

## MỤC LỤC

Mục	Trang
<b>CHƯƠNG I: TỔNG QUAN VỀ VISUAL BASIC 6.0</b>	<b>1</b>
<b>I. Giới thiệu về Visual Basic 6.0</b>	<b>1</b>
<b>II. Cài đặt Visual Basic 6.0</b>	<b>1</b>
<b>III. Làm quen với VB6</b>	<b>1</b>
1. Bắt đầu một dự án mới với VB6	1
2. Tìm hiểu các thành phần của IDE	3
3. Sử dụng thanh công cụ trong IDE của VB	3
4. Quản lý ứng dụng với Project Explorer	5
5. Cửa sổ Properties	5
6. Cửa sổ Form Layout	6
7. Biên dịch đề án thành tập tin thực thi	6
<b>CHƯƠNG II: BIỂU MẪU VÀ MỘT SỐ ĐIỀU KHIỂN THÔNG DỤNG</b>	<b>7</b>
<b>I. Các khái niệm</b>	<b>7</b>
<b>II. Biểu mẫu (Form)</b>	<b>9</b>
1. Khái niệm	9
2. Thuộc tính	9
3. Phương thức	9
4. Sự kiện	9
<b>III. Nhãn (Label)</b>	<b>10</b>
1. Khái niệm	10
2. Thuộc tính	10
3. Phương thức	10
4. Sự kiện	10
<b>IV. Khung (Frame)</b>	<b>10</b>
1. Khái niệm	11
2. Thuộc tính	11
3. Phương thức	11
4. Sự kiện	11
<b>V. Nút lệnh (Command Button)</b>	<b>11</b>
1. Khái niệm	11
2. Thuộc tính	12
3. Phương thức	12
4. Sự kiện	12
<b>VI. Ô nhập liệu (TextBox)</b>	<b>12</b>

<b>1. Khái niệm</b>	<b>12</b>
<b>2. Thuộc tính</b>	<b>13</b>
<b>3. Phương thức</b>	<b>14</b>
<b>4. Sự kiện</b>	<b>14</b>
<b>CHƯƠNG III: LẬP TRÌNH TRONG VISUAL BASIC</b>	<b>15</b>
<b>I. Môi trường lập trình</b>	<b>15</b>
<b>1. Soạn thảo chương trình</b>	<b>15</b>
<b>2. Các chức năng tự động</b>	<b>15</b>
<b>II. Kiểu dữ liệu</b>	<b>16</b>
<b>1. Khái niệm</b>	<b>16</b>
<b>2. Các kiểu dữ liệu cơ sở trong Visual Basic</b>	<b>16</b>
<b>III. Hằng số</b>	<b>17</b>
<b>1. Khái niệm</b>	<b>17</b>
<b>2. Khai báo hằng</b>	<b>17</b>
<b>V. Biểu thức</b>	<b>18</b>
<b>1. Khái niệm</b>	<b>18</b>
<b>2. Các loại phép toán</b>	<b>18</b>
<b>VI. Câu lệnh</b>	<b>18</b>
<b>1. Khái niệm</b>	<b>18</b>
<b>2. Lệnh gán</b>	<b>19</b>
<b>3. Lệnh rẽ nhánh If</b>	<b>20</b>
<b>4. Lệnh lựa chọn Select Case</b>	<b>20</b>
<b>5. Cấu trúc lặp</b>	<b>21</b>
<b>VII. Chương trình con</b>	<b>23</b>
<b>1. Khái niệm</b>	<b>23</b>
<b>2. Thủ tục</b>	<b>23</b>
<b>3. Hàm</b>	<b>25</b>
<b>VIII. Truy xuất dữ liệu trong Visual Basic</b>	<b>26</b>
<b>1. Các khái niệm</b>	<b>26</b>
<b>2. Biến toàn cục</b>	<b>26</b>
<b>3. Biến cục bộ</b>	<b>26</b>
<b>4. Biến Module</b>	<b>26</b>
<b>5. Truyền tham số cho chương trình con</b>	<b>27</b>
<b>IX. Bẫy lỗi trong Visual Basic</b>	<b>28</b>
<b>CHƯƠNG IV: CÁC KIỂU DỮ LIỆU CÓ CẤU TRÚC</b>	<b>29</b>
<b>I. Kiểu chuỗi ký tự (String)</b>	<b>29</b>
<b>1. Khai báo</b>	<b>29</b>
<b>2. Các hàm xử lý chuỗi</b>	<b>29</b>

<b>II. Kiểu ngày tháng (Date)</b>	<b>30</b>
<b>1. Khái niệm</b>	<b>30</b>
<b>2. Ví dụ</b>	<b>30</b>
<b>III. Kiểu số</b>	<b>30</b>
<b>1. Các hàm chuyển đổi chuỗi sang số</b>	<b>30</b>
<b>2. Ví dụ</b>	<b>31</b>
<b>IV. Kiểu Object</b>	<b>31</b>
<b>V. Kiểu Variant</b>	<b>31</b>
<b>VI. Kiểu Mảng</b>	<b>32</b>
<b>2. Khai báo</b>	<b>33</b>
<b>3. Một số thao tác trên mảng</b>	<b>34</b>
<b>VII. Kiểu do người dùng định nghĩa - Kiểu mẫu tin</b>	<b>34</b>
<b>CHƯƠNG V: THIẾT KẾ BIỂU MẪU DÙNG CÁC ĐIỀU KHIỂN</b>	<b>35</b>
<b>I. Phân loại điều khiển</b>	<b>35</b>
<b>II. Sử dụng các điều khiển</b>	<b>35</b>
<b>1. Điều khiển danh sách các lựa chọn (List Box)</b>	<b>35</b>
<b>2. Điều khiển hộp lựa chọn (Combo Box)</b>	<b>39</b>
<b>3. Điều khiển hộp đánh dấu (Check Box)</b>	<b>39</b>
<b>4. Điều khiển nút lựa chọn (Option Button)</b>	<b>40</b>
<b>5. Điều khiển thanh cuộn ngang (HScrollBar)</b>	<b>41</b>
<b>6. Điều khiển thanh cuộn đứng (VScrollBar)</b>	<b>42</b>
<b>7. Điều khiển hộp hình ảnh (Picture Box)</b>	<b>42</b>
<b>8. Điều khiển hình ảnh (Image)</b>	<b>43</b>
<b>9. Điều khiển hình dạng (Shape)</b>	<b>43</b>
<b>10. Điều khiển thời gian (Timer)</b>	<b>44</b>
<b>11. Điều khiển danh sách ổ đĩa (DriveListbox), danh sách thư mục (DirListbox), danh sách tập tin (FileListbox)</b>	<b>45</b>
<b>CHƯƠNG VI: LẬP TRÌNH XỬ LÝ GIAO DIỆN &amp; ĐỒ HỌA</b>	<b>47</b>
<b>I. Menu</b>	<b>47</b>
<b>1. Khái niệm</b>	<b>47</b>
<b>2. Các thuộc tính của Menu</b>	<b>47</b>
<b>3. Các sự kiện</b>	<b>47</b>
<b>4. Cách tạo Menu</b>	<b>47</b>
<b>II. Hộp thoại</b>	<b>49</b>
<b>1. Khái niệm</b>	<b>49</b>
<b>2. Hộp thông điệp</b>	<b>49</b>

<b>3. Hộp nhập</b>	<b>51</b>
<b>4. Các hộp thoại thông dụng</b>	<b>52</b>
<b>III. Xử lý các sự kiện chuột và bàn phím</b>	<b>56</b>
<b>1. Sự kiện chuột</b>	<b>56</b>
<b>2. Sự kiện bàn phím</b>	<b>59</b>
<b>IV. Xử lý đồ họa và giao diện</b>	<b>60</b>
<b>1. Hiển thị hình ảnh</b>	<b>60</b>
<b>2. Xử lý đồ họa</b>	<b>60</b>
<b>CHƯƠNG VII: CÁC KHÁI NIỆM CƠ BẢN VỀ CƠ SỞ DỮ LIỆU</b>	<b>64</b>
<b>I. Cơ sở dữ liệu</b>	<b>64</b>
<b>1. Khái niệm</b>	<b>64</b>
<b>2. Bộ máy (Engine) cơ sở dữ liệu</b>	<b>64</b>
<b>3. Bảng (Table) và trường (Field)</b>	<b>64</b>
<b>4. Tập mẫu tin (Recordset)</b>	<b>64</b>
<b>II. Truy xuất cơ sở dữ liệu trong Visual Basic 6.0</b>	<b>64</b>
<b>1. Sử dụng cửa sổ cơ sở dữ liệu</b>	<b>64</b>
<b>2. Dùng Visual Data Manager để tạo giao diện</b>	<b>67</b>
<b>III. Sử dụng cửa sổ xem dữ liệu (Data View)</b>	<b>68</b>
<b>IV. Sử dụng điều khiển dữ liệu để tạo giao diện người sử dụng</b>	<b>69</b>
<b>1. Kết nối với cơ sở dữ liệu và làm việc với các mẫu tin thông qua điều khiển ADO Data</b>	<b>70</b>
<b>2. Kết nối với cơ sở dữ liệu và làm việc với các mẫu tin thông qua điều khiển Data (DAO Data Control)</b>	<b>73</b>

# **CHƯƠNG I: TỔNG QUAN VỀ VISUAL BASIC 6.0**

## **I. Giới thiệu về Visual Basic 6.0**

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

- Với VB6, chúng ta có thể :
  - + Khai thác thế mạnh của các điều khiển mở rộng.
  - + Làm việc với các điều khiển mới (ngày tháng với điều khiển MonthView và DateTimePicker, các thanh công cụ có thể di chuyển được CoolBar, sử dụng đồ họa với ImageCombo, thanh cuộn FlatScrollBar, ...).
  - + Làm việc với các tính năng ngôn ngữ mới.
  - + Làm việc với DHTML.
  - + Làm việc với cơ sở dữ liệu.
  - + Các bổ sung về lập trình hướng đối tượng.

## **II. Cài đặt Visual Basic 6.0**

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

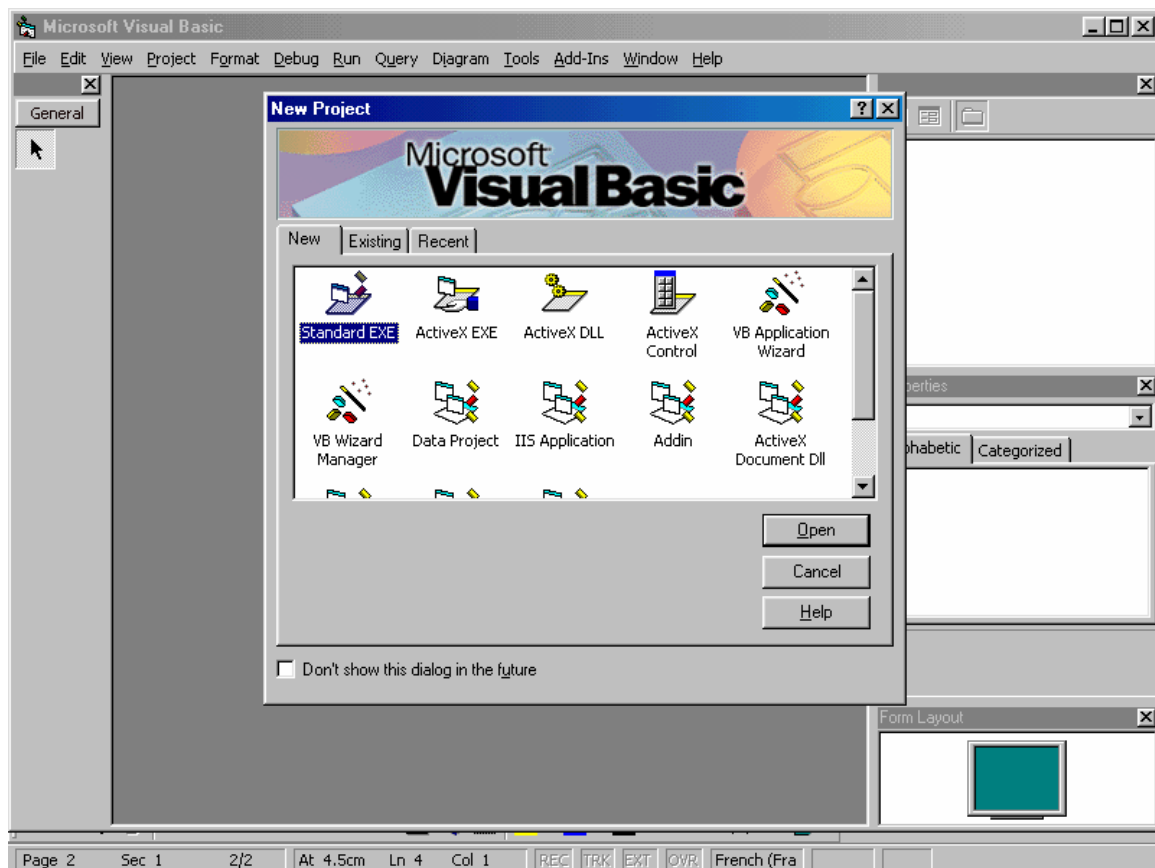
- Để cài đặt VB6, người dùng nên kiểm tra máy tính của mình đảm bảo được cấu hình tối thiểu. Các yêu cầu hệ thống tối thiểu:

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

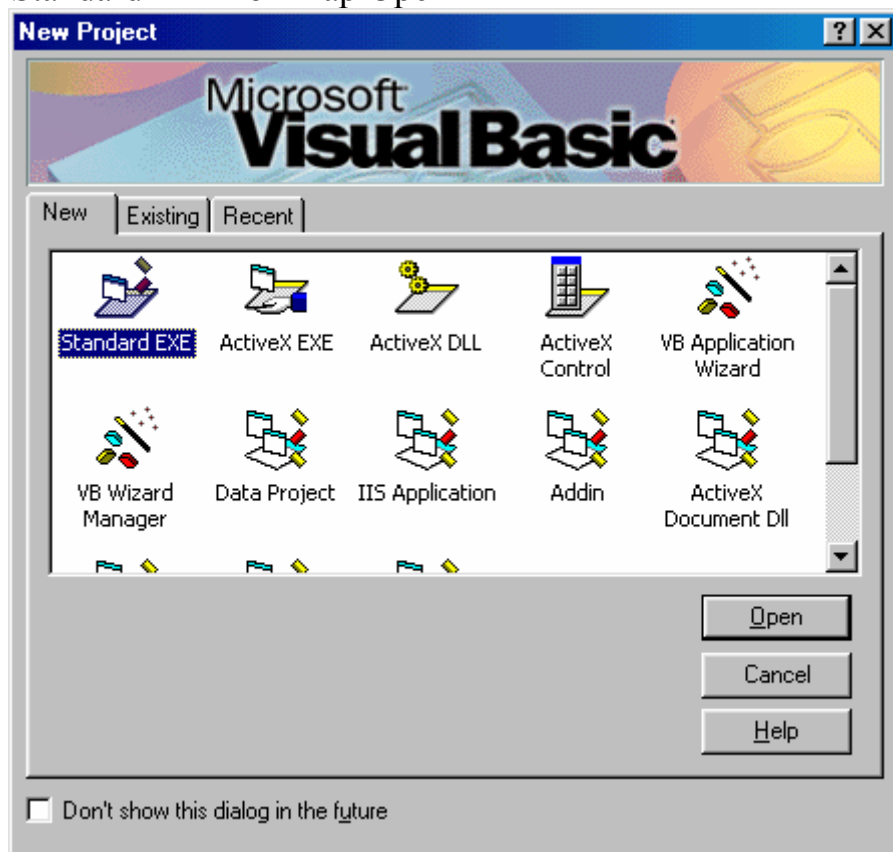
## **III. Làm quen với VB6**

### **1. Bắt đầu một dự án mới với VB6**

- Từ menu Start chọn Programs, Microsoft Visual Basic 6.0. Khi đó người dùng sẽ thấy màn hình đầu như hình dưới đây.



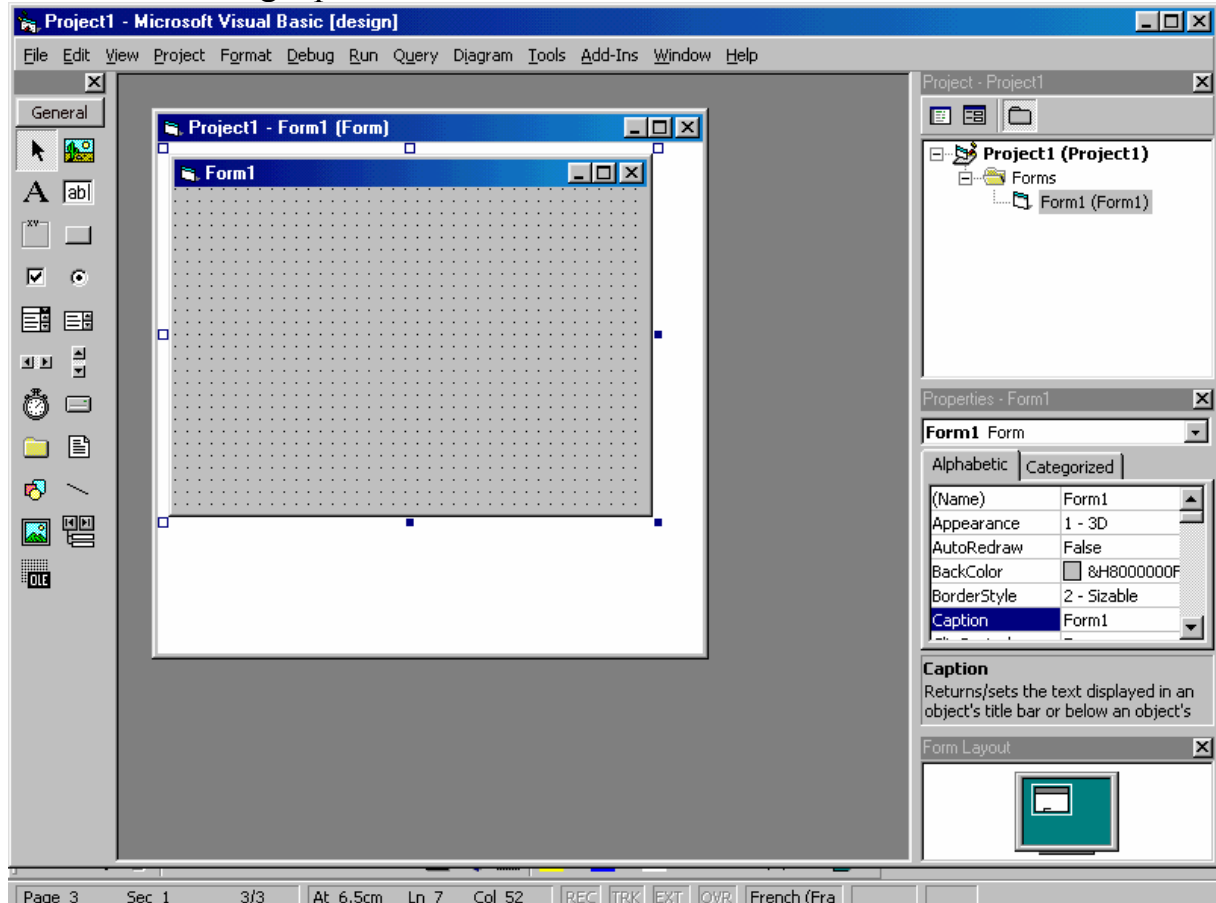
- Ở đây, người dùng có thể chọn tạo mới một dự án thực thi được bằng cách chọn Standard EXE rồi nhấp Open



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

## 2. Tìm hiểu các thành phần của IDE

- IDE là tên tắt của môi trường phát triển tích hợp (Integrated Development Environment), đây là nơi tạo ra các chương trình Visual Basic.
- IDE của Visual Basic là nơi tập trung các menu, thanh công cụ và cửa sổ để tạo ra chương trình. Mỗi một thành phần của IDE có các tính năng ảnh hưởng đến các hoạt động lập trình khác nhau.



