ThS NGUYỄN ĐÌNH NAM

GIÁO TRÌNH TỰ HỌC

LẬP TRÌNH CĂN BẢN
Visual Basic 2008

Tái bản dành cho học sinh-Sinh viên

TẬP 2

NHÀ XUẤT BẢN HỒNG ĐỨC

MỤC LỤC

Chương 1
Giới thiệu ngôn ngữ Visual Basic.

1. Giới thiệu

VB được giới thiệu lần đầu tiên vào năm 1991, tiền thân là ngôn ngữ lập trình Basic trên HĐH DOS. Tuy nhiên, lúc bấy giờ VB chưa được nhiều người người tiếp nhận. Mãi cho đến năm 1992, khi phiên bản 3.0 ra đời với rất nhiều cải tiến so với các phiên bản trước đó, VB mới thật sự trở thành một trong những công cụ chính để phát triển các ứng dụng trên Windows.

Các phiên bản sau đó của VB, như phiên bản 4.0 ra đời năm 1995, phiên bản 5.0 ra đời năm 1996 và gần đây nhất là phiên bản 6.0 ra đời năm 1998 với các tính năng ngày càng được nâng cao đã khiến mọi người công nhận VB hiện là một trong những công cụ chính để phát triển các ứng dụng trên Windows.

Visual Basic 6.0 cho phép người dùng tiếp cận nhanh cách thức lập trình trên môi trường Windows.

* Ưu điểm:

- Tiết kiệm được thời gian và công sức so với một số ngôn ngữ lập trình có cấu trúc khác vì bạn có thể thiết lập các hoạt động trên từng đối tượng được VB cung cấp.
- Khi thiết kế chương trình có thể thấy ngay kết quả qua từng thao tác và giao diện khi thi hành chương trình.
- Cho phép chỉnh sửa dễ dàng, đơn giản.
- Làm việc với các điều khiển mới (ngày tháng với điều khiển MonthView và DataTimePicker, các thanh công cụ có thể di chuyển được CoolBar, sử dụng đồ họa với ImageCombo, thanh cuộn FlatScrollBar,…).
- Làm việc với cơ sở dữ liệu.
- Các bổ sung về lập trình hướng đối tượng.
- Khả năng kết hợp với các thư viện liên kết động DLL.

* Nhược điểm:

- Yêu cầu cấu hình máy khá cao.
- Chỉ chạy được trên môi trường Win95 trở lên.

2. Cấu hình máy

Cấu hình máy 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.

3. Cài đặt Visual Basic

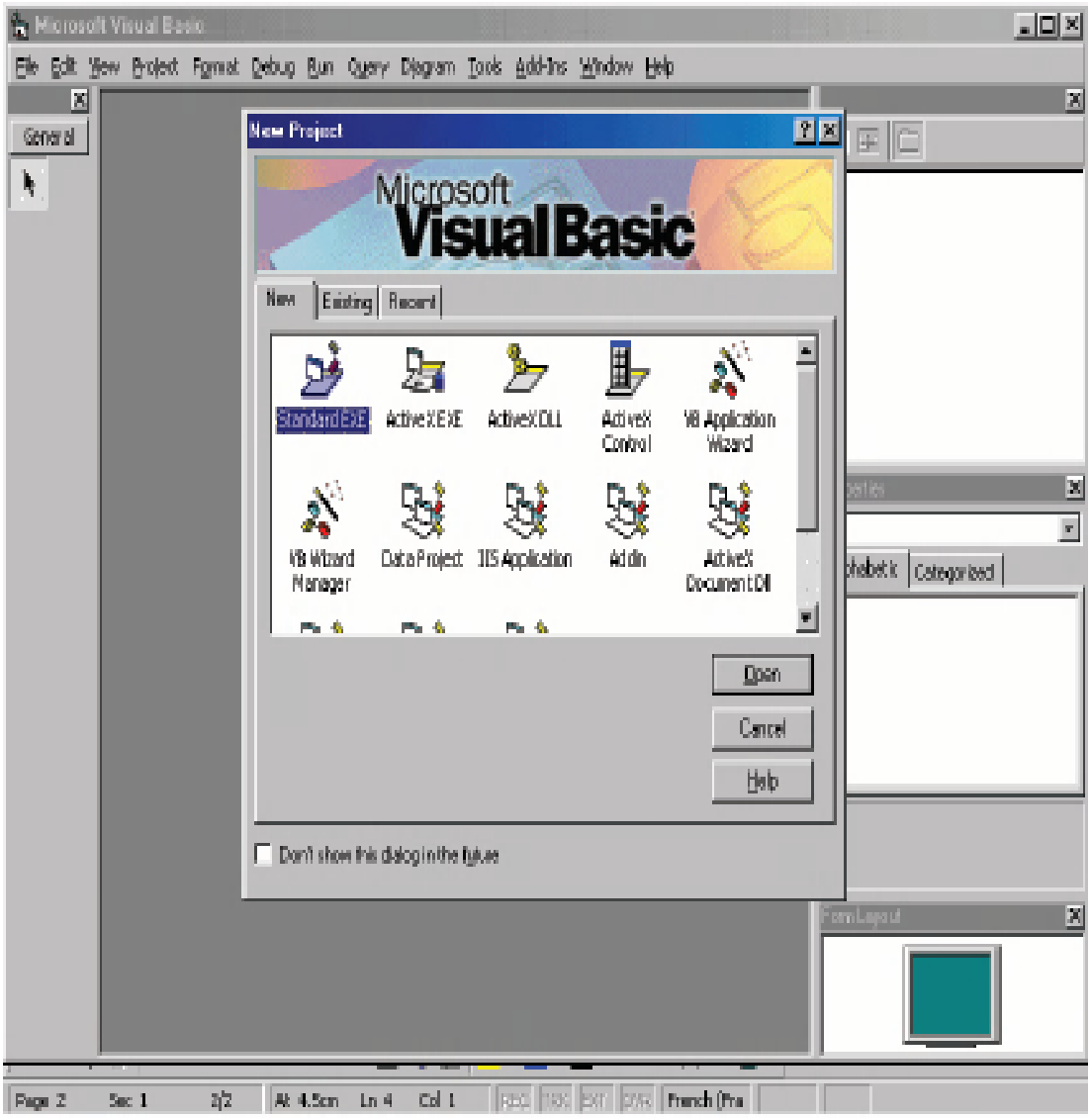
– Sử dụng chương trình Setup để cài đặt VB6.

– Chương trình Setup này còn cài đặt các tập tin cần thiết để xem tài liệu trên đĩa CD MSDN (Microsoft Developer Network).

– Nếu cần, người dùng có thể cài đặt riêng phần tài liệu và ví dụ mẫu của Visual Basic lên máy tính.

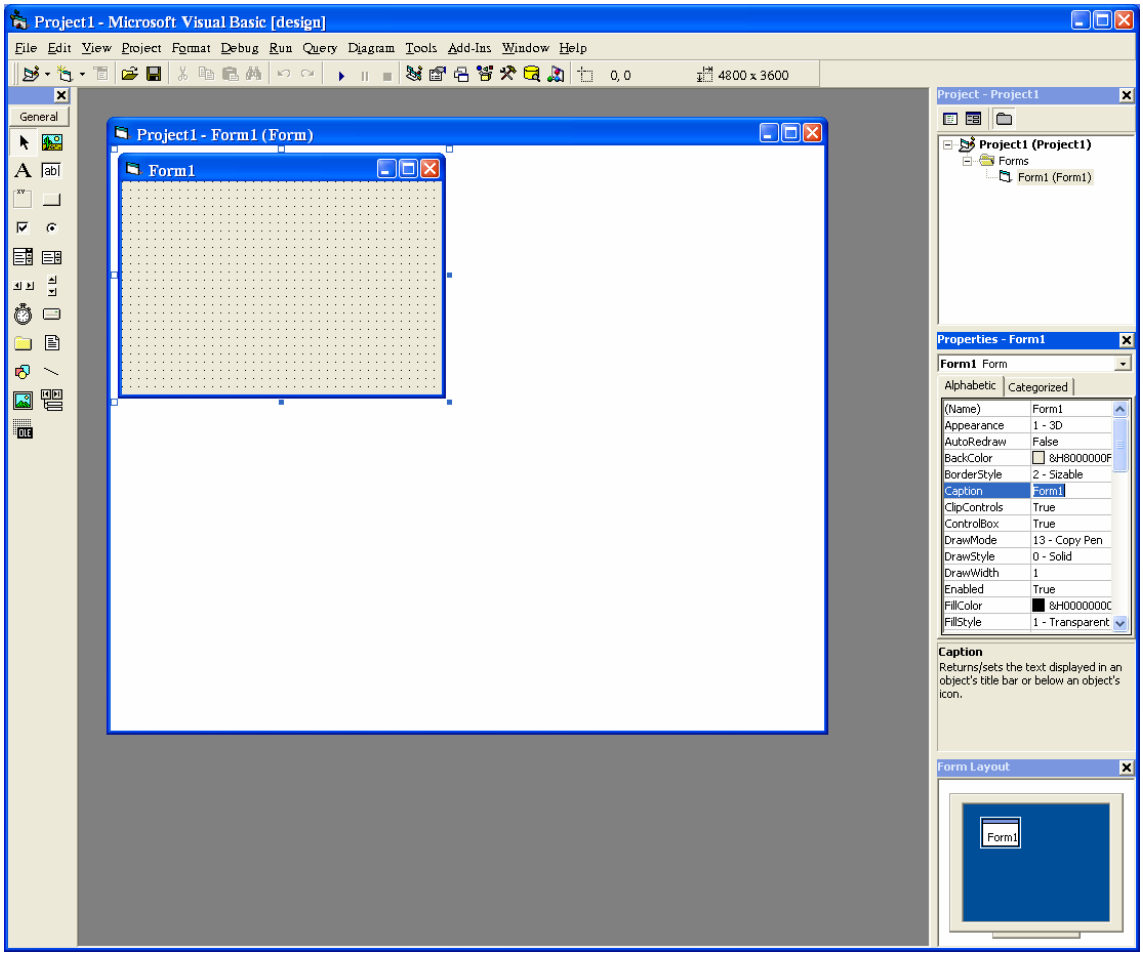
4. Khởi động

Từ menu Start chọn Programs, Microsoft Visual Basic 6.0 chọn Microsoft Visual Basic 6.0. Khi đó màn hình đầu tiên hiển thị như hình dưới đây:



Hình 1-1. Màn hình khởi động

5. Cửa sổ làm việc khi chọn Standard.exe



Hình 1-2. Cửa sổ làm việc của VB khi chọn Standard.exe

5.1. Thanh tiêu đề

5.2. Thanh Menu

5.3. Thanh công cụ

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 ...).

