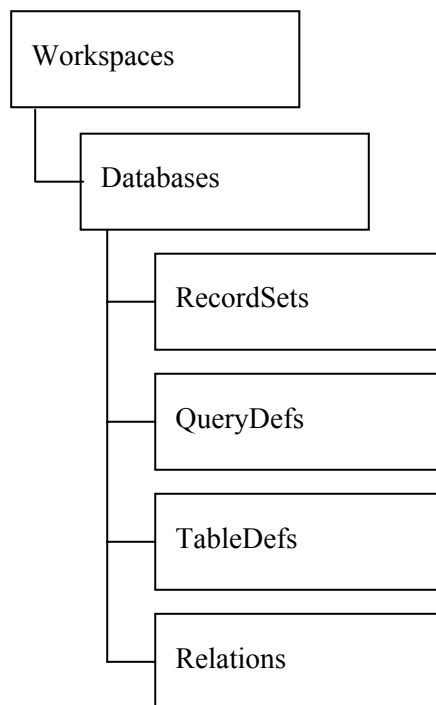
Trong đó:

- Tầng ứng dụng: bao gồm những giao diện người sử dụng cũng như những công cụ đơn giản mà người lập trình có thể dùng để xử lý dữ liệu theo các bài toán;
- Tầng Kết nối dữ liệu: bao gồm tập hợp các công cụ, phương thức để kết nối tới những dữ liệu cần làm việc trong CSDL. Ở đây, tầng kết nối bao gồm các chuẩn Microsoft Jet 3.51 và các lớp đối tượng DAO;
- Tầng Cơ sở dữ liệu: bao gồm các bảng, các query trong cơ sở dữ liệu thực tại.

Như vậy để lập trình trên một CSDL phải sử dụng các đối tượng, các phương thức ở tầng kết nối như là những công cụ để có thể truy cập được vào CSDL tác nghiệp xử lý. Tầng kết nối đó chính là Jet 3.51 và DAO 3.6 sẽ trình bày dưới đây.

1.1 Lớp đối tượng DAO

Cấu trúc một CSDL bao gồm nhiều thành phần, đòi hỏi việc lập trình cũng cần có những thành phần tương ứng để làm việc. Lớp các thành phần tương ứng để có thể lập trình được trên toàn bộ cấu trúc CSDL là lớp các đối tượng DAO. Chúng có tên gọi, có những tập thuộc tính, các phương thức làm việc và có quan hệ mật thiết với nhau. Cây phân cấp lớp các đối tượng DAO sau đây thể hiện điều đó:



Trong đó:

- **Workspaces** – định nghĩa tập hợp các vùng làm việc. Đây có thể coi là lớp làm việc cao nhất. Về lý thuyết có thể khai báo một vài vùng làm việc (Workspace), nhưng trên thực tế chỉ cần khai báo một vùng làm việc và

vùng này luôn được khai báo ngầm định cho CSDL hiện tại. Nên sẽ không cần bàn nhiều đến lớp các WorkSpace này;

- **Databases** - định nghĩa tập hợp các CSDL Access cần làm việc trên một dự án;
- **RecordSets**- định nghĩa các tập hợp bản ghi (Records) cần làm việc;
- **QueryDefs** - định nghĩa tập hợp các Query để làm việc. Querydefs và Recordsets là khả năng truy xuất, xử lý dữ liệu (Data Manipulation) của DAO;
- **TableDefs** - định nghĩa tập hợp các bảng (Table) cần làm việc. Đây là khả năng định nghĩa dữ liệu (Data-Definition Language);
- **Relations** - định nghĩa tập hợp các quan hệ (Relationship) cần làm việc;

Mỗi lớp các đối tượng trên sẽ bao gồm tất cả các đối tượng cùng loại trong một đối tượng mẹ đang mở. Ví dụ:

- Databases sẽ bao gồm tất cả các CSDL đang được mở trong vùng làm việc hiện tại;

- RecordSets sẽ bao gồm tập hợp tất cả các Recordset đang được mở trên CSDL hiện tại.

Khi đó, để tham chiếu đến một đối tượng cụ thể cần làm việc, có thể dùng chỉ số (số thứ tự của đối tượng đó trên tập hợp tất cả các đối tượng đó) hoặc dùng tên gọi đối tượng đó để tham chiếu. Ví dụ sau liệt kê tên của tất cả các Recordset đang sử dụng trong CSDL db.

```
Dim db As DAO.Database
'-----
'các câu lệnh tiếp theo ở đây..
'-----

'-----
'Liệt kê tên tất cả các recordset đang được mở trên db
'
For i = 0 To db.Recordsets.Count
    MsgBox db.Recordsets(i).Name
Next
```

Để làm việc tới một đối tượng cụ thể, cần phải tham chiếu từ lớp các đối tượng mẹ của nó.

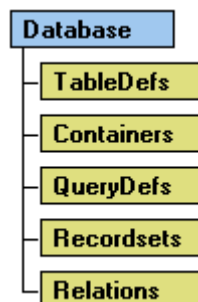
Ví dụ: Để hiển thị giá trị của trường (Field) hoten trên tập hợp các bản ghi (Recordset) rs1 làm như sau:

```
MsgBox rs1.Fields("hoten").Value
' hoặc
MsgBox rs1.Fields![hoten].Value
```

1.2 Đối tượng Database

Database là đối tượng dùng làm việc với một CSDL (trong trường hợp này có thể hiểu một CSDL như một tệp Access .MDB).

Lớp các đối tượng con của Database được thể hiện qua sơ đồ sau:



Khai báo

```
Dim db As DAO.Database
' Gán db cho một CSDL cụ thể
Set db = OpenDatabase("C:\Baitap\qlbh.mdb")
```

Khi không làm việc với CSDL nào đó, có thể ra lệnh đóng để giải phóng bộ nhớ bằng câu lệnh:

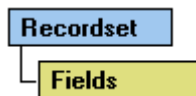
```
db.Close
```


Sau khi lệnh này thực thi, tất cả các đối tượng con của db nếu đang mở sẽ được đóng lại để giải phóng bộ nhớ. Bản thân db cũng được giải phóng bộ nhớ (là Nothing), tất nhiên tệp CSDL và dữ liệu vẫn còn nguyên trên đĩa.

1.3 Đối tượng RecordSet

Recordset là đối tượng dùng để miêu tả tập hợp các bản ghi của một bảng, của một query hoặc tập các bản ghi kết quả của việc thi hành một câu lệnh SQL nào đó.

Lớp các đối tượng con của Recordset được thể hiện qua sơ đồ sau:



Khai báo

```
Set rs=db.OpenRecordset (<Name>)
```

Trong đó:

- `Set rs = db.OpenRecordset` là lệnh để tạo ra tập hợp các bản ghi từ CSDL db gán vào biến kiểu recordset `rs`;
- `<Name>` là một xâu ký tự chỉ ra nguồn dữ liệu sẽ trả về cho Recordset. Xâu này có thể là tên một bảng, một Query hoặc một câu lệnh SQL;

Mỗi biến Recordset khi làm việc, phải được chỉ ra Database xuất xứ của nó (phải được tham chiếu từ một biến kiểu Database đã được khai báo). Sau đây là các ví dụ:

Ví dụ 1:

Gán tập hợp các bản ghi từ một bảng vào biến Recordset (ở đây là bảng *canbo*).

```
Dim rs As DAO.Recordset

Set rs = db.OpenRecordset ("canbo")
```

Ví dụ 2:

Gán tập hợp các bản ghi từ một câu lệnh chọn dữ liệu SQL vào biến Recordset (ở đây là các thông tin *hoten*, *ngaysinh* của tất cả các cán bộ nữ từ bảng *canbo*).

```
Dim rs As DAO.Recordset

Set rs = db.OpenRecordset ("SELECT hoten, ngaysinh FROM canbo
WHERE gioitinh = False")
```

Một số thuộc tính của Recordset**Thuộc tính Name**

Trả về chuỗi ký tự trong tham số *<name>* của lệnh gọi Recordset. Ví dụ: lệnh sau sẽ cho biết chuỗi ký tự tạo nguồn dữ liệu cho Recordset là gì?

```
MsgBox rs.Name
```

Thuộc tính AbsolutePosition

Cho biết vị trí bản ghi hiện tại (được tính từ 0). Trong trường hợp không có bản ghi nào trên recordset hoặc con trỏ bản ghi đang nằm ở EOF- sẽ không thể lấy được giá trị thuộc tính này. Do vậy để sử dụng thuộc tính này thường phải đi kèm thuộc tính kiểm tra có tồn tại bản ghi nào hay không (`RecordCount > 0`) và con trỏ bản ghi có ở cuối tệp chưa (`EOF = False`).

Thuộc tính RecordCount

Cho biết tổng số bản ghi trả về trên Recordset

Thuộc tính EOF

Cho biết con trỏ bản ghi hiện tại có nằm ở EOF hay không? Nếu có giá trị thuộc tính này là True, trái lại là False.

Thuộc tính Fields

Dùng tham chiếu tới các trường (Fields) trên tập hợp các bản ghi mà Recordset trả về. Thực tế Field cũng là một đối tượng và cũng có bộ thuộc tính và các phương thức của nó. Với Field của Recordset thông thường người ta hay sử dụng thuộc tính *Value*. Nếu không chỉ định thuộc tính cụ thể nào cho Field, VB vẫn hiểu ngầm định đó là Value.

Ví dụ: Hiển thị giá trị trường *hoten* trong Recordset *rs*

```
Msgbox rs.Fields("hoten").Value  
  
'hoặc  
Msgbox rs.Fields("hoten")
```

Một số phương thức của Recordset

Phương thức Close

Để đóng Recordset, giải phóng bộ nhớ. Chỉ thực hiện hành động này khi không làm việc với Recordset nào đó.

Các phương thức di chuyển bản ghi của Recordset

Phương thức MoveFirst

Để chuyển con trỏ về bản ghi đầu tiên

Phương thức MoveLast

Để di chuyển con trỏ về bản ghi cuối cùng

Phương thức MoveNext

Dịch đến bản ghi kế sau

Phương thức MovePrevious

Dịch đến bản ghi kế trước

Ví dụ 3:

Ví dụ sau duyệt và hiển thị toàn bộ *Hoten* của bảng *canbo*

```
Dim rs As DAO.Recordset
```

```

Set rs = db.OpenRecordset("canbo")

If rs.RecordCount > 0 Then
    rs.MoveFirst
    While rs.EOF = False
        MsgBox rs.Fields("hoten").Value
        rs.MoveNext
    Wend
End If

```

Phương thức AddNew, Update

Để thêm mới một bản ghi vào Recordset. Qui trình thêm một bản ghi mới như sau:

1. Ra lệnh Addnew
2. Gán giá trị cho các trường của bản ghi mới
3. Ra lệnh Update

Dưới đây là ví dụ thêm mới một hồ sơ cán bộ mới vào bảng *canbo*.

Ví dụ 4:

```

Dim rs As DAO.Recordset

Set rs = db.OpenRecordset("canbo")

'-----
'Ra lệnh thêm mới bản ghi
'
rs.AddNew

'-----
'Gán giá trị cho các trường của bản ghi mới
'
rs.Fields("canboID") = "CB00565"
rs.Fields("hoten") = "Nguyễn Sơn Hải"
rs.Fields("ngaysinh") = #2/11/1975#
rs.Fields("gioitinh") = True
rs.Fields("chucvuID") = "CV002"

'-----
'Ra lệnh ghi lại dữ liệu
'
rs.Update

```

Phương thức Edit, Update

Phương thức Edit để sửa dữ liệu một bản ghi nào đó trên recordset. Qui trình để sửa một bản ghi như sau:

1. Định vị tới bản ghi cần sửa trên recordset
2. Ra lệnh Edit
3. Gán giá trị mới cho các trường cần sửa
4. Ra lệnh Update

Dưới đây là ví dụ về sửa hồ sơ cán bộ có mã *CB000565*

Ví dụ 5:

```
Dim rs As DAO.Recordset

'-----
'Định vị tới bản ghi cần sửa
'
Set rs = db.OpenRecordset("SELECT * FROM canbo WHERE
canboID='CB0565'")

If rs.RecordCount > 0 Then
    rs.MoveFirst

    '-----
    'Ra lệnh sửa bản ghi
    '
    rs.Edit

    '-----
    'Thực hiện sửa các trường dữ liệu cần thiết
    '
    rs.Fields("hoten") = "Nguyễn Văn Hải"
    rs.Fields("ngaysinh") = #22/11/1975#

    '-----
    'Ra lệnh ghi lại dữ liệu vừa sửa
    '
    rs.Update
End If
```

Phương thức Delete

Để xoá bản ghi hiện tại ra khỏi Recordset. Khi đó bản ghi hiện tại sẽ bị xoá bỏ khỏi CSDL. Cần thận trọng mỗi khi ra lệnh này. Thông thường các lệnh một nút Xoá bản ghi của một mẫu nhập liệu (nhập vào biến Recordset *rs*) như sau:

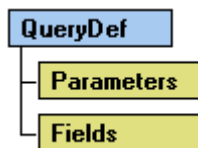
Ví dụ 6:

```
Private Sub cmDelete_Click()
    Dim tbao

    tbao = MsgBox("Đã chắc chắn xoá chưa?", vbYesNo + vbCritical)
    If tbao = vbYes Then
        rs.Delete
        rs.MoveNext
    End If
End Sub
```

1.4 Đối tượng QueryDef

Đối tượng Querydef dùng để tham chiếu tới các Query có sẵn (Buil-in) trên CSDL Access, hoặc cũng có thể lập trình tạo các Query từ các câu lệnh SQL.



Để tạo và kích hoạt một query trên VB bằng cách thực thi câu lệnh SQL, làm theo hướng dẫn sau:

```
'-----
' Khai báo một biến kiểu Database và một biến kiểu QueryDef
Dim qr As DAO.QueryDef

'-----
' Ra lệnh tạo một Query mới, có tên rỗng (chỉ ở trong bộ
nhớ)

Set qr = db.CreateQueryDef(<tên query>)

'-----
' Gán chuỗi lệnh SQL vào thuộc tính SQL của query
'
qr.SQL = "Gõ lệnh SQL cần thi hành vào đây"
```

```

'-----
'Ra lệnh thi hành query
'
qr.Execute

'-----
'giải phóng bộ nhớ

qr.Close

```

Trong đó:

- Bắt buộc phải khai báo một biến kiểu QueryDef để làm việc (biến *qr*);
- Phải có một biến Database đã được khai báo sẵn (biến *db*);
- Lệnh **Set qr = db.CreatQueryDef(<tên query>)** để tạo một query mới lên CSDL. <tên query> sẽ được hiển thị trên danh sách trong thẻ Queries trên cửa sổ Database. Nếu <tên query>="", query này sẽ chỉ tồn tại trong bộ nhớ. Tùy thuộc vào mục đích công việc mà có đặt tên query hay không, nếu chỉ đơn thuần tạo một query để xử lý công việc nào đó rồi giải phóng, nên đặt <tên query>="";
- Lệnh **qr.SQL=<câu lệnh SQL>** để gán lệnh SQL cần thực thi vào Query. Tùy thuộc vào câu lệnh SQL này mà query sẽ thực hiện những gì. Ví dụ: **qr.SQL = "DELETE * FROM canbo"** lệnh này sẽ xoá tất cả các bản ghi trên bảng cán bộ;
- Lệnh **qr.Excute** để thi hành câu lệnh SQL đã được thiết lập. Lệnh này tương đương nhấn nút Run đối với một query trên chế độ thiết kế;
- Lệnh **qr.Close** để đóng query hiện tại và giải phóng bộ nhớ khi không cần làm việc đến nữa.

Sau đây là một số ví dụ về sử dụng biến QueryDef để tạo ra một số loại query xử lý dữ liệu.

Ví dụ 1: Tạo DELETE query để xoá danh sách những cán bộ có tuổi lớn hơn 60 ra khỏi bảng *canbo* (cán bộ đã nghỉ hưu)

```
Dim qr As DAO.QueryDef
```

```

Set qr = db.CreateQueryDef("")

qr.SQL="DELETE * FROM canbo WHERE Year(Date())- " _
    & " Year(Ngaysinh)>=60"

qr.Execute

qr.Close
    
```

Ví dụ 2: Giả sử đã thêm một trường mới trên bảng cán bộ có tên *luongchinh*. Tạo UPDATE query để tính giá trị cho trường này = **hesoluong * 290000**.

```

Dim db As DAO.Database
Dim qr As DAO.QueryDef

Set db = CurrentDb

Set qr = db.CreateQueryDef("")

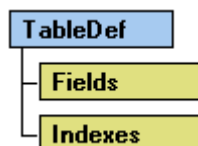
qr.SQL = "UPDATE canbo SET canbo.luongchinh = hesoluong *
290000"

qr.Execute

qr.Close
    
```

1.5 Đối tượng TableDef

Đối tượng TableDef được dùng để tham chiếu tới các bảng dữ liệu (Table) trên CSDL. Thông qua đối tượng này có thể thiết kế, chỉnh sửa được cấu trúc các bảng dữ liệu trong chế độ Run-time của VB như trên chế độ thiết kế bảng Design View trên Access.



Một số thuộc tính quan trọng của TableDef

Thuộc tính Name

Cho biết tên bảng được gán vào biến kiểu TableDef

Thuộc tính RecordCount

Cho biết tổng số bản ghi hiện có trên bảng được gán bởi biến TableDef

Thuộc tính DateCreated

Cho biết thời gian tạo ra bảng được gán vào biến kiểu TableDef

Thuộc tính Fields

Để tham chiếu tới các trường của bảng. Đây là thuộc tính hay được sử dụng nhất đối với TableDef. Thực chất, Field ở đây là một đối tượng, do đó cũng có tập các thuộc tính và phương thức riêng cho thuộc tính này.

Dưới đây là thủ tục hiển thị tên của tất cả các trường trong một bảng nào đó (ngầm định trên một CSDL đã được khai báo và gán biến *db* - kiểu Database).

Ví dụ 1:

```
Sub LietKeTenTruong(tenbang As String)
    Dim tbl As DAO.TableDef

    Set tbl = db.TableDefs(tenbang)

    For i = 0 To tbl.Fields.Count - 1
        MsgBox tbl.Fields(i).Name
    Next

End Sub
```

Một số phương thức của TableDefPhương thức CreateTableDef

Để tạo ra một bảng mới từ VB. Cú pháp tạo bảng mới như sau:

```
Set tbl = db.CreateTableDef(<Tên bảng mới>)
'-----
'....Các thủ tục tạo trường mới cho bảng
'-----

db.TableDefs.Append tbl
```

Trong đó:

- *db* – là biến kiểu Database đã được gán bởi CSDL cần làm việc (bảng mới sẽ được tạo ra trên CSDL này);
- *<Tên bảng mới>* là tên bảng cần tạo.
- Lệnh **db.TableDefs.Append tbl** là lệnh ghi cấu trúc bảng đang khai báo lên CSDL đã chỉ định.

Phương thức CreateField

Để tạo ra các trường cho một bảng kiểu TableDef nào đó. Để thêm một trường mới lên bảng, sử dụng cú pháp sau:

```
tbl.Fields.Append tbl.CreateField (<tên trường>, <KiểuDL>, <độlớn>)
```

Trong đó:

- *<tên trường>* - tên trường mới cần tạo;
- *<KiểuDL>* - là một tùy chọn để khai báo kiểu dữ liệu của trường cần tạo. Kiểu dữ liệu được khai báo theo các hằng số như sau:

Giá trị: Tương ứng với kiểu

- dbBoolean* Boolean
- dbByte* Byte
- dbChar* Char
- dbCurrency* Currency
- dbDate* Date/Time
- dbDecimal* Decimal
- dbDouble* Double
- dbFloat* Float
- dbGUID* GUID
- dbInteger* Integer
- dbLong* Long
- dbMemo* Memo

dbNumeric Numeric

dbSingle Single

dbText Text

dbTime Time

- <Độ lớn> là một tùy chọn để khai báo độ lớn dữ liệu nếu cần.