- Thanh menu cho phép bạn tác động cũng như quản lý trực tiếp trên toàn bộ ứng dụng.
- Thanh công cụ cho phép truy cập các chức năng của thanh menu thông qua các nút trên thanh công cụ.
- Các biểu mẫu (Form) - Vùng để thiết kế, xây dựng chương trình chính của VB.
  - Hộp công cụ để thêm các điều khiển vào các biểu mẫu của đề án.
- Cửa sổ Project Explorer hiển thị các đề án khác nhau mà người dùng đang làm cũng như các phần của đề án.
- Cửa sổ dùng để duyệt và cài đặt các thuộc tính của điều khiển, biểu mẫu và module trong cửa sổ Properties.
- Cửa sổ dùng để xem xét và bố trí một hoặc nhiều biểu mẫu trên màn hình thông qua cửa sổ Form Layout.

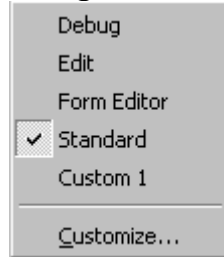
## 3. Sử dụng thanh công cụ trong IDE của VB

### 3.1. Thanh công cụ (Tools bar)

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

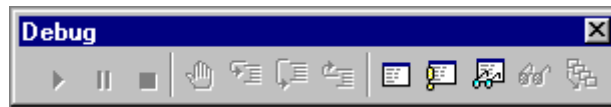


- Hơn nữa, người dùng có thể kéo rê thanh công cụ trên IDE đến vị trí bất kỳ nào đó thuận tiện cho việc sử dụng.
- Người dùng có thể thêm hay xóa thanh công cụ trên IDE:
  - + Chọn Toolbars từ menu View hoặc ấn chuột phải vào điểm bất kỳ nào trên thanh menu, một popup menu bật ra.
  - + Chọn loại thanh công cụ mà ta muốn thêm vào hoặc xóa đi. Nếu có đánh dấu check ở bên trái thì loại công cụ đó đang được chọn.



### 3.2. Thanh gỡ rối (debug)

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



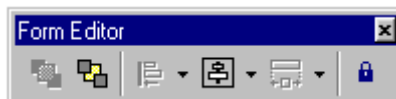
### 3.3. Thanh công cụ soạn thảo (edit)

- Thanh công cụ Edit được dùng để viết chương trình trong cửa sổ Code, thanh công cụ Edit có đầy đủ các tính năng của menu Edit. Ngoài ra người sử dụng có thể sử dụng chức năng viết chương trình tự động như là Quick Info.
- Thanh công cụ Edit của VB6 có tính năng lý thú đó là tự hoàn tất các từ khóa. Tính năng này rất hữu dụng giúp cho người dùng tránh các lỗi mắc phải do gõ sai từ khóa.



### 3.4. Thanh công cụ Form Editor

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



### 3.5. Hộp công cụ (Toolbox)

- Hộp công cụ là nơi chứa các điều khiển được dùng trong quá trình thiết kế biểu mẫu. Các điều khiển được chia làm hai loại: Điều khiển có sẵn trong VB và các điều khiển được chứa trong tập tin với phần mở rộng là .OCX.
- Đối với các điều khiển có sẵn trong VB thì ta không thể gỡ bỏ khỏi hộp công cụ, trong khi đó đối với điều khiển nằm ngoài ta có thêm hoặc xóa bỏ khỏi hộp công cụ.



- Một điều khiển có thể được đưa vào biểu mẫu bằng cách chọn điều khiển đó và đưa vào biểu mẫu (chi tiết ở chương II).

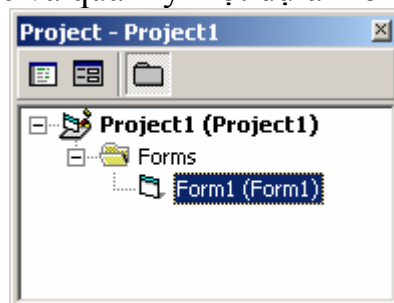


#### 4. Quản lý ứng dụng với Project Explorer

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

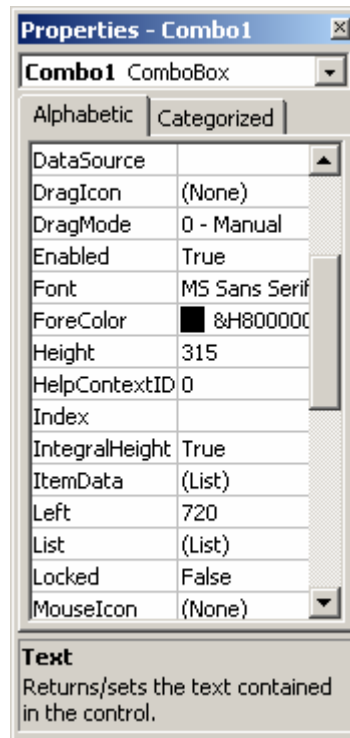
- Project Explorer có cấu trúc cây phân cấp như cây thư mục trong cửa sổ Explorer của hệ điều hành Windows. Các đề án có thể được coi là gốc của cây, các thành phần của đề án như biểu mẫu, module ... là các nút của cây. Khi muốn làm việc với thành phần nào thì ta có thể nhấn đúp lên thành phần đó trên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

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



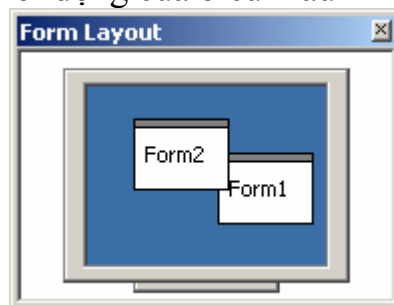
#### 5. Cửa sổ Properties

Mỗi một thành phần, điều khiển đều có nhiều thuộc tính. Mỗi một thuộc tính lại có một hoặc nhiều giá trị. Cửa sổ Properties cho phép người dùng xem, sửa đổi giá trị các thuộc tính của điều khiển nhằm giúp điều khiển hoạt động theo đúng ý đồ của người sử dụng.



## 6. Cửa sổ Form Layout

- Đây chính là cửa sổ trình bày biểu mẫu cho phép xác định vị trí của một hoặc nhiều biểu mẫu trên màn hình khi chương trình ứng dụng được thi hành.
- Ta định vị một biểu mẫu trên màn hình bằng cách dùng chuột di chuyển biểu mẫu trong cửa sổ Form Layout.
- Sử dụng cửa sổ Form Layout không đơn giản như các cửa sổ khác vì nó không được kích hoạt sẵn, người dùng cần phải chạy ứng dụng sau đó mới có thể bố trí được các biểu mẫu thông qua Form Layout.
- Nếu ta không định vị các biểu mẫu thì vị trí của biểu mẫu trên màn hình lúc thiết kế cũng là vị trí khởi động của biểu mẫu khi thực thi.



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

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

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

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

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

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

## CHƯƠNG II: BIỂU MẪU VÀ MỘT SỐ ĐIỀU KHIỂN THÔNG DỤNG

### I. Các khái niệm

**1. Điều khiển:** Các thành phần có sẵn để người lập trình tạo giao diện tương tác với người dùng. Mỗi điều khiển thực chất là một đối tượng, do vậy nó sẽ có một số điểm đặc trưng cho đối tượng, chẳng hạn như các thuộc tính, các phương thức & các sự kiện.

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

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

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

Ví dụ:

- Nhấn phím bất kỳ trên bàn phím.
- Nhấp chuột:

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

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

### 5. Lập trình sự kiện

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

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

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

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

- Ví dụ:

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

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

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

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

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

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

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

- + Tên có thể dài từ 1 - 40 ký tự.

- + Tên phải bắt đầu với ký tự chữ, có thể chữ hoa hay thường.

- + Sau ký tự đầu tiên, tên có thể chứa ký tự, số hay dấu gạch dưới.

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

## II. Biểu mẫu (Form)

### 1. Khái niệm

Chương trình ứng dụng giao tiếp với người dùng thông qua các biểu mẫu (hay còn gọi là cửa sổ hay Windows); Trên biểu mẫu thường gồm các điều khiển (Control) để người dùng thao tác với chương trình.

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

### 2. Thuộc tính

- Name: Thuộc tính này như là một định danh nhằm xác định tên của biểu mẫu là gì. Người dùng sẽ sử dụng thuộc tính này để truy xuất đến các thuộc tính khác cùng với phương thức có thể thao tác được trên biểu mẫu.

- Caption: Chuỗi hiển thị trên thanh tiêu đề của biểu mẫu.

- Icon: hình icon được dùng trong thanh tiêu đề của biểu mẫu, nhất là khi biểu mẫu thu nhỏ lại.

- WindowState: Xác định biểu mẫu sẽ có kích thước bình thường (Normal=0), hay Minimized (=1), Maximized =(2).

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

- BorderStyle: Xác định dạng (kiểu) của biểu mẫu.

### 3. Phương thức

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

### 4. Sự kiện

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

- **Form\_Load**: Sự kiện này xảy ra mỗi lần ta gọi thể hiện một biểu mẫu. Nếu ta chỉ dùng một thể hiện duy nhất của một biểu mẫu trong chương trình thì **Form\_Load** coi như tương đương với **Form\_Initialize**. Ta dùng sự kiện **Form\_Load** để khởi tạo các biến, điều khiển cho các thể hiện của biểu mẫu này.

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

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

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


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

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

### III. Nhãn (Label)

#### 1. Khái niệm

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

Biểu tượng (shortcut) trên hộp công cụ: 

#### 2. Thuộc tính

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

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

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

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

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

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

- **BackStyle, BackColor**: **BackStyle** quy định là nhãn trong suốt hay không. **BackColor** quy định màu nền của nhãn trong trường hợp không trong suốt.

#### 3. Phương thức

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

#### 4. Sự kiện

- **Change**: Xảy ra mỗi khi nhãn thay đổi giá trị.

- **Click**: Mỗi khi nhãn được chuột nhấp lên, sự kiện này xảy ra.

- **DbClick**: Xảy ra khi người sử dụng nhấp đúp chuột lên điều khiển nhãn.

### IV. Khung (Frame)

## 1. Khái niệm

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

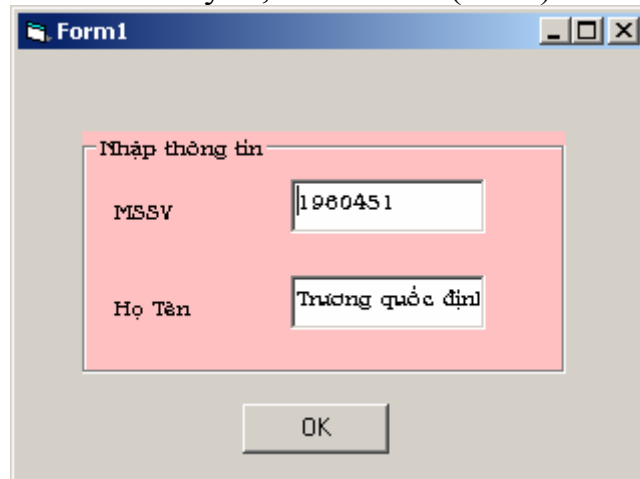
- Biểu tượng (shortcut) trên hộp công cụ:



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

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

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



## 2. Thuộc tính

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

## 3. Phương thức

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

## 4. Sự kiện

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

## V. Nút lệnh (Command Button)

### 1. Khái niệm

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

Biểu tượng (shortcut) trên hộp công cụ:



### 2. Thuộc tính

- Name: sử dụng như một định danh nhằm xác định tên của nút lệnh.
- Caption: Dùng để hiển thị một chuỗi nào đó trên nút lệnh.

- Default: Nếu giá trị của thuộc tính này là True thì ta có thể chọn nút lệnh bằng cách nhấn phím Enter.

- Cancel: Nếu giá trị của thuộc tính này là True thì ta có thể chọn nút lệnh nào đó bằng cách nhấn phím ESC.

- Enabled: Trong một biểu mẫu, có thể có nhiều nút lệnh để thực hiện nhiều công việc khác nhau và tại một thời điểm nào đó ta chỉ được phép thực hiện một số công việc. Nếu giá trị thuộc tính Enabled là False thì nút lệnh đó không có tác dụng. Giá trị mặc định của thuộc tính này là True. Ta có thể thay đổi giá trị của thuộc tính tại thời điểm chạy ứng dụng.

- ToolTipText: cho phép hiển thị một đoạn văn bản chú thích công dụng của nút lệnh khi người sử dụng dùng chuột rê trên nút nhấn.

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

### 3. Phương thức

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

### 4. Sự kiện

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

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

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




## VI. Ô nhập liệu (TextBox)

### 1. Khái niệm

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



dụng. Còn thao tác nhận thông tin do người dùng nhập vào dĩ nhiên là được thực hiện tại thời điểm chạy ứng dụng.

- Biểu tượng (shortcut) trên hộp công cụ 

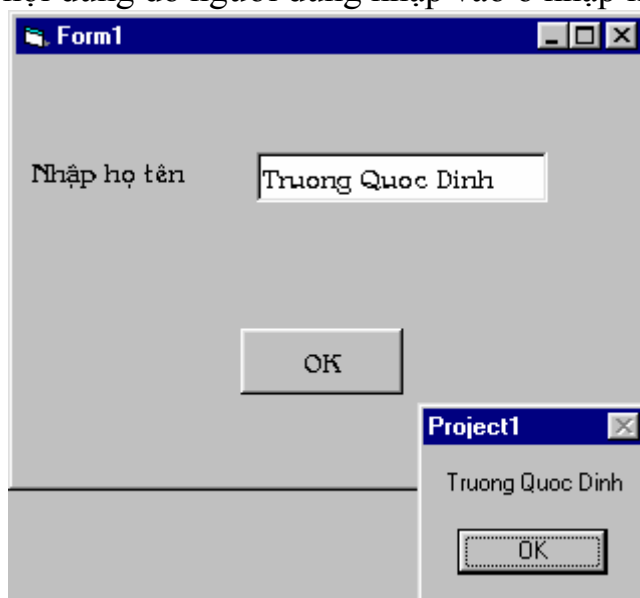
## 2. Thuộc tính

- Name: Đây là tên của ô nhập liệu, được sử dụng như một định danh.
- MaxLength: Thuộc tính quy định số ký tự tối đa có thể nhập vào ô nhập liệu. Nếu số ký tự nhập vào vượt quá số ký tự tối đa thì chỉ có đúng số ký tự tối đa được ghi nhận vào trong thuộc tính Text.
- Text: Dùng để nhập vào thông tin cần hiển thị trong Textbox tại thời điểm thiết kế hoặc nhận giá trị do người dùng nhập vào tại thời điểm chạy ứng dụng.

**Ví dụ:**

*MsgBox Text1.Text*

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



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

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

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

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

### 3. Phương thức

- Move: Di chuyển ô nhập liệu đến tọa độ X, Y: Move X, Y.
- SetFocus: Phương thức này nhằm mục đích thiết lập cho điều khiển ô nhập liệu nhận được Focus, nghĩa là nó sẵn sàng được tương tác bởi người sử dụng.

### 4. Sự kiện

- KeyPress: Xảy ra khi người sử dụng chương trình nhấn một phím. Đối với điều khiển TextBox, ta thường dùng nó để lọc (filter out) các phím không chấp nhận. Sự kiện KeyPress cho ta một mã Ascii (là một số có giá trị từ 0 đến 255 của phím vừa nhấn)

**Trong ví dụ dưới đây**, TextBox Text1 sẽ chỉ nhận biết các phím là số (0 - 9), không nhận biết các phím khác:

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

- KeyDown, KeyUp: Mỗi sự kiện KeyPress lại cho ta một cặp sự kiện KeyDown/KeyUp. Sự kiện KeyDown/KeyUp có 2 tham số là KeyCode và Shift. Sự kiện này cho phép ta nhận biết được các phím đặc biệt trên bàn phím.