5.4. Hộp công cụ (ToolBox)

Hộp công cụ chứa các biểu tượng tương ứng với những đối tượng điều khiển chuẩn bao gồm nhãn, hộp văn bản, nút lệnh…

Ngoài các điều khiển có sẵn trong VB còn có các điều khiển mở rộng khác được chứa trong tập tin với phần mở rộng là .OCX.

Các điều khiển chuẩn có sẵn trong VB không thể gỡ bỏ khỏi ToolBox, các điều khiển mở rộng có thể được thêm vào và được gỡ khỏi ToolBox.

Hình 1-3. Hộp công cụ ToolBox

5.5. Cửa sổ Properties Window.

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 mục đích của người sử dụng.

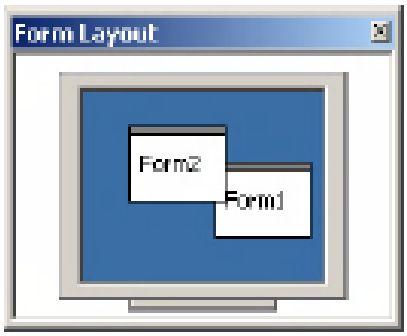
Hình 1-4. Cửa sổ Properties

5.6. Form Layout Window.

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

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

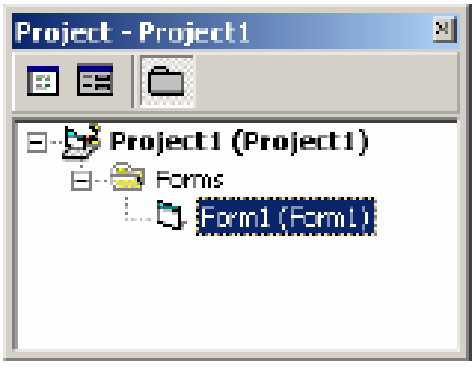


Hình 1-5. Cửa sổ Form Layout

5.7. Project Explorer Window.

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. 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.



Hình 1-6. Cửa sổ Project Explorer

6. Các lệnh trong menu File.

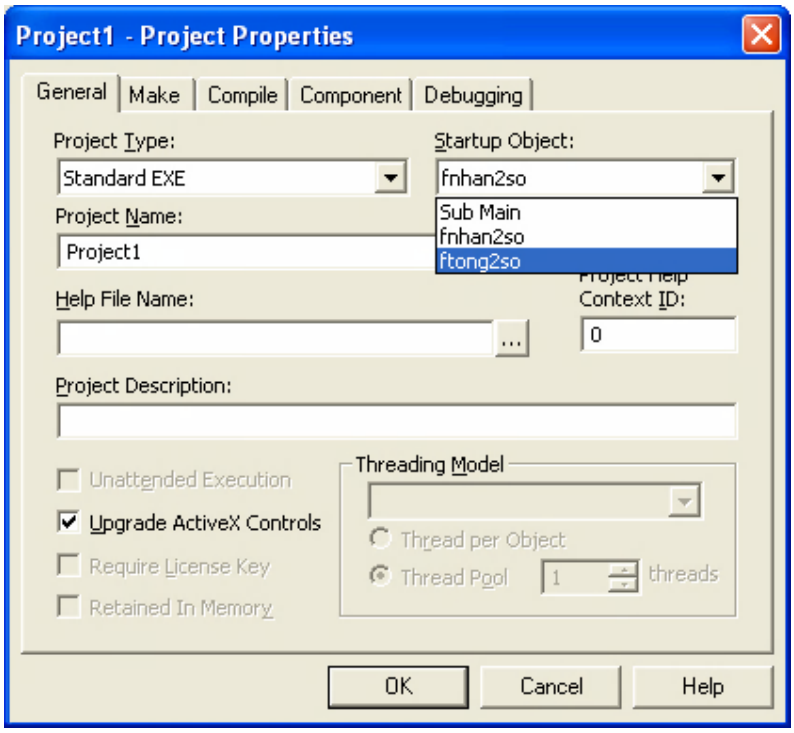
- **New Project**: Mở một đề án mới.
- **Open Project**: Mở một đề án đã tồn tại.

- **Add Project**: Thêm vào đề án một đề án đã có hoặc một đề án mới.
- **Remove Project**: Gỡ bỏ đề án đang làm việc.
- **Save Form1**: Lưu Form1 dưới dạng tập tin *.Frm.
- **Save Form1 As**: Lưu tập tin Form1 với một tập tin mới dưới dạng *.Frm.
- **Save Project**: Lưu đề án thành tập tin *.vbp.
- **Save Project as**: Lưu đề án thành một tập tin mới *.vbp.
- **Print**: Thực hiện in Form chương trình.
- **Print Setup**: Định dạng trang in cho Form.
- **Make …exe**: Dịch một chương trình ra tập tin thi hành .exe
- **Make Project Group**: Nhóm nhiều đề án lại thành một nhóm.
- **Exit**: Thoát khỏi Visual Basic

7. Biên dịch chương trình thành file *.exe

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

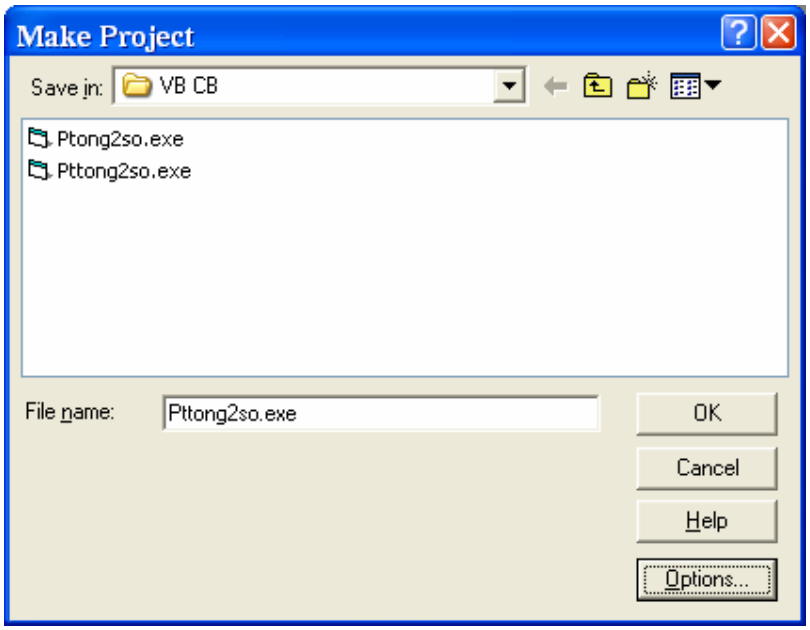
– Bước 1: Chọn Form mở đầu cho ứng dụng bằng cách: từ **menu Project** chọn **Project Properties**, một hộp thoại xuất hiện:



Hình 1-7. Hộp thoại Project Properties

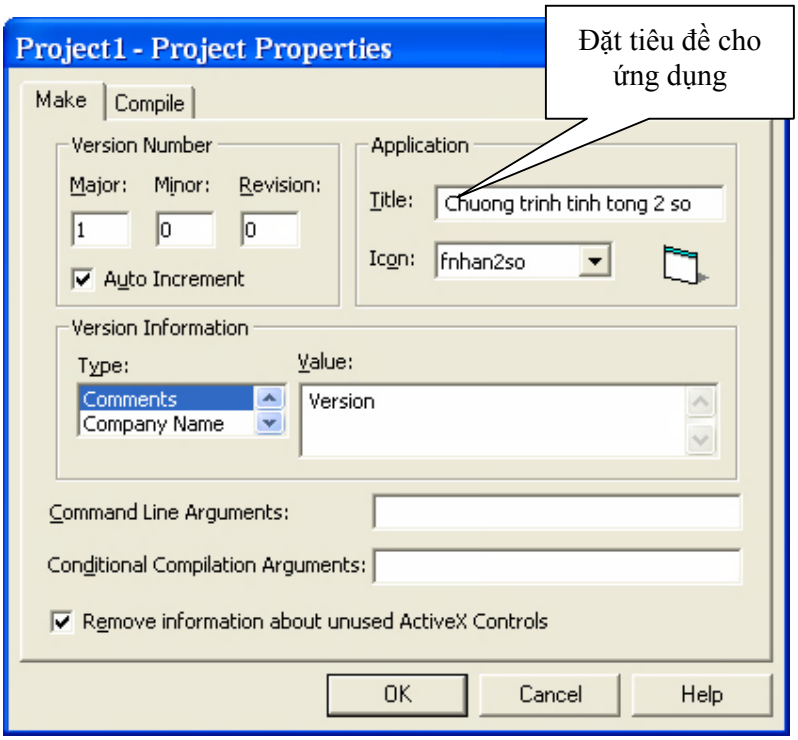
– Chọn **Tab General**, chọn Form khởi động ứng dụng trong combo box **Startup Object.** Ví dụ chọn form ftong2so.