Tiếp theo là ví dụ minh họa cách tạo cấu trúc một bảng dữ liệu tổng hợp những hướng dẫn đã trình bày trên.

Ví dụ 2:

```
Sub TaoBangMoi()
On Error GoTo Loi
Dim tbl As DAO.TableDef

Set tbl = db.CreateTableDef("NewTable")

tbl.Fields.Append tbl.CreateField("ID", dbInteger)
tbl.Fields.Append tbl.CreateField("Name", dbText)
tbl.Fields.Append tbl.CreateField("Age", dbByte)
tbl.Fields.Append tbl.CreateField("DateBirth", dbDate)
tbl.Fields.Append tbl.CreateField("Comment", dbMemo)

db.TableDefs.Append tbl

Exit Sub

Loi:
If Err.Number = 3010 Then
MsgBox "Đã tồn tại bảng có tên " + tbl.Name
End If
End Sub
```

1.6 Đối tượng Relation

Đối tượng Relation dùng để tạo kết nối (Relationship) giữa 2 bảng trong CSDL Access. Dưới đây là một ví dụ tạo kết nối giữa 2 bảng hoadon và khách trong CSDL Quản lý bán hàng.

```
Sub CreatRelationship()
On Error GoTo Loi
Dim rls As DAO.Relation
```

```

Set rls = db.CreateRelation("TaoQuanHe", "khach",
"hoadon", dbRelationUpdateCascade)

rls.Fields.Append rls.CreateField("khachID")
rls.Fields("khachID").ForeignName = "khachID"
db.Relations.Append rls

Loi:
If Err.Number = 3012 Then
    MsgBox "Đã tồn tại quan hệ này !"
End If

End Sub

```

Trong trường hợp đã tồn tại kết nối này, một thông báo lỗi tiếng Việt "*Đã tồn tại quan hệ này !*" xuất hiện.

1.7 Sử dụng Data Form Wizard

Data Form Wizard là một trình tiện ích dùng để tạo nhanh một giao diện cập nhật dữ liệu cho một bảng dữ liệu trên một CSDL. Chúng được dùng trong công cụ VisData được tích hợp sẵn trong VB. Tiếp theo sẽ hướng dẫn sử dụng tiện ích này để tạo một số form nhập dữ liệu cho các bảng trên CSDL Quản lý lương cán bộ.

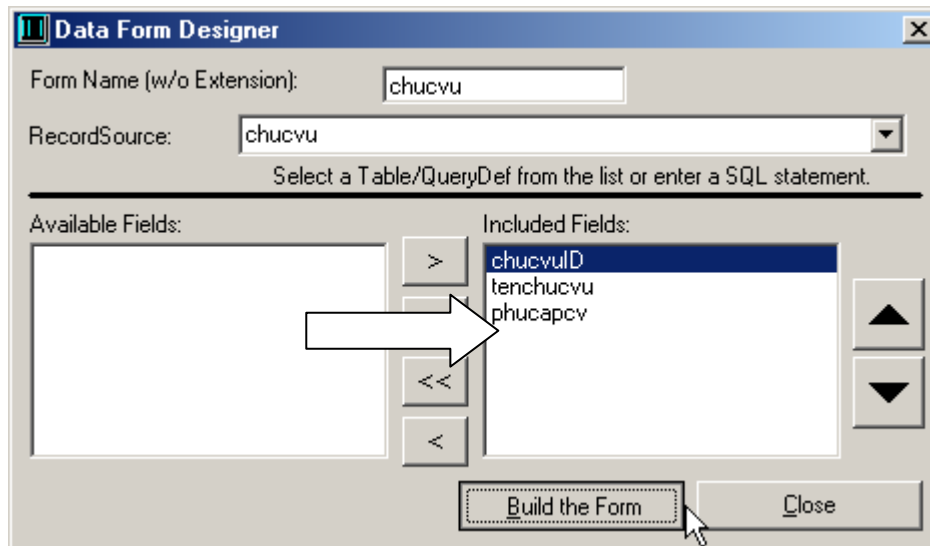
Yêu cầu: tạo form cập nhật danh mục chức vụ cho CSDL Quản lý lương cán bộ, có sử dụng Data Form Wizard.

Bước 1: Tạo một Project trên VB

Bước 2: Thực hiện lệnh Add-in Visual Data manager. Cửa sổ ứng dụng VisData xuất hiện.

Bước 3: mở tệp CSDL Access cần làm việc (CSDL Quản lý lương) bằng cách: **File | Open Database | Microsoft Access...** Tiếp theo tìm đến tệp CSDL Quản lý lương và **Open**.

Bước 4: Tạo form nhập dữ liệu bằng cách: **Utility | Data Form Designer**. Một hộp thoại xuất hiện, hãy thiết lập như sau:

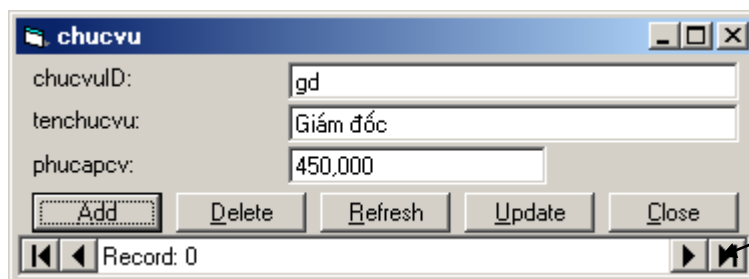


Cách thiết lập:

- Gõ tên form vào hộp **Form Name**: chú ý tên form không chứa dấu cách và một số ký tự đặc biệt;
- Hộp **Record Source**, chọn tên bảng cần nhập dữ liệu. Ở đây chọn bảng *CHUCVU*;
- Tiếp theo sử dụng các nút >, >> hoặc <, << để đưa các trường của bảng dữ liệu cần nhập lên form từ danh sách **Available Fields** sang danh sách **Included Fields**;
- Cuối cùng nhấn nút **Build the Form** để kết thúc.

Đến đây, khi quay sang Project trên VB sẽ thấy xuất hiện một form nhập dữ liệu cho bảng CHUCVU đã được tự động tạo vào có thể sử dụng.

Khi thi hành project, form này có giao diện như sau:



Thanh định vị bản ghi
Đối tượng Data Control

Để thêm một bản ghi:

- Nhấn nút **Add**;
- Tiếp theo điền dữ liệu bản ghi mới lên các hộp nhập dữ liệu của form;
- Nhấn nút **Update** để ghi dữ liệu.

Để sửa dữ liệu một bản ghi:

- Sử dụng thanh định vị bản ghi để chuyển đến bản ghi cần sửa;
- Thực hiện sửa dữ liệu;
- Nhấn nút **Update** để ghi nhận dữ liệu.

Để xóa một bản ghi:

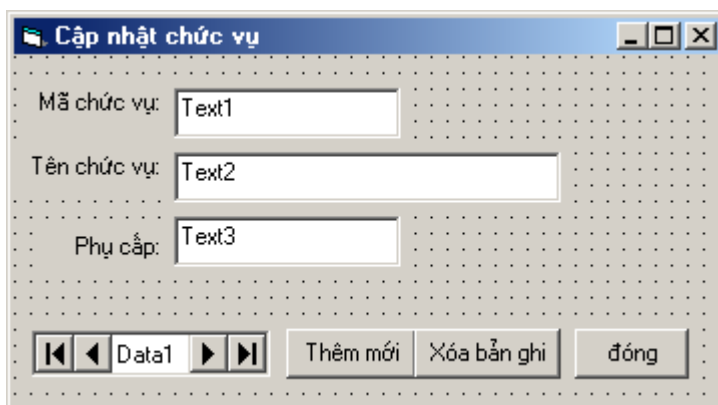
- Sử dụng thanh định vị bản ghi để chuyển đến bản ghi cần xóa;
- Nhấn nút **Delete** để xóa bản ghi đang hiển thị.

1.8 Bài toán cập nhật dữ liệu

Nhập dữ liệu là yêu cầu gần như bắt buộc đối với bất kỳ ứng dụng nào. Với công nghệ DAO, Data Control là điều khiển rất phù hợp cho bài toán này. Dưới đây minh họa ví dụ tạo form nhập dữ liệu cho bảng CHUCVU sử dụng công nghệ này:

Ví dụ 1: thiết kế form nhập dữ liệu đơn giản

Bước 1: Thiết kế giao diện nhập dữ liệu cho form như sau:



Bước 2: thiết lập một số thuộc tính cho các điều khiển trên form như sau:

Điều khiển Data1

- Thuộc tính *DatabaseName* – hãy trỏ tới tệp CSDL Quản lý lương;
- Thuộc tính *RecordSource* - chọn bảng cần nhập dữ liệu (bảng CHUCVU).

Nút Thêm mới, thuộc tính *name* là *cmdThemMoi*;

Nút Xóa bản ghi, thuộc tính *name* là *cmdXoa*;

Nút Đóng, thuộc tính *name* là *cmdDong*;

Ô nhập Mã chức vụ:

Mỗi một ô để nhập dữ liệu trên form yêu cầu tối thiểu phải thiết lập 2 thuộc tính là:

- *DataSource* - để khai báo nguồn dữ liệu cho ô này lấy từ đâu (bảng nào)? Thông thường lấy từ điều khiển Data Control liên kết tới bảng dữ liệu;
- *DataField* – để khai báo trường dữ liệu sẽ liên kết tới ô nhập dữ liệu hiện tại. Thuộc tính này phải được thiết lập sau thuộc tính *DataSource* mới có tác dụng.

Với form trên, 3 đối tượng dùng nhập dữ liệu phải thiết lập các thuộc tính như sau:

Đối tượng	DataSource	DataField
Text1	<i>Data1</i>	<i>chucvuID</i>
Text2	<i>Data1</i>	<i>Tenchucvu</i>
Text3	<i>Data1</i>	<i>Phucapcv</i>

Bước 3: viết mã lệnh điều khiển cho các nút lệnh trên form như sau:

```

Private Sub cmdThemMoi_Click()
    '-----
    'lện thêm bản ghi mới
    '
    Data1.Recordset.AddNew
    '-----
    
```

```

' trước khi nhập dữ liệu, chuyển con trỏ tới ô mã chức vụ
,
Text1.SetFocus

End Sub

Private Sub cmdXoa_Click()
    Dim thongbao

    thongbao = MsgBox("Đã chắc chắn xóa chưa?", vbYesNo)
    If thongbao = vbYes Then

        '-----
        'lệnh xóa bản ghi hiện tại trên Data1
        '
        Data1.Recordset.Delete
    End If
End Sub

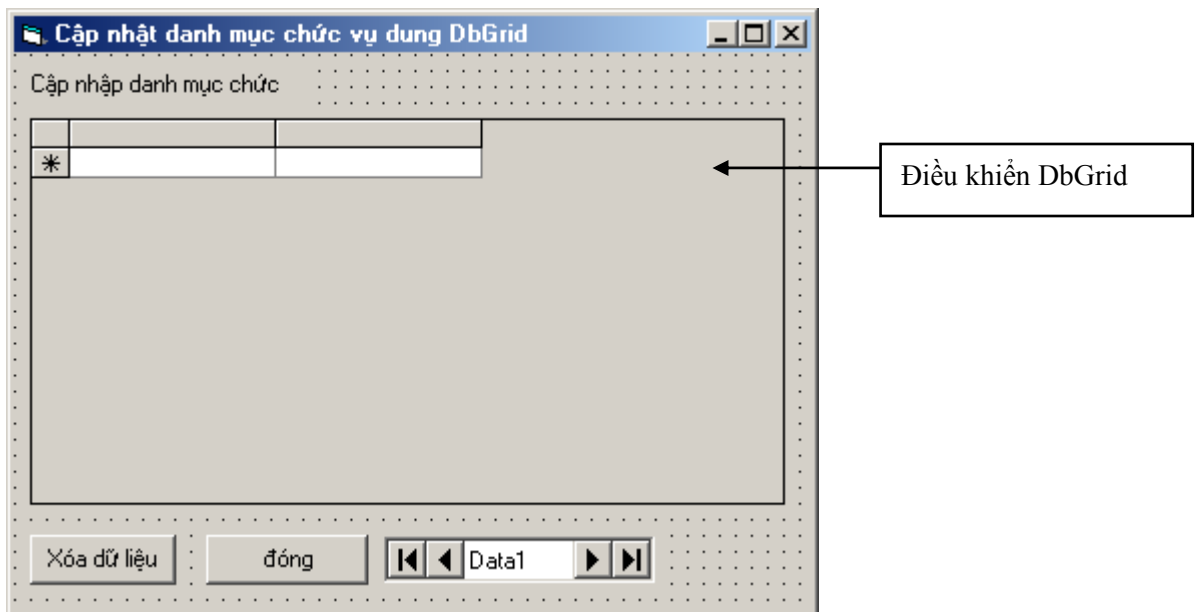
Private Sub cmdDong_Click()
    Unload Me
End Sub

```

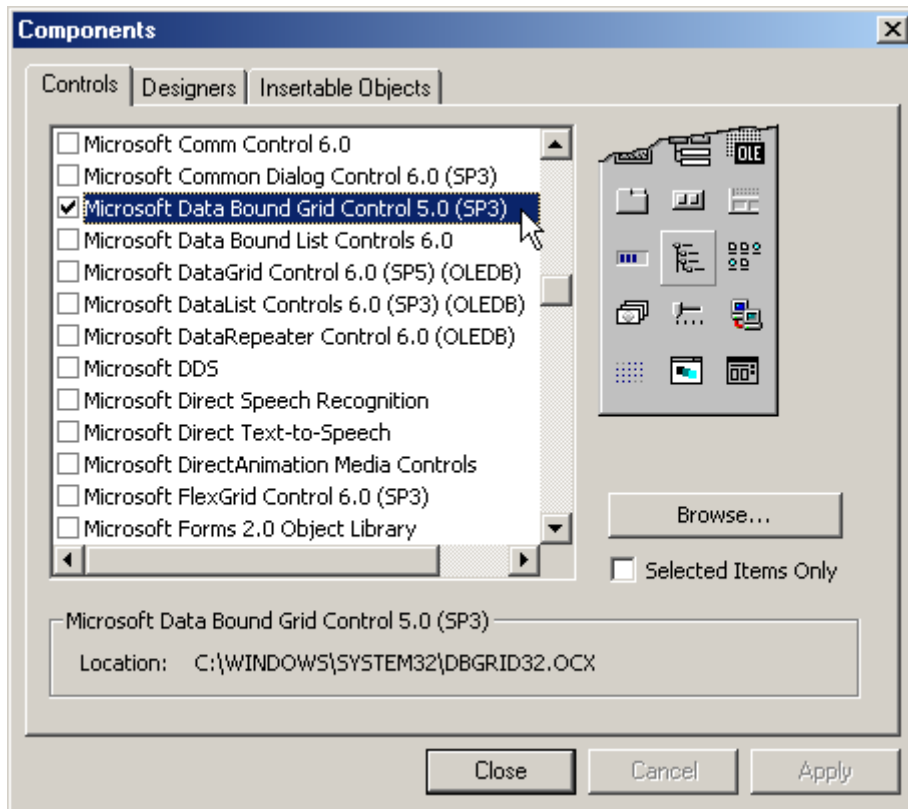
Ví dụ 2: Sử dụng DbGrid

Với DbGrid, việc thiết kế một giao diện nhập dữ liệu đơn giản hơn và dễ sử dụng hơn. Dưới đây giới thiệu cách sử dụng điều khiển này.

Bước 1: thiết kế giao diện form như sau:



Trong trường hợp không thấy điều khiển DbGrid trên thanh ToolBar, hãy kích hoạt nó bởi cửa sổ Components như sau:



Bước 2: thiết lập các thuộc tính phù hợp cho các đối tượng:

Điều khiển Data1

- Thuộc tính *DatabaseName* – hãy trỏ tới tệp CSDL Quản lý lương;
- Thuộc tính *RecordSource* - chọn bảng cần nhập dữ liệu (bảng CHUCVU);
- Thuộc tính *Visible* = *False* (để ẩn điều khiển Data này khi chạy chương trình).

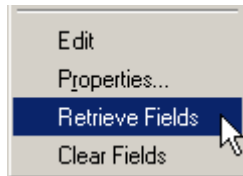
Nút Thêm mới, thuộc tính *name* là *cmdThemMoi*;

Nút Đóng, thuộc tính *name* là *cmdDong*;

Điều khiển DbGrid1

- Thuộc tính *DataSource* – chọn nguồn dữ liệu sẽ hiển thị lên DbGrid (chọn Data1);
- Thuộc tính *AllowAddNew* = *True*;
- Thuộc tính *AllowDelete* = *True*;
- Thuộc tính *AllowUpdate* = *True*;

- Thuộc tính *DataMode = 0 – Bound* (tức là tự động gắn kết dữ liệu tới bảng dữ liệu);
- Cuối cùng nhấn phải chuột lên DbGrid chọn thực đơn *Retrieve Fields* để tự động thiết lập tiêu đề các cột và định vị thứ tự các cột trong bảng dữ liệu lên các cột tương ứng trên DbGrid.



Hơn nữa, nếu chọn mục *Properties* để có thể thiết lập được nhiều hơn các thuộc tính khác làm DbGrid trở nên đẹp hơn. Việc này bạn đọc tự nghiên cứu.

Bước 3: Viết lệnh cho form như sau:

```

Private Sub cmdXoa_Click()
    Dim thongbao

    thongbao = MsgBox("Đã chắc chắn xóa chưa?", vbYesNo)
    If thongbao = vbYes Then

        '-----
        'lệnh xóa bản ghi hiện tại trên Data1
        '
        Data1.Recordset.Delete
    End If
End Sub

Private Sub cmdDong_Click()
    Unload Me
End Sub

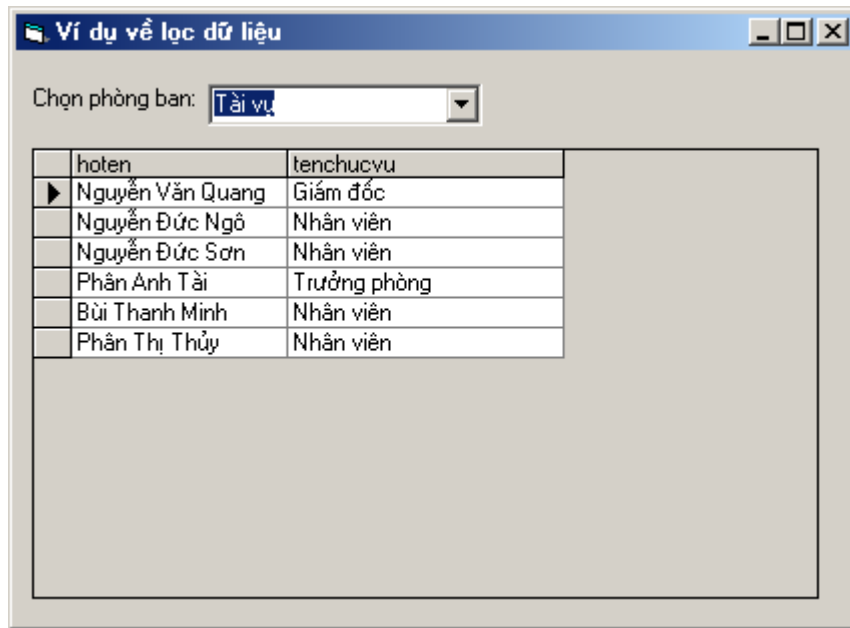
```

Với phương pháp này không cần sử dụng phương thức thêm bản ghi mới (bởi trên DbGrid đã tự động cho phép làm việc này).