**Trong ví dụ dưới đây**, ta hiển thị tên các phím chức năng mà người sử dụng chương trình nhấn vào:

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

## CHƯƠNG III: LẬP TRÌNH TRONG VISUAL BASIC

### I. Môi trường lập trình

#### 1. Soạn thảo chương trình

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

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

- Các phím tắt trong cửa sổ mã lệnh:

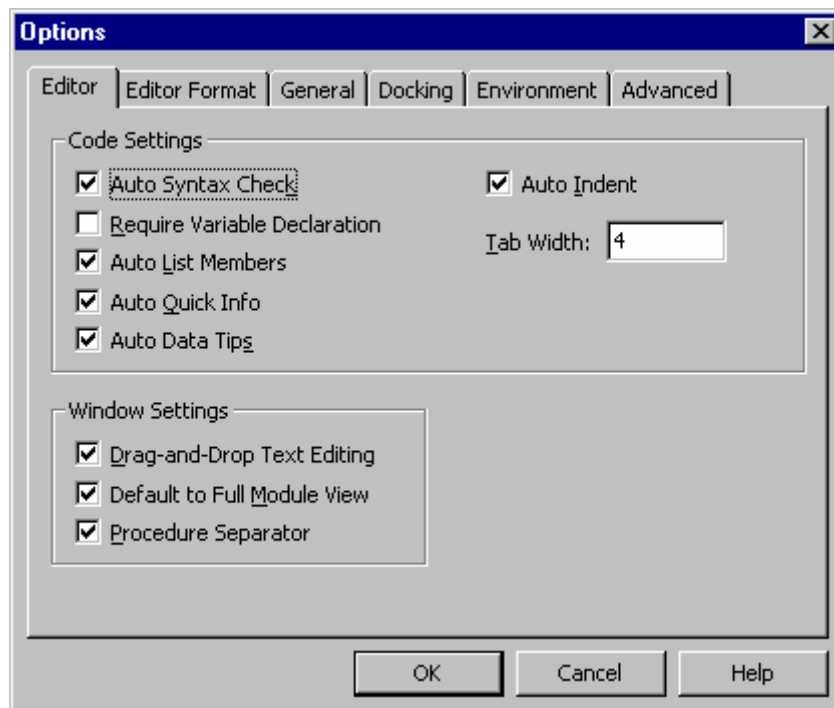
Chức năng	Phím tắt
Xem cửa sổ Code	F7
Xem cửa sổ Object Browser	F2
Tìm kiếm	CTRL+F
Thay thế	CTRL+H
Tìm tiếp	SHIFT+F4
Tìm ngược	SHIFT+F3
Chuyển đến thủ tục kế tiếp	CTRL+DOWN ARROW
Chuyển đến thủ tục trước đó	CTRL+UP ARROW
Xem định nghĩa	SHIFT+F2
Cuộn xuống một màn hình	CTRL+PAGE DOWN
Cuộn lên một màn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

#### 2. Các chức năng tự động

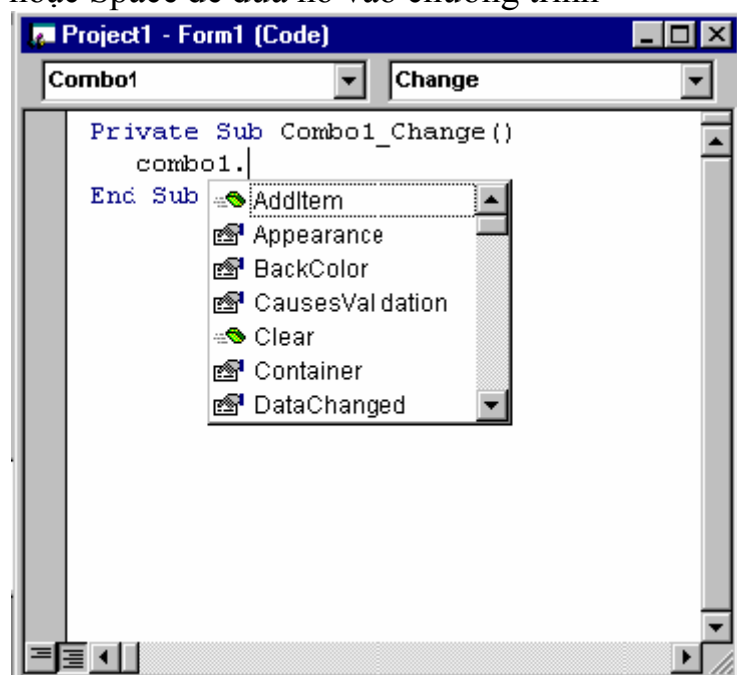
- Tự động kiểm tra cú pháp (Auto Syntax Check)

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

- Yêu cầu khai báo biến (Require Variable Declaration): VB sẽ thông báo lỗi khi một biến được dùng mà không khai báo và sẽ chỉ ra vị trí của biến đó.



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



## II. Kiểu dữ liệu

### 1. Khái niệm

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

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

Kiểu dữ liệu	Mô tả
Boolean	Gồm 2 giá trị: TRUE & FALSE.
Byte	Các giá trị số nguyên từ 0 – 255
Integer	Các giá trị số nguyên từ -32768 đến 32767

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

### III. Hằng số

#### 1. Khái niệm

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

#### 2. Khai báo hằng

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

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

**Ví dụ:**

*Const g = 9.8*

*Const Num As Integer = 4\*5*

### IV. Biến

#### 1. Khái niệm

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

- Biến có 2 đặc điểm:

+ Mỗi biến có một tên.

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

#### 2. Khai báo

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

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

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

- Khai báo ngầm: Đây là hình thức không cần phải khai báo một biến trước khi sử dụng. Cách dùng này có vẻ thuận tiện nhưng sẽ gây một số sai sót,

chẳng hạn khi ta đánh nhầm tên biến, VB sẽ hiểu đó là một biến mới dẫn đến kết quả chương trình sai mà rất khó phát hiện.

**Ví dụ:**

*Dim Num As Long, a As Single*

*Dim Age As Integer*

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

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

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

## V. Biểu thức

### 1. Khái niệm

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

- Toán hạng: Là giá trị dữ liệu (biến, hằng...).

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

### 2. Các loại phép toán

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

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

### 2.2. Các phép toán so sánh

- < : Phép toán so sánh nhỏ hơn
- <= : Phép toán so sánh nhỏ hơn hoặc bằng
- > : Phép toán so sánh lớn hơn
- >= : Phép toán so sánh lớn hơn hoặc bằng
- = : Phép toán so sánh bằng
- <> : Phép toán so sánh khác

## VI. Câu lệnh

### 1. Khái niệm

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

## 2. Lệnh gán

**Cú pháp:**

$\langle \text{Tên biến} \rangle = \langle \text{Biểu thức} \rangle$

**Ví dụ:**

Giả sử ta có khai báo sau:

*Dim TodayTemp As Single, MinAge As Integer*

*Dim Sales As Single, NewSales As Single, FullName As String*

Các lệnh sau gán giá trị cho các biến trên:

*TodayTemp = 30.5*

*MinAge = 18*

*Sales = 200000*

*NewSales = Sales \* 1.2*

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

*FullName = txtName.Text*

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

## 3. Lệnh rẽ nhánh If

- Dạng một dòng lệnh:

*If <điều kiện> Then <dòng lệnh>*

- Dạng nhiều dòng lệnh:

*If <điều kiện> Then*

*Các dòng lệnh*

*End If*

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

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

- Dạng đầy đủ: *If ... Then ... Else*

*If <điều kiện 1> Then*

*[Khối lệnh 1]*

*ElseIf <điều kiện 2> Then*

*[Khối lệnh 2]...*

*[Else*

*[Khối lệnh n]]*

*End If*

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

Ví dụ:

*If (TheColorYouLike = vbRed) Then*

```

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

```

#### 4. Lệnh lựa chọn Select Case

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

```

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

```

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

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

```

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

```

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

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

**Ví dụ:**

```

Select Case Tuoi
Case Is <18

```



```

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

```

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

## 5. Cấu trúc lặp

### 5.1. Lặp không biết trước số lần lặp

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

- Kiểu 1:

```

Do While <điều kiện>
    <khởi lệnh>
Loop

```

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

- Kiểu 2:

```

Do
    <khởi lệnh>
Loop While <điều kiện>

```

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

- Kiểu 3:

```

Do Until <điều kiện>
    <khởi lệnh>
Loop

```

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

- Kiểu 4:

```

Do
    <khởi lệnh>
Loop Until <điều kiện>

```

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

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

```

Dim i As Integer
i = 2
Do While (i <= Sqr(N)) And (N Mod i = 0)
    i = i + 1
Loop

```

```

If (i > Sqr(N)) And (N <> 1) Then
    MsgBox Str(N) & " là số nguyên tố"
Else
    MsgBox Str(N) & " không là số nguyên tố"
End If

```

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

## 5.2. Lặp biết trước số lần lặp

- Cấu trúc: **For ... Next** là cấu trúc biết trước số lần lặp, ta dùng biến đếm tăng dần hoặc giảm dần để xác định số lần lặp.

```

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

```

Biến đếm, điểm đầu, điểm cuối, bước nhảy là những giá trị số (Integer, Single,...). Bước nhảy có thể là âm hoặc dương. Nếu bước nhảy là số âm thì điểm đầu phải lớn hơn điểm cuối, nếu không khối lệnh sẽ không được thi hành. Khi Step không được chỉ ra, VB sẽ dùng bước nhảy mặc định là một.

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

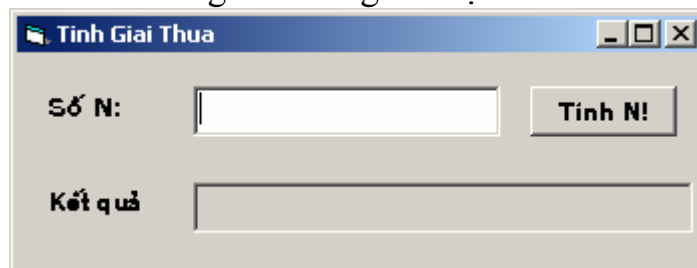
```

Private Sub Form_Click( )
    Dim i As Integer
    For i = 0 To Screen.FontCount
        MsgBox Screen.Fonts(i)
    Next
End Sub

```

- **Ví dụ:** Tính N!

Bước 1: Thiết kế chương trình có giao diện:



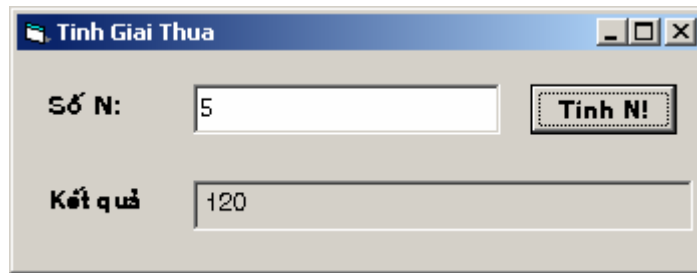
Bước 2: Sự kiện Command1\_Click được xử lý:

```

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

```

Bước 3: Lưu dự án và chạy chương trình ta được kết quả như hình dưới:



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

*For Each <phần tử> In <nhóm>*

*<khối lệnh>*

*Next <phần tử>*

Lưu ý:

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

Phần tử trong mảng chỉ có thể là biến Variant không chứa kiểu tự định nghĩa.

## VII. Chương trình con

### 1. Khái niệm

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

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

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

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

+ Ta muốn nhận lại một kết quả (chỉ một mà thôi) khi gọi chương trình con.

+ Ta cần dùng tên chương trình con (có chứa kết quả) để viết trong các biểu thức.

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

### 2. Thủ tục

#### 2.1. Khái niệm

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

#### 2.2. Khai báo thủ tục

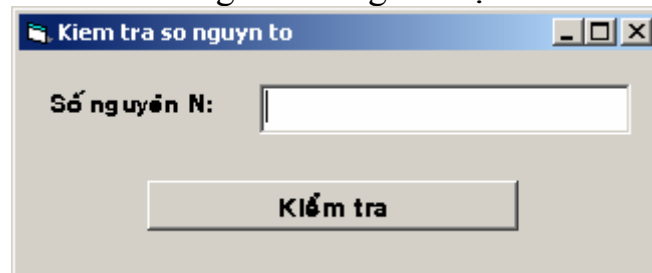
[Private | Public] [Static] Sub <tên thủ tục> [(<tham số>[As <Kiểu tham số>])]

<Các dòng lệnh> hay <Các khai báo>

End Sub

Trong đó:

- <Tên thủ tục>: Đây là một tên được đặt giống quy tắc tên biến, hằng,...
- <tham số>[: <Kiểu tham số>]: có thể có hay không. Nếu có nhiều tham số thì mỗi tham số phân cách nhau dấu phẩy. Nếu không xác định kiểu tham số thì tham số có kiểu Variant.
- Để gọi thủ tục để thực thi, ta có 2 cách:
  - + Cách 1: <Tên thủ tục> [<Các tham số thực tế>]
  - + Cách 2: Call <Tên thủ tục> ([<Các tham số thực tế>])
- **Ví dụ:** Thiết kế chương trình kiểm tra xem số nguyên N có phải là số nguyên tố hay không?
- + Bước 1: Thiết kế chương trình có giao diện



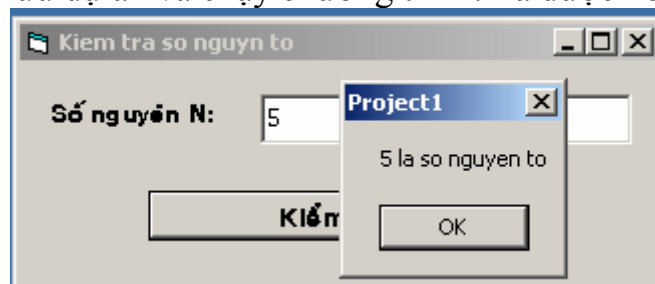
- + Bước 2: Viết thủ tục KtraNgTo trong phần mã lệnh của Form Sub

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

- + Bước 3: Xử lý sự kiện Command1\_Click; trong thủ tục xử lý sự kiện này ta có gọi thủ tục KtraNgTo như sau:

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

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



- **Ghi chú:** Trong ví dụ trên thay vì gọi thủ tục bằng lời gọi:

*KTraNgTo Val(txtNum.Text)*

- Ta có thể sử dụng cách khác:

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

### 3. Hàm

#### 3.1. Khái niệm

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

#### 3.2. Khai báo hàm

*[Private | Public | Static] Function <Tên hàm> [( <tham số> [As <Kiểu tham số>])] \_  
[As <KIỂU DỮ LIỆU>]*

*<Các dòng lệnh> hay <Các khai báo>*

*End Function*

- Trong đó:

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

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

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

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

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

- Ví dụ: Tính N!

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



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

```
Function Giaithua(N As Integer) As Long
```

```
Dim i As Integer, Kq As Long
```

```
Kq = 1
```

```
For i = 1 To n
```

```
Kq = Kq * i
```

```
Next
```

```
Giaithua = Kq
```

```
End Function
```

```
Private Sub Command1_Click()
```

```
Dim n As Integer
```

```
n = Val(txtNum.Text)
```

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

```
End Sub
```

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

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

## VIII. Truy xuất dữ liệu trong Visual Basic

### 1. Các khái niệm

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

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

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

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

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

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

- + Phạm vi biến cục bộ.
- + Phạm vi biến module.
- + Phạm vi biến toàn cục.

### 2. Biến toàn cục

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

- Khai báo:

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

### 3. Biến cục bộ

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

- Khai báo:

*Dim <Tên biến> [As <Kiểu dữ liệu>]*

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

#### 4. Biến Module

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

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

- Ví dụ: *Private Num As Integer*

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

Ví dụ: *Public Num As Integer*

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

#### 5. Truyền tham số cho chương trình con

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

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

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

**Ví dụ:**

*Sub Twice (ByVal Num As Integer)*

*Num = Num \* 2*

*Print Num*

*End Sub*

*Private Sub Form\_Click()*

*Dim A As Integer*

*A = 4*

*Print A*

*Twice A*

*Print A*

*End Sub*

- Truyền tham số bằng địa chỉ: Truyền tham số theo địa chỉ cho phép chương trình con truy cập vào giá trị gốc của biến trong bộ nhớ. Vì thế, giá trị của biến có thể sẽ bị thay đổi bởi đoạn mã lệnh trong chương trình con. Mặc

nhien, trong VB6 các tham số được truyền theo địa chỉ; tuy nhiên ta có thể chỉ định một cách tường minh nhờ vào từ khóa ByRef.

**Ví dụ:**

```
Sub Twice (Num As Integer)
    Num = Num * 2
    Print Num
End Sub
Private Sub Form_Click()
    Dim A As Integer
    A = 4
    Print A
    Twice A
    Print A
End Sub
```

## IX. Bẫy lỗi trong Visual Basic

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

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

- Cú pháp:

Dạng 1:

```
On Error GoTo <Tên nhãn>
<Các câu lệnh có thể gây ra lỗi>
<Tên nhãn>:
<Các câu lệnh xử lý lỗi>
```

Ý nghĩa:

<Tên nhãn>: là một tên được đặt theo quy tắc của một danh biểu.

Nếu một lệnh trong <Các câu lệnh có thể gây ra lỗi> thì khi chương trình thực thi đến câu lệnh đó, chương trình sẽ tự động nhảy đến đoạn chương trình định nghĩa bên dưới <Tên nhãn> để thực thi.

Dạng 2:

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

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



## CHƯƠNG IV: CÁC KIỂU DỮ LIỆU CÓ CẤU TRÚC

### I. Kiểu chuỗi ký tự (String)

#### 1. Khai báo

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

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

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

**Ví dụ:**

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

```
Dim A As String
```

#### 2. Các hàm xử lý chuỗi

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

**Ví dụ:**

```
Dim FirstWord As String, SecondWord As String
```

```
Dim Greeting As String
```

```
FirstWord = "Hello"
```

```
SecondWord = "World"
```

```
Greeting = FirstWord & SecondWord
```

```
' Greeting bây giờ là "HelloWorld"
```

- Hàm Len: Trả về chiều dài một chuỗi được chỉ định.

**Ví dụ:**

```
Greeting = "Hi John!"
```

```
Dim iLen As Integer
```

```
iLen = Len(Greeting) ' iLen bây giờ bằng 8
```

- Hàm Left: Trích chuỗi con từ phần đầu chuỗi gốc Left (String, [length]).

- Hàm Right: Trích chuỗi con từ phần đuôi chuỗi gốc Right (String, [length])

- Hàm Mid: Trích chuỗi con từ giữa chuỗi gốc

- Hàm Mid(String, Start As Long, [length])

**Ví dụ:**

```
Dim Today As String, StrDay As String, StrMonth As String
```

```
Dim StrYear As String, StrMonthYear As String
```

```
Today = "24/05/2001"
```

```
' Lấy ra 2 ký tự từ bên trái của chuỗi Today
```

```
StrDay = Left(Today, 2) ' StrDay bây giờ bằng "24"
```

```
' Lấy ra 4 ký tự từ bên phải của String
```

```
Today StrYear = Right(Today, 4) ' StrYear bây giờ bằng "2001"
```

```
' Lấy ra 2 characters bắt đầu từ ký tự thứ tư của chuỗi
```

```
StrMonth = Mid(Today, 4, 2) ' StrMonth bây giờ bằng "05"
```

```
' Lấy ra phần còn lại bắt đầu từ ký tự 4 của chuỗi Today
```

```
StrMonthYear = Mid(Today, 4) ' StrMonthYear bằng "05/2001"
```

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

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

+ Trong đó:

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

String1: Biểu thức chuỗi để so sánh.

String2: Chuỗi cần tìm.

Compare: Xác định kiểu so sánh chuỗi. Giá trị: vbTextCompare, vbBinaryCompare.

**Ví dụ:**

*Dim myString As String, Position As Integer*

*myString = "The \*rain in Spain mainly..."*

*Position = Instr(myString, "\*") ' Position sẽ là 5*

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

- Hàm Replace: Tìm và thay thế chuỗi.

+ Cú pháp:

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

+ Trong đó:

Expression: Biểu thức chuỗi chứa chuỗi cần thay thế.

find: Chuỗi cần tìm.

replace: Chuỗi thay thế chuỗi tìm được.

start: Tương tự như hàm InStr.

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

compare: Tương tự như hàm InStr.

- Hàm LTrim(string): Cắt tất cả các khoảng trắng bên trái của chuỗi

- Hàm RTrim(string): Cắt tất cả các khoảng trắng bên phải của chuỗi

- Hàm UCase(string): Đổi chuỗi sang chuỗi gồm các ký tự là chữ hoa.

- Hàm Asc(string): Cho mã Ascii của một ký tự.

- Hàm Chr(ascii): Trả về ký tự ứng với mã Ascii được chỉ định.

- Hàm InstrRev: Tương tự như InStr nhưng việc tìm kiếm được tiến hành từ phải sang.

- Hàm Val(string): Hàm đổi chuỗi sang số.

- Hàm Str(number): Hàm đổi số sang chuỗi.