– Bước 2: Từ menu File, chọn Make ... EXE... Một hộp thoại xuất hiện cho phép nhập vào tên của tập tin thực thi (chỉ cần gõ tên tập tin, VB sẽ tự động thêm phần mở rộng .EXE)



Hình 1-8. Hộp thoại Make Project

– 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.



Hình 1-9. Hộp thoại Project Properties – Đặt tiêu đề và phiên bản cho ứng dụng

– Cuối cùng, nhấn OK để trở về hộp thoại Make Project và nhấn OK để tạo file *.exe cho ứng dụng.

Chương 2
Biểu mẫu và một số điều khiển thông dụng

1. Các khái niệm cơ bản.

- **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 và các sự kiện.

- **Thuộc tính**: Các đặc trưng của một điều khiển tạo nên dáng vẻ của điều khiển đó.
- **Phương thức**: Các điều khiển có thể thực thi một số tác vụ nào đó, các tác vụ này được định nghĩa sẵn bên trong các phương thức (còn gọi là chương trình con: hàm, thủ tục), người lập trình có thể gọi thực thi các phương thức này nếu cần.
- **Sự kiện**: là hành động của người dùng tác động lên ứng dụng đang thực thi.

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 nhấp chuột vào button, lúc đó button nhận biết được sự kiện này; hay như textbox nhận biết được sự kiện bàn phím tác động lên nó.

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

- Lập trình sự kiện:

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

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

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

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

- 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 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 và 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.

2. Biểu mẫu (Form)

2.1. Khái niệm

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

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

2.2. Thuộc tính

- **Name**: thuộc tính này là một định danh nhằm xác định tên của biểu mẫu là gì? 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 hiển thị trên thanh tiêu đề của biểu mẫu, nhất là khi biểu mẫu thu nhỏ lại.
- **WindowState**: xác định biểu mẫu sẽ có kích thước bình thường (Normal=0), hay Minimized (=1), Maximized (=2).
- **Font**: xác lập Font cho biểu mẫu. Thuộc tính này sẽ được các điều khiển nằm trên nó thừa kế. Tức là khi ta đặt một điều khiển lên biểu mẫu, thuộc tính Font của điều khiển ấy sẽ tự động trở nên giống y của biểu mẫu.
- **BorderStyle**: xác định dạng của biểu mẫu.

2.3. Phương thức

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

2.4. **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.

3. **Các bước xây dựng một chương trình**

Để xây dựng một chương trình ứng dụng cần thực hiện theo các bước sau đây:

– Bước 1: Phân tích bài toán

Là quá trình tìm hiểu bài toán, xác định các dữ kiện nhập, dữ kiện xuất và đi tìm một giải thuật thích hợp nhất. Bước này cần thực hiện trên giấy cho rõ ràng để tạo thói quen lập trình tốt.

– Bước 2: Thiết kế giao diện

Người lập trình phải thiết kế giao diện thích hợp cho việc nhập, xuất dữ liệu, cần chú ý đến cách trang trí, cách bố trí, thứ tụ, màu sắc, …

– Bước 3: Thiết kế chương trình

Là bước viết chương trình dựa trên giải thuật đã xây dựng ở bước 1, chạy thử chương trình để kiểm tra, phát hiện các lỗi đặc biệt và sửa chữa.

– Bước 4: **Cải tiến**

Đây là bước hoàn thiện chương trình ở mức độ cao hơn.

4. Một số điều khiển thông dụng

4.1. Nhãn (Label)

Nhãn (còn gọi là Label) thường được dùng để vẽ những chuỗi ký tự hằng trên Form nhằm tạo ra các màn hình giao tiếp với người dùng. Các thuộc tính quan trọng liên quan đến đối tượng nhãn bao gồm:

Thuộc tính	Ý nghĩa
Name	Tên của nhãn. Khi mới tạo sẽ tự động có tên là Label1,…
Caption	Chuỗi ký tự nội dung
AutoResize	Tự động thay đổi kích thước khi chuỗi nội dung vượt quá kích thước hiển thị
ForeColor	Màu chữ
BackColor	Màu nền
Font	Kiểu chữ
WordWrap	Tự động xuống dòng khi chuỗi nội dung vượt quá độ rộng label

4.2. Hộp văn bản (Textbox)

Đối tượng hộp văn bản được sử dụng để hiển thị dữ liệu kết quả của các xử lý hay dùng để cho phép người sử dụng nhập liệu vào hệ thống. Ngoài những thuộc tính định dạng như màu chữ, màu nền,… thuộc tính **Text** là thuộc tính thường được sử dụng đối với điều khiển này. Thuộc tính này cho phép chúng ta truy xuất nội dung của hộp văn bản.

4.3. Nút lệnh (command button)

Đối tượng **nút lệnh** là điều khiển được dùng để thực hiện các xử lý của chương trình. Nút lệnh chỉ có một vài thuộc tính thường dùng, đó là:

Thuộc tính	Ý nghĩa
Name	Tên của nút lệnh
Caption	Chuỗi ký tự hiển thị trong nút lệnh
Enabled	Mờ hay sáng nút lệnh
Visible	Ẩn hay hiện nút lệnh
Cancel	Nút sẽ được chọn khi phím Esc được nhấn. Chỉ có một nút duy nhất trên màn hình Form có thuộc tính này là True
Default	Nút sẽ được chọn khi phím Enter được nhấn. Chỉ có một nút có thuộc tính Default là True

Ngoài những thuộc tính nêu trên, nút lệnh còn có phương thức và biến cố liên quan đó là:

Phương thức **SetFocus**: Di chuyển con trỏ hiện hành đến đối tượng nút lệnh.

Biến cố **Click**: Biến cố phát sinh khi nút lệnh được nhấn.

5. Tạo và chạy chương trình

Để tạo một chương trình ứng dụng trong VB, chúng ta cần lần lượt các bước sau:

- Bước 1: Phân tích bài toán
- Bước 2: Thiết kế giao diện
  - Vẽ các điều khiển lên màn hình Form
  - Đặt tên, giá trị những thuộc tính cần thiết cho điều khiển trên Form.
- Bước 3: Thiết kế chương trình
  - Thêm lệnh cho các thủ tục xử lý biến cố.

Sau khi xây dựng hoàn tất chương trình theo ba bước trên, có thể chạy và kiểm tra lỗi chương trình bằng cách nhấn phím F5 hay nhấn chuột tại nút ▶ trên thanh công cụ Toolbar.

- Bước 4: Cải tiến chương trình

Ví dụ minh họa:

Viết chương trình cho phép nhập vào hai số, sau đó tính và xuất ra tổng của hai số vừa nhập vào.

Hình 2-1. Giao diện chương trình tính tổng 2 số khi thực thi

Yêu cầu: Khi nhập vào số thứ nhất, số thứ 2 và click vào nút Cộng thì kết quả sẽ hiển thị ở textbox còn lại; click vào nút Tiếp tục sẽ xóa hết kết quả trên 3 textbox và đặt con trỏ vào ô số thứ nhất. Click vào nút Thoát sẽ thoát khỏi ứng dụng.

Hình 2-2. Giao diện chương trình tính tổng 2 số khi thiết kế.

Bước 1: Phân tích

Để thực hiện được yêu cầu bài toán, cần hai textbox để nhập liệu cho 2 số và một textbox để hiển thị giá trị tổng của hai số.

Bước 2: Thiết kế giao diện

Vẽ giao diện và đặt thuộc tính cho các điều khiển

Control	Name	Caption
Form	fTong	Tinh tong 2 so
CommandButton	cmdTinh	Tính
	cmdTiep	Tiếp
	cmdThoat	Thoát
TextBox	txtSo1	
	txtSo2	
	txtTong	

Bước 3: Thiết kế chương trình

Thêm các lệnh cho các thủ tục xử lý biến cố

```
Private Sub cmdTiep_Click()
  txtSo1.Text = ""
  txtSo2.Text = ""
  txtTong.text = ""
  txtSo1.SetFocus
End Sub
Private Sub cmdTinh_Click()
```

```
  txtTong.Text = Val(txtSo1.Text) + Val(txtSo2.Text)
End Sub
Private Sub cmdThoat_Click()
  Unload Me
End Sub
```

Bước 4: Cải tiến chương trình

Chương 3
Các phép toán và kiểu dữ liệu cơ bản

1. Các phép toán và các ký hiệu

1.1. Phép gán

Đây là toán tử cơ sở của hầu hết các ngôn ngữ lập trình. Toán tử dùng để gán giá trị cho các biến có kiểu dữ liệu cơ sở trong VB là dấu (=). Cú pháp chung lệnh gán có dạng sau:

<tên biến> = <biểu thức>

Biểu thức ở phần bên phải của cú pháp trên có thể là một giá trị hằng, một biến hay một biểu thức tính toán. Khi đó, VB sẽ thực hiện việc tính giá trị của *biểu thức* trước rồi sau đó mới gán giá trị có được cho biến. Ví dụ dòng lệnh gán sau đây sẽ tăng giá trị biến k thêm 1:

k = k + 1

Thông thường, giá trị của *biểu thức* và *biến* trong cú pháp lệnh gán phải cùng kiểu dữ liệu, tuy nhiên chúng ta vẫn có thể gán biểu thức số vào một biến kiểu chuỗi. Trong trường hợp này, VB sẽ tự động đổi giá trị biểu thức thành chuỗi sau đó mới gán vào biến.