1.9 Bài toán tìm và lọc dữ liệu

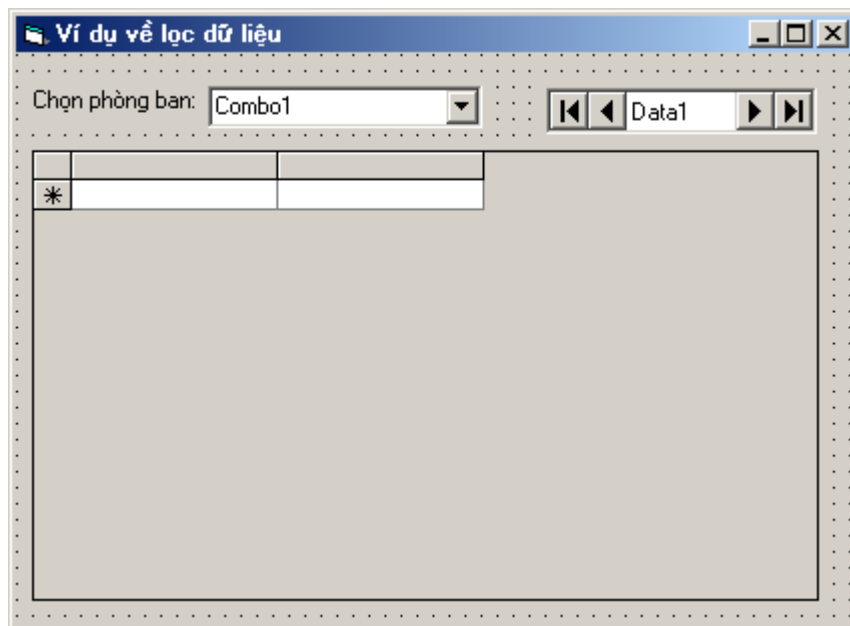
Lọc và tìm kiếm là dạng bài toán cơ bản trong lập trình CSDL. Nguyên tắc hoạt động của dạng bài toán này là như nhau, sẽ chỉ khác về yêu cầu, giao diện sử dụng.

Ví dụ dưới đây là một bài toán lọc ra danh sách cán bộ của một phòng ban nào đó sau khi tên phòng ban được chọn từ một hộp Combobox:



Cách làm:

Bước 1: Thiết kế giao diện form như sau:



Bước 2: thiết lập các thuộc tính phù hợp cho các điều khiển như sau:

Điều khiển Data1

- Thuộc tính *DatabaseName* – hãy trỏ tới tệp CSDL Quản lý lương;
- Thuộc tính *Visible* = *False* (để ẩn điều khiển Data này khi chạy chương trình).

Điều khiển DbGrid1

- Thuộc tính *DataSource* – chọn nguồn dữ liệu sẽ hiển thị lên DbGrid (chọn Data1);

Bước 3: viết lệnh cho form như sau:

```

Private Sub Form_Load()
    Dim rs As DAO.Recordset

    '-----
    'mở tệp CSDL gán vào biến db
    '
    Set db = OpenDatabase(Data1.DatabaseName)

    '-----
    'mở danh sách các phòng ban và đẩy lên Combo1
    '
    Set rs = db.OpenRecordset("phongban")
    While rs.EOF = False
        Combo1.AddItem rs.Fields("tenphongban").Value
        rs.MoveNext
    Wend
End Sub

Private Sub Combo1_Click()
    '-----
    'thủ tục này xảy ra sau khi chọn một phòng ban từ combo1
    '

    Dim strSQL As String

    'Bước 1 -----
    'xác định câu lệnh SQL để lọc ra danh sách cán bộ thỏa mãn
    'phòng ban đã chọn ở Combo1
    '
    strSQL = "SELECT canbo.hoten, chucvu.tenchucvu "
    + " FROM (canbo INNER JOIN chucvu ON canbo.chucvuID = "
    + " chucvu.chucvuID) INNER JOIN phongban ON = "
    + " canbo.phongbanID = phongban.phongbanID "
    + " WHERE phongban.tenphongban = '" + Combo1.Text + "'"

    'Bước 2-----
    'gán cấu lệnh SQL trên cho Data1
    '
    Data1.RecordSource = strSQL

    '-----
    'Lệnh này thực hiện đổ dữ liệu từ Data1 lên DbGrid
    '
    Data1.Refresh

End Sub

```



Chú ý

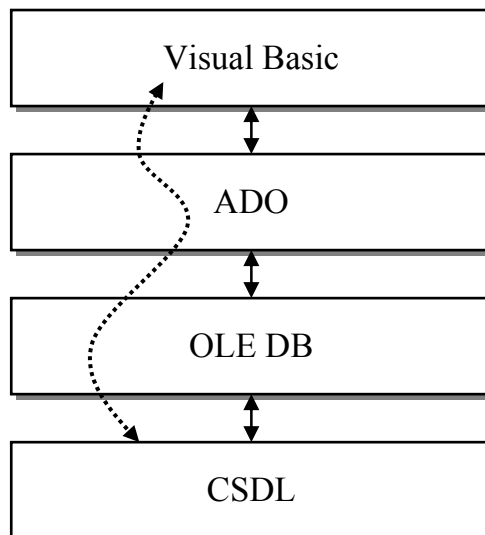
Việc viết ra câu lệnh SQL đúng là rất khó, đòi hỏi lập trình viên phải có nhiều kinh nghiệm. Để làm tốt được việc này, nên thiết kế một query ở chế độ design view; rồi sử dụng tính năng Copy, Paste để dán câu lệnh SQL mà query đã tạo lên nơi soạn thảo lệnh VB và chỉnh sửa cho phù hợp. Thông thường sửa mệnh đề WHERE của câu lệnh.

2. Kỹ thuật ADO

Phần 1 của chương này đã giới thiệu công nghệ lập trình CSDL DAO trên VB, đó là công nghệ khá dễ học, phổ biến. ADO (ActiveX Data Objects) là công nghệ mới hơn, khắc phục được một số các nhược điểm của DAO và thích nghi trên nhiều hệ CSDL. Hiện nay, ADO.NET là công cụ cực mạnh và đang được phát triển rất phổ biến. Tuy nhiên ADO.NET chỉ có thể chạy trên nền .NET. Để trở thành nhà phát triển chuyên nghiệp, chọn ADO là một hướng đi đúng.

2.1 Kiến trúc ADO

Trước khi đi vào tìm hiểu kỹ thuật lập trình ADO hãy tìm hiểu kiến trúc và cách thức sử dụng VB lập trình CSDL dùng ADO.



Như vậy, để có thể sử dụng VB lập trình CSDL sử dụng công nghệ ADO ta cần qua 2 lớp: ADO và OLE DB.

- ADO cung cấp tập hợp các đối tượng, công cụ để có thể lập trình xử lý hệ CSDL;

- Trong khi đó, OLE DB là một trình cung cấp các dịch vụ giúp điều khiển các yêu cầu xử lý dữ liệu từ ADO. Trong thực tế với nguồn dữ liệu là Access, còn cần phải qua lớp điều khiển Jet trước khi tập hợp các lệnh từ ADO có thể thâm nhập vào CSDL.

Tuy nhiên, các lập trình viên cũng chưa cần quan tâm nhiều đến cấu trúc của OLE DB cũng như Jet mà chỉ cần sử dụng tốt các đối tượng ADO vào lập trình CSDL vì OLE DB và Jet chỉ đơn giản là việc khai báo chúng.

2.2 Đối tượng Connection

Đối tượng Connection dùng để kết nối tới nguồn dữ liệu vật lý cần làm việc. Tất cả các đối tượng của ADO đều phải làm việc trên một Connection nào đó đã được thiết lập.

Để thiết lập một Connection cần phải khai báo một biến có kiểu ADODB.Connection;

Các thuộc tính kết nối tới CSDL được thể hiện qua một chuỗi kết nối (thường gọi chuỗi này là Connection String). Trên chuỗi này có chỉ định các thuộc tính như:

Provider – tên trình cung cấp dịch vụ truy cập CSDL. Ở đây dùng *Microsoft.JET.OLEDB.3.51*;

Data Source - chỉ định nguồn dữ liệu truy cập. Ở đây chỉ định đường dẫn tới tệp CSDL Access cần làm việc. Ví dụ: *C:\Project\QLTV\Data\qltv.mdb*

Khi đó, việc mở một kết nối trong ADO được thực hiện bởi thủ tục sau:

```
Dim cnn As New ADODB.Connection
cnn.Open "Provider=Microsoft.Jet.OLEDB.3.51;"
        & "Data Source=C:\Project\QLTV\Data\qltv.mdb"
```

Khi không muốn làm việc với kết nối nào đó (cnn), nên ngắt bỏ kết nối để giải phóng bộ nhớ bằng việc gọi phương thức Close như sau:

```
cnn.Close
```

Sau lệnh ngắt kết nối này, tất cả các đối tượng khác của ADO có sử dụng Connection *cnn* cũng sẽ tự động được giải phóng khỏi bộ nhớ.

2.3 Đối tượng Command

Đối tượng Command trong ADO thường dùng để thực thi một câu lệnh truy vấn SQL nào đó. Ví dụ như: việc xóa, thêm, cập nhật các bản ghi trong CSDL. Cách sử dụng đối tượng này như sau:

Bước 1: Khai báo một biến đối tượng Command như sau:

```
Dim cmd As New ADODB.Command
```

Bước 2: Thiết lập chuỗi kết nối cho Command này thông qua thuộc tính ActiveConnection. Chuỗi này hệt như chuỗi đã sử dụng để khởi tạo Connection. Ví dụ: với việc kết nối tới CSDL Quản lý thư viện như trên, chuỗi kết nối sẽ như sau:

```
cmd.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.3.51;"  
& "Data Source= " + App.Path + "\qltv97.mdb"
```

Hoặc có thể sử dụng thông qua Connection đã khởi tạo như sau:

```
cmd.ActiveConnection = cnn.ConnectionString
```

Bước 3: Gán câu lệnh SQL cần thực thi thông qua thuộc tính CommandText của đối tượng Command. Giả sử muốn chèn thêm một bản ghi mới lên bảng NXB có giá trị (*nxbID = '05', tenNXB='Nhà xuất bản khoa học kỹ thuật', diachi='Hà nội'*):

```
cmd.CommandText = "INSERT INTO nxb (nxbID, tenNXB, diachi)"  
& "VALUES ('05', 'Nhà xuất bản khoa học kỹ thuật', 'Hà  
nội')"
```

Bước 4: Ra lệnh thực thi Command bởi phương thức Execute

```
cmd.Execute
```

Về cơ bản, đối tượng Command trong ADO gần giống với đối tượng QueryDef trong DAO!

2.4 Đối tượng Recordset

Đối tượng Recordset trong ADO cũng gần giống như trong DAO, tức là có thể trích lọc dữ liệu từ CSDL. Điểm khác biệt chủ yếu giữa Recordset trong ADO và DAO là: Recordset trong ADO chỉ có khả năng chỉ đọc, không thể cập nhật được dữ liệu. Như vậy những phương thức như Delete, Addnew hay Update là không có trên đối tượng recordset trong ADO.

Khác với Command, Recordset cần hoạt động trên một Connection đã được kích hoạt. Giả sử đã có Connection cnn được kích hoạt và kết nối tới CSDL Quản lý thư viện. Các bước sau đây hướng dẫn tạo một Recordset:

Bước 1: Khai báo một biến đối tượng RecordSet như sau:

```
Dim rs As New ADODB.RecordSet
```

Bước 2: Ra lệnh thực thi câu lệnh lựa chọn SQL và gán tập hợp các bản ghi trả về vào biến Recordset này. Giả sử muốn lấy ra các thông tin về sách như: *sachID*, *tensach*, *namxb*, *tentacgia*, *tennxb*. Thủ tục như sau:

```
rs.Open "SELECT sach.sachID, sach.tensach, sach.namXuatban, "
      & " tacgia.tentacgia, nxb.tenNxb FROM tacgia INNER JOIN "
      & " (nxb INNER JOIN sach ON nxb.nxbID = sach.nxbID) ON "
      & " tacgia.tacgiaID = sach.tacgiaID;", cnn
```

Đến đây có thể sử dụng Recordset này như trong DAO, chỉ khác là không thể thực hiện các chức năng về cập nhật dữ liệu. Ví dụ dưới đây thực hiện duyệt và in ra *Tensach*, *tentacgia*, *tennxb*, *Namxb* của tất cả các bản ghi Recordset trên:

```
While rs.EOF = False
```

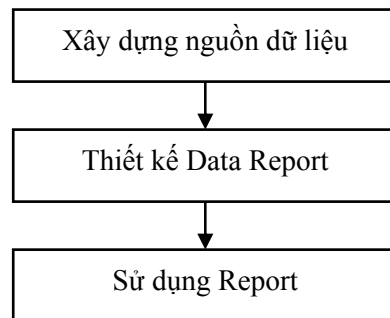


```
    MsgBox rs.Fields("tensach") & " - " &  
rs.Fields("tentacgia")  
    rs.MoveNext  
Wend
```

3. Data Report

Data Report là công cụ thiết kế in báo cáo trong VB. Đây là công cụ không thực sự chuyên nghiệp, nhưng nó rất quan trọng để tiếp cận tới các công cụ thiết kế in báo cáo chuyên nghiệp khác.

Quy trình thiết kế và sử dụng một Data Report thường phải qua 3 bước sau:



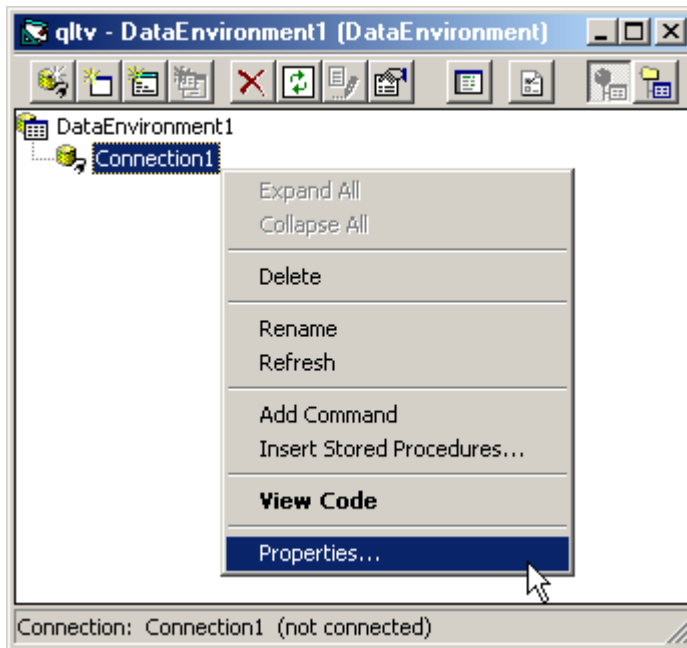
Dưới đây sẽ trình bày kỹ từng qui trình tạo ra và sử dụng một Data Report.

3.1 Xây dựng nguồn dữ liệu

Xây dựng nguồn dữ liệu là xây dựng tập hợp dữ liệu cần in lên báo cáo thông qua một truy vấn dữ liệu (query). Cách làm như sau:

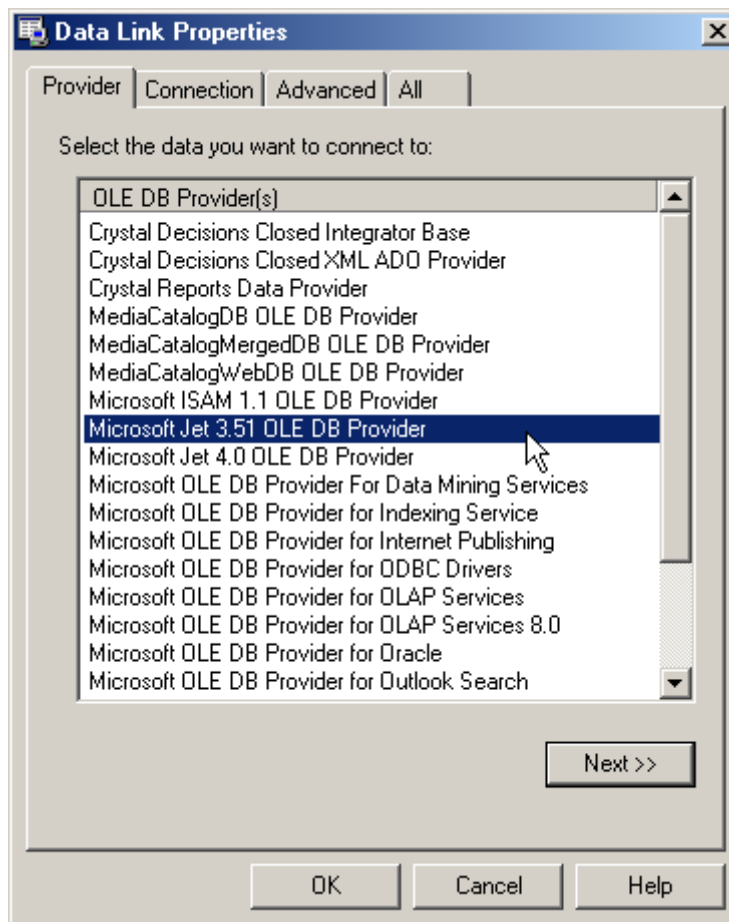
Bước 1: Tạo môi trường dữ liệu lên Project (trong trường hợp đã có một môi trường dữ liệu rồi, bước này là không cần phải làm lại).

Ra lệnh **Project \ Data Environment**. Một Data Environment sẽ xuất hiện trên Project như sau:



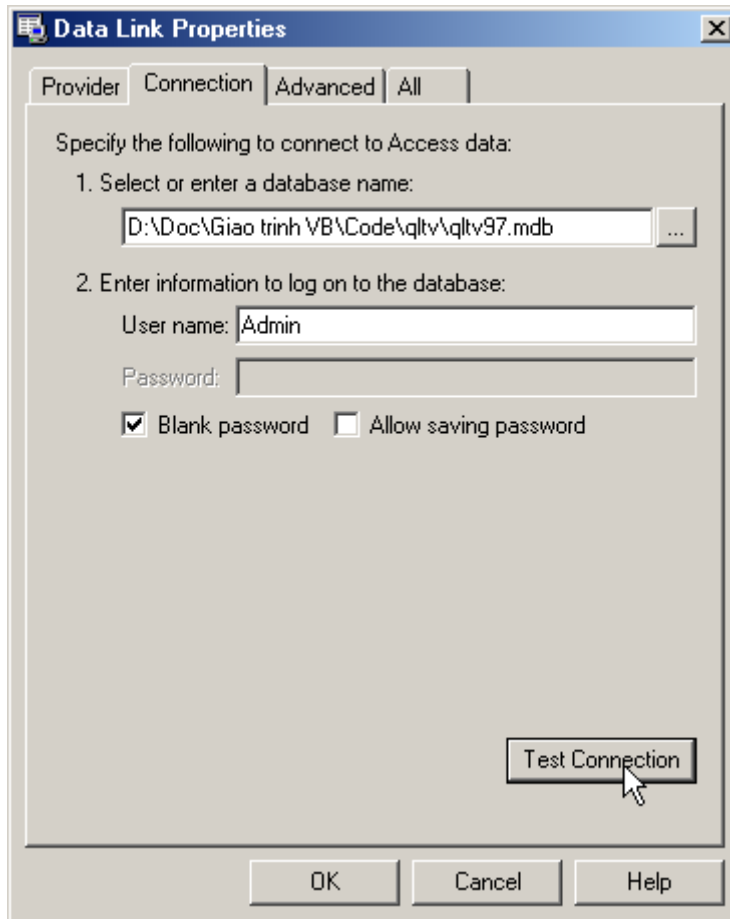
Bước 2: Thiết lập kết nối tới tệp CSDL bằng cách:

Nhấn phải chuột lên mục **Connection** trên Data Environment (hình trên) và chọn **Properties**, hộp thoại **Data Link Properties** xuất hiện như sau:

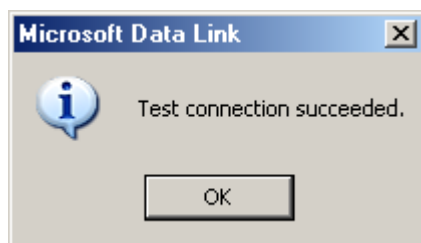


Ở thẻ **Provider** hãy chọn dịch vụ *Microsoft Jet 3.51 OLE DB Provider* (hình trên).