## **II. Kiểu ngày tháng (Date)**

### **1. Khái niệm**

- Là kiểu mà các biến của nó chứa giá trị ngày tháng.

- Để cho VB biết dữ liệu là kiểu Date ta cần đặt giữa hai dấu # (hoặc cặp "").

### **2. Ví dụ**

*Dim D As Date*

*D = #01/02/98# ' Hay "01/02/98"*

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

## **III. Kiểu số**

### **1. Các hàm chuyển đổi chuỗi sang số**

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

## 2. Ví dụ

### - Ví dụ 1

*Dollars = "500"*

*ExchangeRatePerDollar = "7000"*

*tempValue = Val(Dollars) \* Val(ExchangeRatePerDollar)*

*VNDong = CStr(tempValue)*

*MsgBox "Amount in VN Dong is " & VNDong*

### - Ví dụ 2:

*Dollars = "500.0" ExchangeRatePerDollar = "7000.0"*

*'Dùng hàm CSng để đổi chuỗi ra Single*

*tempValue = CSng(Dollars) \* CSng(ExchangeRatePerDollar)*

*'Dùng hàm Format để có các dấu phẩy ở ngàn và triệu*

*' và phải có 2 chữ số sau dấu chấm thập phân.*

*VNDong = Format (tempValue, "#,###,###.00")*

*MsgBox "Amount in VN Dong is " & VNDong*

## IV. Kiểu Object

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

*Dim ObjDb As Object*

*Set ObjDb = OpenDatabase("d:\tqding\thu.mdb")*

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

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

## V. Kiểu Variant

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

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

- Hàm VarType dùng để kiểm tra kiểu dữ liệu.

Hằng	Giá trị	Diễn giải
vbEmpty	0	Không chứa gì cả
vbNull	1	Dữ liệu không hợp lệ
vbInteger	2	Dữ liệu kiểu Integer chuẩn
vbLong	3	Dữ liệu kiểu Long Integer
vbSingle	4	Dữ liệu kiểu dấu chấm động Single
vbDouble	5	Dữ liệu kiểu dấu chấm động Double
vbCurrency	6	Kiểu Currency
vbDate	7	Kiểu Date
vbString	8	Kiểu String
vbObject	9	Kiểu Object
vbError	10	Có một đối tượng lỗi
vbBoolean	11	Kiểu giá trị Boolean chuẩn

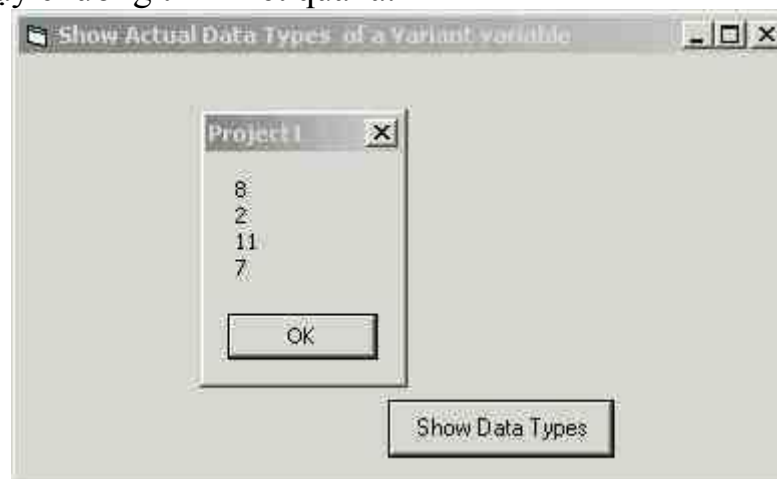
vbVariant	12	Kiểu Variant
vbDataObject	13	Kiểu DAO chuẩn (data access object)
vbDecimal	14	Giá trị thuộc hệ thập phân
vbByte	17	Kiểu Byte
vbUserDefinedType	36	Kiểu do người dùng định nghĩa
vbArray	<b>8192</b>	Kiểu mảng

- Một số chú ý khi dùng biến kiểu Variant:
  - + Nếu muốn thi hành các hàm toán học, Variant phải chứa giá trị kiểu số.
  - + Nếu muốn nối chuỗi, dùng toán tử & thay vì toán tử +.
- Giá trị Empty:
  - + Đây là giá trị đặc biệt xuất hiện khi một biến chưa được gán trị. Ta dùng hàm IsEmpty để kiểm tra giá trị Empty.
  - + Giá trị Empty: Biến mất khi có một giá trị bất kỳ được gán cho biến Variant, để trở về giá trị Empty, ta gán từ khoá Empty cho biến Variant.
  - + Giá trị Null: Biến Variant chứa giá trị Null trong trường hợp những ứng dụng cơ sở dữ liệu thể hiện không có dữ liệu hoặc dữ liệu không xác định.
  - + Giá trị Error: Trong một biến kiểu Variant, Error là một giá trị đặc biệt cho biết đã có một lỗi đã xảy ra bên trong thủ tục.

**- Ví dụ:**

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

Khi chạy chương trình kết quả là:



## VI. Kiểu Mảng

## 1. Khái niệm

- Mảng là tập hợp các phần tử có cùng một kiểu.
- Dùng mảng sẽ làm cho chương trình đơn giản và gọn hơn vì ta có thể sử dụng vòng lặp. Mảng sẽ có biên trên và biên dưới, trong đó các thành phần của mảng là liên tiếp trong khoảng giữa hai biên này.
- Có hai loại biến mảng: Mảng có chiều dài cố định và mảng có chiều dài thay đổi lúc thi hành.

## 2. Khai báo

### 2.1. Mảng có chiều dài cố định

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

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

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

- Ví dụ:

*' Khai báo một biến mảng 15 phần tử kiểu Integer*

*Dim Counters(14) As Integer*

*' Khai báo một biến mảng 21 phần tử kiểu Double*

*Public Sums(20) As Double*

*' Khai báo một biến mảng 10 phần tử kiểu chuỗi ký tự*

*Dim List (1 To 10) As String \* 12*

- Hàm UBound trả về biên trên của một mảng.
- Hàm LBound trả về biên dưới của một mảng.

### 2.2. Mảng động

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

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

- Ví dụ:

*Dim DynArray() As Integer*

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

+ Cú pháp: ReDim <Tên mảng>(N) ' Trong đó N là một biểu thức kiểu Integer.

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

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

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

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

### 3. Một số thao tác trên mảng

- Truy xuất từng phần tử trong mảng: <Tên mảng>(<Vị trí>)
- Sao chép mảng: Đối với VB6, ta có thể gán một mảng cho một mảng khác, hoặc kết quả trả về của một hàm có thể là một mảng.

Ví dụ:

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

```
    New = old
```

```
End Sub
```

- Tuy nhiên, cách này cũng chỉ áp dụng được cho mảng khai báo động mà thôi. Khi gán biến, có một số quy luật mà ta cần lưu ý: Đó là quy luật về kiểu dữ liệu và quy luật về kích thước và số chiều của mảng. Lỗi khi gán mảng có thể xảy ra lúc biên dịch hoặc khi thi hành. Ta có thể thêm bẫy lỗi để đảm bảo rằng hai mảng là tương thích trước khi gán.

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

```
Public Function ArrayFunction (b As Byte) As Byte()
```

```
    Dim x(2) As Byte
```

```
    x(0) = b
```

```
    x(1) = b + 2
```

```
    x(2) = b + b
```

```
    ArrayFunction = x
```

```
End Function
```

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

## VII. Kiểu do người dùng định nghĩa - Kiểu mẫu tin

- Cú pháp:

```
Type <tên kiểu>
```

```
    <Tên trường 1> : <Kiểu trường 1>
```

```
    <Tên trường 2> : <Kiểu trường 2>
```

```
    :
```

```
    <Tên trường n> : <Kiểu trường n>
```

```
End Type
```

**Ví dụ:**

```
Type Profile
```

```
    Fullname As String
```

```
    Salary As Single
```

```
    Age As Integer
```

```
End Type
```

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

Cách truy xuất từng trường của kiểu mẫu tin: <Tên biến mẫu tin>.<Tên trường>

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

*Dim e As Profile*

*Ta có thể gán:*

*e.Fullname = "Nguyen Van An"*

*e.Salary = 300000.00*

*e.Age = 26*

- Câu lệnh *With*:

+ Công dụng: Được sử dụng để viết gọn hơn khi thao tác với dữ liệu kiểu mẫu tin.

+ Cú pháp:

*With <Tên biến mẫu tin>*

*[ Truy xuất đến từng trường của mẫu tin theo dạng:*

*.<Tên trường>*

*]*

*End With*

+ Ví dụ: *Dim e As Profile*

*Ta có thể gán:*

*With e*

*.Fullname = "Nguyen Van An"*

*.Salary = 300000.00*

*.Age = 26*

*End With*

# CHƯƠNG V: THIẾT KẾ BIỂU MẪU DÙNG CÁC ĐIỀU KHIỂN

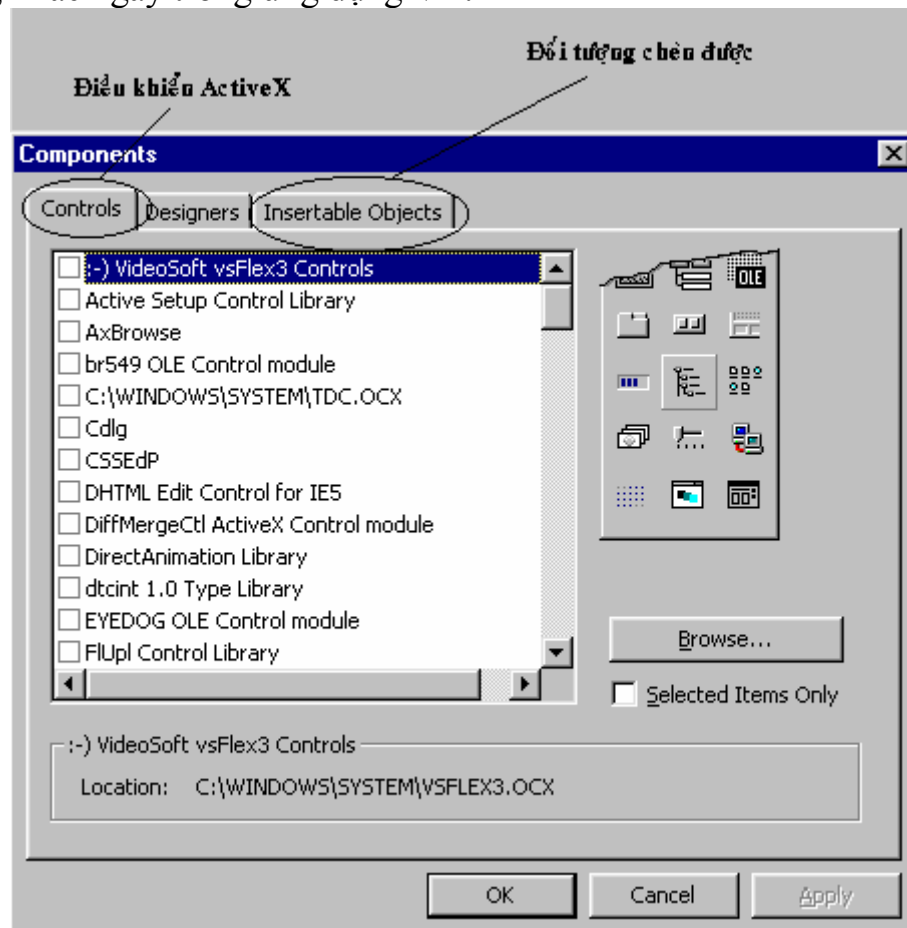
## I. Phân loại điều khiển

Có 3 nhóm điều khiển trong Visual Basic:

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

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

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




## II. Sử dụng các điều khiển

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

#### 1.1. Khái niệm

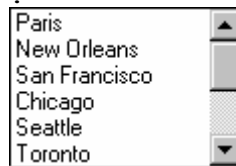
- Điều khiển này hiển thị một danh sách các đề mục mà ở đó người dùng có thể chọn lựa một hoặc nhiều đề mục.

- Biểu tượng (Shortcut) trên hộp công cụ 

- List Box giới thiệu với người dùng một danh sách các lựa chọn. Một cách mặc định, các lựa chọn hiển thị theo chiều dọc trên một cột và bạn có thể thiết lập là hiển thị theo nhiều cột. Nếu số lượng các lựa chọn nhiều và không



thể hiện thì hết trong danh sách thì một thanh trượt sẽ tự động xuất hiện trên điều khiển. Dưới đây là một ví dụ về danh sách các lựa chọn đơn cột.



## 1.2. Thuộc tính

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

## 1.3. Phương thức

- AddItem: Thêm một phần tử vào List Box.

Cú pháp:

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

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

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

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

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

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

**Ví dụ:** List1.AddItem "Japan", 0

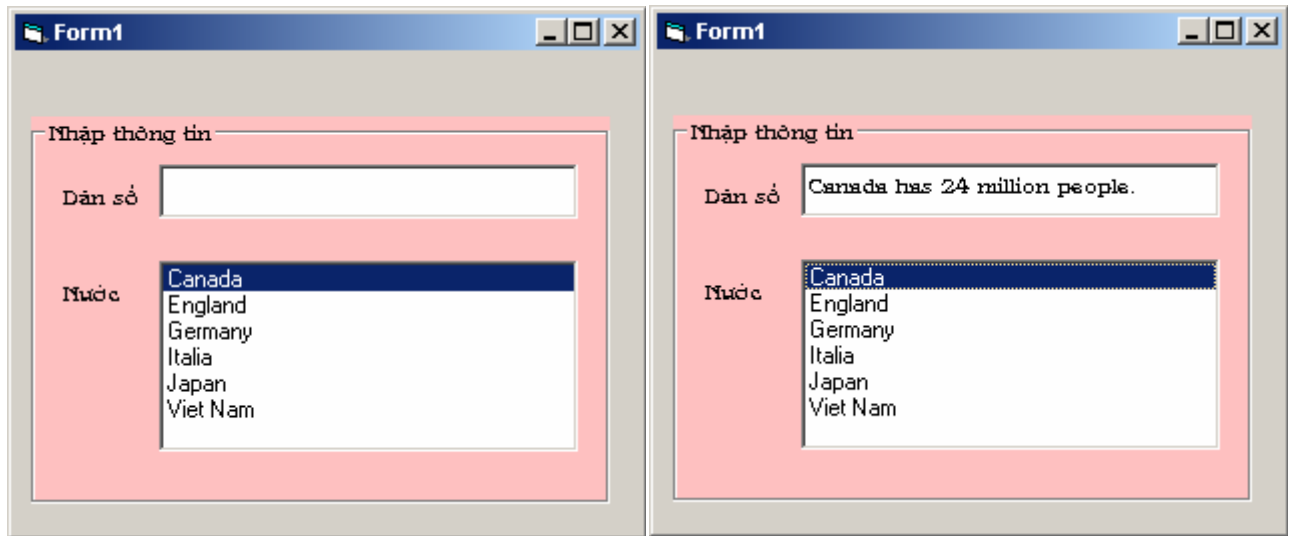
- Thêm mới đề mục tại thời điểm thiết kế: Sử dụng thuộc tính List của điều khiển List Box, ta có thể thêm mới các đề mục và dùng tổ hợp phím CTRL+ENTER để bắt đầu thêm vào đề mục mới trên dòng khác. Khi đã thêm xong danh sách các đề mục, ta có thể sắp xếp lại các đề mục bằng cách sử dụng thuộc tính Sorted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

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

**Cú pháp:** <Name>.RemoveItem Index

- + Tham số Name và Index giống như ở trường hợp thêm vào một đề mục.
- Clear: Xóa tất cả các mục trong List Box. Cú pháp: <Name>.Clear
- Text: Nhận giá trị từ List Box khi một đề mục được chọn. Chẳng hạn đoạn mã sau đây sẽ cho biết dân số của Canada khi người dùng chọn Canada từ List Box.

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



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

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

<Name>.List(Index)

**Ví dụ:** Text1.Text = List1.List(2)

#### 1.4. Sự kiện

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

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

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

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

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

```
Private Sub List1_DblClick ()
```

*Command1.Value = True*

*End Sub*

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

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

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



- Biểu tượng short cut trên hộp công cụ:

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

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



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

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

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

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

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

### 3.1. Khái niệm

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

- Biểu tượng shortcut trên hộp công cụ



### 3.2. Thuộc tính

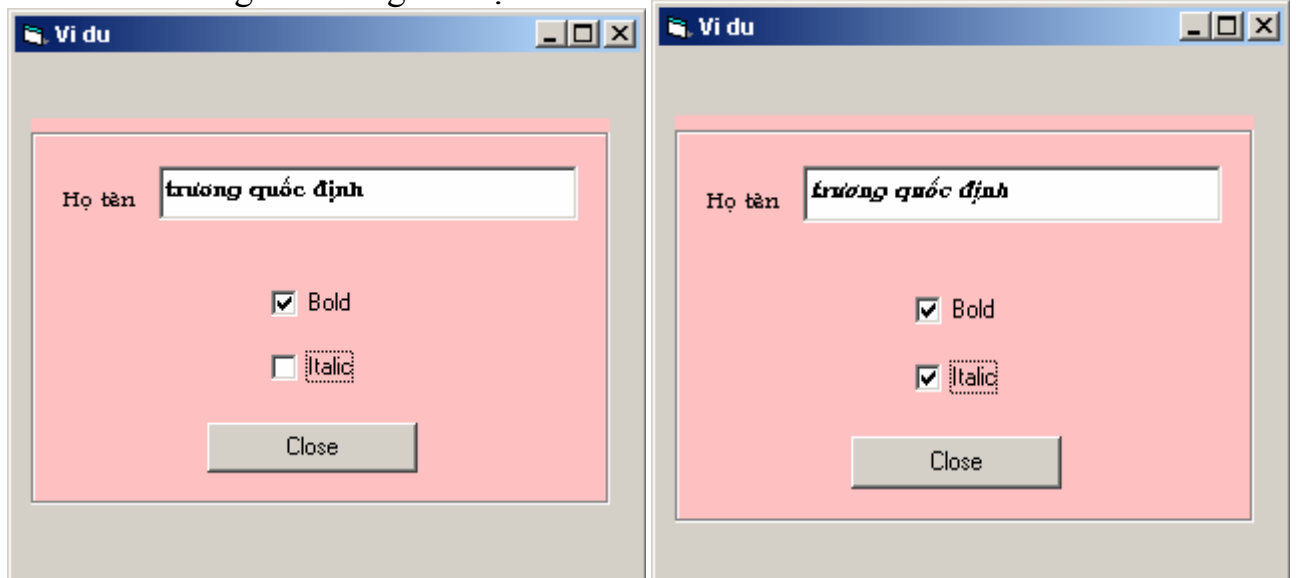
- Name: thuộc tính tên.
- Value: Giá trị hiện thời trên Check Box. Có thể nhận các giá trị: vbChecked, vbUnchecked, vbGrayed.