Với các biến có kiểu dữ liệu tổng quát, để gán giá trị cho biến chúng ta phải dùng lệnh Set theo cú pháp dưới đây:

Set <tên biên> = <biểu thức>

1.2. 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

1.3. Các phép toán luận lý

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

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

1.4. Các phép toán so sánh

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

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

1.5. Phép &

Đây là toán tử cơ sở dùng để nối các chuỗi dữ liệu lại với nhau. Ví dụ trong dòng lệnh dưới đây

s = “Visual” &” “& “Basic”

biến chuỗi s sẽ có giá trị là “Visual Basic”.

Tương tự như lệnh gán chuỗi, khi chúng ta nối chuỗi với các biểu thức, VB sẽ tự động thực hiện việc chuyển kiểu dữ liệu chuỗi thành số trước rồi sau đó mới nối. Với dòng lệnh tiếp theo:

s = s & 1

Giá trị của biến s sau lệnh gán sẽ là “Visual Basic 1”

1.6. Phép like

So sánh sự giống nhau giữa chuỗi với Mẫu và cho ra kết quả True hoặc False.

Kết quả = Chuỗi like Mẫu.

Ví dụ: Kt= “Visual Basic” like “Visual Basic” (Kt có giá trị là True)

1.7. Các ký hiệu

Các ký hiệu qui ước có thể dùng trong các biểu thức tính toán.

“ “: rào một chuỗi.

#: rào một chuỗi Date.

%: đại diện cho một nhóm ký tự bất kỳ.

?: đại diện cho một ký tự bất kỳ.

2. Các kiểu dữ liệu cơ bản

Tuỳ theo từng loại ứng dụng, người lập trình sẽ dùng các kiểu dữ liệu khác nhau có sẵn của VB. Ngoài những kiểu dữ liệu đặc thù cho từng loại ứng dụng, giống như những ngôn ngữ lập trình khác, VB hỗ trợ một tập hợp các kiểu dữ liệu thường dùng bao gồm *các kiểu dữ liệu cơ sở* như kiểu số nguyên, số thực, luận lý, chuỗi,… và các *kiểu dữ liệu tổng quát*.

Tên kiểu	Tiền tố	Hậu tố	Mặc định	Đặc điểm
Byte	by		0	Kiểu dữ liệu nhị phân
Integer	n, i	%	0	Số nguyên 2 byte
Long	l	&	0	Số nguyên 4 byte
Single	f	!	0	Số thực 4 byte
Double	d	#	0	Số thực 8 byte
String	s, str	$	Chuỗi rỗng	Chuỗi các ký tự
Currency	c	@	0	Kiểu số
Boolean	b		No	Luận lý (Yes/No)
Date	dt			Dữ liệu ngày tháng năm
Control	ctl			Đối tượng điều khiển
Object	ob			Đối tượng chung
Variant				

Ngoài các kiểu dữ liệu cơ sở khá quen thuộc có trong bảng trên như kiểu số, chuỗi,… chúng ta còn thấy ba kiểu dữ liệu tổng quát cũng thường được sử dụng đó là **Control**, **Object** và **Variant**.

Một biến có kiểu dữ liệu **Control** sẽ được dùng tương ứng với một đối tượng điều khiển bất kỳ có trên màn hình giao tiếp Form. Đối tượng này có thể là hộp văn bản, nhãn, nút lệnh,… Khởi tạo giá trị cho các biến **Control** (tương tự với **Object**) phải dùng lệnh Set chứ

không thể dùng lệnh gán (=) như các kiểu dữ liệu cơ sở. Đoạn chương trình sau minh hoạ việc khai báo, gián giá trị và thao tác trên một biến control.

```
Dim ctlTextBox As Control

‘Khởi tạo ctlTextBox là hộp txtNoidung

Set ctlTextBox = Me.txtNoidung

‘Di chuyển điểm nháy đến txtNoidung

ctlTextBox.SetFocus

‘Chọn khối phần văn bản của txtNoidung

ctlTextBox.SelStart = 0

ctlTextBox.SelLength = Len(ctlTextBox.Text)
```

Đoạn chương trình trên sẽ khai báo và khởi tạo biến ctlTextBox là hộp văn bản txtNoidung có trên màn hình hiện hành. Sau đó di chuyển điểm nháy đến hộp văn bản này và chọn khối hết toàn bộ nội dung văn bản có trong đó.

Kiểu dữ liệu **Object** được dùng để tham chiếu đến một đối tượng bất kỳ có trong ứng dụng như màn hình giao tiếp (Form), các điều khiển,… Thực chất, kiểu dữ liệu **Object** là một vùng nhớ có kích thước 4 byte chứa địa chỉ của đối tượng mà nó tham chiếu. Đoạn chương trình sau đây cũng có cùng tác dụng như lệnh trên đây:

```
Dim obControl As Object, obForm As Object

Set obForm = Me

Set obControl = obForm.txtNoidung

obControl.SetFocus

obControl.SelStart = 0

obControl.SelLength = Len(obControl.Text)
```

Variant là một kiểu dữ liệu tổng quát có thể đại diện cho một kiểu dữ liệu cơ sở bất kỳ như Integer, Single,… Tuy vậy, để tốc độ chương trình được nhanh hơn cần hạn chế dùng các kiểu dữ liệu tổng quát mà nên dùng các kiểu dữ liệu cụ thể. Ví dụ như trong hai mẫu ví dụ trên có thể dùng các biến có kiểu dữ liệu cụ thể là TextBox hay Form thay vì dùng Control hay Object.

Mặc nhiên khi khai báo biến mà không chỉ ra kiểu dữ liệu thì VB sẽ dựa vào ký tự đặc biệt cuối tên biến (còn được gọi là hậu tố) để xác định kiểu dữ liệu cho biến. Nếu hậu tố của tên biến không là các ký tự đặc biệt như đã được trình bày trong bảng các kiểu dữ liệu thường dùng trên thì biến sẽ có kiểu dữ liệu mặc nhiên là Variant. Ví dụ hai dòng khai báo biến dưới đây là tương đương nhau: trong đó biến m có kiểu Variant, I có kiểu số nguyên Integer và s có kiểu chuỗi String.

Dim m, I, As Integer, s As String

Dim m, i%, s$

3. Biến

3.1. Khái niệm

Trước tiên ta tìm hiểu khái niệm Module.

o 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).

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.

- 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.

3.2. Phân loại biến

3.2.1 Biến toàn cục

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

o Khai báo:

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

3.2.2 Biến cục bộ

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

o Khai báo:

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

o Lưu ý:

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

3.2.3 Biến Module

o ***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.

o Khai báo:

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

Ví dụ:

Private Num As Integer

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

Ví dụ:

Public Num As Integer

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

3.3. Khai báo biến

Có hai chế độ khai báo và sử dụng biến trong VB. Đó là khai báo tường minh và khai báo không tường minh.

3.3.1 Khai báo không tường minh

Trong chế độ khai báo không tường minh, chúng ta không cần phải khai báo biến trước khi sử dụng. Tự bản thân hệ thống VB sẽ cấp phát biến khi gặp một tên biến mới. Ví dụ trong hàm *MySqr* dưới đây, biến TempVal được sử dụng mà chưa khai báo trước.

```
Function MySqr(num)

TempVal = Abs(num)

MySqr = Sqr(TempVal)

End Function
```

Khi đó, hệ thống sẽ tự động tạo biến TempVal khi gặp dòng lệnh này. Đầu tiên, ai cũng cảm thấy thích chế độ khai báo và sử dụng biến không tường minh như thế. Tuy nhiên, chúng ta, những lập trình viên chuyên nghiệp, không nên sử dụng chế độ này vì đôi khi nó sẽ gây ra nhiều lỗi không phát hiện nổi do đánh nhầm tên biến. Thật vậy, cũng với hàm như trên nhưng nếu chúng ta nhập vào như sau:

```
Function MySqr(num)

TempVal = Abs(num)

MySqr = Sqr(TemVal)

End Function
```

Thoạt nhìn có thể nghĩ hai hàm trên đây giống nhau, kỳ thật là kết quả của hàm thứ hai lại luôn là 0. Đó chính là vì biến TempVal đã bị nhập sai ở dòng lệnh thứ 2 là TemVal. Khi ấy, VB sẽ tự động tạo ra một biến mới có tên là TemVal và có giá trị mặc nhiên là 0. Điều này sẽ cho kết quả của hàm luôn là 0. Trong những chương trình phức tạp, có rất nhiều dòng lệnh thì việc phát hiện ra những lỗi như thế là rất khó.

3.3.2 Khai báo tường minh

Để tránh những lỗi chương trình xảy ra do nhập sai tên biến, chúng ta có thể sử dụng chế độ khai báo tường minh. Với chế độ này, mỗi biến sử dụng cần phải được khai báo trước. Những biến nào chưa khai báo, VB sẽ báo lỗi khi thực thi chương trình. Chúng ta có thể sử dụng một trong hai cách dưới đây để sử dụng chế độ khai báo biến tường minh:

Cách 1:

Trong cửa sổ lệnh, đặt dòng lệnh sau đây **`Option Explicit`** ở đầu phần **Declarations** của màn hình giao tiếp (**Form**), lớp (**Class**) hay thư viện (**Module**).