Ở thẻ **Connection**, hộp **Select or enter a databasename:** hãy chọn đến tệp CSDL Access (hình dưới):



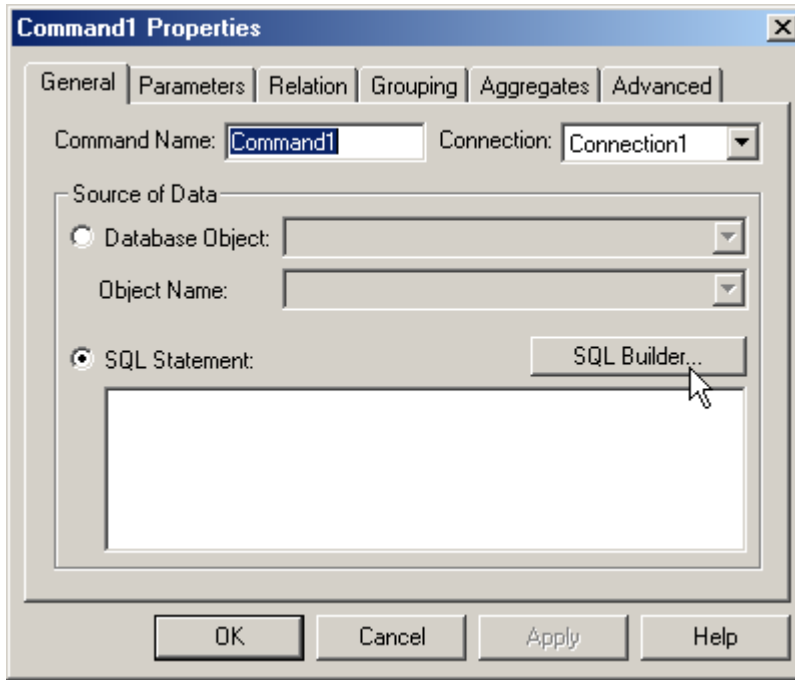
Nhấn nút **Test Connection** để kiểm tra kết quả kết nối, nếu hộp thoại dưới đây hiển thị:



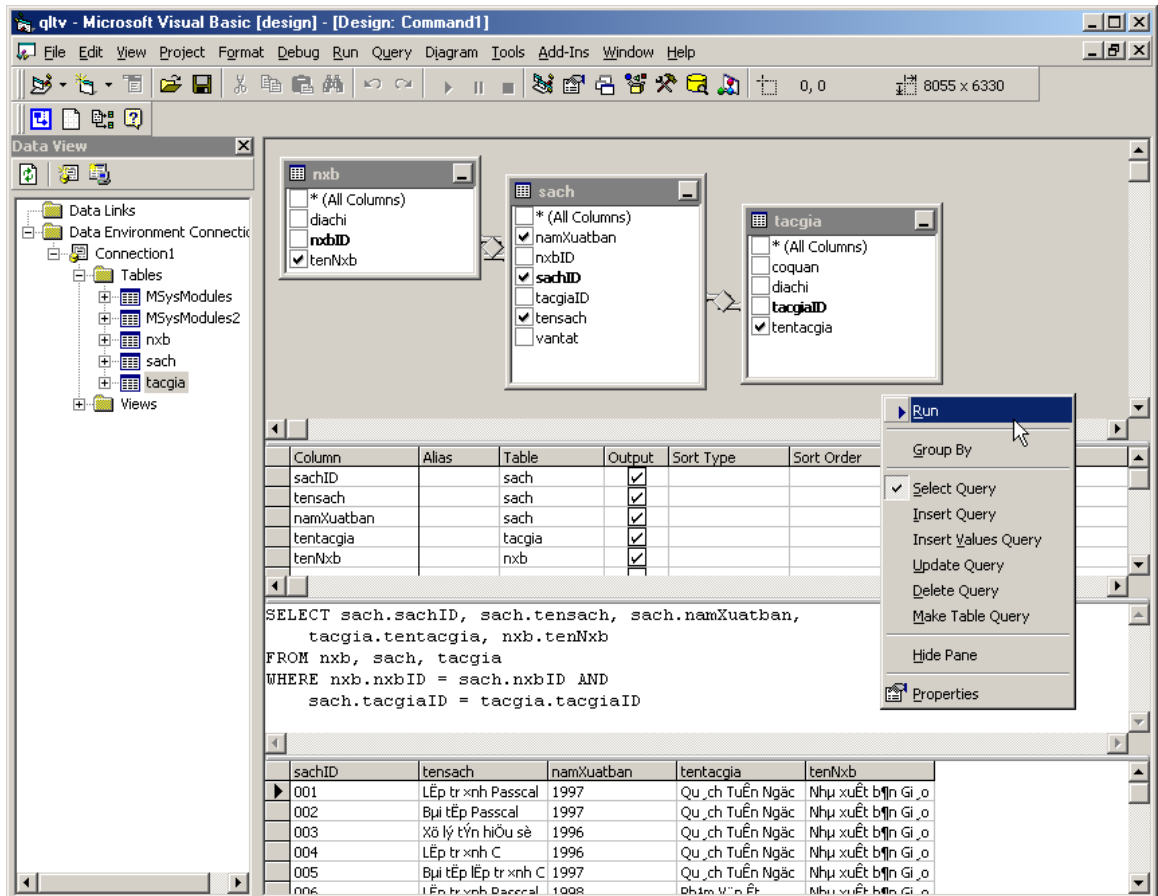
Tức là đã kết nối thành công tới tệp CSDL.

Bước 3: tạo truy vấn dữ liệu cần in ra báo cáo.

Nhấn phải chuột lên Connection vừa làm việc, chọn lệnh Add Command, hộp thoại sau xuất hiện:



Tiếp theo nhấn nút lệnh SQL Builder để xây dựng truy vấn cho Command này, một màn hình thiết kế query (gần giống trong Access) xuất hiện cho phép thiết kế một truy vấn dữ liệu đến CSDL:

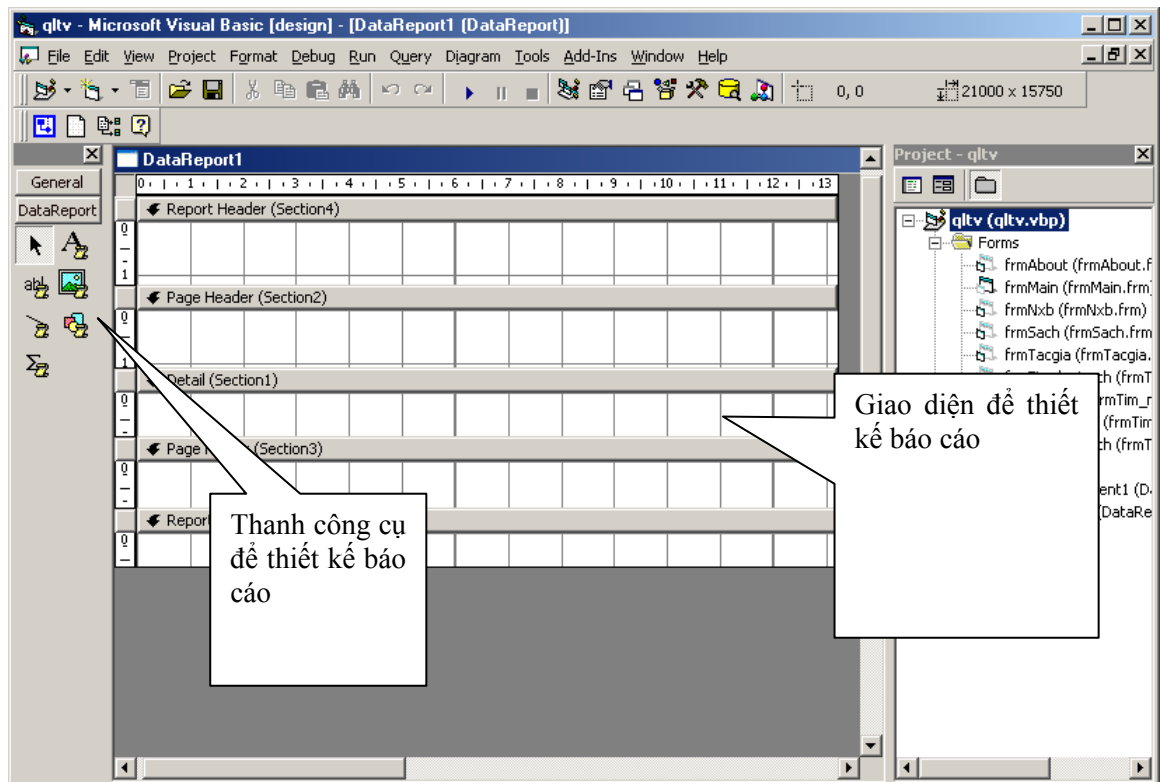


Trên hộp thoại trên có thể:

- Đưa các bảng lên query (Add Table) bằng cách dùng chuột kéo bảng cần chọn từ cửa sổ Data View sang phần thiết kế query;
- Chọn một trường dữ liệu nào đó để truy vấn bằng cách tích lên hộp chọn trường đó trên bảng thiết kế;
- Xem dữ liệu đã truy vấn được bằng cách nhấn phải chuột lên hộp thoại thiết kế query, chọn lệnh Run.

3.2 Thiết kế Data Report

Bước 1: Để đưa một Data Report mới lên Project, ra lệnh Project \ Data Report. Màn hình thiết kế Report xuất hiện như sau:



Màn hình thiết kế report gồm 2 phần:

- Thanh công cụ để thiết kế report- chứa các điều khiển cần thiết cho việc thiết kế dữ liệu lên report. Bao gồm các điều khiển như:



RptLabel – dùng tạo ra các nhãn văn bản trên báo cáo (giống như Label trên form).



RptTextbox - để in dữ liệu từ các trường (dat field) ra report



RptLine - để kẻ đường thẳng



RptImage - để chèn các hình ảnh lên report



RptFunction - để đưa các hàm tổng hợp lên report



RptShape - để vẽ các khối hình lên report.

- Giao diện để thiết kế báo cáo. Đó cũng thể hiện cấu trúc hiển thị dữ liệu trên report. Thông thường, mỗi report gồm 5 phần cơ bản:

Phần 1: Page Header - mỗi report sẽ chỉ có duy nhất một Page Header, đó là phần dữ liệu hiển thị trên cùng (Header) của mỗi trang in;

Phần 2: Page Footer - mỗi report sẽ chỉ có duy nhất một Page Footer, đó là phần dữ liệu hiển thị dưới cùng (Footer) của mỗi trang in;

Phần 3: Detail – là phần hiển thị dữ liệu chi tiết từ các bản ghi trong truy vấn dữ liệu ra report. Truy vấn được bao nhiêu bản ghi, Detail sẽ gồm chừng đó lượt in dữ liệu ra báo cáo. Mỗi report chỉ gồm duy nhất một Detail.

Phần 4: Report Header – là tiêu đề đầu của Report. Dữ liệu của phần này hiển thị ngay sau dữ liệu phần Page Header của trang in đầu tiên.

Phần 5: Report Footer – là tiêu đề cuối của Report. Dữ liệu của phần này hiển thị ngay trước dữ liệu phần Page Footer của trang in cuối cùng.

Bước 2: Thiết lập một số thuộc tính cho Data Report

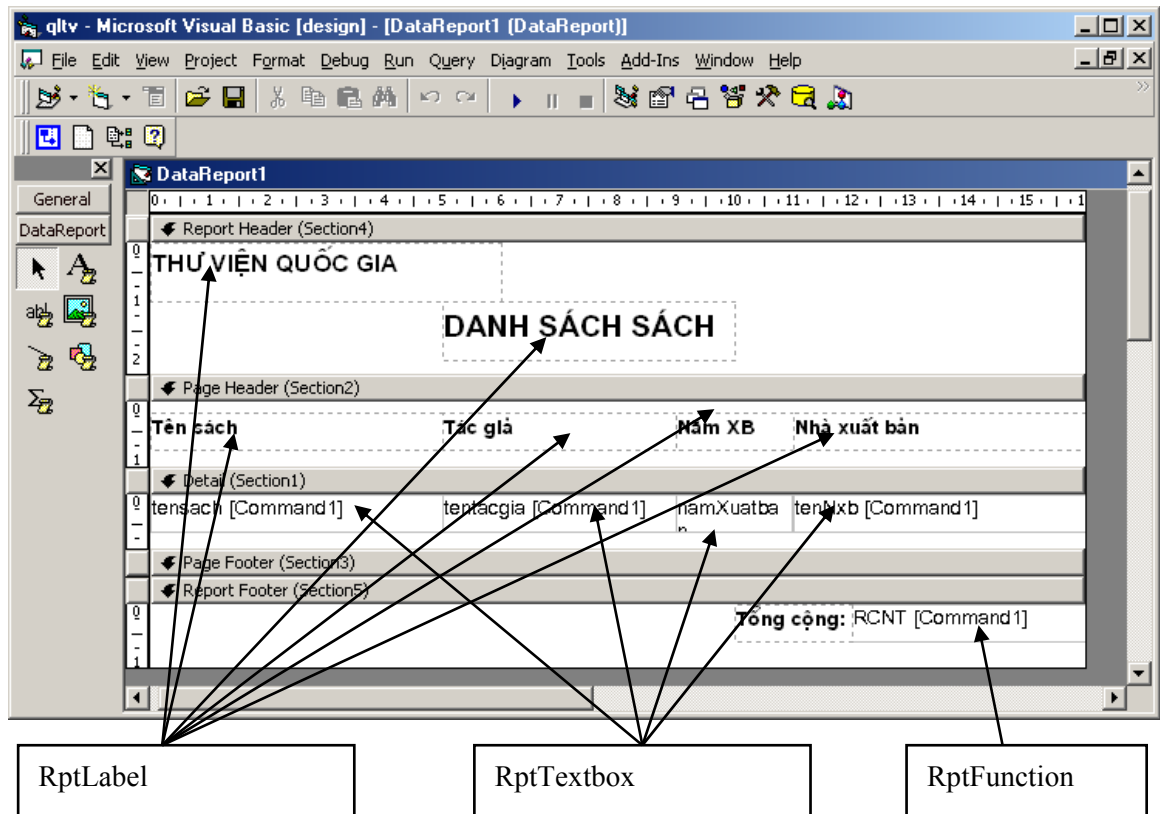
Thuộc tính DataSource - chỉ ra môi trường dữ liệu (Data Environment) nào sẽ cung cấp dữ liệu cho report;

Thuộc tính Datamember - chỉ ra truy vấn dữ liệu (Command) nào sẽ cung cấp dữ liệu cho report.

Bước 3: Thiết kế report

Sử dụng các điều khiển trên thanh công cụ phù hợp để thiết kế và định dạng report.

Hình dưới là một thiết kế report in ra thông tin các cuốn sách từ CSDL Quản lý thư viện.



3.3 Sử dụng Data Report

Gọi một Data Report để xem và in dữ liệu cách làm như với một form. Cú pháp lệnh là:

```
<Tên Data Report>.Show
```

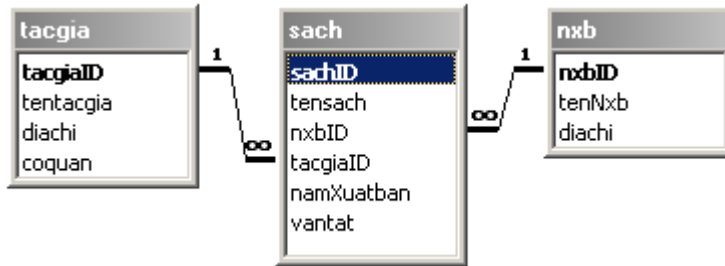
Sau khi nhận được lệnh này trong chế độ Running Time, màn hình xem dữ liệu trước khi in của Data Report xuất hiện như sau:



Đến đây có thể thực hiện việc in dữ liệu một cách dễ dàng như trên Word và Excel.

Bài tập chương 3

1. Thiết kế CSDL Quản lý thư viện trên Access như sau:



Các công việc tiếp theo được thực hiện trên một Project của VB:

2. Thiết kế form cập nhật dữ liệu bảng NXB với giao diện sử dụng như sau:

The form is titled "Danh mục NXB" and contains a table for updating publisher information. The table has the following data:

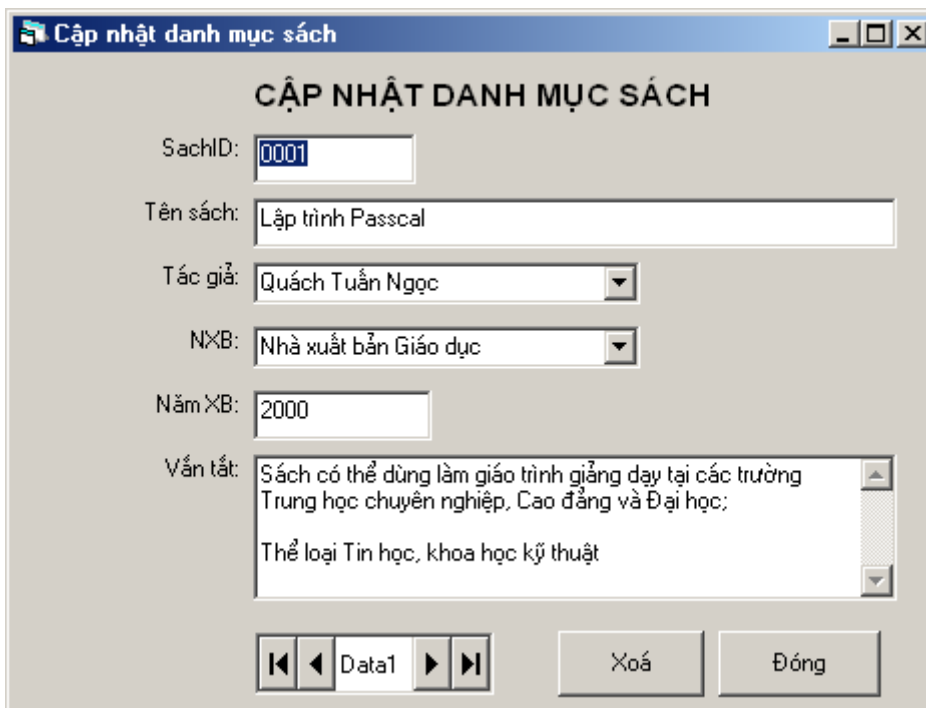
Mã NXB	Tên NXB	Địa chỉ
01	Nhà xuất bản Giáo dục	
02	Nhà Xuất bản Thống kê	
03	Nhà xuất bản Khoa học kỹ thuật	
04	Nhà xuất bản Thanh niên	
*		

At the bottom of the form, there are three buttons: "Xóa", "In", and "Đóng".