### 3.3. Sự kiện

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

### 3.4. Ví dụ

Thiết kế chương trình có giao diện:




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

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

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

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

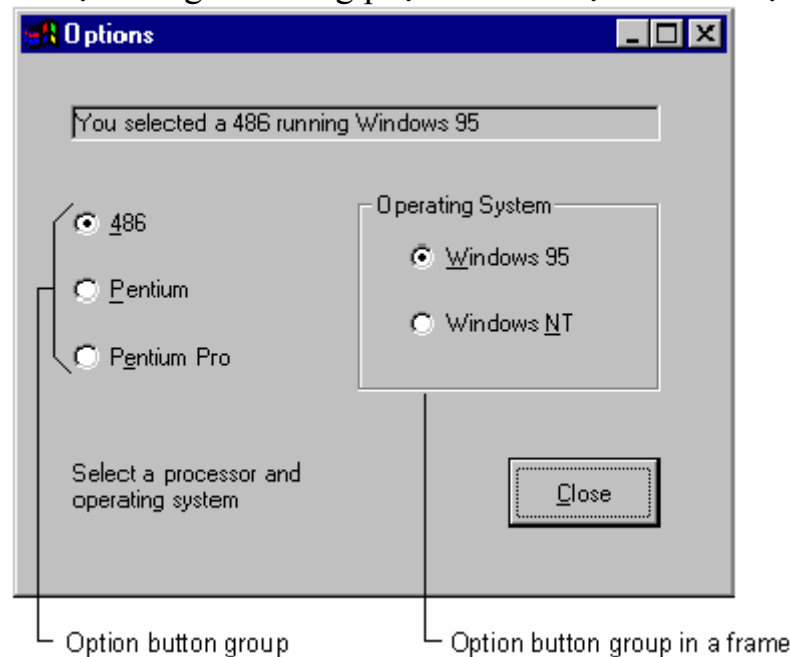
### 4.1. Khái niệm

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

Biểu tượng Shortcut trên hộp công cụ

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

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



## 4.2. Thuộc tính

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

## 4.3. Sự kiện

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

## 5. Điều khiển thanh cuộn ngang (HScrollBar)

### 5.1. Khái niệm

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

- Biểu tượng Shortcut trên hộp công cụ

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

### 5.2. Thuộc tính

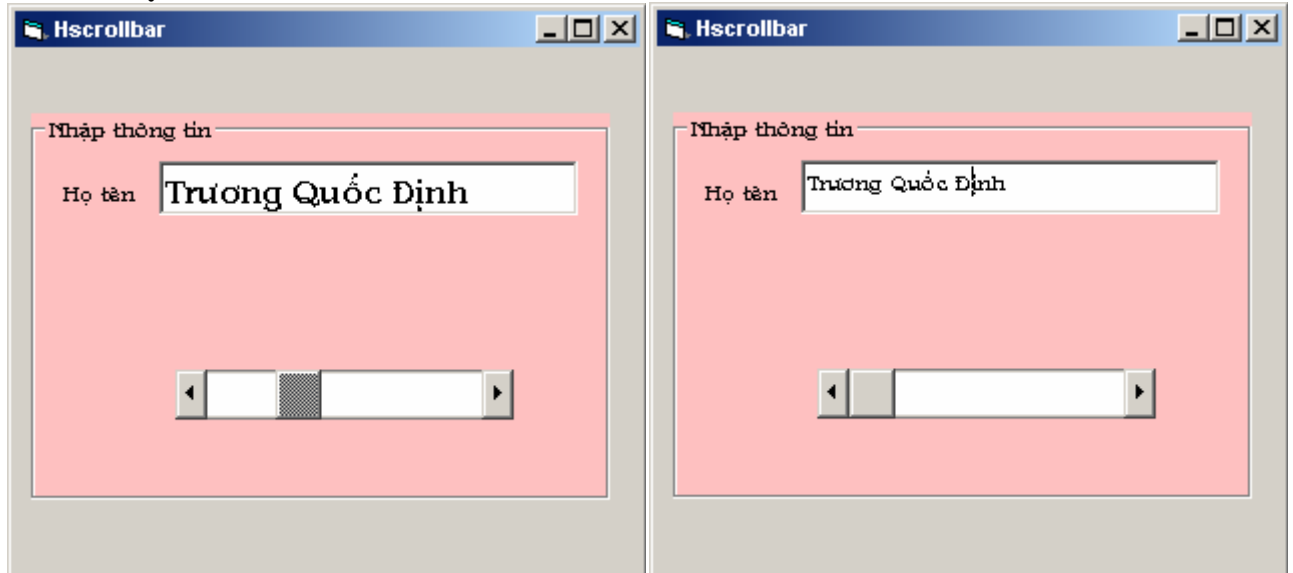
- Name: Tên của thanh cuộn.
- Min: Là giá trị nhỏ nhất trên thanh cuộn.
- Max: Là giá trị lớn nhất của thanh cuộn.
- Large change: Thuộc tính này dùng để xác định khoảng thay đổi khi người dùng ấn chuột lên Hscrollbar.
- Small change: Thuộc tính này dùng để xác định khoảng thay đổi khi người dùng ấn lên mũi tên phía cuối thanh cuộn.

- Value: Thuộc tính này trả về giá trị tại một thời điểm của thanh cuộn nằm trong khoảng giá trị [Min, Max] mà người dùng đã xác định.

### 5.3. Sự kiện

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


### 5.4. Ví dụ



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

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


## 6. Điều khiển thanh cuộn đứng (VScrollBar)

- Biểu tượng shortcut trên hộp công cụ 
- Các thuộc tính và công dụng của VScrollBar cũng tương tự như HScrollBar.

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

### 7.1. Khái niệm

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

- Biểu tượng Shortcut trên hộp công cụ 

### 7.2. Thuộc tính

- Name: tên của điều khiển Picture Box.
- Picture: Đây là thuộc tính cho phép xác định hình ảnh nào sẽ được hiển thị bên trong Picture box. Bao gồm tên tập tin hình ảnh và cả đường dẫn nếu có.
- Để hiển thị hoặc thay thế một hình ảnh tại thời điểm chạy chương trình thì người dùng có thể dùng phương thức LoadPicture để đặt lại giá trị của thuộc tính Picture với cú pháp như trong ví dụ dưới đây:

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

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

nên cẩn thận khi sử dụng thuộc tính này vì khi điều khiển Picture Box thay đổi kích thước, nó không quan tâm đến vị trí của các điều khiển khác.

### 7.3. Sự kiện

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

### 7.4. Lưu ý

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

## 8. Điều khiển hình ảnh (Image)

### 8.1. Khái niệm

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

- Biểu tượng Shortcut trên hộp công cụ



### 8.2. Thuộc tính

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

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

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

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

### 8.3. Sự kiện

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

## 9. Điều khiển hình dạng (Shape)

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

- Biểu tượng Shortcut trên hộp công cụ

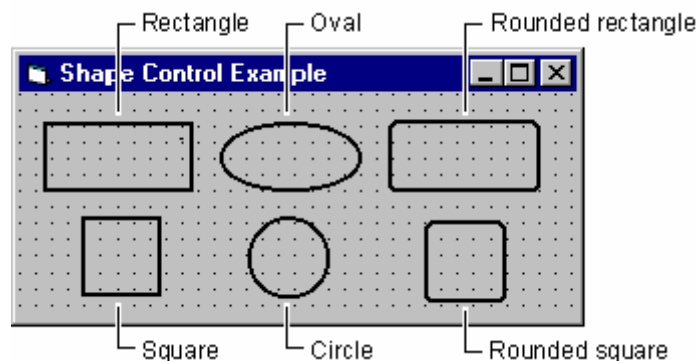


- Thuộc tính Shape cho phép người dùng chọn 1 trong 6 dạng như đã nêu ở trên. Sau đây là bảng giá trị của thuộc tính này:

Giá trị của thuộc tính này	Hình dạng	Giá trị	Hằng
Rectangle		0	vbShapeRectangle
Square		1	vbShapeSquare
Oval		2	vbShapeOval
Circle		3	vbShapeCircle



Rounded Rectangle	4	vbShapeRoundedRectangle
Rounded Square	5	vbShapeRoundedSquare




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

## 10. Điều khiển thời gian (Timer)

### 10.1. Khái niệm

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

- Biểu tượng Shortcut trên hộp công cụ 

- Việc đưa một điều khiển Timer và trong một biểu mẫu cũng tương tự như những điều khiển khác. Ở đây, ta chỉ có thể quan sát được vị trí của điều khiển Timer tại giai đoạn thiết kế, khi chạy ứng dụng điều khiển Timer coi như không có thể hiện trên biểu mẫu.

### 10.2. Thuộc tính

- `Name`: Tên của điều khiển Timer.

- `Interval`: Đây là thuộc tính chỉ rõ số ms giữa hai sự kiện kế tiếp nhau. Trừ khi nó bị vô hiệu hóa, mỗi điều khiển Timer sẽ luôn nhận được một sự kiện sau một khoảng thời gian đều nhau. Thuộc tính `Interval` nhận giá trị trong khoảng 0...64.767 ms có nghĩa là khoảng thời gian dài nhất giữa hai sự kiện chỉ có thể là khoảng một phút (64.8 giây).

- `Enabled`: nếu giá trị là `True` nghĩa là điều khiển Timer được kích hoạt và ngược lại.

### 10.3. Sự kiện

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

### 10.4. Sử dụng điều khiển Timer

- Khởi tạo một điều khiển Timer: Nếu lập trình viên muốn điều khiển Timer hoạt động ngay tại thời điểm biểu mẫu chứa nó được nạp thì đặt thuộc tính `Enable` là `TRUE` hoặc có thể dùng một sự kiện nào đó từ bên ngoài để kích hoạt điều khiển Timer.

- Lập trình đáp ứng sự kiện trả về từ điều khiển Timer: Ta sẽ đưa mã lệnh của công việc cần thực hiện vào trong sự kiện Timer của điều khiển Timer. Sau đây là ví dụ khởi tạo một đồng hồ số nhờ vào điều khiển Timer.



```

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

```

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

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

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

### **11.1. Khái niệm**

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

Biểu tượng shortcut (DriveListbox) trên hộp công cụ 

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

Biểu tượng Shortcut (DirListbox) trên hộp công cụ 

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

Biểu tượng shortcut (FileListbox) trên hộp công cụ 

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

### **11.2. Ví dụ**

Các thuộc tính được thiết lập tại thời điểm thiết kế

```

FileListbox: Partten = "*.doc"
DriveListbox, DirListbox
FileListbox: FontName = "Vncourier New".

```

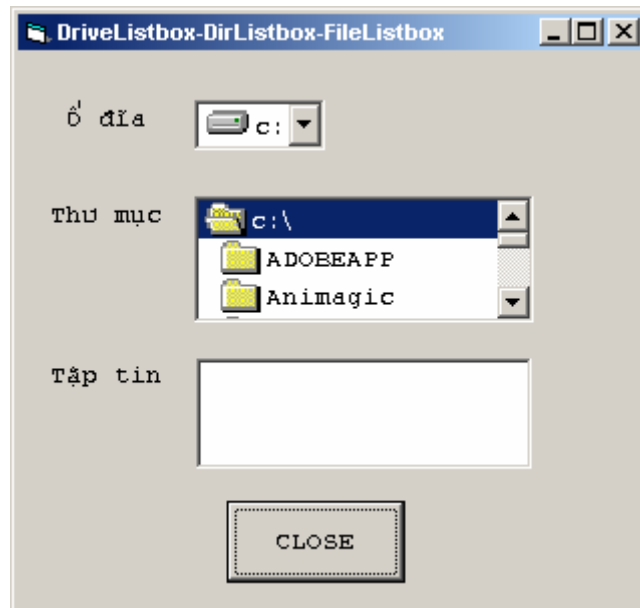
Tức là điều khiển FileListbox chỉ hiển thị các tập tin của Microsoft Word.

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

```

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

```



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

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

```
Private Sub Dir1_Change()  
    File.Path = Dir1.Path  
End Sub  
Private Sub Drive1_Change()  
    Dir1.Path = Drive1.Drive  
End Sub
```

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

# CHƯƠNG VI: LẬP TRÌNH XỬ LÝ GIAO DIỆN & ĐỒ HỌA

## I. Menu

### 1. Khái niệm

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

- Có 2 loại menu:

+ Menu thả xuống (Drop-Down Menu): là dạng menu thông dụng nhất.

+ Menu bật ra (Pop-Up Menu): thường hiển thị khi ta ấn nút phải chuột.

### 2. Các thuộc tính của Menu

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

- Caption: Là chuỗi hiển thị trên menu.

- Name: Phải duy nhất và dễ nhớ. Đây là tên để nhận biết thành phần nào của Menu được chọn.

- Shortcut: dùng để thiết lập các phím tắt (Shortcut key).

- WindowList: Dùng trong các ứng dụng MDI (Multi Document Interface). Đây là ứng dụng có một biểu mẫu chính và nhiều biểu mẫu con. Thuộc tính này ra lệnh cho VB hiển thị tiêu đề của các biểu mẫu con trên menu.

- Checked: Nếu chọn thuộc tính này thì sẽ có một dấu hiển thị bên cạnh trái, nhưng thuộc tính này không được áp dụng cho những mục menu có chứa menu con.

- Enabled: Nếu thuộc tính này không được chọn thì mục này sẽ bị xám đi và người dùng không thể chọn.

- Visible: Nếu thuộc tính này không được chọn thì mục này sẽ không được hiển thị.

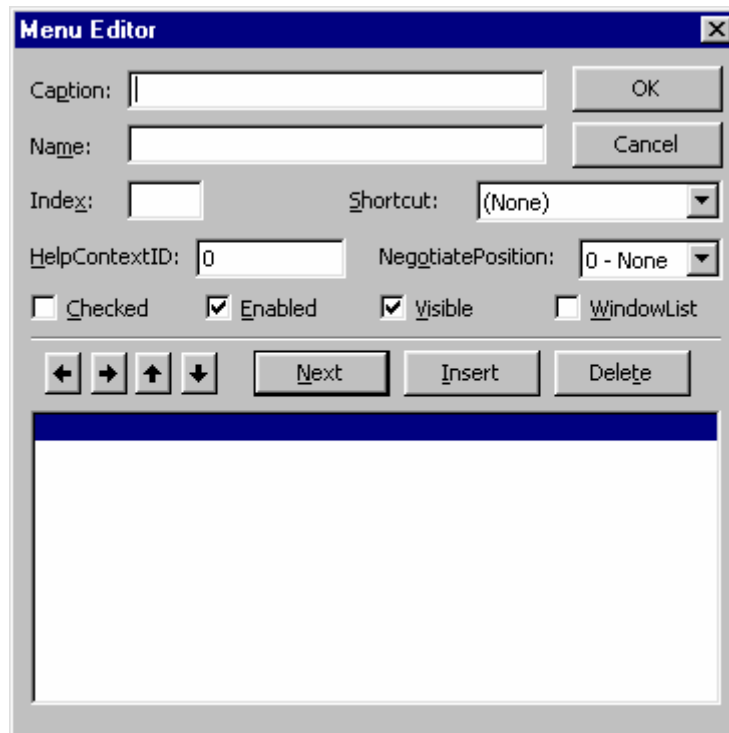
- NegotiatePosition: Quản lý vị trí gắn menu trong trường hợp sử dụng các đối tượng ActiveX.

### 3. Các sự kiện

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

### 4. Cách tạo Menu

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



- Menu không chứa trong hộp công cụ mà được thiết kế từ trình soạn thảo menu. Trong Visual Basic 6.0 IDE, chọn Tools, rồi Menu Editor (hoặc ấn Ctrl - E) để mở chương trình này.

- Ví dụ: Tạo một Drop-Down Menu.

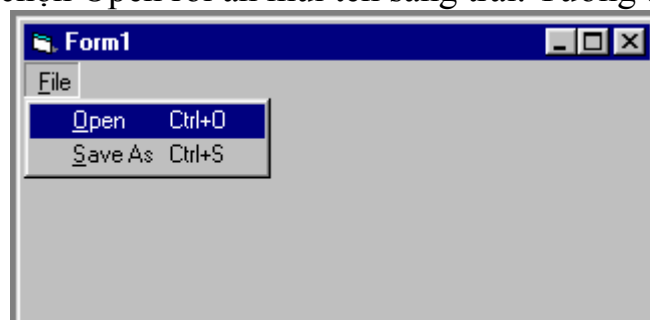
+ Tạo một đề án mới.

+ Ấn Ctrl-E để mở Menu Editor

+ Ta sẽ tạo một Menu File với Menu con là Open và Save As.

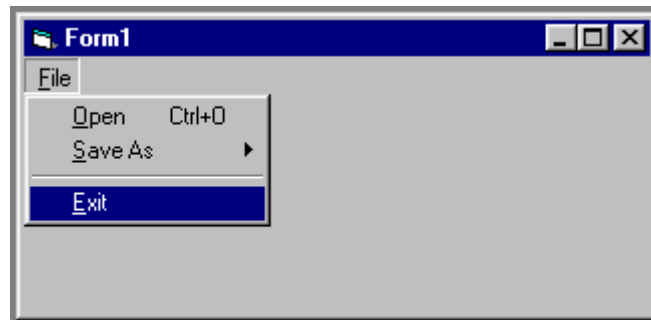
+ Trước tiên ta nhập vào &File trong ô Caption và nhập một tên bất kỳ vào ô Name (chẳng hạn là mfile). Ký tự & trước chữ F cho biết chữ F sẽ là phím tắt (ấn Ctrl-F) coi như chọn menu File.

+Tiếp theo ta nhập &Open và &Save As. Để Open và Save As là Sub menu của File, ta chọn Open rồi ấn mũi tên sang trái. Tương tự đối với Save As.



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

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



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

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

- + Chạy thử ứng dụng, khi ta ấn chuột phải, một menu sẽ bật ra.
- + Lệnh PopupMenu cho biết tên menu cần bật ra, đó là tên mà ta đã đặt trong trình soạn thảo MenuEditor, ở đây là mfile.
- Kể đến, đó là tham số xác định cách hiển thị menu: vbPopupMenuLeftAlign, vbPopupMenuRightAlign, vbPopupMenuCenterAlign.
- Sau khi đã thiết kế xong menu, ta sẽ viết các đoạn mã để VB sẽ thi hành một công việc nào đó tương ứng với mục được chọn. Công việc thi hành sẽ được viết bên trong sự kiện Click của mục đó.