Cách 2:

Chọn **Tools\Options\Editor** và sau đó chọn **Require Variable Declaration**. Từ thời điểm này trở đi, các màn hình lớp hay thư viện được tạo ra sẽ được mặc nhiên là có sẵn dòng lệnh Option Explicit trong phần Declaration. Với các màn hình giao tiếp, lớp hay thư viện đã được tạo trước đó, chúng ta sẽ phải tự thêm vào dòng lệnh này như cách 1.

Tuỳ theo phạm vi biến cần sử dụng, chúng ta có thể dùng các cấu trúc lệnh sau để khai báo biến. Để khai báo biến **cục bộ** của một thủ tục, hàm, màn hình (Form) hay thư viện chúng ta có thể dùng cú pháp:

Dim Tên_biến [As Kiểu dữ liệu]

Để khai báo các biến **toàn cục** cho toàn bộ ứng dụng. Các biến toàn cục thường được khai báo trong một thư viện.

Puclic Tên_biến [As Kiểu_dữ_liệu]

Tên biến là một chuỗi ký tự thoả các điều kiện sau:

- Bắt đầu bằng ký tự. Tuỳ thuộc vào kiểu dữ liệu của biến, người lập trình thường dùng các ký tự trong bộ ký pháp Hungary làm các ký tự đầu (tiền tố) cho các tên biến. Các tiền tố này sẽ giúp nhận biết một biến có kiểu dữ liệu là gì trong quá trình lập trình. Ví dụ với biến Socong có kiểu dữ liệu số nguyên thường được đặt tên là nSocong. Phần dưới đây sẽ trình bày các tiền tố trong bộ ký pháp Hungary thường được dùng.
- Các ký tự có trong tên biến chỉ có thể là các ký tự chữ cái, ký tự số hay ký tự (_). Tuy nhiên, VB cũng cho phép ký tự cuối cùng của tên biến (hậu tố) là ký tự đặc biệt (xác định kiểu dữ liệu) như ký tự %, #, $... (Xem thêm phần Các kiểu dữ liệu).
- Tên biến dài không quá 255 ký tự.
- Không trùng với các tên biến khác trong cùng phạm vi khai báo như thủ tục, hàm (*Sub*, *Function*), màn hình (*Form*), thư viện (*Module*).
- Không được trùng với các từ khóa của Visual Basic.

Ví dụ dòng lệnh sau khai báo hai biến nSocong va fDongia

Dim nSocong As Integer, fDongia As Single

Trong quá trình hoạt động mỗi biến sẽ có một kiểu dữ liệu nào đó. Kiểu dữ liệu sẽ quy định các giá trị sẽ được lưu trữ trong biến.

4. Hằng

4.1. Khái niệm

Giống như tên gọi, **hằng** là đại lượng có giá trị không thể thay đổi trong quá trình thực hiện chương trình.

4.2. Khai báo hằng

Chúng ta có thể dùng hằng để thay thế những giá trị không gợi nhớ trong chương trình. Ví dụ, thay vì dùng giá trị khó hiểu 3.1416 trong các lệnh tính chu vi, diện tích một hình tròn chúng ta có thể khai báo một hằng với tên gợi nhớ là Pi bằng 3.1416 và sau đó dùng hằng Pi này để tính chu vi và diện tích hình tròn.

Để khai báo một hằng, chúng ta dùng cấu trúc sau:

Const Tên_hằng [As <kiểu dữ liệu>] = <Biểu thức>

Ví dụ:

Const A = 5

Const B As Single = A/2

Đoạn lệnh trên định nghĩa hai hằng số, hằng số A có giá trị là 5, hằng số B kiểu số thực và có giá trị là 2.5.

Để phân biệt với các hằng kiểu số, các giá trị hằng chuỗi phải được biểu diễn trong cặp ký tự ‘ ‘ hay “ “ và hằng kiểu ngày tháng phải được đặt trong cặp ký tự # #.

Const TenDV = “Trung Tam Tin Hoc – DHKHTN”

Const NgayBatDau = #10/24/86#

5. Mảng

- Mảng là tập hợp các phần tử có cùng một kiểu dữ liệu và được chứa trong một biến.
- 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.
- Phần mảng sẽ được đề cập chi tiết ở môn Lập trình nâng cao.

6. Cú pháp lập trình

Ngoài các cú pháp lệnh, hàm, phép toán, khi viết chương trình cần tôn trọng cú pháp lập trình sau:

- Mỗi lệnh phải viết trên một dòng bất kể ngắn hay dài, không được xuống dòng khi chưa hết lệnh.
- Muốn viết nhiều lệnh trên một dòng phải phân cách các lệnh bằng dấu hai chấm (:).
- Dòng lệnh có màu đỏ là dòng lệnh sai cần sửa lỗi.

Chương 4
Các lệnh và hàm cơ bản

1. Lệnh rẽ nhánh

1.1. Lệnh If

o Một dòng lệnh:

If <điều kiện> Then <*dòng lệnh*>

o Nhiều dòng lệnh:

If <điều kiện> Then

Các dòng lệnh

End If

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.

o 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
```

1.2. Lệnh Select Case

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

Select Case <biểu thức kiểm tra>

Case <Danh sách kết quả biểu thức 1>

[Khối lệnh 1]

Case <Danh sách kết quả biểu thức 2>

[Khối lệnh 2]

.

.

.

[Case Else

[Khối lệnh n]]

End Select

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

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

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

Toán tử Is & To

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

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

Ví dụ:

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

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

2. Lệnh lặp

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

2.1. Lệnh For

2.1.1 For ... Next

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

For <biến đếm> = <giá trị đầu> **To** <giá trị cuối> [**Step** <bước nhảy>]

[khối lệnh]

Next

Biến đếm, giá trị đầu, giá trị 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 giá trị đầu phải lớn hơn giá trị 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

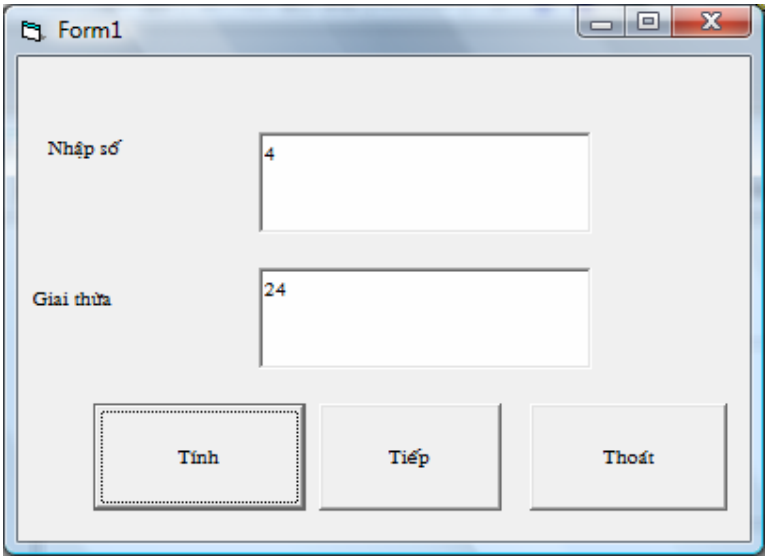
MsgBox Screen.Fonts(I)

Next

End Sub

Ví dụ: Tính N!

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



Hình 4-1. Giao diện chương trình tính giai thừa

o Bước 2: Sự kiện Command1_Click được xử lý như sau:

```
Private Sub cmdTinh_Click()
    Dim i As Integer, gt As Long, n As Integer
    n = Val(txtSo.Text)
    gt = 1
    For i = 2 To n
        gt = gt * CLng(i)
    Next
    txtgt.Text = gt
```

```
End Sub
```

Lưu dự án và chạy chương trình ta được kết quả như hình trên.

2.1.2 For Each ... Next

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

For Each <phần tử> **In** <nhóm>

<khối lệnh>

Next <phần tử>

Lưu ý:

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

- Phần tử trong mảng chỉ có thể là biến Variant.

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

2.2. Lệnh Do

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

Kiểu 1:

Do While <điều kiện>

<khối lệnh>

Loop

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

Kiểu 2:

Do

<khối lệnh>

Loop While <điều kiện>

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

Kiểu 3:

Do Until <điều kiện>

<khối lệnh>

Loop

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

Kiểu 4:

Do

<khối lệnh>

Loop Until <điều kiện>

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

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

Dim i As Integer

i = 2

Do While (i <= Sqr(N)) **And** (N Mod i = 0)

i = i + 1

Loop

If (i > Sqr(N)) And (N <> 1) **Then**

MsgBox Str(N) & " la so nguyen to"

Else

MsgBox Str(N) & " khong la so nguyen to"

End If

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

2.3. Lệnh While

Tương tự vòng lặp Do...While, nhưng ta khôing thể thoát vòng lặp bằng lệnh Exit.

Vì vậy, vòng lặp kiểu này chỉ thoát khi biểu thức điều kiện sai.

While <điều kiện>

<khối lệnh>

Wend

3. Các lệnh và hàm cơ bản

3.1. Lệnh End

Dùng để kết thúc chương trình

Cú pháp: End

3.2. Lệnh Exit

Để thoát khỏi cấu trúc ta dùng lệnh Exit, Exit for cho phép thoát khỏi vòng For, exit Do cho phép thoát khỏi vòng lặp Do, exit sub cho phép thoát khỏi Sub, exit function thoát khỏi Function.

Cú pháp: Exit For | Do|Sub|Function.

Ví dụ: Đây là ví dụ minh học một dạng thoát khỏi vòng lặp Do không điều kiện.