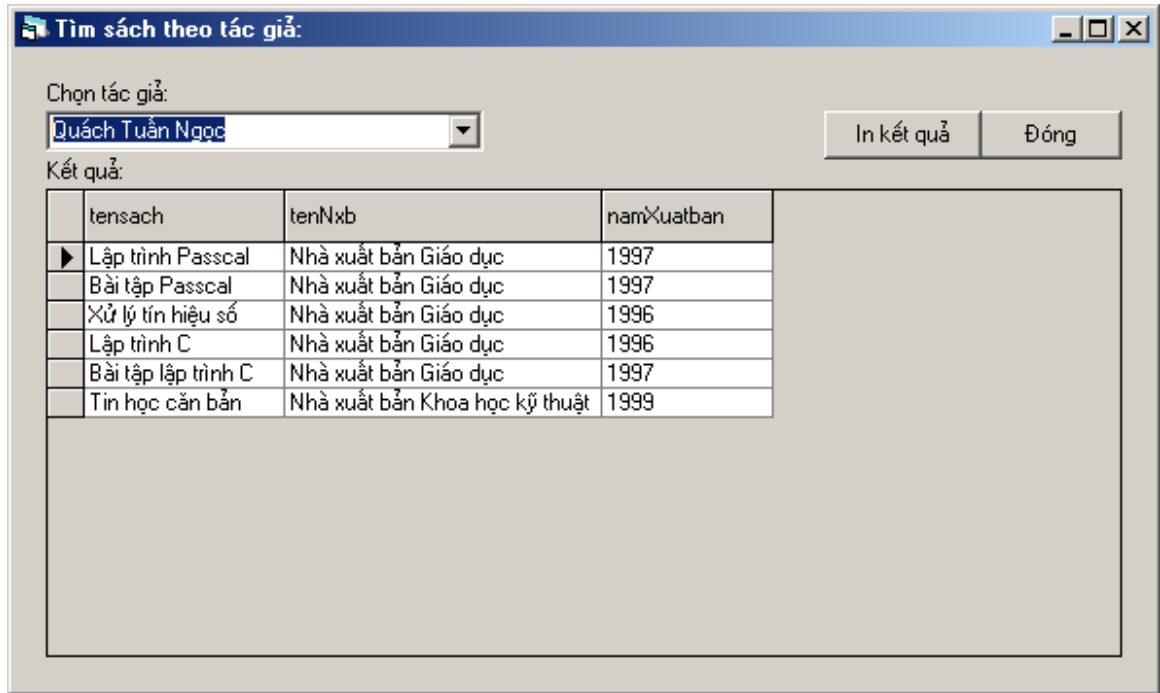
3. Thiết kế form cập nhật dữ liệu cho bảng TACGIA với giao diện sử dụng như sau:



4. Thiết kế giao diện nhập dữ liệu cho bảng SÁCH với giao diện sử dụng như sau:



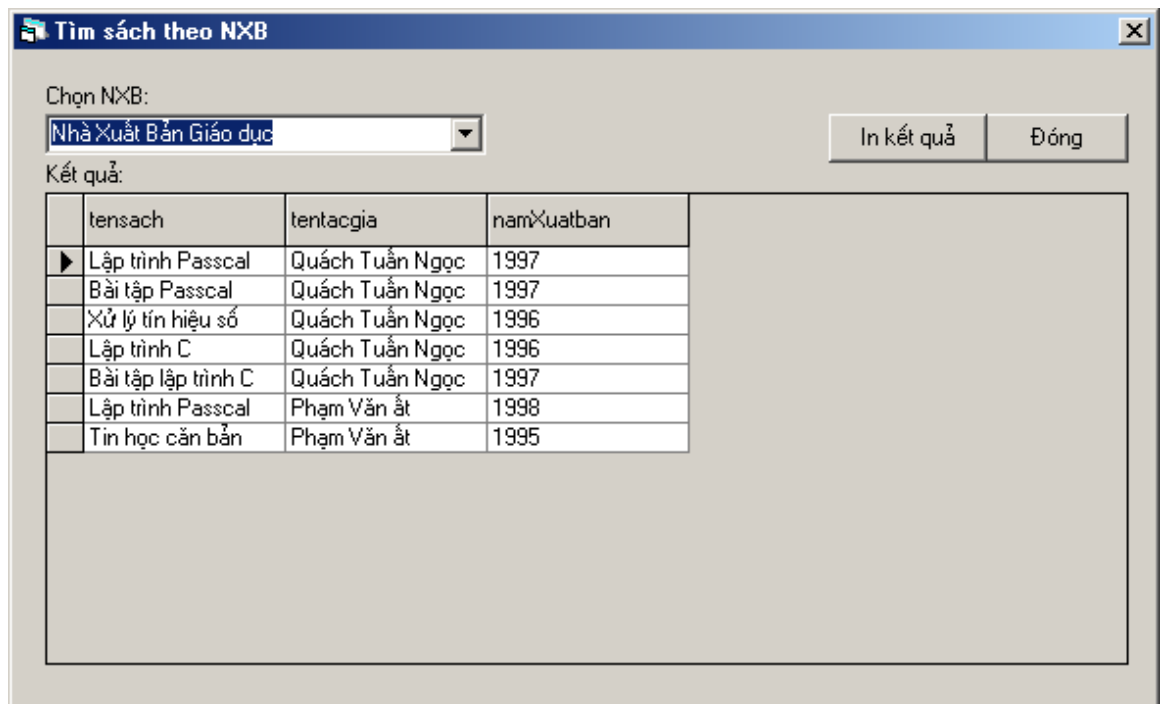
5. Thiết kế form phục vụ tra cứu sách theo một tác giả nào đó như sau:



Sau khi chọn tên một tác giả, thông tin về các cuốn sách của tác giả đó được hiển thị trên DbGrid;

Nhấn nút **In kết quả**, thông tin các kết quả này sẽ được in ra một Data Report.

6. Thiết kế form tra cứu sách theo một nhà xuất bản nào đó như sau:



Sau khi chọn một Nhà xuất bản, thông tin về các cuốn sách của nhà xuất bản đó sẽ được hiển thị lên một DbGrid;

Nhấn chuột lên nút **In kết quả**, dữ liệu tìm được sẽ in ra một Data Report.

7. Thiết kế form tìm kiếm sách theo tên gần đúng. Tên gần đúng là tên có chứa trong trường Tensach. Giao diện sử dụng form yêu cầu như sau:

Gõ vào tên sách cần tìm:

Tin học

Tìm..

In kết quả

Đóng

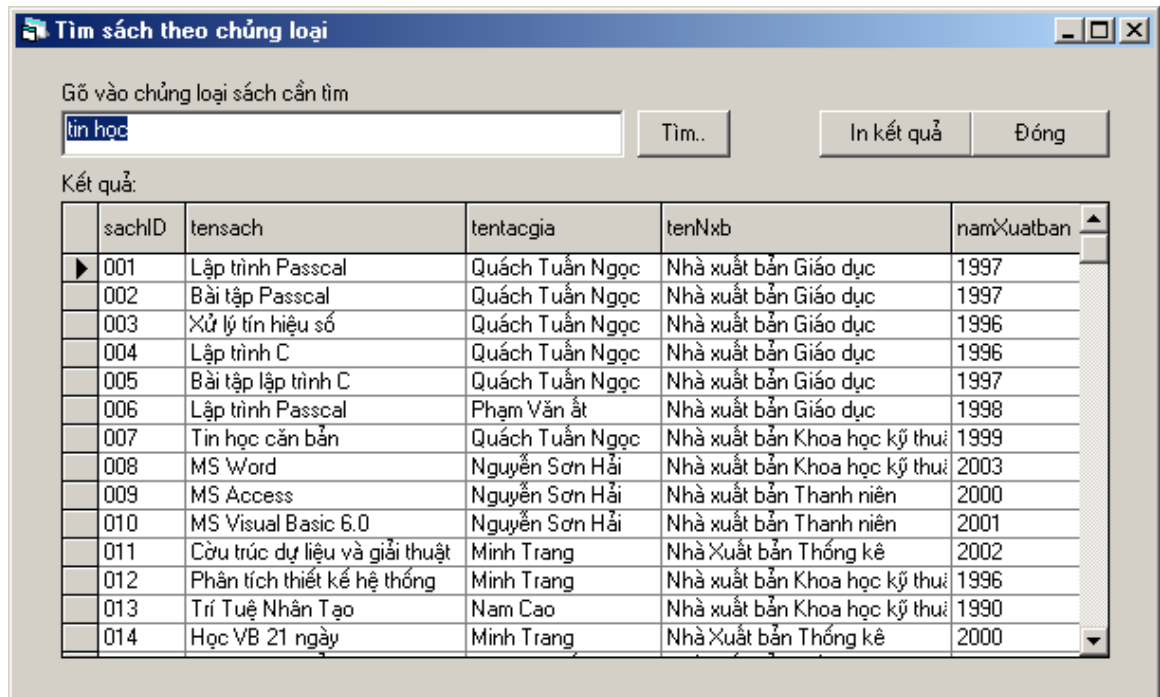
Kết quả:

	sachID	tensach	tentacgia	tenNxb	namXuatban
▶	007	Tin học căn bản	Quách Tuấn Ngọc	Nhà xuất bản Khoa học kỹ thuật	1999
	015	Tin học căn bản	Phạm Văn ắt	Nhà xuất bản Giáo dục	1995

Sau khi nhập một tên cần tìm, nhấn nút **Tìm**. Danh sách các cuốn sách mà tên sách có chứa cụm từ vừa gõ vào sẽ được hiển thị lên DbGrid;

Nhấn nút **In kết quả**, thông tin này sẽ được in ra một Data Report.

8. Thiết kế form tìm kiếm sách theo chủng loại (thông tin chủng loại của mỗi cuốn sách được lưu vào trường VANTAT) như sau:



Sau khi gõ vào tên chủng loại cần tìm, nhấn nút **Tìm**, phần mềm sẽ liệt kê các cuốn sách có chứa cụm từ vừa nhập vào trong trường VANTAT;

Nếu nhấn nút **In kết quả**, toàn bộ kết quả tìm được sẽ được in ra một Data Report.

CHƯƠNG 4

HOÀN THIỆN DỰ ÁN

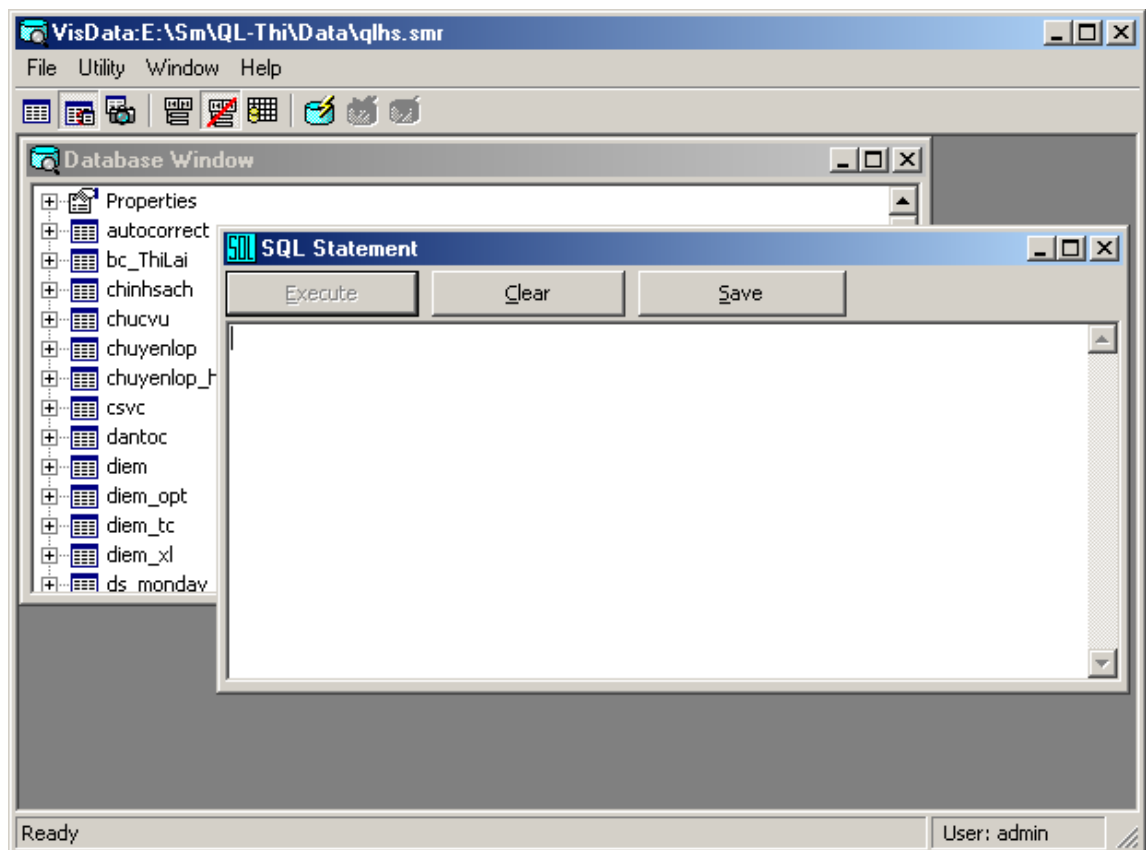
Chương này trang bị những kiến thức cũng như kỹ thuật thiết kế giao diện chính cho một ứng dụng và dịch, đóng gói dự án thành bộ cài đặt ứng dụng mang tính thương mại. Nội dung của chương tập trung vào các vấn đề:

- Thiết kế giao diện chính ứng dụng;
- Thiết kế Menu, Toolbar;
- Dịch ứng dụng ra dạng .exe;
- Đóng gói dự án thành bộ cài đặt phần mềm.

1. Thiết kế MDI Form

1.1 Một số khái niệm

MDI Form (Multiple Document Interface Form) là một kiểu giao diện ứng dụng được dùng khá phổ biến, ví dụ như: Microsoft Word, Microsoft Excel, Microsoft Powerpoint, Microsoft Visual Basic, ... Trong một ứng dụng, MDI là form giao diện chính, nơi chứa hệ thống thực đơn (menu), thanh công cụ (Toolbar) để có thể truy cập được đến các chức năng của ứng dụng. Đây có lẽ là giải pháp tốt nhất để quản lý việc thực hiện các mô đun chức năng trong một ứng dụng. Hình dưới minh họa một ứng dụng sử dụng giải pháp MDI Form cho ứng dụng:



Trong VB, mỗi Project chỉ tồn tại nhiều nhất một MDI form. Khác với form thông thường, trên MDI chỉ có thể thiết kế được một số các điều khiển như:

Menu, ToolBar, Picture Box, ListImage, StatusBar, Timer và Common Dialog. Đa số các điều khiển còn lại không thể đặt trực tiếp lên MDI form được, nếu muốn phải đặt chúng trên một điều khiển dạng Container như Picture Box.

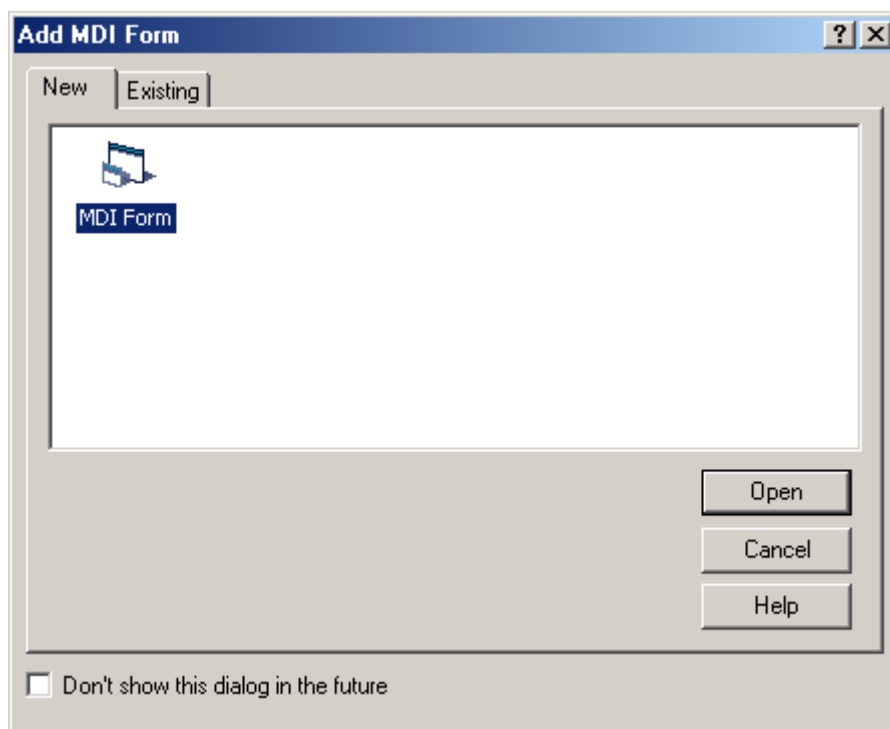
Những form được quản lý bởi MDI form được gọi là MDI Child forms (form con). Khi MDI form đóng lại, toàn bộ các form con đang mở cũng sẽ bị giải phóng khỏi bộ nhớ.

Với hình trên, cửa sổ có tiêu đề VisData ... là MDI form, 2 cửa sổ còn lại: Database Window và SQL Statement đều là các MDI Child forms.

1.2 Sử dụng MDI form

Bước 1: Đưa một MDI form lên project. Cách làm như sau:

- Thực đơn **Project | Add MDI form**. Một hộp thoại xuất hiện:

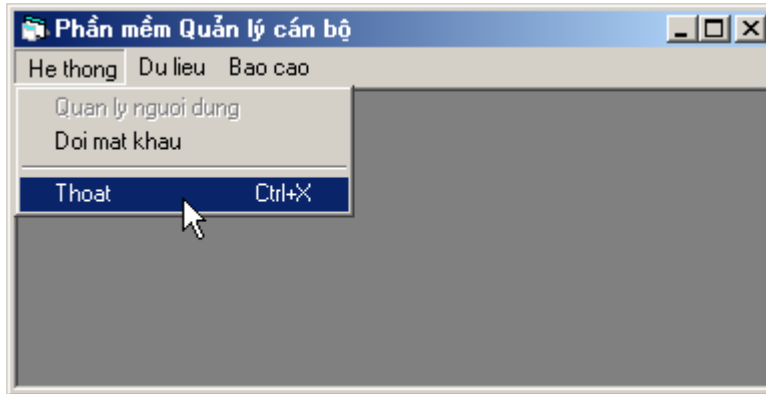


- Chọn **MDI Form**, nhấn **Open**, một MDI form được chèn vào Project;
- Tiếp theo có thể thiết kế menu, toolbar và các giao diện cần thiết khác lên MDI và gài đặt các thủ tục cần thiết (tìm hiểu tiếp ở các mục sau).

Bước 2: Thiết lập các MDI Child form bằng cách: thiết lập thuộc tính **MDIChild = True** cho các form muốn làm MDI Child.