## II. Hộp thoại

### 1. Khái niệm

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

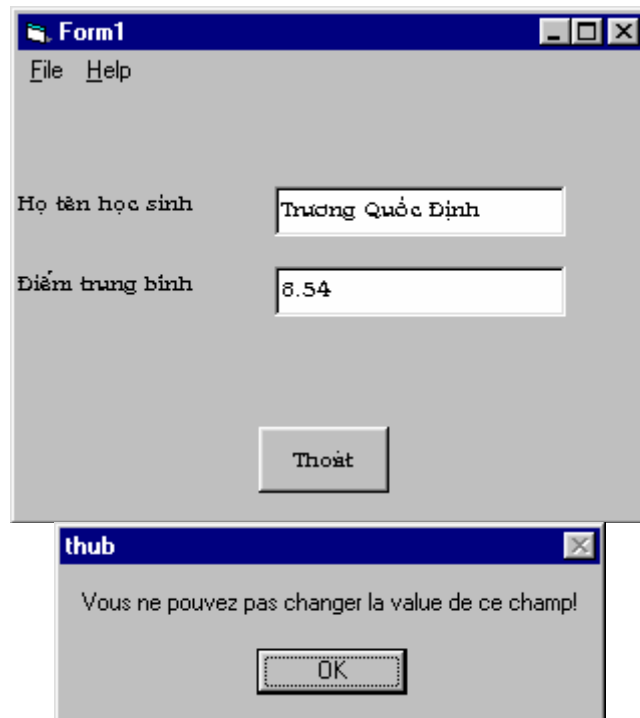
- Hộp thông điệp (Message Box).
- Hộp nhập (Input Box).
- Các hộp thoại thông dụng (Common Dialog)
- Hộp thoại hiệu chỉnh (Custom Dialog).

### 2. Hộp thông điệp

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

#### 2.1. Loại chỉ xuất thông báo

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



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

```
Private Sub Text2_Click()
```

```
    MsgBox "Vous ne pouvez pas changer la value de ce champ!"
```

```
End Sub
```

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

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

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

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

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

- Các loại biểu tượng trên hộp công cụ

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

## 2.2. Loại tương tác với người dùng

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

- Cú pháp:

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

Hằng số	Giá trị	Nút
vbOK	1	OK
vbCancel	2	Cancel
vbAbort	3	Abort
vbRetry	4	Retry
vbIgnore	5	Ignore
vbYes	6	Yes
vbNo	7	No

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

## 3. Hộp nhập

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

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

- Thông tin được nhập là rất ít.

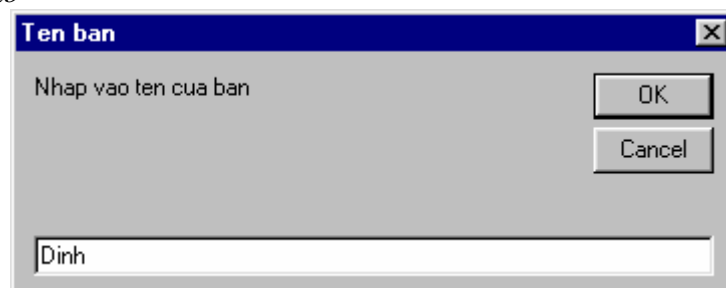
- Sau đây là một ví dụ về hộp nhập:

```
Public Sub Main ()
```

```
Dim ReturnString As String
```

```
ReturnString = InputBox("Nhập vào tên của bạn", "Tên bạn")
```

```
End Sub
```



#### 4. Các hộp thoại thông dụng

Có 6 loại hộp thoại thông dụng:

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

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

##### 4.1. Hộp thoại mở và lưu tập tin

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

- Các thuộc tính quan trọng:

+ Name: tên của Common Dialog.

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

+ FilterIndex: nếu có nhiều phần mở rộng của tên tập tin được mô tả trong thuộc tính Filter thì thuộc tính này xác định mặc định loại tập tin nào được chọn (là một số nguyên).

+ FileName: trả về tên tập tin sau khi người sử dụng hộp thoại chọn một tập tin nào đó.

+ CancelError: nếu TRUE thì trả về giá trị lỗi khi người dùng chọn nút Cancel, mặc nhiên giá trị này là False.

- Phương thức:

+ ShowOpen: mở ra hộp thoại mở tập tin.

+ ShowSave: mở ra hộp thoại lưu tập tin.

- Ví dụ:

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

```
ErrHandler:  
    MsgBox Err.Description  
End Sub
```

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

All Files: (\*.\*)

Text Files: (\*.txt)



Batch Files: (\*.bat)

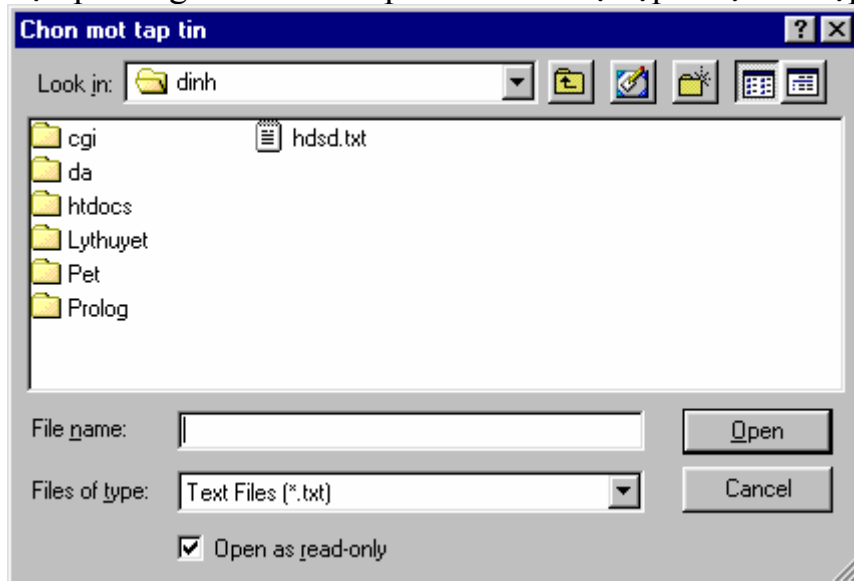
- + Các nhóm tập tin được thể hiện trong thuộc tính Filter. Mỗi nhóm tập tin cách nhau bởi dấu phân cách |.

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

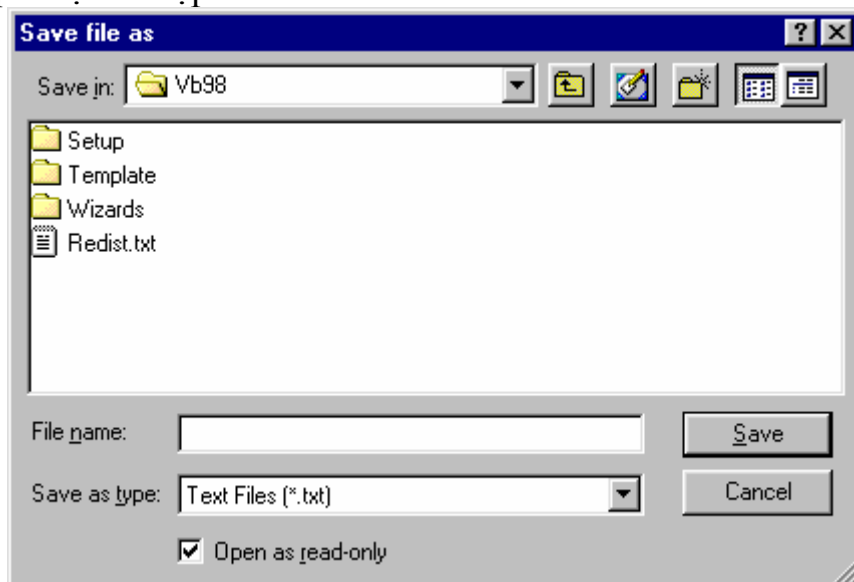
- + Sau khi đã chọn một tập tin và nhấn nút Open, ta sử dụng thuộc tính FileName để nhận về tên tập tin đã chọn.

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

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



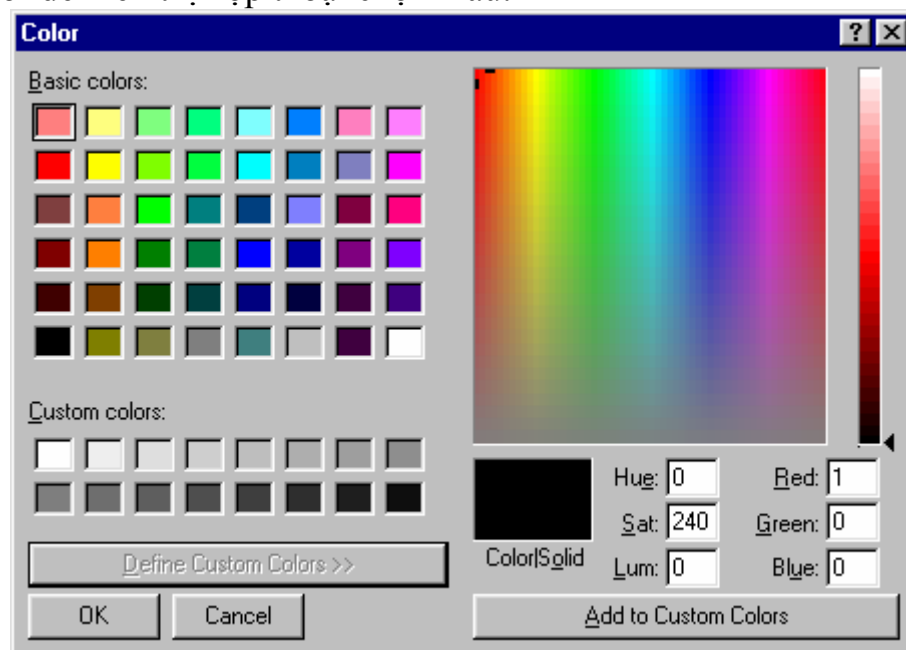
- + Các thuộc tính cũng tương tự đối với hộp thoại lưu tập tin, ta chỉ cần thay đổi tiêu đề của Dialog và dùng phương thức ShowSave. Dưới đây là minh họa cho hộp thoại lưu tập tin.



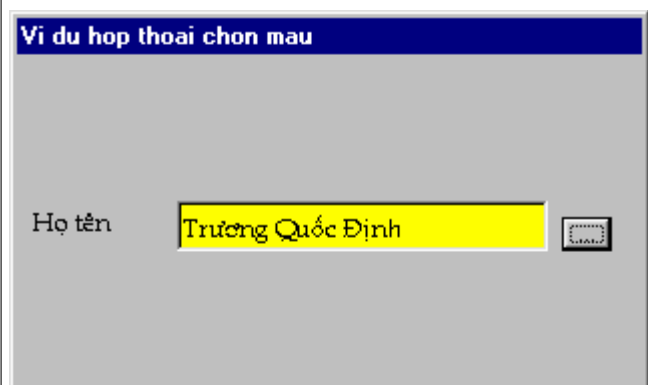
## 4.2. Hộp thoại chọn màu

- Đây là hộp thoại cho phép người dùng chọn và hiển thị các màu có sẵn trong bảng màu của Windows cũng như thiết lập thêm nhiều màu mới. Một thuộc tính quan trọng đối với hộp thoại chọn màu đó là thuộc tính Color, thuộc

tính này trả về giá trị của màu đã được chọn. Ta sẽ dùng phương thức ShowColor để hiển thị hộp thoại chọn màu.



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



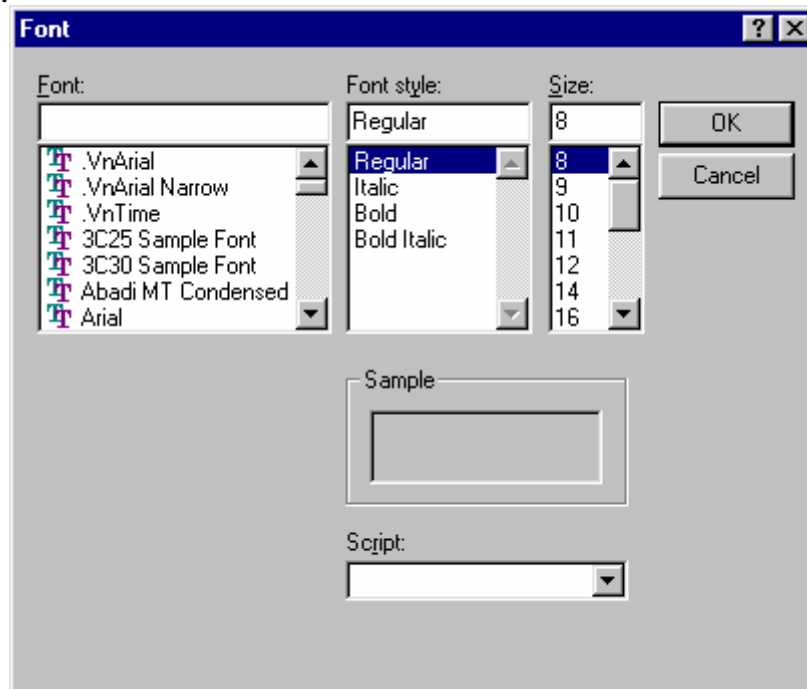
- Ta sẽ thiết kế một nút nhấn nhỏ bên cạnh điều khiển TextBox, nút nhấn này cho phép người sử dụng chọn màu nền của TextBox. Ta có đoạn mã lệnh sau:

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

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

#### 4.3. Hộp thoại chọn Font chữ

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



Thuộc tính	Giải thích
<code>Color</code>	Lưu giữ giá trị của màu được chọn
<code>FontBold</code>	TRUE nếu người dùng chọn chế độ đậm (Bold) và FALSE nếu ngược lại.
<code>FontItalic</code>	TRUE nếu người dùng chọn chế độ nghiêng (Italic) và FALSE nếu ngược lại.
<code>FontStrikeThrough</code>	TRUE nếu chọn chế độ gạch ngang các ký tự.
<code>FontUnderLine</code>	TRUE nếu chọn chế độ gạch dưới
<code>FontName</code>	Tùy ý chọn tên Font chữ
<code>Max</code>	Kích cỡ lớn nhất của Font được hiển thị
<code>Min</code>	Kích cỡ nhỏ nhất của font được hiển thị
<code>FontSize</code>	Kích cỡ của Font được chọn

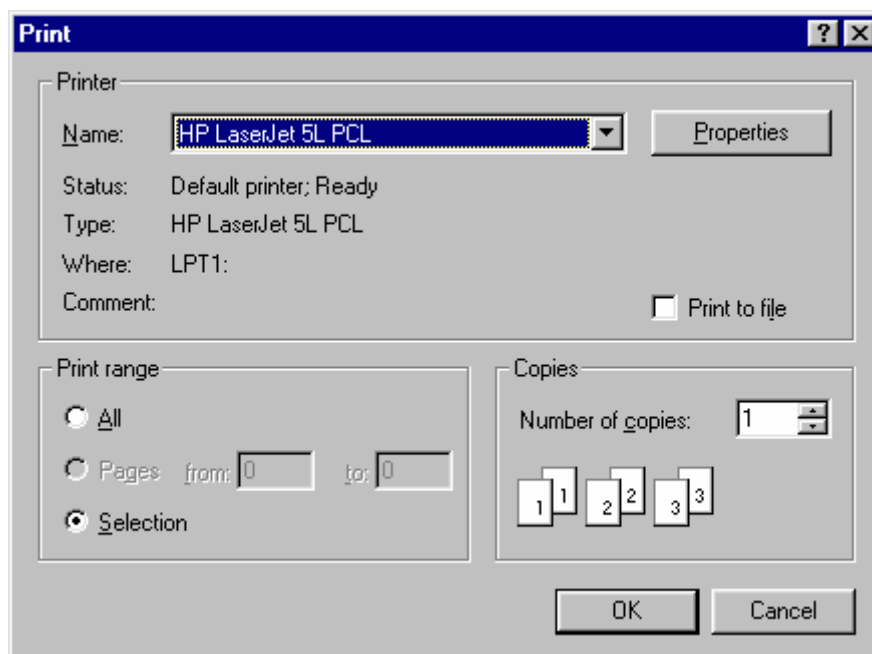
- Nếu muốn chọn màu cho Font, ta thêm 256 vào giá trị của thuộc tính `Flags`. Nếu không có điều này, ta chỉ thấy tên Font, kiểu Font và kích cỡ Font mà thôi.

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

#### 4.4. Hộp thoại in ấn

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

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



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

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

### III. Xử lý các sự kiện chuột và bàn phím

#### 1. Sự kiện chuột

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

- Có 3 sự kiện chuột chủ yếu, đó là:

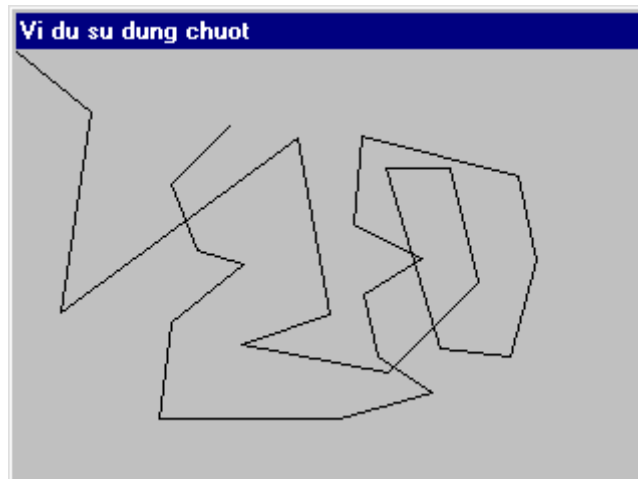
Sự kiện	Giải thích
MouseDown	Xảy ra khi người sử dụng ấn chuột (chuột trái hoặc phải)
MouseUp	Xảy ra khi người sử dụng thả một nút chuột bất kỳ
MouseMove	Xảy ra khi con trỏ chuột di chuyển đến một điểm mới trên màn hình.

- Các tham số

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

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

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

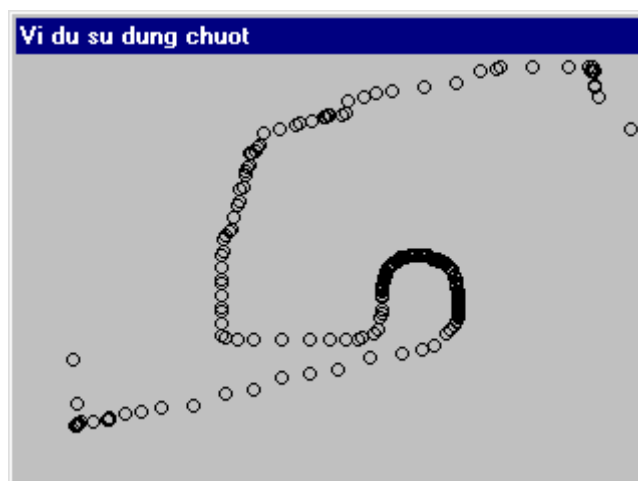


- Ví dụ 2: Sử dụng sự kiện MouseUp để hiển thị một thông điệp cho biết nút chuột nào vừa được thả. Sự kiện Form\_MouseUp được xử lý:

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

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

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



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

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

Hằng	Giá trị	Diễn giải
<code>ccDefault</code>	0	(Default) Shape determined by the object.
<code>ccArrow</code>	1	Arrow.
<code>ccCross</code>	2	Cross (cross-hair pointer).
<code>ccIbeam</code>	3	I Beam.
<code>ccIcon</code>	4	Icon (small square within a square).
<code>ccSize</code>	5	Size (four-pointed arrow pointing north, south, east, and west).
<code>ccSizeNESW</code>	6	Size NE SW (double arrow pointing northeast and southwest).
<code>ccSizeNS</code>	7	Size N S (double arrow pointing north and south).
<code>ccSizeNWSE</code>	8	Size NW, SE.
<code>ccSizeEW</code>	9	Size E W (double arrow pointing east and west).
<code>ccUpArrow</code>	10	Up Arrow.
<code>ccHourglass</code>	11	Hourglass (wait).
<code>ccNoDrop</code>	12	No Drop.
<code>ccArrowHourglass</code>	13	Arrow and hourglass.
<code>ccArrowQuestion</code>	14	Arrow and question mark.
<code>ccSizeAll</code>	15	Size all.
<code>ccCustom</code>	99	Custom icon specified by the <code>MouseIcon</code> property.