```
Do
        …
        Exit Do
        …
Loop
```

3.3. Lệnh Msgbox

MsgBox <Thông báo> [, <Loại thông báo> [, Tiêu đề]]

Trong cú pháp sử dụng này, thành phần **Thông báo** chính là chuỗi nội dung sẽ hiển thị của lệnh.

Giá trị của thành phần **Loại thông báo** sẽ quy định hình ảnh và những nút sẽ hiển thị trong thông báo.

Các hằng số liên quan đến hình ảnh được hiển thị gồm:

vbQuestion 

vbCritical 

vbInformation 

vbExclamation 

Hằng số quy định các nút sẽ hiển thị gồm: vbOKOnly, vbOKCancel, vbYesNoCancel, vbYesNo, vbAbortRetryIgnore.

Tiêu đề là chuỗi ký tự sẽ xuất hiện trên thanh tiêu đề của cửa sổ thông báo.

Ví dụ để hiển thị giá trị của biến k chúng ta có thể dùng câu lệnh như sau:

MsgBox "k= " & Format(k, "0.0") & vbCrLf & "Khong hop le! Bien k phai khac 0", vbOKOnly + vbCritical, "Thong bao loi"

3.4. Go Sub … Return

Chuyển điều khiển đến một nhãn trong chương trình và trở về (lệnh rẽ nhánh trở về).

Cú pháp:

GoSub Nhãn

……

……

Nhãn:

Các lệnh trong nhãn

……

Return

Trong đó:

– Nhãn là một thường trình trong chương trình, một chương trình có thể có nhiều thường trình, mỗi thường trình có một Nhãn phân biệt. Nhãn là một tên có độ dài.

– Return: là lệnh đặc biệt cho biết kết thúc một nhãn và thực hiên quay trở về lệnh đứng sau lệnh GoSub

3.5. Goto

Được dùng cho bẫy lỗi.

On Error Goto ErrorHandler

Khi có lỗi, chương trình sẽ nhảy đến nhãn ErrorHandler và thi hành lệnh ở đó.

3.6. On Error Goto nhãn

Lệnh On Error dùng trong hàm hay thủ tục báo cho Visual basic biết cách xử lý khi lỗi xảy ra.

On Error GoTo <Nhãn>

Dùng On error Goto 0 tắt xử lý lỗi

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.

3.7. Các hàm chuyển kiểu

- Cbool(biểu thức): trả ra giá trị Boolean bằng cách chuyển đổi luận lý biểu thức.

  Ví dụ: A = 6; B = 7

  Check = (A = B) (Check = False)

- Cbyte(biểu thức): trả ra số nguyên Byte bằng cách chuyển biểu thức ra Byte.

  Ví dụ: X = 126.234

  N = Cbyte(X) (N = 126)

- Cint(biểu thức): trả ra số nguyên Integer bằng cách chuyển biểu thức ra Integer.

  Ví dụ: X = 12245.323

  M = Cint(X) (M = 12245)

- Clng(biểu thức): trả ra số nguyên Long bằng cách chuyển biểu thức ra Long.

  Ví dụ: MyDouble = 12145.4324

  X = Clng(X) (X = 12145)

- Csng(biểu thức): trả ra số thực Single bằng cách chuyển biểu thức ra Single.

  Ví dụ: MyDouble = 12145.432416934

  X = Csng(MyDouble) (MyDouble = 12145.43242)

- Cdbl(biểu thức): trả ra số thực Double bằng cách chuyển biểu thức ra Double.
- Ccur(biểu thức): trả ra số Curency bằng cách chuyển biểu thức ra Currency.
- Cvar(biểu thức): trả ra giá trị kiểu Variant bằng cách chuyển biểu thức ra Variant.
- Cstr(biểu thức): trả ra Chuỗi bằng cách chuyển biểu thức ra Chuỗi.
- Cvdate(biểu thức): trả ra chuỗi Date bằng cách chuyển biểu thức ra Date.
- Chr(mã ký tự): trả ra một ký tự bằng cách chuyển mã ký tự ra ký tự tương ứng theo bảng mã Ascii. Mã ký tự: là giá trị số từ 0 đến 255

  Ví dụ: C = Chr(65) (C = "A")

- Val(số): trả ra một số chứa trong chuỗi.

  Ví dụ: MyValue = Val("2457") (MyValue = 2457)

  MyValue = Val("2 4 5 7") (MyValue = 2457)

  MyValue = Val("24 and 57") (MyValue = 24)

3.8. Các hàm toán học

- Atn(Số): trả về giá trị Arctangent của số tính theo đơn vị Radians. Giá trị trả về trong khoảng từ -Pi/2 đến Pi/2.
- Cos(số): trả về giá trị Cosine của số tính theo đơn vị Radians. Giá trị trả về trong khoảng từ -1 đến 1.
- Sin(Số): trả về giá trị Sine của số tính theo đơn vị Radians. Giá trị trả về trong khoảng từ -1 đến 1.
- Tan(Số): trả về giá trị Tangent của số tính theo đơn vị Radians.
- Exp(Số): trả về giá trị $e^{S\sigma}$, với hằng số e = 2.718282.
- Log(Số): trả về giá trị Logarithm tự nhiên của số với số >0. (Logarithm của e = 2.718282).
- Sqr(Số): trả về căn bậc hai của số, với số >=0.

Ví dụ: A = Sqr(4) (A = 2)

- Randomize: thực hiện khởi động bộ tạo số ngẫu nhiên.
- Rnd: trả về một số ngẫu nhiên có giá trị <=1 và >=0.
- Abs(Số): trả về giá trị tuyệt đối của số.
- Sgn(Số): trả về một số nguyên cho biết dấu của số.
  - Giá trị trả về = 1: số là số dương.
  - Giá trị trả về = 0: số = 0.
  - Giá trị trả về = -1: số là số âm.

Ví dụ: A = 23.454; B= -34.65

N = Int(A) (N = 23)

M = Fix(A) (M = 23)

X = Int(B) (X = -35)

Y = Fix(B) (Y = -34)

3.9. Các hàm kiểm tra kiểu dữ liệu

- IsDate(biểu thức): trả về giá trị True | False cho biết biểu thức có phải là Date không.

Ví dụ:

MyDate = “February 12, 1969”: YourDate = #2/12/69#: NoDate = “Hello”.

MyCheck = IsDate(MyDate) (MyCheck = True)

MyCheck = IsDate(YourDate) (MyCheck = True)

MyCheck = IsDate(NoDate) (MyCheck = False)

- IsEmpty(biểu thức): trả về giá trị True | False cho biết biểu thức đã được khởi tạo chưa.
- IsNull(biểu thức): trả về giá trị True | False cho biết biểu thức có phải là Null không.
- IsNumeric(biểu thức): trả về giá trị True | False cho biết biểu thức có phải là số không.
- IsArray(tên biến): trả về giá trị True | False cho biết biến có phải là mảng không.
- VarType(tên biến): trả về số nguyên cho biết kiểu dữ liệu của biến.

Giá trị trả về	Mô tả kiểu dữ liệu
0	Empty (chưa khởi tạo)
1	Null
2	Integer
3	Long
4	Single
5	Double
6	Currency
7	Date
8	String
9	OLE Automation Object
10	Error (biến lỗi)
11	Boolean
12	Mảng Variant
13	None OLE Automation Object
17	Byte
8192	Mảng

3.10. **Các hàm thời gian**

3.10.1 Lệnh gán giá trị

Với biến d được khai báo là có kiểu dữ liệu ngày tháng, chúng ta có thể khởi tạo giá trị cho d bằng những lệnh sau:

```
Dim d As Date

'Khoi tao d bang ngay gio hien tai:

d = Now

'Khoi tao d bang ngay hien tai:

d = Date

'Khoi tao d bang gia tri ngay thang:
```

```
d = #12/24/2000#
'Khoi tao d bang 3 gia tri ngay, thang, nam:
d = DateSerial(nam, thang, ngay)
```

3.10.2 Lệnh xử lý ngày tháng

Lấy riêng giá trị ngày của d

Hàm Day(d) As Variant(Integer)

Lấy riêng giá trị tháng của d

Hàm Month(d) As Variant(Integer)

Lấy riêng giá trị năm của d

Hàm Year(d) As Variant(Integer)

Tính thứ trong tuần của ngày d

Hàm WeekDay(d)

(1 = vbSunday, 2 = vbMonday,...)

Cộng giá trị ngày d với k(tháng, ngày, tuần,...)

Hàm DateAdd("Đơn vị", k, d) As Date

(Đơn vị được dùng có thể là:

"d": tương ứng với ngày

"w": tương ứng với tuần

"m": tương ứng với tháng

"yyyy": tương ứng với năm

Ví dụ dưới đây sẽ sử dụng các hàm về ngày tháng trong VB để xác định sinh nhật lần thứ n của bạn là thứ mấy trong tuần.

```
Dim d As Date, d1 As Date, n As Integer
Dim s As String, thu As String
s = InputBox("Nhap ngay sinh cua ban", "Nhap thong tin")
n = InputBox("Nhap n", "Nhap thong tin")
```

```
d = CDate(s)

d1 = DateAdd("yyyy", n, d)

Select Case Weekday(d1)

  Case 1

    thu = "Chu Nhat"

  Case 2

    thu = "Thu Hai"

  ...