2. Thiết kế Menu

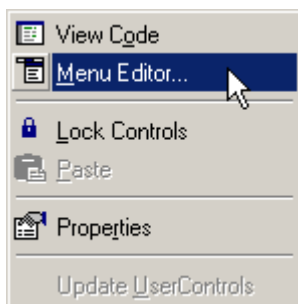
Menu là loại điều khiển phổ biến dùng để gọi các thủ tục, các giao diện hoặc chức năng của một ứng dụng ra làm việc. Thông thường, menu được thiết kế ngay trên MDI form của mỗi ứng dụng.



Trong VB, muốn tạo ra menu cho ứng dụng, cần sử dụng trình Menu Editor. Tiếp theo đây, sẽ hướng dẫn cách sử dụng Menu Editor để thiết kế hệ thống menu trên một MDI form.

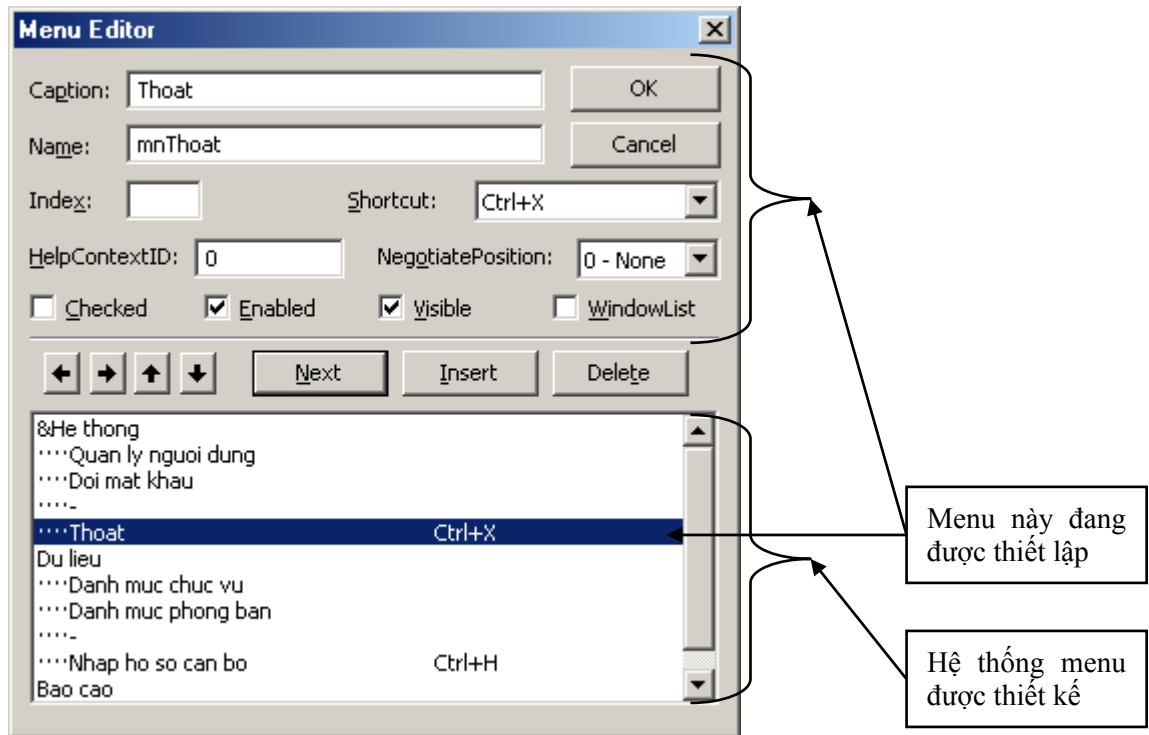
Bước 1: Tạo một MDI Form – là giao diện chính của ứng dụng, nơi sẽ chứa menu cần tạo (hình trên);

Bước 2: Gọi trình Menu Editor bằng cách: nhấn phải chuột lên MDI Form, tiếp theo chọn **Menu Editor** (hình dưới):




→ trình **Menu Editor** xuất hiện!

Bước 3: Thực hiện thiết kế cấu trúc menu trên hộp thoại Menu Editor như sau:



Toàn bộ hệ thống menu trên MDI Form được thiết kế trên một hộp thoại Menu Editor (một phần của chúng như hình trên). Cách thức: xây dựng từng mục của menu. Mỗi mục của menu có thể phải thiết lập các thông tin sau:

- **Caption** – nhãn thể hiện của mục menu (bắt buộc phải nhập). Đặc biệt:
 - Để tạo đường phân cách (=====) thiết lập Caption là – (gạch ngang);
 - Để thiết lập phím nóng cho mục menu, gõ ký tự & lên trước ký tự muốn thiết lập phím nóng (chú ý ký tự H của mục &He thong).
- **Name** - định tên cho mỗi mục menu. Giá trị này phải duy nhất và bắt buộc phải nhập (Name không được trùng nhau);
- **Shortcut** - chọn phím nóng cho mục menu này (với những mục menu cấp một như: Hệ thống hoặc Dữ liệu hoặc Báo cáo ở trên không thể thiết lập được phím nóng, chúng chỉ dành cho các mục menu cấp 2 trở đi);
- **Enable** – cho phép làm việc hoặc cấm (không cho phép làm việc) mục menu này. Như hình trên, mục menu Quan ly nguoi dung là đang bị cấm;

- **Visible** – cho phép hiển thị hoặc ẩn mục menu. Nếu không chọn thuộc tính này, mục menu đó sẽ không hiển thị trên hệ thống menu khi chương trình hoạt động;
- Nút **Next** để chuyển đến mục menu tiếp theo sau;
- Nút **Insert** - để chèn thêm một mục menu trống;
- Nút **Delete** - để xóa mục menu đang chọn;
- Hệ thống các nút  để dịch chuyển vị trí cũng như thứ cấp của các mục menu;
- Thiết lập xong nhấn **OK** để chấp nhận.

Bước 3: viết các thủ tục đáp ứng sự kiện cho các mục của hệ thống menu (đó là các lệnh gọi thi hành các thủ tục hoặc Form trong Project). Thông thường, viết thủ tục đáp ứng sự kiện Click cho các mục bằng cách: tại cửa sổ thiết kế MDI Form, dùng chuột trở đến mục menu cần viết thủ tục đáp ứng sự kiện, màn hình soạn thảo chương trình sẽ xuất hiện để làm việc.

Lệnh hiển thị một Form ra để làm việc là:

```
<tên form>.Show
```

Ví dụ: lệnh sau gọi form *Cập nhật hồ sơ* có Name là *frmCapNhatHoSo* được viết như sau:

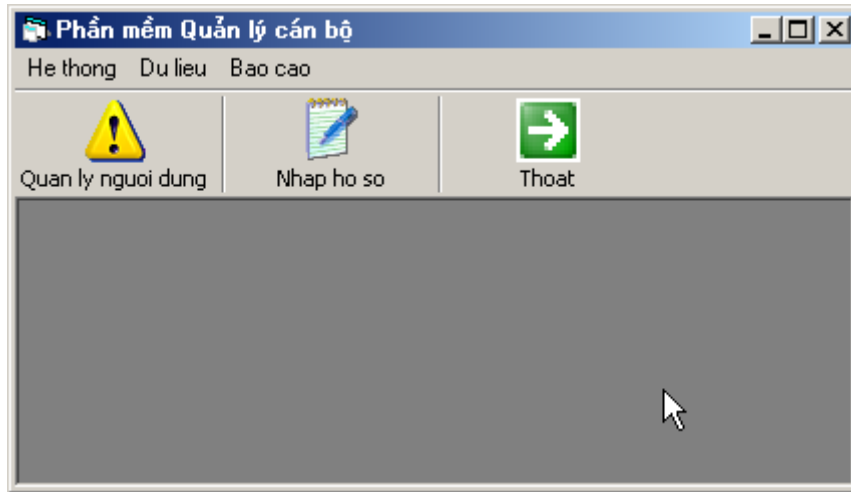
```
frmCapNhatHoSo.Show
```

3. Thiết kế ToolBar

Cùng với Menu, ToolBar là một điều khiển quan trọng để thiết kế giao diện chính cho các ứng dụng. Thông thường, mỗi ứng dụng đều sử dụng cả Menu lẫn ToolBar. Menu là công cụ chính để gọi các chức năng của ứng dụng, còn

ToolBar được dùng để gọi những tính năng hay sử dụng nhất cho mỗi ứng dụng. Bởi vì, dùng ToolBar thao tác nhanh hơn Menu.

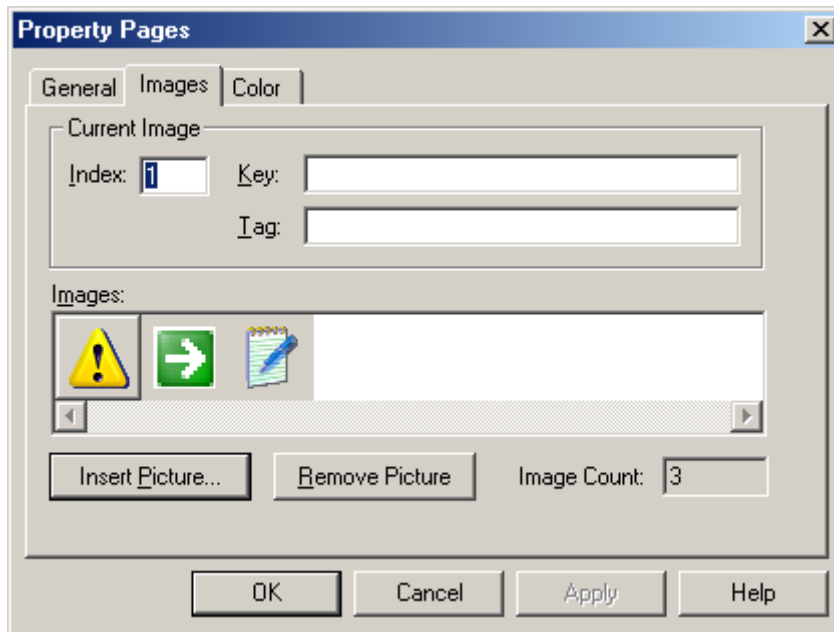
Dưới đây là một giao diện chính của ứng dụng sử dụng cả Menu và ToolBar:



Dưới đây sẽ hướng dẫn cách thiết kế và cài đặt ToolBar trên theo đúng mục đích của nó.

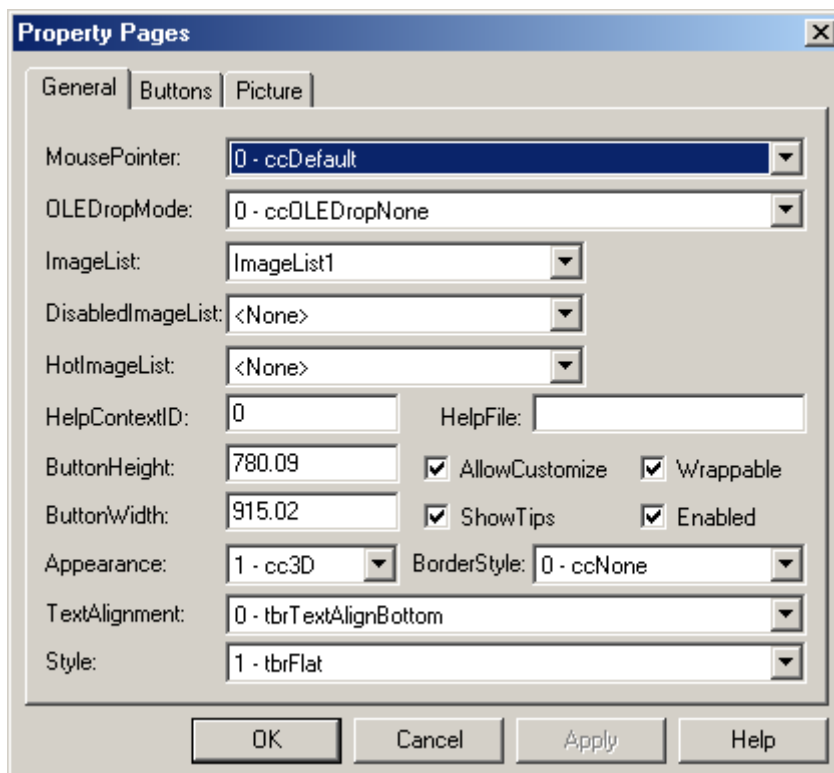
Bước 1: Một Toolbar có thể được trang trí những nút lệnh bằng hình ảnh (như hình trên), khi đó cần tạo một điều khiển ImageList trên form có chứa Toolbar này. Điều khiển ImageList (cả Toolbar) cùng nằm trong thư viện **Microsoft Windows Common Controls 6.0**. Nên phải khai báo thư viện này trên cửa sổ Components trước khi có thể sử dụng chúng. Cách thiết kế ImageList như sau:

- Đưa một ImageList lên Form (giả sử có tên ImageList1);
- Nhấn nút phải chuột lên ImageList, chọn **Properties**, hộp thoại **Property Page** xuất hiện như sau:



- Sử dụng nút **Insert Picture** để đưa danh sách các ảnh sẽ sử dụng vào thiết kế ToolBar lên ImageList này (hình trên đã đưa được 3 ảnh).

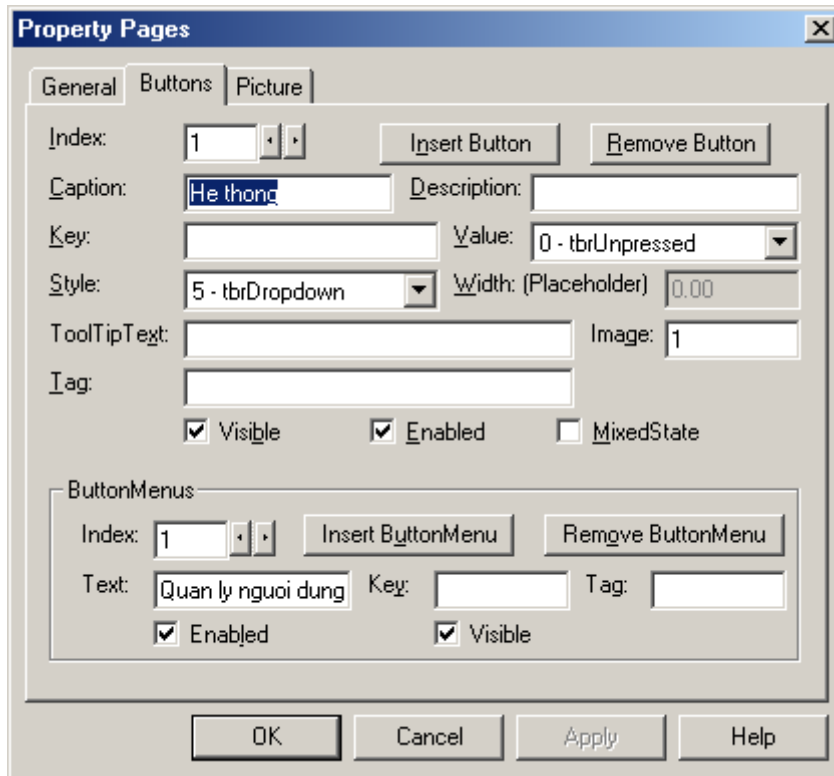
Bước 2: Đưa đối tượng ToolBar lên form. Nhấn phải chuột lên ToolBar, chọn **Properties**, hộp thoại thiết kế ToolBar xuất hiện như sau:



Thẻ **General** chứa các thuộc tính chung cho ToolBar đó là:

- **ImageList** - chọn điều khiển ImageList1 đã tạo ra (tuy nhiên thuộc tính này có thể bỏ qua);
- **Style** – hình dáng của các nút trên Toolbar kiểu *1* – *tbrFlat* (kiểu phẳng) hoặc *0* – *3D* (3 chiều);

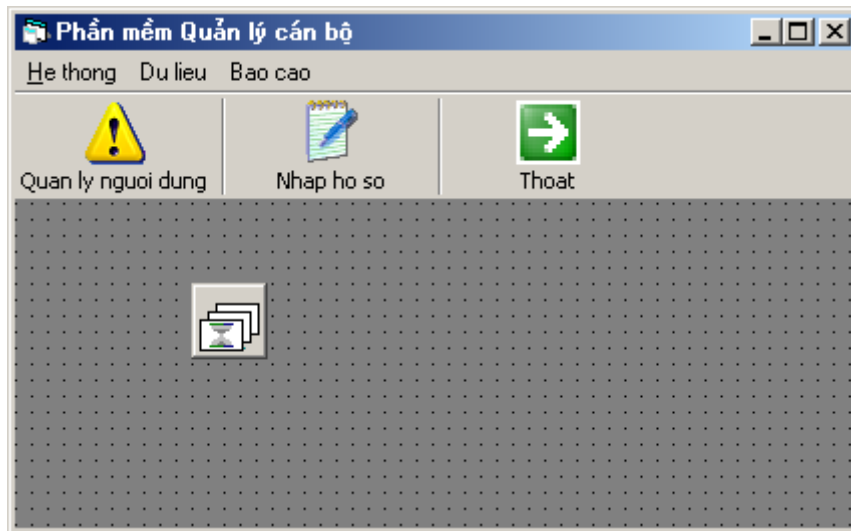
Thẻ **Buttons** để thiết kế hệ thống các nút cho Toolbar:



Để tạo mới một nút lên toolbar nhấn nút **Insert Button**. Tiếp theo khai báo các thuộc tính cho nút vừa tạo bao gồm:

- **Caption** – nhãn tiêu đề của nút;
- **Key** – khóa của nút. Giá trị khóa giữa các nút là duy nhất. Thuộc tính này cũng có thể dùng để nhận biết các nút khi dùng;
- **Style** - kiểu nút. Mỗi nút lệnh trên Toolbar có thể nhận 1 trong các kiểu như là: *0* – *tbrDefault* - kiểu nút lệnh thông thường; *1* – *tbrCheck* - kiểu nút 2 trạng thái ấn xuống và không; *3* – *tbrSeparator* – làm vách ngăn giữa các nút;...
- **ToolTip Text** - thiết lập chuỗi ký tự nhắc mỗi khi di chuột lên nút này;

- **Image** - để chọn ảnh cho nút lệnh (nếu cần). Ảnh được lấy từ điều khiển ImageList. Giá trị gõ vào đây là thứ tự của ảnh cần hiển thị lên nút trong danh sách ImageList.