## 2. Sự kiện bàn phím

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

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

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

- + Khi người dùng bấm một tổ hợp phím SHIFT, CTRL và ALT.

- + Phím định hướng.

- + PAGEUP và PAGEDOWN.

- + Phân biệt được phím số ở bên phải bàn phím và phím số ở bên trái bàn phím.

- + Đáp ứng khi thả phím.

- + Phím chức năng không trùng với menu.

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

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

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

- Sau đây là ví dụ về điều này:

```
Private Sub Form_KeyPress (KeyAscii As Integer)
```

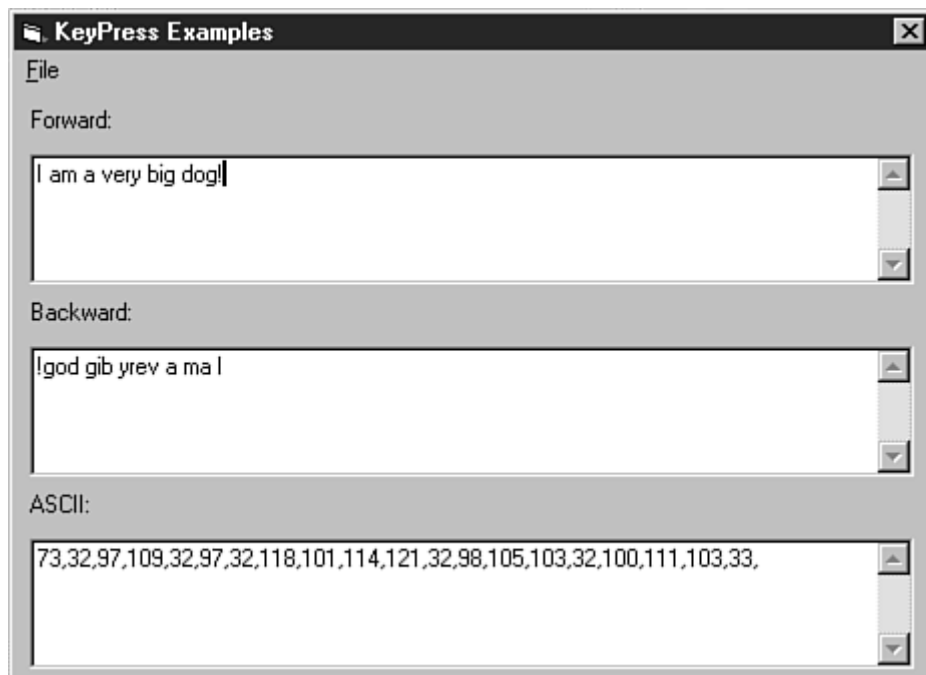
```
    ' Gửi điều khiển đến textbox đầu tiên
```

```
    txtForward.SetFocus
```

```
    txtBackward.Text = Chr(KeyAscii) & txtBackward.Text
```

```
    txtAscii.Text = txtAscii.Text & CStr(KeyAscii) & ", "
```

```
End Sub
```



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

#### **IV. Xử lý đồ họa và giao diện**

##### **1. Hiện thị hình ảnh**

###### **1.1. Sử dụng Picture Box**

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

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

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

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

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

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

###### **1.2. Sử dụng Image Control**



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

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

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

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

## **2. Xử lý đồ họa**

### **2.1. Tọa độ màn hình**

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

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

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

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

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

### **2.2. Các phương thức đồ họa**

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

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

Cú pháp:

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

object	Object là đối tượng mà phương thức sẽ làm việc, nó có thể là biểu mẫu, hộp hình hay đối tượng máy in.
picture	Hình ảnh nguồn sẽ được vẽ lên đối tượng phải được chỉ rõ bởi thuộc tính Picture của biểu mẫu hoặc hộp hình.
x1, y1	Giá trị chỉ định vị trí của hình ảnh trên đối tượng. Thuộc tính ScaleMode xác định hệ tọa độ nào được sử dụng.
width1	Giá trị xác định độ rộng của hình ảnh, nếu bỏ qua thì mặc định là độ rộng của ảnh nguồn.
height1	Giá trị xác định độ cao của hình ảnh, nếu bỏ qua thì mặc định là độ

	cao của ảnh nguồn.
x2, y2	Các giá trị xác định hình ảnh sẽ được vẽ lại từ vị trí nào. Nếu bỏ qua thì giá trị mặc định là 0, tức toàn bộ hình ảnh được vẽ lại.
width2	Tương tự như Width1, nhưng ở đây là tác động đến ảnh nguồn.
height2	Tương tự như Height1, nhưng ở đây là tác động đến ảnh nguồn.
opcode	Đây là tùy chọn và chỉ có tác dụng với ảnh Bitmap

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

*Dim re*

*Private Sub Form\_Load()*

*re = False*

*End Sub*

*Private Sub Form\_MouseDown(Button As Integer, & \_*

*Shift As Integer, X As Single, Y As Single)*

*re = True*

*End Sub*

*Private Sub Form\_MouseMove(Button As Integer, & \_*

*Shift As Integer, X As Single, Y As Single)*

*If re Then*

*Form1.PaintPicture Image1.Picture, X, Y, & \_*

*Image1.Width, Image1.Height*

*End If*

*End Sub*

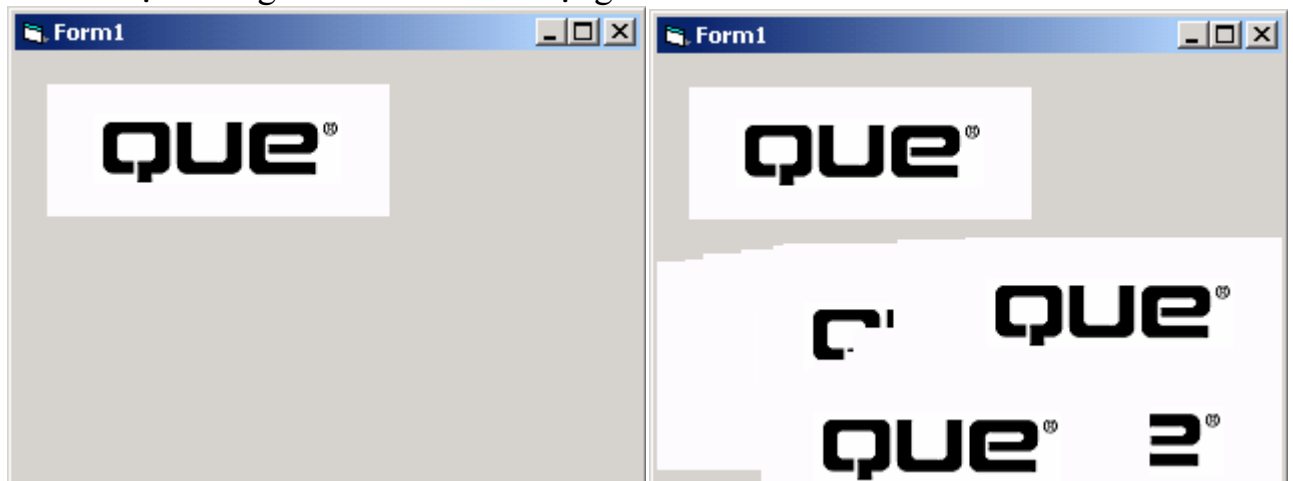
*Private Sub Form\_MouseUp(Button As Integer, & \_*

*Shift As Integer, X As Single, Y As Single)*

*re = False*

*End Sub*

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



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

+ Cú pháp : Object.PSet [Step] (x, y), [color]

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

Color	Màu của điểm đó.
-------	------------------

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

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

## **CHƯƠNG VII: CÁC KHÁI NIỆM CƠ BẢN VỀ CƠ SỞ DỮ LIỆU**

### **I. Cơ sở dữ liệu**

#### **1. Khái niệm**

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

- Một cơ sở dữ liệu quan hệ:

+ Chứa dữ liệu trong các bảng, được cấu tạo bởi các dòng còn gọi là các mẫu tin, và cột còn gọi là các trường.

+ Cho phép lấy về (hay truy vấn) các tập hợp dữ liệu con từ các bảng.

+ Cho phép nối các bảng với nhau cho mục đích truy cập các mẫu tin liên quan với nhau chứa trong các bảng khác nhau.

#### **2. Bộ máy (Engine) cơ sở dữ liệu**

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

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

#### **3. Bảng (Table) và trường (Field)**

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

- Bảng: chứa các mẫu tin là các mẫu dữ liệu riêng rẽ bên trong phân nhóm dữ liệu.

- Mẫu tin: chứa các trường, mỗi trường thể hiện một bộ phận dữ liệu trong một mẫu tin. Ví dụ như mỗi mẫu tin thể hiện một mục trong danh bạ địa chỉ chứa các trường tên và họ, địa chỉ, thành phố, số điện thoại...

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

#### **4. Tập mẫu tin (Recordset)**

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

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

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

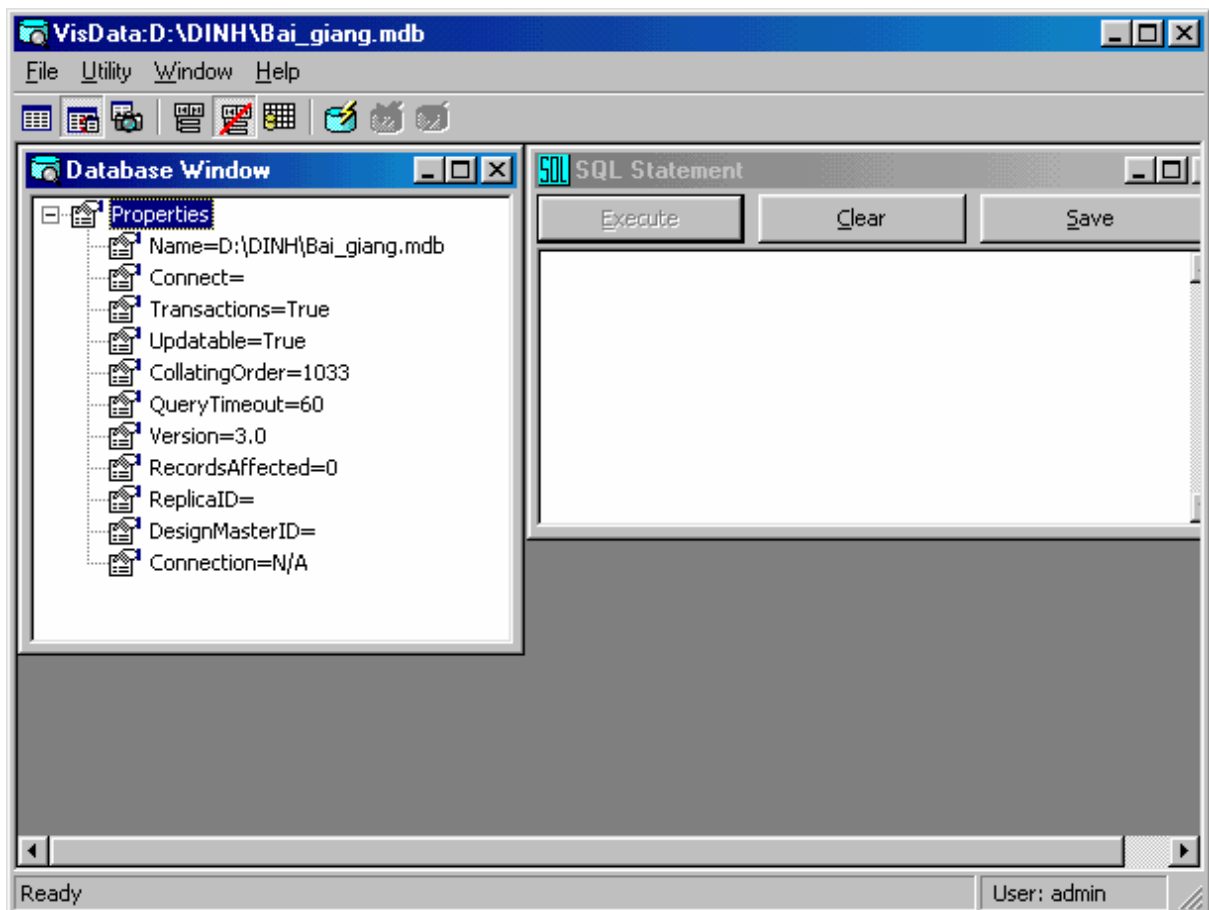
Visual Basic cung cấp kèm theo nó một bộ máy cơ sở dữ liệu có thể hiểu được dữ liệu của Microsoft Access gọi là Joint Engine Technology (viết tắt là JET).

#### **1. Sử dụng cửa sổ cơ sở dữ liệu**

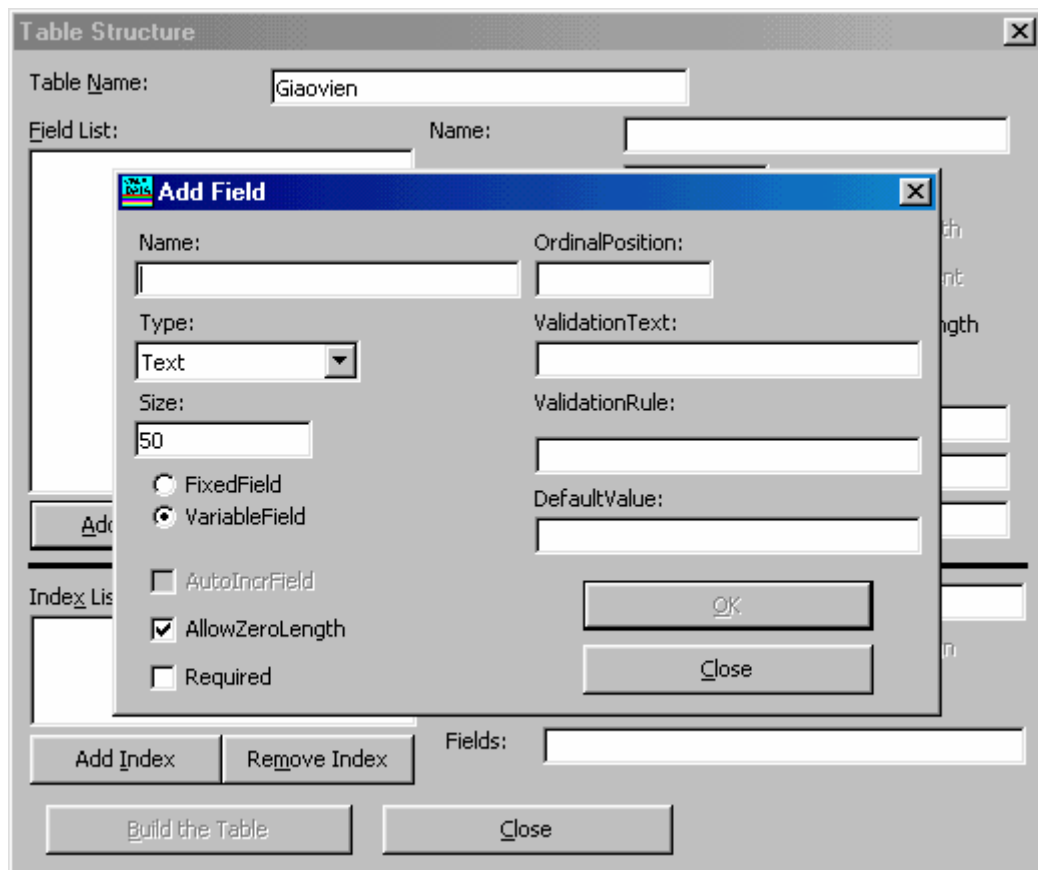
- Từ Menu của VB6, chọn mục Add-Ins, Visual Data Manager. Cửa sổ Visual Data Manager sẽ xuất hiện.

- Chọn mục File -> New -> MicroSoft Access -> Version 7.0 MDB.

- Chọn thư mục ta muốn lưu cơ sở dữ liệu và tên của cơ sở dữ liệu.



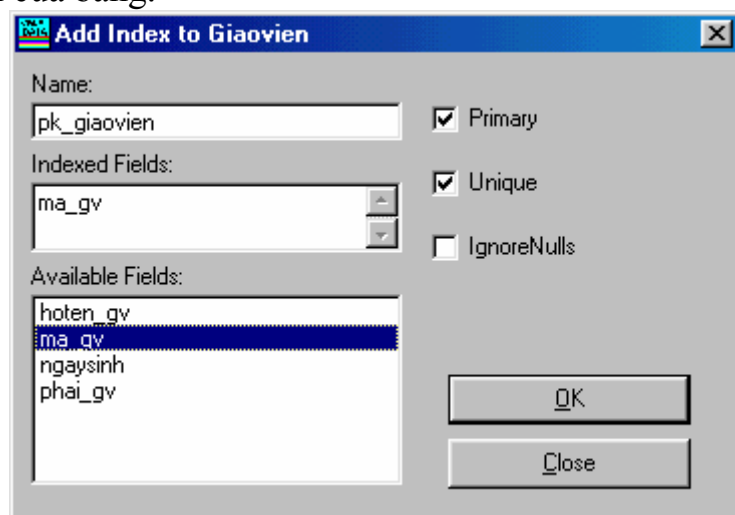
- Tạo bảng
- + Để tạo mới một bảng, ta chọn Properties trong cửa sổ Databases, nhấp chuột phải, chọn New Table, đặt tên cho Table tại ô Table Name, ấn Add Field để tạo mới các trường cho bảng.



+ Ta sẽ nhập tên trường tại ô Name, chọn kiểu của trường tại Combo Type, tùy chọn FixedField và VariableField xác định độ dài của trường là cố định hay thay đổi.