  Case 7

    thu = "Thu Bay"

End Select

MsgBox "Sinh nhat thu " & Str(n) & " cua ban la ngay " & thu
```

3.11. Các hàm xử lý chuỗi

3.11.1 Hàm Len

Hàm này dùng để tính chiều dài của một chuỗi nào đó. Cú pháp sử dụng của hàm có dạng sau:

```
dodai = Len(chuoi)
```

trong đó dodai phải là một biến kiểu số nguyên đã được khai báo. Câu lệnh dưới đây sẽ duyệt qua từng ký tự của chuỗi s:

```
Dim I As Integer

For I = 1 to Len(s)

  'Xu ly tren tung ky tu cua chuoi s

  Print Mid(s, I, 1)

Next
```

3.11.2 Hàm InStr

Hàm *InStr* dùng để xem một chuỗi s có chứa chuỗi con s1 hay không. Nếu tìm thấy, hàm sẽ có giá trị là vị trí được tìm. Ngược lại hàm sẽ có giá trị là 0. Cú pháp sử dụng của hàm có dạng sau:

Dim tim As Integer

tim = InStr([vt = 1,] chuỗi s, chuỗi con s1, [tuỳ chọn = vbBinaryCompare]) As Integer

Trong đó:

vt là một thành phần có thể có hay không. Giá trị của thành phần này là vị trí bắt đầu thực hiện việc tìm kiếm trong chuỗi s. Nếu chúng ta không chỉ ra thành phần này khi sử dụng InStr, VB sẽ thực hiện tìm từ đầu chuỗi (**vt là 1**).

Tuỳ chọn tìm cũng là một thành phần có thể dùng hoặc không. Khi được sử dụng thành phần này có thể sẽ là một trong những giá trị sau:

vbTextCompare: Không phân biệt chữ hoa hay thường .

vbBinaryCompare: So sánh có phân biệt hoa thường.

vbUseCompareOption: Dùng chế độ hiện hành được đặt của hệ thống.

Ví dụ:

Dim s As String, s1 As String

s = “Chuong trinh Visual Basic 1”

s1 = “Visual Basic”

If InStr(s, s1, vbTextCompare) > 0 Then

MsgBox “Tim thay s1 trong s”

End If

3.11.3 Lệnh Replace

Lệnh Replace dùng để tìm và thay thế chuỗi ký tự sTim có trong chuỗi s bằng chuỗi thay thế sThayThe. Cú pháp của lệnh có dạng sau:

Replace(s, sTim, sThayThe [, vị trí đầu = 1] [, số lần thay thế = 0] [, tuỳ chọn = vbBinaryCompare]) As String

Mặc nhiên số lần thay thế có giá trị là 0, khi ấy hàm sẽ thay thế tất cả chuỗi sTim bằng sThayThe có trong s. Kết quả trả về là chuỗi đã được thay thế.

3.11.4 Các hàm trích chuỗi

Hàm Left(chuỗi s, n) As String

Hàm Right(chuỗi s, n) As String

Hàm Mid(chuỗi s, bắt đầu, [n]) As String

Trong cú pháp các hàm trên, than số **n** chính là số ký tự cần trích. Với hàm **Mid**, nếu tham số này được bỏ qua thì chuỗi kết quả trả về sẽ được trích từ vị trí bắt đầu đến cuối chuỗi s.

3.11.5 Các lệnh cắt khoảng trắng

Cắt các khoảng thừa bên trái của chuỗi s:

LTrim(chuỗi s)

Cắt các khoảng thừa bên phải của chuỗi s:

RTrim(chuỗi s)

Cắt các khoảng thừa bên trái và bên phải của chuỗi s:

Trim(chuỗi s)

3.11.6 Các hàm định dạng

Đổi chuỗi s thành chuỗi chữ hoa

Hàm UCase(chuỗi s)

Đổi chuỗi s thành chuỗi chữ thường:

Hàm LCase(chuỗi s)

Đổi biểu thức thành dạng chuỗi có định dạng

Hàm Format(<biểu thức s>, chuỗi định dạng)

Ví dụ: hàm Format(10, “0.0”) sẽ trả về chuỗi “10.0”

3.12. Các hàm khác

3.12.1 Hàm MsgBox

Trong trường hợp cần hỏi đáp với người sử dụng, chúng ta có thể dùng hàm MsgBox theo cú pháp:

MsgBox(Thông báo, Loại, Tiêu đề)

Ví dụ:

Dim TraLoi As Integer

TraLoi = MsgBox(Thông báo, Loại, Tiêu đề)

Kết quả trả về trong biến TraLoi sẽ chỉ là số của nút mà người dùng đã nhấn. Có thể dùng chỉ số các nút này là các hằng số vbOK, vbYes, vbCancel.

3.12.2 Hàm InputBox

Hàm InputBox này sẽ hiển thị một hộp thoại để người dùng nhập giá trị cho một biến nào đó của chương trình. Đây là một trong những lệnh nhập xuất cơ sở của VB. Cú pháp của hàm như sau:

InputBox (Thông báo, Tiêu đề) As String

Ví dụ:

Để yêu cầu người sử dụng nhập giá trị cho một biến n trong chương trình chúng ta có thể ra lệnh

n = InputBox(“Nhap gia tri so n”,”Nhap lieu””)

Chương 5
Thủ tục và hàm

1. Thủ tục

1.1. Khái niệm

Thủ tục là một dạng chương trình con cho phép khai báo tập hợp các lệnh tương ứng với một đơn vị xử lý nào đó mà đơn vị xử lý này không có giá trị trả về. Thủ tục có thể có hay không có tham số.

1.2. Phân loại

Thủ tục có thể được chia làm 2 loại: thủ tục sự kiện và thủ tục dùng chung.

Thủ tục sự kiện: là các thủ tục được viết cho một sự kiện của Form hoặc Control. Thủ tục loại này sẽ tự thực hiện khi sự kiện xảy ra.

Thủ tục dùng chung: là các thủ tục được viết ở cấp Module hoặc ở phần General cấp Form. Các thủ tục này có tính tổng quát và được gọi sử dụng từ các thủ tục, hàm khác.

1.3. Cấu trúc một thủ tục

[Private | Public] Sub <tên thủ tục> (các tham số)

Tập hợp lệnh

[Exit Sub]

Tập hợp lệnh

End Sub

Giải thích các từ khóa:

- **Private**: Thủ tục chỉ có thể được gọi thực hiện trong cùng màn hình giao tiếp (form), thư viện (module) hiện hành.
- **Public**: Thủ tục có thể được gọi thực hiện từ một màn hình, thư viện khác. Các khai báo thủ tục không chỉ ra phạm vi là Private hay Public sẽ có phạm vi mặc nhiên là Public.
- **Sub … End Sub**: là cặp từ khoá khai báo bắt đầu và kết thúc một thủ tục.
- **Tên thủ tục**: Cũng giống như tên biến, tên thủ tục là một chuỗi ký tự liên tục không trùng với các đối tượng khác trong cùng phạm vi.

Với các thủ tục xử lý biến cố của một đối tượng nào đó, tên của các thủ tục sẽ do chính VB tạo ra theo quy định tênđốitượng_biếncố().

- **Các tham số**: Danh sách tên các biến "hình thức" (còn thường được gọi là tham số hình thức) được sử dụng để giao tiếp dữ liệu với đơn vị chương trình gọi.
- Khác với các ngôn ngữ lập trình khác, những thủ tục không có tham số trong VB cũng phải được khai báo có cặp ngoặc ().

- **Exit Sub**: Mặc nhiên thủ tục sẽ chấm dứt khi thực hiện đến lệnh End Sub. Tuy nhiên chúng ta cũng có thể dùng lệnh Exit Sub để thoát khỏi thủ tục khi cần thiết..

1.4. Xây dựng một thủ tục

1.4.1 Thủ tục dùng chung

Có 2 trường hợp: cấp Form và cấp Module

Cấp Form:

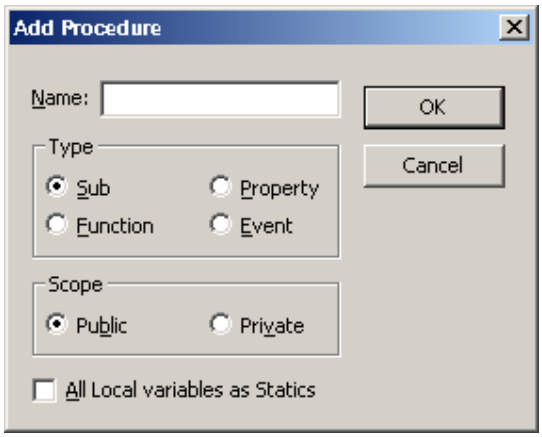
Từ Form ta nhấn F7, xuất hiện khung chương trình, chọn mục General tại hộp Object, nhập vào dòng [Private | Public] [Static] Sub Tên thủ tục [(Danh số các tham số)], sẽ xuất hiện dòng End Sub. Ta thực hiện viết khối lệnh bên trong.