Bước 3: gài đặt thủ tục cho Toolbar để gọi đến các chức năng ứng dụng như đã thiết kế. Bắt sự kiện Click của Toolbar để gài đặt các thủ tục gọi chức năng ứng dụng tương ứng. Trong sự kiện này, tham số **Button** cho biết nút vừa được nhấn chuột. Có thể thông qua giá trị **Key** của Button để biết vừa bấm lên nút nào. Với Toolbar trên, thủ tục này sẽ như sau:

```

Private Sub Tbr1_ButtonClick(ByVal Button As MSComctlLib.Button)
    '-----
    'Sử dụng thuộc tính Key để nhận biết các nút
    '
    Select Case Button.Key
        '-----
        'Key = m1 - đã nhấn lên nút Quan lý nguoi dung
        '
        Case "m1"
            frmQuanlynguoidung.Show
        '-----
        'Key = m2 - đã nhấn lên nút Nhap ho so
        '
        Case "m2"
            frmQuanlyhoso.Show
        '-----
        'Key = m3 - đã nhấn lên nút Thoat
        '
        Case "m3"
            End
    End Select
End Sub
    
```

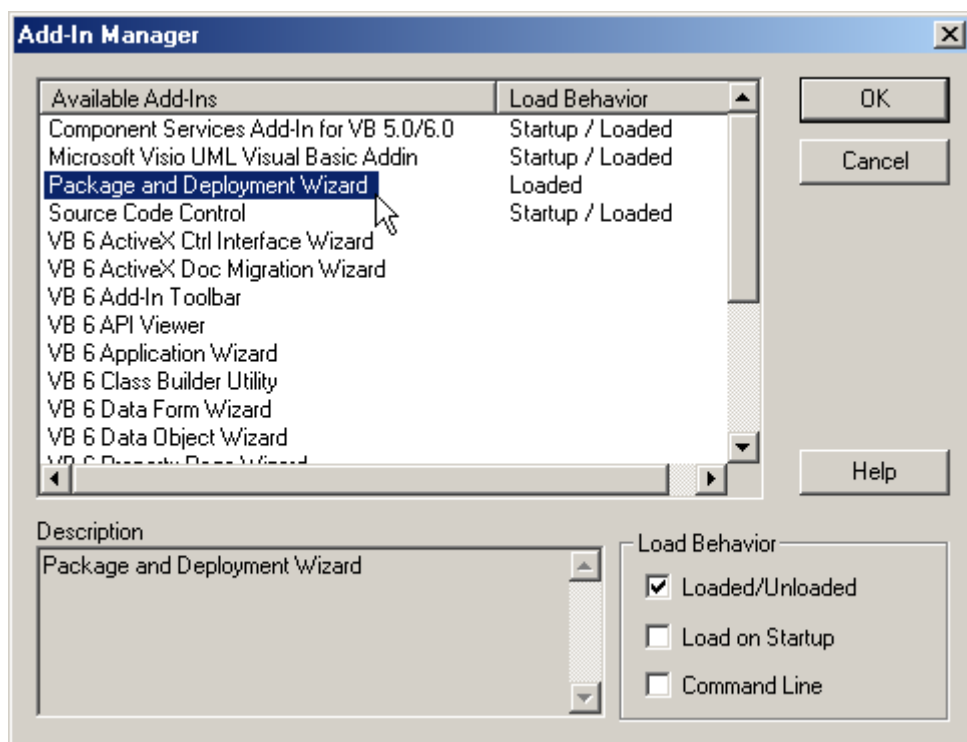

4. Đóng gói dự án


Đóng gói dự án là qui trình cuối cùng để dịch dự án viết trên VB thành bộ gài đặt phần mềm - một sản phẩm có thể chuyển giao đến khách hàng. Bước này sẽ bao gồm các công việc dịch và đóng gói toàn bộ các tài nguyên cần thiết để dự án có thể chạy độc lập trên bất kỳ một máy tính nào, kết quả sẽ là một bộ gài đặt phần mềm (Setup). Các bước thực hiện tạo một bộ gài đặt như sau:

Bước 1: Dịch Project ra tệp có thể thực thi được (tệp .exe) bằng cách ra lệnh **File \ Make <project>.exe...** Nếu không có lỗi gì, toàn bộ dự án VB sẽ được dịch ra một tệp có tên <tên project>.exe.

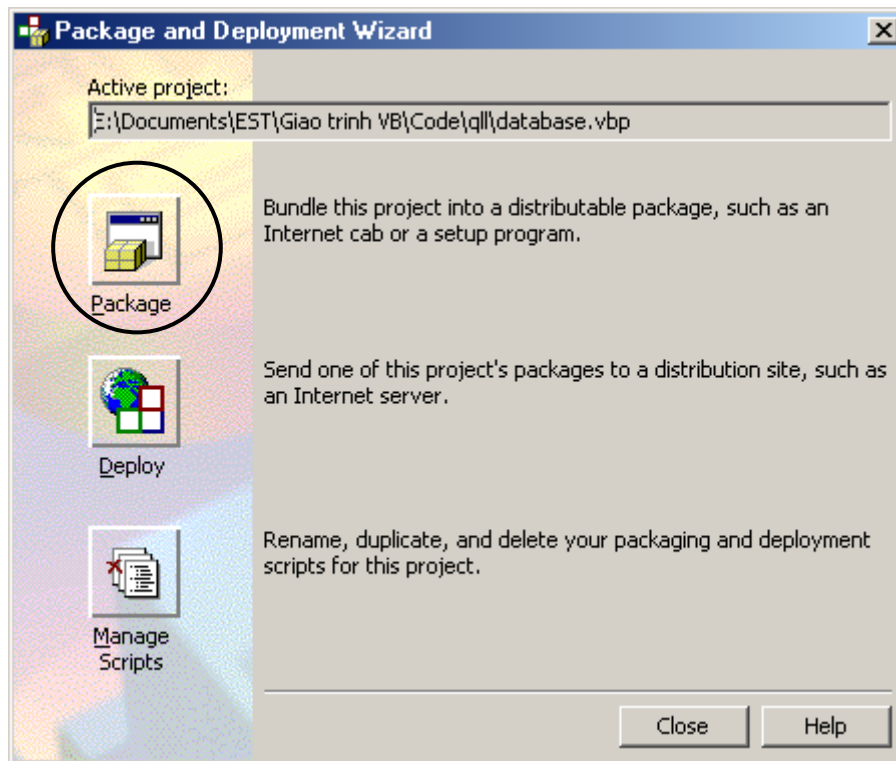
Bước 2: Kích hoạt trình Package and Deployment Wizard của VB - một công cụ dùng đóng gói dự án đi kèm VB6 bằng cách:

- Ra lệnh **Add-in \ Add – in Manager**;
- Tiếp theo kích hoạt trình Package and Deployment Wizard bằng cách nhấn đúp chuột lên mục **Package and Deployment Wizard**:

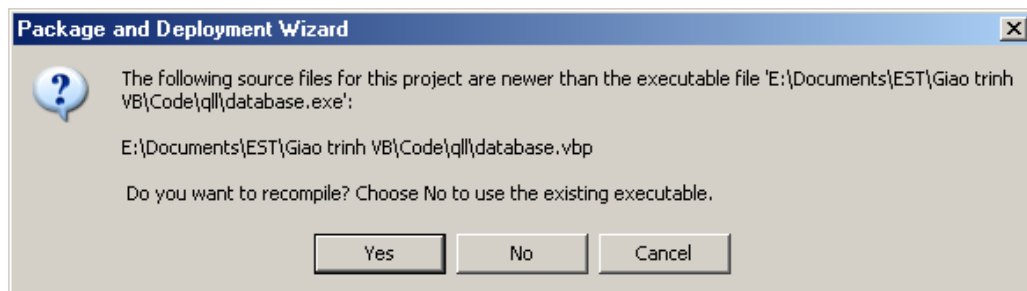


- Nhấn **Ok** để đóng hộp thoại này;
- Tiếp theo ra lệnh **Add – Ins** |  Package and Deployment Wizard... Trình Package and Deployment Wizard xuất hiện để từng bước giúp đóng gói dự án.

Bước 3: từng bước thực hiện đóng gói dự án:

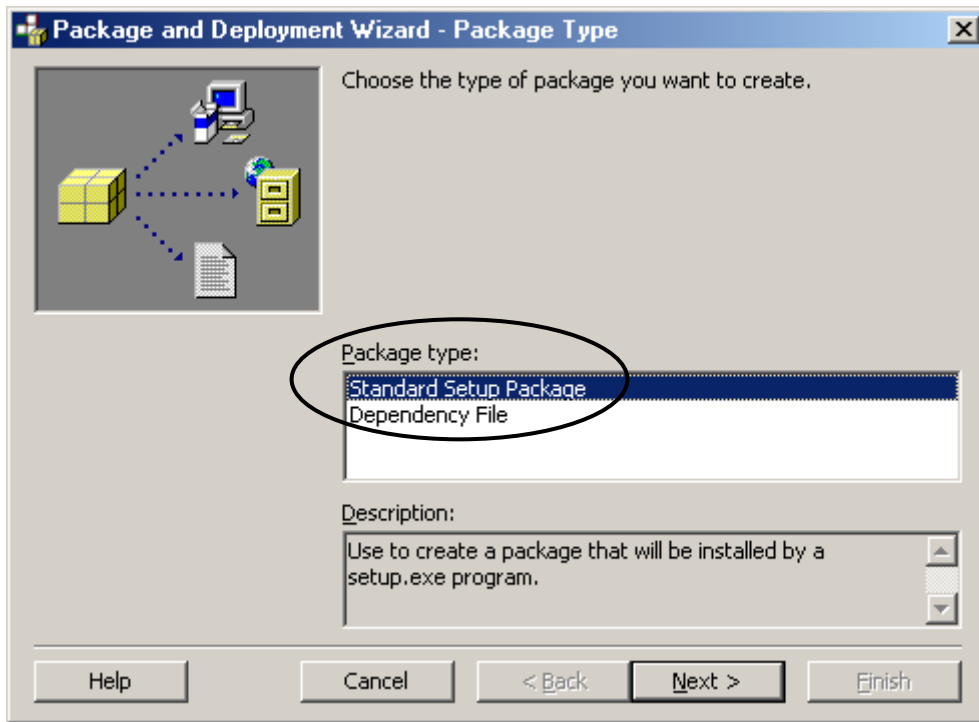


Ở hộp thoại trên, nhấn nút **Package** để tiếp tục. Hộp thoại sau xuất hiện để hỏi lại một lần nữa có muốn dịch lại dự án hay không?



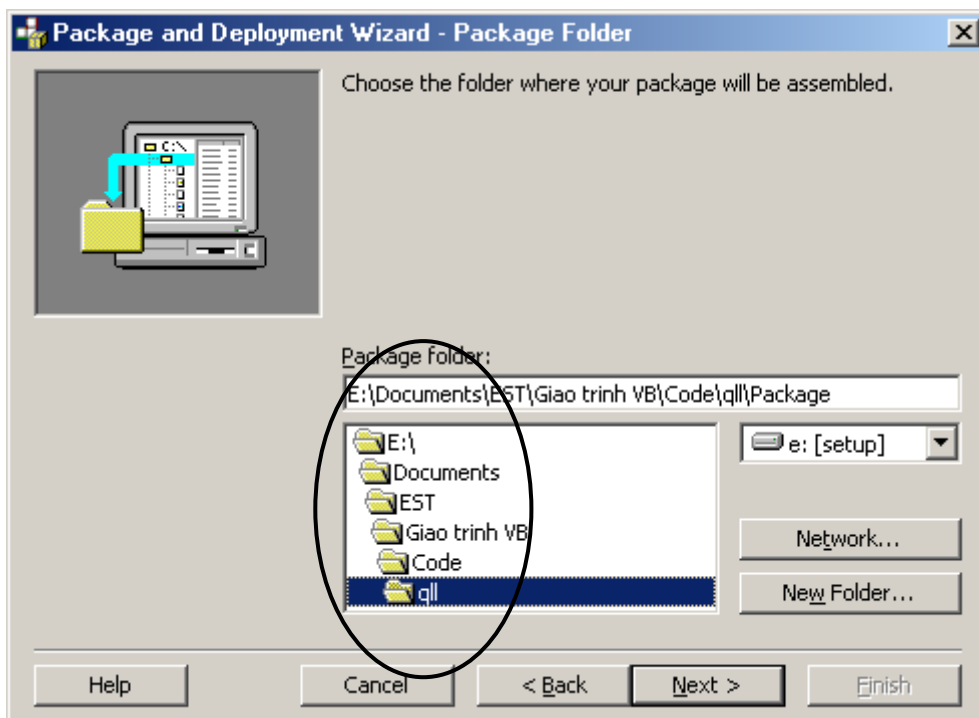
Nhấn **Yes** nếu muốn dịch lại, trái lại nhấn **No** để tiếp tục.

Hộp thoại tiếp theo xuất hiện để hỏi chọn kiểu đóng gói. Thông thường chọn kiểu **Standard Setup Package**:



Nhấn **Next** để tiếp tục:

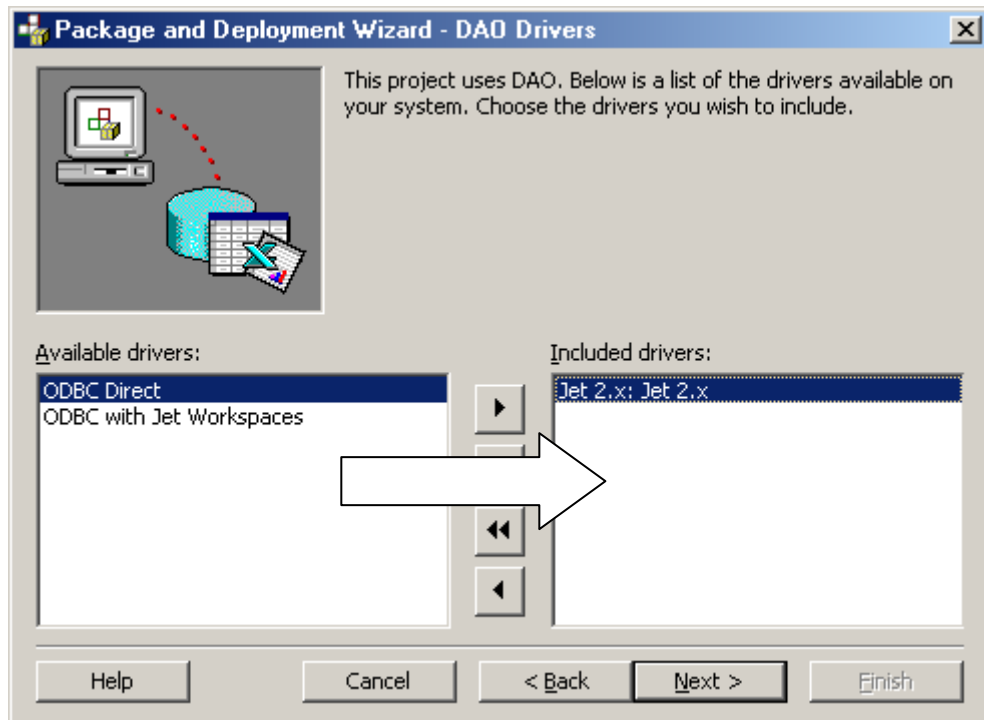
Hộp thoại tiếp theo yêu cầu xác định thư mục trên ổ đĩa nơi sẽ lưu trữ bộ gài đặt sẽ tạo được sau này:



Chọn thư mục xong, nhấn **Next** để tiếp tục:

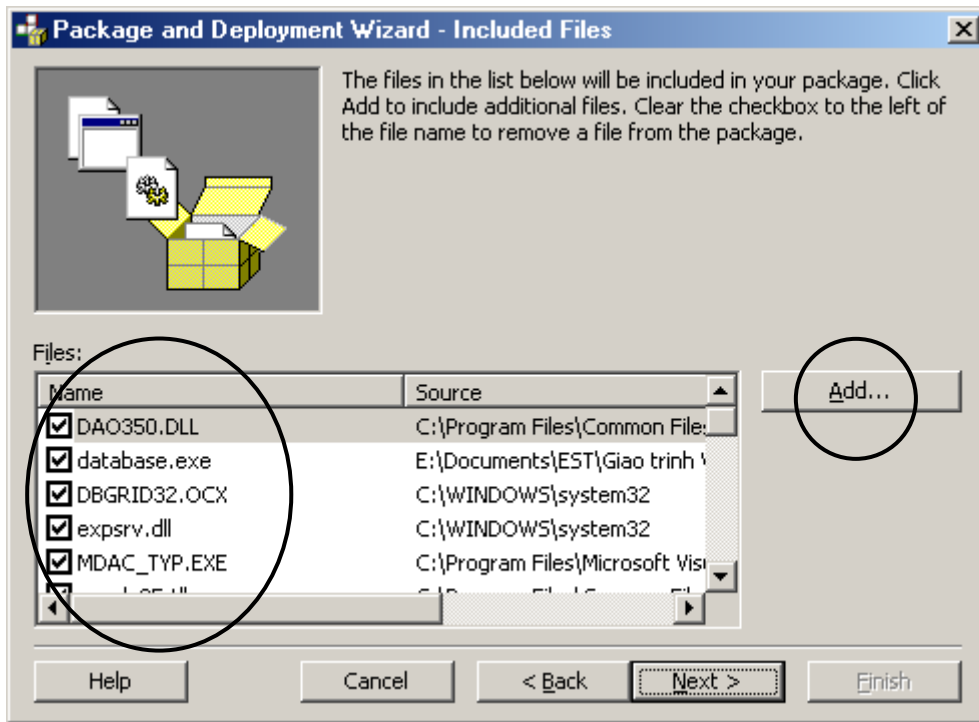
Hộp thoại sau đây xuất hiện cho phép tùy chọn các thư viện điều khiển DAO vào bộ gài đặt. VB sẽ tự động xác định các thư viện cần thiết, cách tốt

nhất là đưa chúng lên bộ gài đặt bằng cách đưa chúng từ danh sách **Available drivers** sang danh sách **Included drivers** (hình dưới):



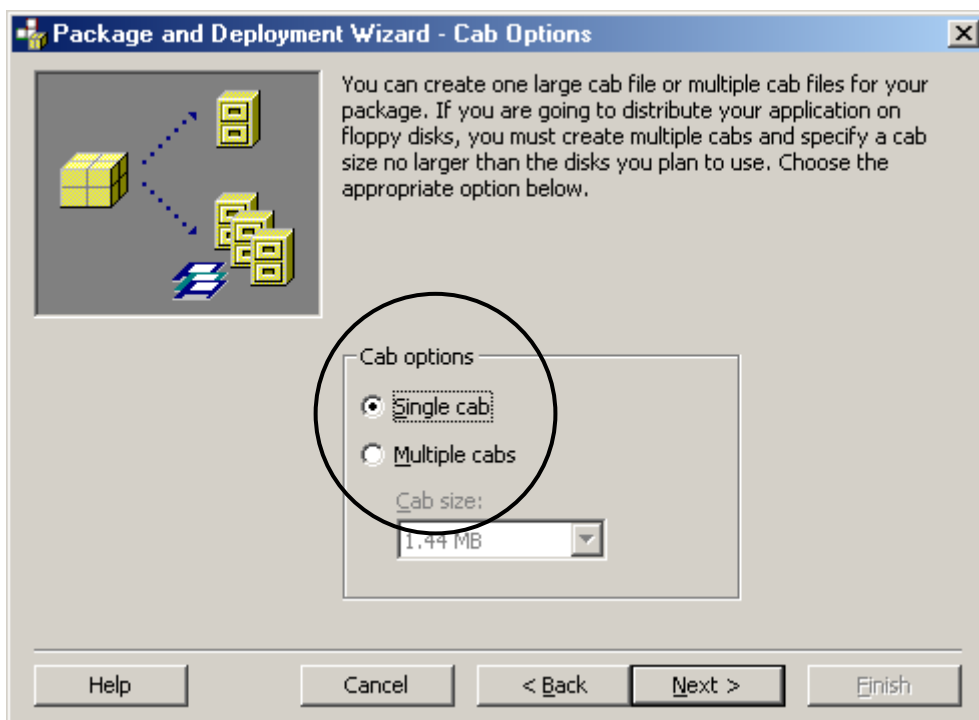
Nhấn **Next** để tiếp tục:

Hộp thoại tiếp theo xuất hiện cho phép thêm (Add) hoặc gỡ bỏ (Remove) các tệp tin trên bộ đóng gói. Thông thường, các tệp CSDL cũng như một số các tệp tin liên quan khác đến dự án VB phải sử dụng tính năng này đưa vào bộ gài đặt. Nút **Add** để thực hiện thêm các tệp tin:



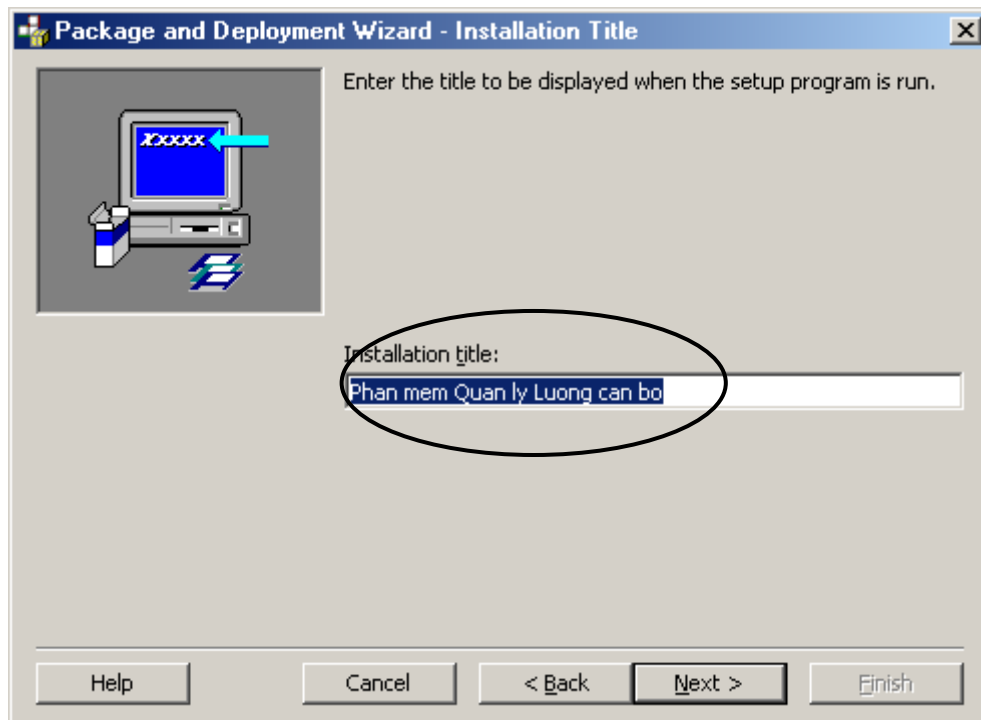
Nhấn **Next** để tiếp tục:

Hộp thoại tiếp theo để lựa chọn cách tạo ra các tệp tin trong bộ gài đặt. Có hai tùy chọn là: **Single cab** – toàn bộ đĩa gài đặt chỉ gom lại thành một tệp tin Setup.exe; hoặc **Multiple cab** - sẽ chia bộ đĩa gài đặt thành các tệp tin định dạng nén có dung lượng nhỏ (trước đây thường dùng tùy chọn này để có thể copy bộ gài đặt ra các đĩa mềm).