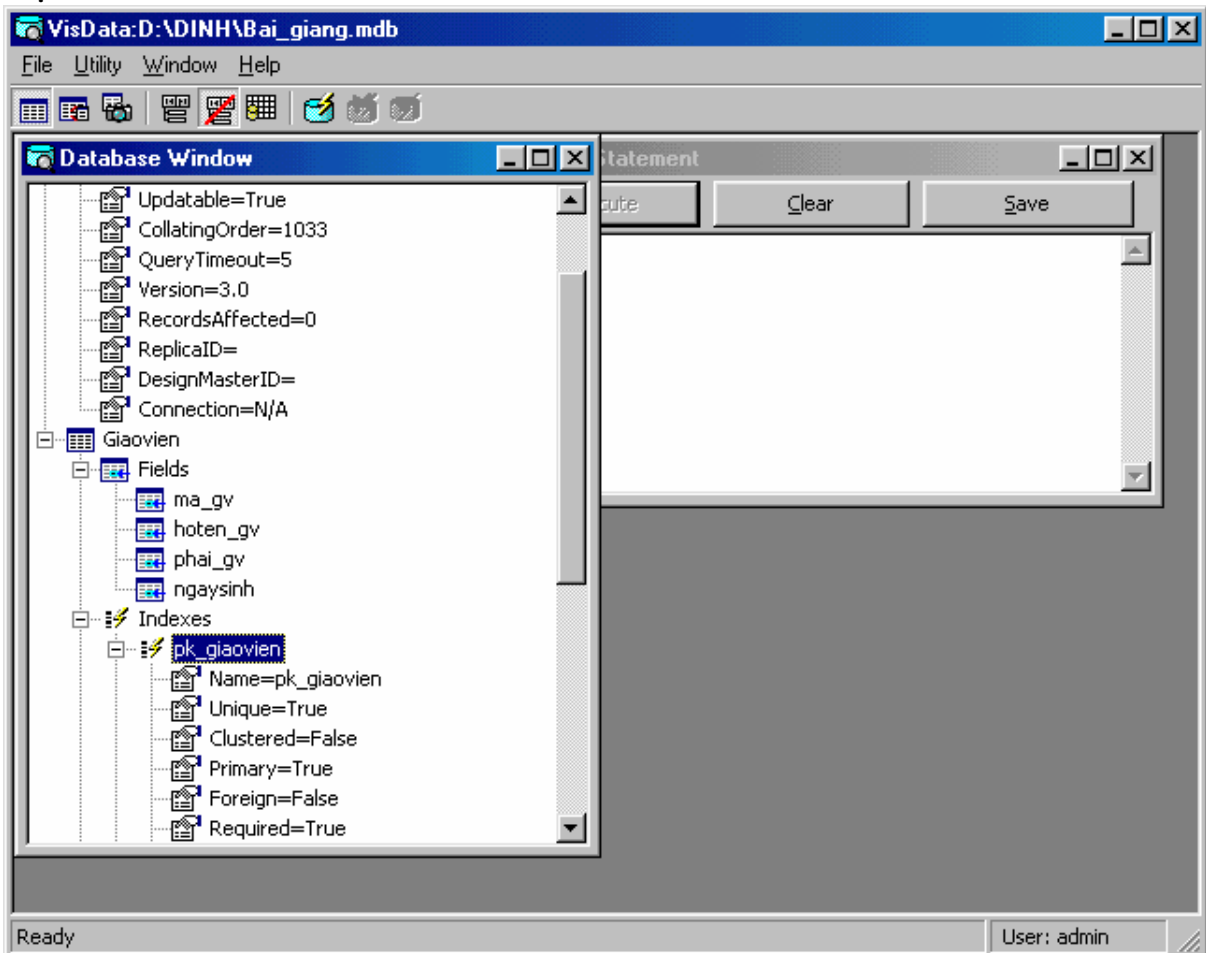
+ Sau khi xác định đầy đủ các thuộc tính của trường, ấn OK và tiếp tục thêm vào các trường khác cho bảng. Nếu đã thêm mới đầy đủ các trường của bảng, ấn Close để quay về cửa sổ Table Structure.

+ Sau khi quay về cửa sổ Table Structure, ta sẽ xác lập các chỉ mục cũng như khóa chính của bảng.



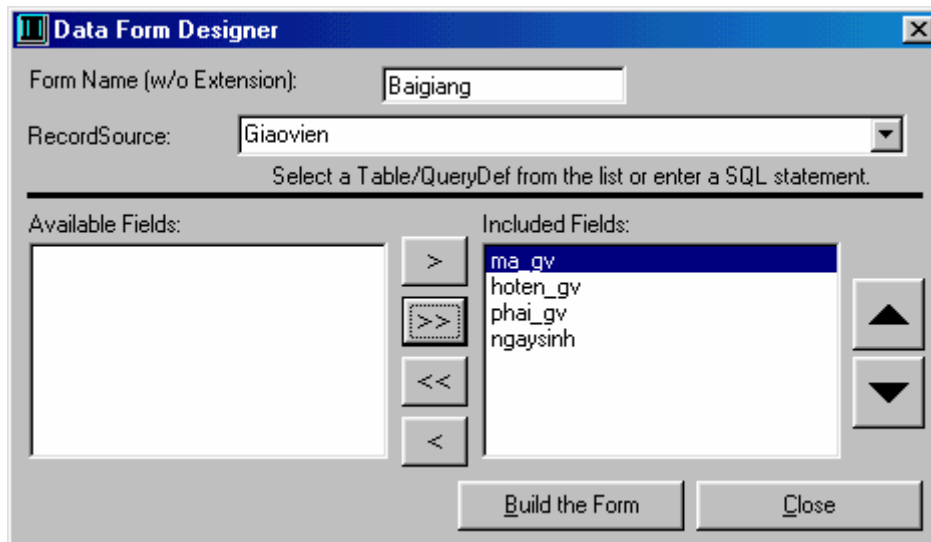
+ Tại ô Name, ta sẽ nhập vào tên của chỉ mục, rồi chọn các trường tham gia vào chỉ mục đó. Nếu ta chọn Primary thì đó chính là các trường cấu thành khóa chính của bảng. Chọn Unique tức là giá trị của chỉ mục đó sẽ không có sự trùng lặp.

- + Ấn Close xác nhận rằng ta đã xây dựng xong tập các chỉ mục của bảng.
- + Sau khi đã hoàn thành tất cả các thao tác trên, để tạo bảng ta ấn Build the Table.
- + Tuy rằng đây là một tính năng mới của VB6, tuy nhiên chúng ta cũng sẽ gặp phải rất nhiều bất tiện khi phải thiết kế một cơ sở dữ liệu hoàn chỉnh cũng như trong quá trình bảo trì và sử dụng (khó khăn trong việc thay đổi các thuộc tính đã xác lập, không tạo liên kết giữa các bảng được ...). Một phương cách tốt nhất đó là nên dùng các hệ quản trị cơ sở dữ liệu chuyên dùng để thực hiện công việc nêu trên.

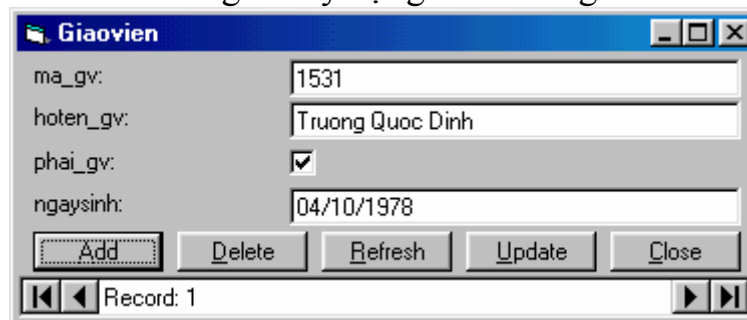


## 2. Dùng Visual Data Manager để tạo giao diện

- Ta có thể thiết kế một Form nhập liệu đơn giản cho một Table từ Visual Data Manager. Các bước tiến hành như sau:
  - + Từ Visual Data Manager chọn Cơ sở dữ liệu cần thao tác.
  - + Chọn Data Form Design từ mục Utility.
  - + Chọn Table cần cho việc tạo Form và các trường hiển thị trên Form (thông thường chúng ta sẽ cho hiển thị tất cả các trường).
  - + Chọn Build the Form, biểu mẫu mới đã được tạo trong đề án của chúng ta.
  - + Kết quả sau khi chúng ta xây dựng Form bằng Visual Data Manager:



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

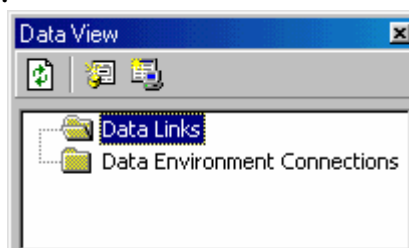


### III. Sử dụng cửa sổ xem dữ liệu (Data View)

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

- Từ Menu của VB chọn Data View hoặc nhấn nút Data View trên thanh công cụ.

- Cửa sổ Data View xuất hiện cho ta hai lựa chọn: Data Links và Data Environment Connections.



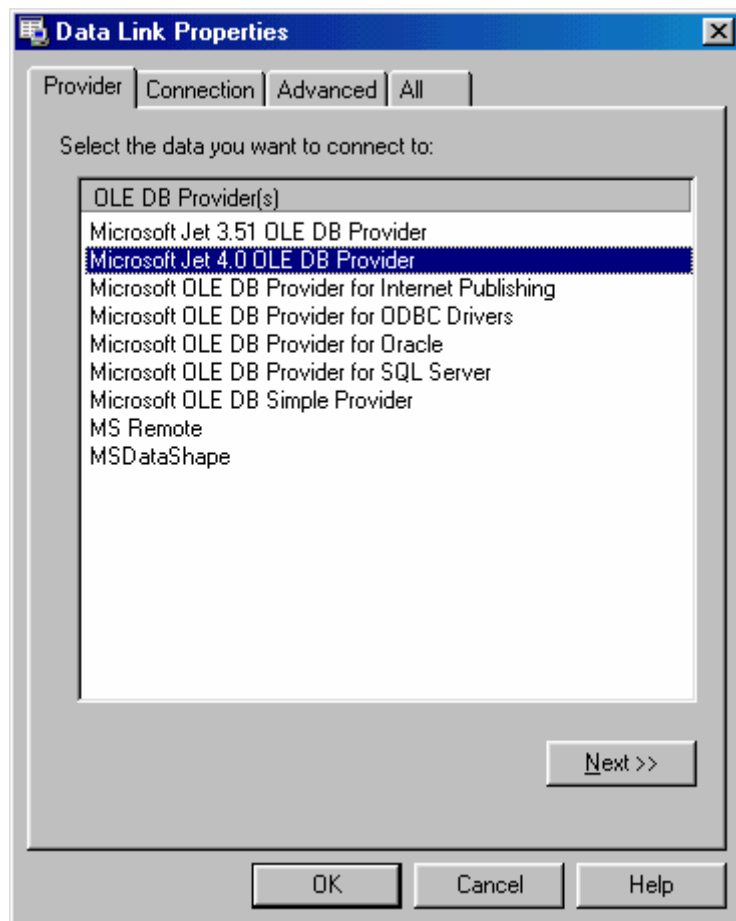
- Liên kết dữ liệu (Data Link) là một cách kết nối môi trường phát triển của VB6 với một cơ sở dữ liệu nào đó. Một kết nối môi trường dữ liệu (Data Environment Connection) là cách thức sử dụng cơ sở dữ liệu trong một đề án cụ thể. Điểm khác biệt giữa hai thành phần này là sự liên quan giữa cơ sở dữ liệu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

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

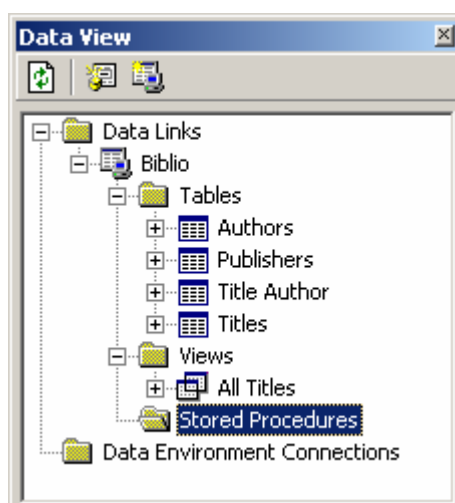
+ Chọn Data Links, ấn chuột phải -> Add a Data Link hoặc ấn nút Data Link trên thanh công cụ của cửa sổ.

+ Cửa sổ Data Link Properties xuất hiện.





- + Chọn trình cung cấp Microsoft Jet, chọn Next.
- + Chọn cơ sở dữ liệu muốn nối kết đến, nhấn OK.
- Liên kết dữ liệu cung cấp một cách nhìn tóm lược về nguồn dữ liệu. Mỗi lần ta tạo một liên kết dữ liệu, ta có thể duyệt bằng cách mở rộng phần tử trong danh sách tóm lược. Thực hiện điều này bằng cách nhấn vào dấu cộng bên trái mỗi phần tử.



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

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

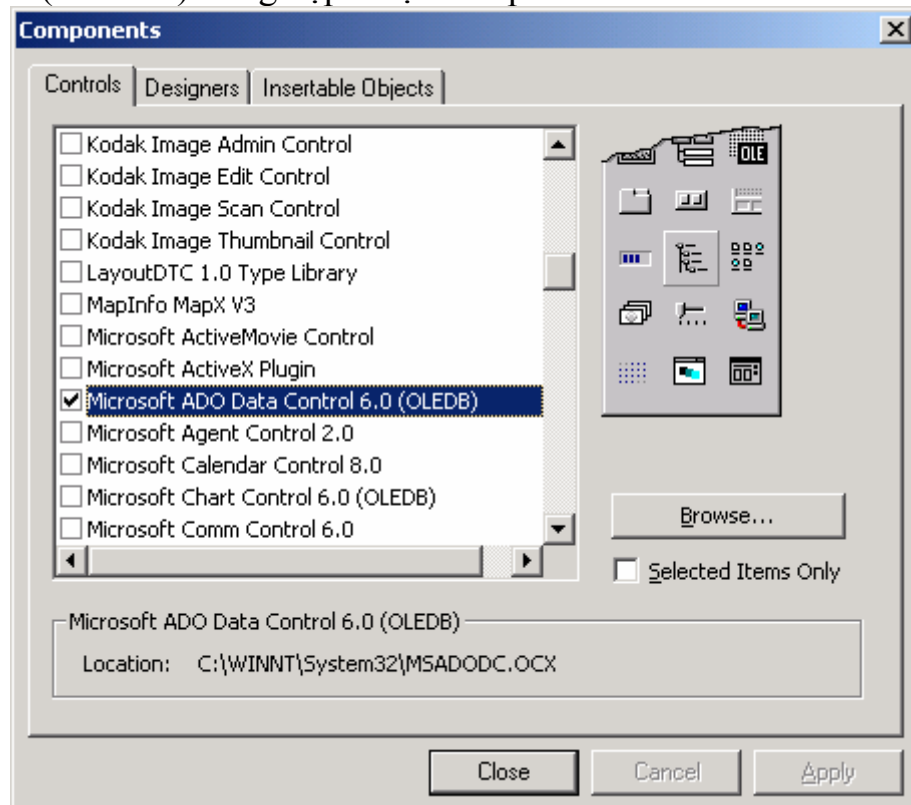
Đối với phiên bản VB6 cung cấp cho chúng ta 3 trình điều khiển dữ liệu: DAO (Data Access Object), RDO (Remote Data Object) và ADO (ActiveX Data Object).

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

### 1.1. Hiển thị dữ liệu

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

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



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

+ ConnectionString: Xác định nguồn dữ liệu cần nối kết, đó chính là chuỗi nối kết chỉ đến cơ sở dữ liệu mà ta thao tác.

+ RecordSource: Xác định xem nối kết của ta đang thao tác trên bảng nào.

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

+ Chọn Use Connection String, ấn Build.

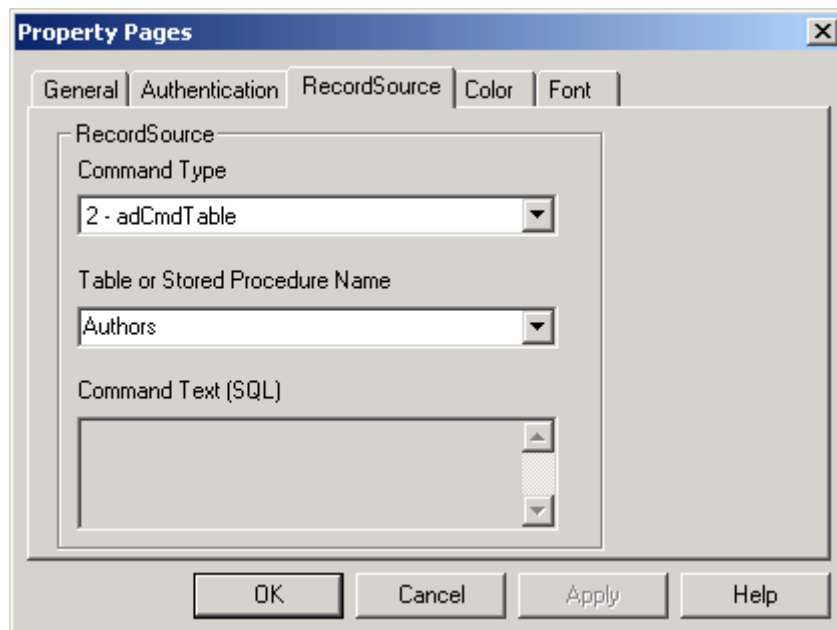
+ Chọn Microsoft Jet 4.0 OLE DB Provider.

+ Chọn cơ sở dữ liệu như ví dụ.

+ Ấn OK.

+ Quay về cửa sổ Property Pages, chọn Tab RecordSource, xác định các tùy chọn như hình vẽ.

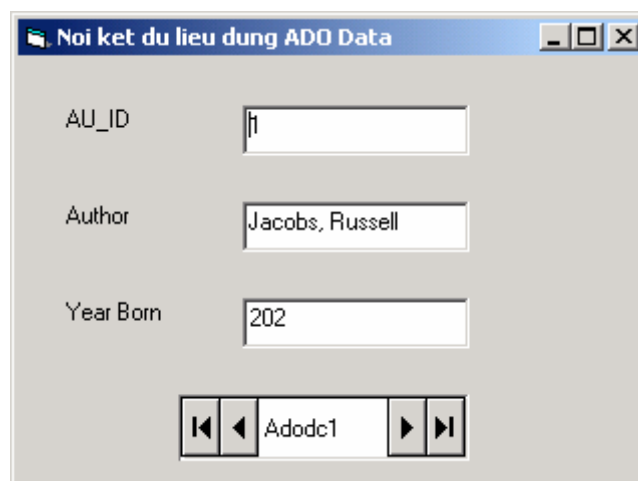
+ Ấn Close.



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

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

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



## 1.2. Cập nhật dữ liệu

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

## 1.3. Thêm mới mẫu tin

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

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

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

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

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

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

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

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

#### 1.5. Xóa mẫu tin

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

- Ví dụ: `Adodc1.Recordset.Delete`

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

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

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

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

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

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

- + Connect xác định cơ sở dữ liệu của ta là loại gì.

- + DatabaseName chỉ đến một cơ sở dữ liệu cụ thể.

- + RecordSource là một bảng dữ liệu trong cơ sở dữ liệu (đối với cơ sở dữ liệu Access).

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

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