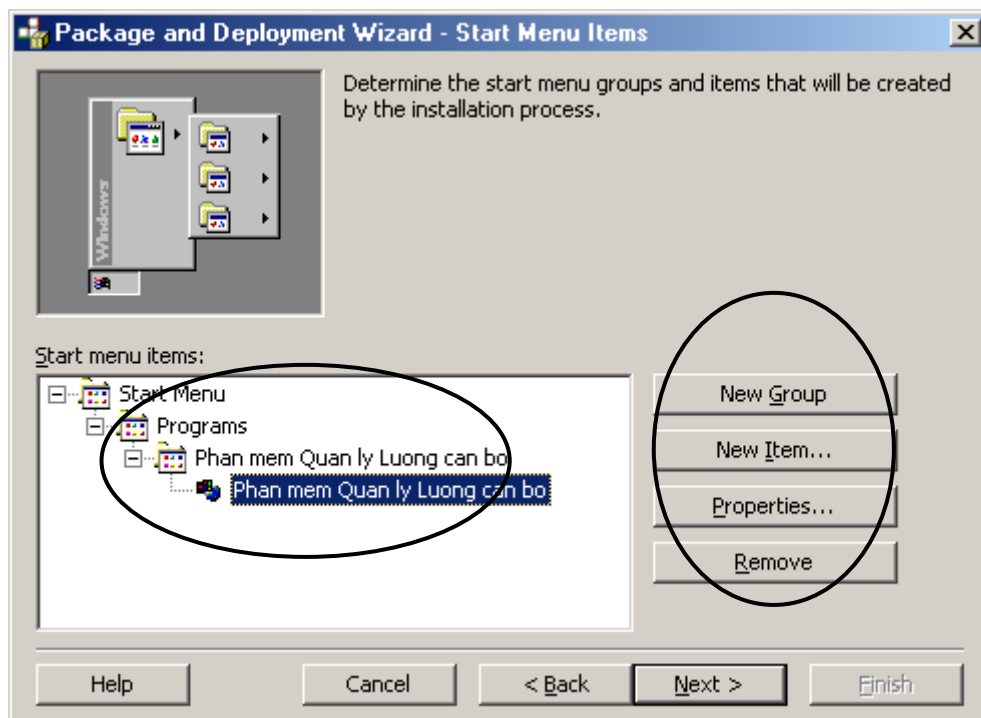
Nhấn **Next** để tiếp tục:

Hộp thoại tiếp theo dùng gõ tiêu đề bộ giải đặt. Hãy gõ tiêu đề phần mềm vào hộp: **Installation title**:



Nhấn **Next** để tiếp tục:

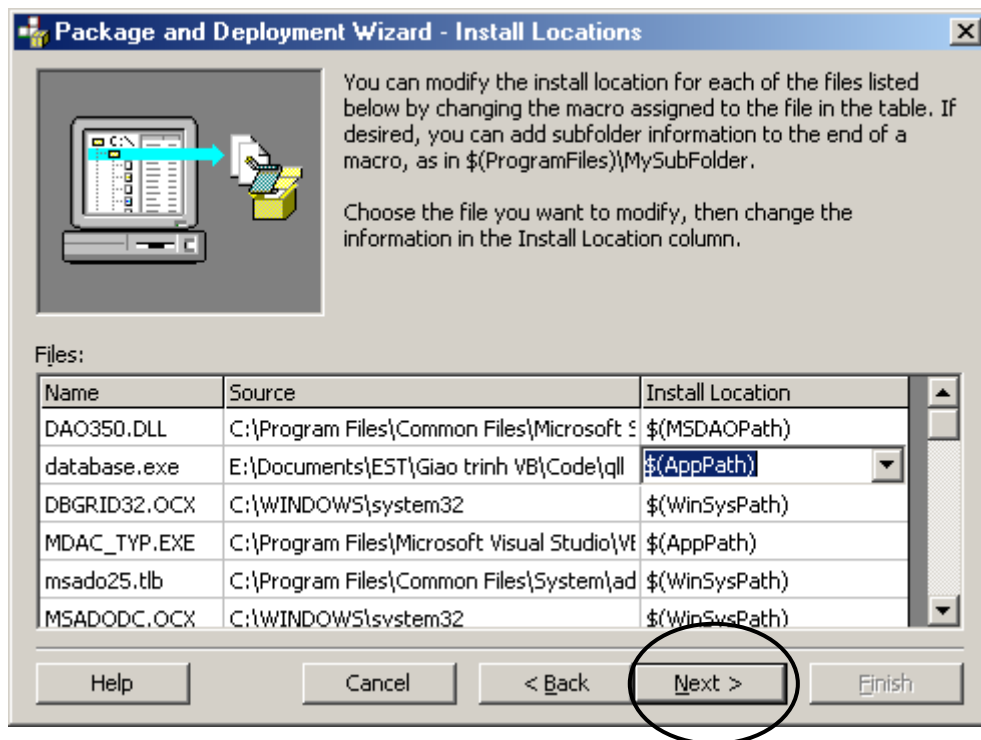
Hộp thoại tiếp theo dùng để quản lý các *Shortcut* của phần mềm trên thanh tác nghiệp *Start* của Windows:



- Nút **New Group** để tạo một nhóm mới (Group) trên mục Programs của thanh Start;
- Nút **New Item** - để tạo một Shortcut mới trên một Group nào đó;
- Nút **Properties** để thiết lập thêm các thuộc tính cho các Group hoặc Items;
- Nút **Remove** để xóa bỏ các Group hoặc Items không cần thiết.

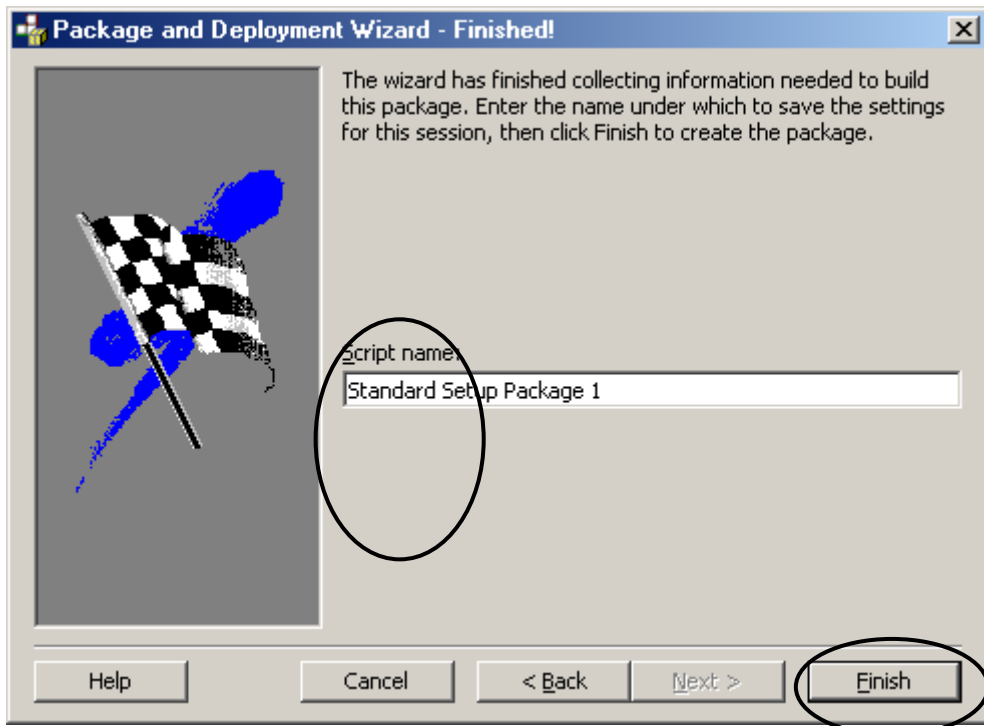
Nhấn **Next** để tiếp tục:

Hộp thoại tiếp theo cho phép khai báo lại vị trí các tệp tin được đóng gói trên ổ đĩa sau khi chúng được cài đặt. Có thể thay đổi vị trí này ở cột **Install Location** trên bảng hiển thị danh sách các tệp tin được đóng gói:



Nhấn **Next** để tiếp tục:

Hộp thoại cuối cùng tiến trình cài đặt xuất hiện. Tại đây có thể nhấn nút **Finish** để kết thúc quá trình đóng gói dự án. Hoặc có thể sử dụng nút **Back** để trở lại các bước đã làm để thiết lập lại.



Bài tập chương 4

1. Với Project Quản lý sách thư viện, hãy tạo MDI form và hệ thống Menu, Toolbar như sau:



2. Sử dụng công cụ Package and Deployment Wizard để đóng gói Project Quản lý sách thư viện thành bộ cài đặt hoàn chỉnh.

TÀI LIỆU THAM KHẢO

1. **Programming Microsoft Visual Basic 6.0**, Francesco Balena, Microsoft Press, 1999;
2. **Advanced Microsoft Visual Basic 6.0**, Second Edition, Wendy Zucker, Microsoft Press, 1998;
3. **MSDN**, Microsoft Corporation, 1997.

DANH SÁCH CÁC TỪ VIẾT TẮT

ADO - ActiveX Data Access

CNTT - Công nghệ thông tin

COM - Common Object Model

CSDL - Cơ sở dữ liệu

DAO - Data Access Objects

GUI - Graphical User Interface

MDI - Multi Document Interface

VB - Visual Basic

THUẬT NGỮ TIẾNG ANH

Connection	Một đối tượng trong ADO dùng để tạo kết nối tới CSDL phục vụ lập trình CSDL.
Control	Điều khiển trong VB - một công cụ lập trình trực quan.
Data	Dữ liệu, dữ kiện, số liệu
Data Environment	Môi trường dữ liệu trong dự án VB. Môi trường có thể bao gồm các kết nối dữ liệu (Connection), các truy vấn dữ liệu (Command) phục vụ lập trình CSDL và in báo cáo.
Data Report	Một công cụ trong VB dùng thiết kế, in báo cáo.
Data source	Nguồn cung cấp dữ liệu, có thể là cung cấp cho một Form, cung cấp dữ liệu cho một Report.
Data type	Kiểu dữ liệu. Dữ liệu trong cơ sở dữ liệu phải được định kiểu: kiểu số, kiểu chữ, kiểu lô gíc, ...- đó là các kiểu dữ liệu
Database	Cơ sở dữ liệu. Nơi chứa toàn bộ dữ liệu cho một mục đích quản lý nào đó, ở đó dữ liệu được thiết kế và lưu trữ theo các cấu trúc tối ưu.
Datasheet	Bảng dữ liệu- nơi dữ liệu hiển thị dưới dạng bảng bao gồm các cột và các dòng dữ liệu.
Design View	Môi trường thiết kế, có thể là thiết kế Form, thiết kế report, query,...
Detete	Chỉ hành động xóa các bản ghi ra khỏi cơ sở dữ liệu
EOF	End Of File – một dấu hiệu cho biết con trỏ bản ghi đã nằm ở vị trí cuối cùng của một recorset.
Field	Trường dữ liệu
Field size	Độ lớn của trường dữ liệu

Form	Đối tượng dùng thiết kế giao diện trong phát triển phần mềm. Kết quả của form khi sử dụng sẽ là các cửa sổ, hộp thoại- nơi mà người dùng có thể sử dụng để thao tác trên phần mềm.
Function	
Group By	Một mệnh đề trong câu lệnh SQL dùng nhóm các bản ghi phục vụ các việc tổng hợp dữ liệu nào đó.
Object	Đối tượng - một thành quan trọng trong lập trình hướng đối tượng, trực quan như VB.
Package & Deployment	Một công cụ dùng để đóng gói dự án, tạo bộ gói đặt trong môi trường phát triển VB.
Project	Một dự án VB – bao gồm nhiều thành phần, nhiều tệp tin và tài nguyên phục vụ xây dựng dự án phần mềm nào đó trên VB.
Query	Một đối tượng dùng truy xuất, xử lý dữ liệu trên các hệ cơ sở dữ liệu
Record	Bản ghi
Recordset	Tập hợp các bản ghi
Relationship	Chỉ quan hệ giữa các bảng trong cơ sở dữ liệu
SQL	Structured Query Language – ngôn ngữ truy vấn dữ liệu có cấu trúc. Một ngôn ngữ xử lý dữ liệu khá phổ dụng trên các hệ cơ sở dữ liệu. Với Access, Query Design chính là nơi để thiết kế tạo ra các câu lệnh SQL thi hành các phép xử lý dữ liệu.
Sub	Từ khóa để khai báo chương trình con dạng thủ tục (Procedure) trong VB.
Wizard	Là trình điều khiển giúp người lập trình đơn giản trong việc xây dựng ứng dụng. Trong VB, chúng ta được tìm hiểu Data Form Wizard.

MỤC LỤC

GIỚI THIỆU	2
BÀI MỞ ĐẦU	4
1. Giới thiệu Visual Basic 6.0.....	4
2. Khởi động	6
3. Lưu trữ.....	8
4. Mở dự án đã có	9
CHƯƠNG 1: LẬP TRÌNH VB CĂN BẢN.....	11
1. Kiểu dữ liệu - biến và hằng	12
1.1 Kiểu dữ liệu.....	12
1.2 Biến	14
<i>a. Biến – khai báo biến.....</i>	<i>14</i>
<i>b. Phạm vi biến</i>	<i>15</i>
1.3 Hằng	16
<i>a. Khai báo hằng.....</i>	<i>16</i>
<i>b. Phạm vi hằng</i>	<i>17</i>
2. Các cấu trúc lệnh VB.....	18
2.1 Cấu trúc IF... END IF	18
2.2 Cấu trúc SELECT CASE .. END SELECT	19
2.3 Cấu trúc FOR ... NEXT.....	22
2.4 Cấu trúc WHILE ... WEND	24
3. Chương trình con.....	26
3.1 Chương trình con dạng hàm.....	26
3.2 Chương trình con dạng thủ tục.....	31
3.3 Sử dụng chương trình con	33
<i>a. Sử dụng thủ tục</i>	<i>33</i>
<i>b. Sử dụng hàm</i>	<i>34</i>
4. Soạn thảo chương trình và xử lý lỗi	35
4.1 Soạn thảo chương trình	35
4.2 Lỗi và xử lý lỗi	38
<i>a. Xử lý lỗi.....</i>	<i>38</i>
<i>b. Bẫy lỗi.....</i>	<i>42</i>
Bài tập chương 1.....	45
CHƯƠNG 2: SỬ DỤNG ĐIỀU KHIỂN.....	47
1. Sơ lược về điều khiển	48
1.1. Tập thuộc tính.....	48
1.2. Tập phương thức	54
1.3. Tập sự kiện	55
2. Một số điều khiển cơ bản	57
2.1 Điều khiển Form.....	57
2.2 Điều khiển Label	59
2.3 Điều khiển Textbox.....	60
2.4 Điều khiển CommandButton.....	63
2.5 Điều khiển Picture.....	65
3. Nhóm điều khiển trình bày giao diện	66
3.1 Điều khiển ComboBox.....	66

3.2 Điều khiển ListBox	68
3.3 Điều khiển CheckBox	69
3.4 Điều khiển OptionButton	71
4. Nhóm điều khiển làm việc thư mục, tệp tin	73
4.1 Điều khiển DriveListbox	74
4.2 Điều khiển DirListBox	74
4.3 Điều khiển FileListBox	75
4.4 Ví dụ tổng hợp.....	75
5. Một số điều khiển chung	78
5.1 Điều khiển MS Common Dialog.....	78
5.2 Điều khiển Treeview	82
Bài tập chương 2.....	88
CHƯƠNG 3: LẬP TRÌNH CƠ SỞ DỮ LIỆU.....	91
1. Kỹ thuật DAO.....	93
1.1 Lớp đối tượng DAO	94
1.2 Đối tượng Database.....	96
1.3 Đối tượng RecordSet.....	97
1.4 Đối tượng QueryDef	102
1.5 Đối tượng TableDef	104
1.6 Đối tượng Relation.....	107
1.7 Sử dụng Data Form Wizard	108
1.8 Bài toán cập nhật dữ liệu.....	110
1.9 Bài toán tìm và lọc dữ liệu	114
2. Kỹ thuật ADO.....	117
2.1 Kiến trúc ADO	117
2.2 Đối tượng Connection.....	118
2.3 Đối tượng Command.....	119
2.4 Đối tượng Recorset	120
3. Data Report.....	122
3.1 Xây dựng nguồn dữ liệu.....	122
3.2 Thiết kế Data Report	126
3.3 Sử dụng Data Report.....	128
Bài tập chương 3.....	130
CHƯƠNG 4: HOÀN THIỆN DỰ ÁN	135
1. Thiết kế MDI Form.....	136
1.1 Một số khái niệm.....	136
1.2 Sử dụng MDI form.....	137
2. Thiết kế Menu.....	138
3. Thiết kế ToolBar.....	140
4. Đóng gói dự án	145
Bài tập chương 4.....	153
TÀI LIỆU THAM KHẢO.....	154
DANH SÁCH CÁC TỪ VIẾT TẮT	155
THUẬT NGỮ TIẾNG ANH	156