```
Public Sub Vidu()

 ' khoi lenh duoc viet o day

End Sub
```

Hoặc ta có thể chọn **Menu Tools \Add Procedure**, sẽ xuất hiện khung đối thoại sau:



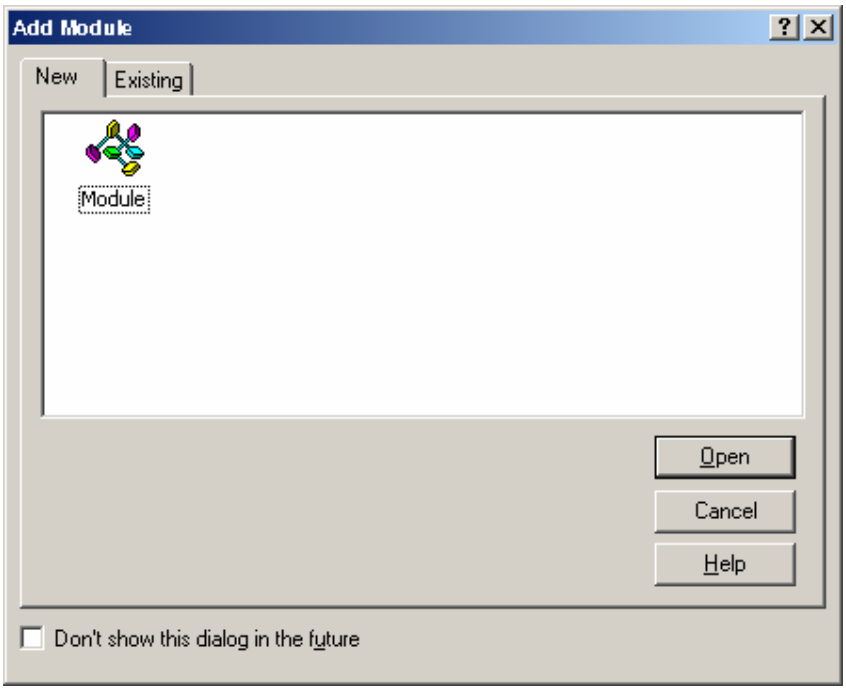
Hình 5-1. Hộp thoại Add Procedure

Chọn □ Sub, quy định □ Public | □ Private, đánh dấu □ All Local variables as Statics để chỉ định (static) cho các biến cục bộ là biến tĩnh, nhập tên thủ tục trong hộp Name, chọn Ok.

Xuất hiện cấu trúc của thủ tục, ta chỉ việc nhập khối lệnh cho thủ tục bên trong Sub… End Sub.

Cấp Module:

Để thêm vào Project một Module chương trình mới (lúc này trên khung Project sẽ có thêm một Module mới) chọn menu Project\chọn Add Module, sẽ xuất hiện hộp thoại sau:



Hình 5-2. Hộp thoại Add Module

Chọn **New** để thêm một **Module** mới, chọn thẻ **Existing** để thêm vào Project một Module đã được xây dựng sẵn.

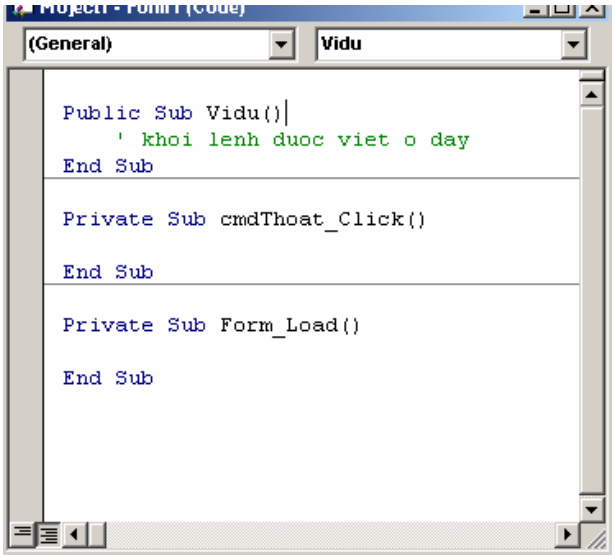
Thao tác thêm mới vào một thủ tục trong Module cũng giống như thao tác thêm mới vào một thủ tục trong Form.

Ví dụ:

```
Private Sub PhucHoi()
  txtSo1.Text = ""
  txtSo2.Text = ""
  txtTong.text = ""
  txtSo1.SetFocus
End Sub
```

1.4.2 Thủ tục sự kiện

Chọn đối tượng cần viết thủ tục, nhấn phím F7 (chọn menu View, Code), sẽ xuất hiện khung chương trình:



Chọn tên một sự kiện cần lập trình tại hộp Proc: sẽ xuất hiện ngăn:

Private Sub ……

End Sub.

Viết khối lệnh bên trong Private Sub … End Sub.

Ghi chú: Để thực hiện viết thủ tục sự kiện cho đối tượng bằng cách Double Click vào đối tượng.

Ví dụ: viết cho nút lệnh thoát

```
Private Sub cmdthoat_Click()
  End
End Sub
```

1.5. Gọi thực hiện thủ tục

Khi đã khai báo một thủ tục, chúng ta có thể gọi thực hiện thủ tục này trong *phạm vi cho phép* theo hai cách sau:

<Tên thủ tục> thamsố1, thamsố2,...

Call <Tên thủ tục> (thamsố1, thamsố2,...)

Ví dụ với thủ tục PhucHoi đã được khai báo trên đây, chúng ta có thể gọi thực hiện như sau:

PhucHoi

Hay: Call PhucHoi()

Tuỳ theo các *tham số hình thức* lúc khai báo, khi gọi thực hiện thủ tục chúng ta phải truyền theo các *thamsố*$_i$ (tham số thực) như trong mô tả cú pháp trên. Các tham số thực này có thể là một giá trị, một biến hay một biểu thức. Với các gọi thực hiện thứ nhất, các thành phần *thamsố*$_i$ nếu có sẽ cách nhau bằng dấu phân cách (,). Với cách gọi thực hiện thứ hai, các tham số thực *luôn phải* được đặt trong dấu ngoặc (). Một điểm cần lưu ý là tên của các tham số hình thức trong khai báo thủ tục và các tham số thực *thamsố*$_i$ không nhất thiết phải giống nhau.

Ví dụ trong khai báo

Sub Dientich (bankinh As Single)

Thì mỗi khi gọi thực hiện, thủ tục sẽ được truyền vào một tham số thực kiểu Single được đại diện bởi một tên chung là bankinh. Khi ấy người lập trình có thể gọi thực hiện với các tham số thực khác như sau:

Call Dientich (3) ‘Tham so thuc la so 3

Call Dientich (r) ‘Tham so thuc la bien r

Trong trường hợp cần gọi thủ tục được khai báo *Public*, từ một màn hình giao tiếp khác ví dụ như Module, chúng ta cần chỉ ra tên của màn hình theo cú pháp:

<Tên màn hình>.<Tên thủ tục>...

2. Hàm

2.1. Định nghĩa

Cũng giống như thủ tục, hàm là một dạng chương trình con có thể nhận vào các giá trị qua danh sách tham số hình thức, thực hiện các lệnh được khai báo, thay đổi các giá trị trong những tham số thực,... Tuy nhiên hàm có giá trị trả về còn thủ tục thì không.

Khác với cú pháp khai báo một thủ tục, khai báo hàm sẽ bắt đầu và kết thúc bằng cặp từ khoá *Function ... End Function*. Ngoài ra, khi khai báo hàm chúng ta còn phải chỉ ra *kiểu dữ liệu trả về* của hàm.

2.2. Cấu trúc một hàm

```
[Private|Public] [Static] Fuction Tên hàm [(Danh số các
tham số)] As Kiểu

  Khối lệnh…

  …….

  Tên hàm = giá trị|biến|biểu thức

End Fuction
```

Giải thích các từ khóa:

As Kiểu: là giá trị của hàm trả ra, có thể là các kiểu sau: Boolean, Byte, Integer, Long, Single, Double, String, Date và Variant.

Tên hàm = giá trị | biến | biểu thức: là lệnh gán đặc biệt dùng gán kết quả tính toán được chứa trong giá trị | biến | biểu thức cho hàm. Nếu không có lệnh này thì hàm không trả ra kết quả nào cả.

End Fuction: là từ khóa cho biết kết thúc hàm.

Ví dụ:

Đây là một hàm tính diện tích Hình Chữ nhật khi biết chiều dài và chiều rộng.

```
Public Function DienTichHinhCN(d As Single, r As
Single) As Single

  Dim dt As Single

  dt = d * r

  DienTichHinhCN = dt

End Function
```

2.3. Xây dựng một hàm

Hàm có thể được xây dựng ở cấp Module hoặc cấp Form:

Nếu hàm được bố trí trong Module thì có thể gọi sử dụng bất kỳ ở mọi thủ tục, hàm khác trong Form kể cả các thủ tục, hàm viết trong các Module khác(trừ trường hợp hàm Private là bị "che").

Nếu hàm được bố trí trong Form thì nó chỉ được gọi sử dụng trong các thủ tục, hàm của Form đó mà thôi.

Cách xây dựng một hàm tương tự như xây dựng thủ tục dùng chung.

2.4. Gọi hàm

Hàm do ta tự xây dựng được sử dụng như các hàm xây dựng sẵn bởi hệ thống. Nó được hiểu như thực hiện một phép toán. Để gọi thực hiện hàm, chúng ta thấy thông thường một hàm khi được sử dụng sẽ thuộc vào những dạng sau:

Tính giá trị và gán cho biến.

Biến = <Tên hàm> (thamsố1, thamsố2,...)

Tham gia vào một biểu thức tính toán.

Biến = Biểu thức có chứa hàm

Trường hợp cần gọi thực hiện hàm nhưng không cần lấy giá trị trả về chúng ta có thể sử dụng cú pháp có dạng:

Call <tên hàm> (thamsố1, thamsố2,...)

Ví dụ:

```
Public Fuction THU (DD As Date) As String
Dim N As Byte, S As String
N = WeekDay(DD)
Select Case (N)
Case 1:
  S = "Chủ Nhật"
Case 2:
  S = "Thứ Hai"
Case 3:
  S = "Thứ Ba"
Case 4:
  S = "Thứ Tư"
Case 5:
  S = "Thứ Năm"
Case 6:
  S = "Thứ Sáu"
Case 7:
  S = "Thứ Bảy"
End Select
THU = S
End Fuction
```

Hàm trên là hàm "công cộng" cho mọi thủ tục, hàm khác; Hàm thực hiện tính thứ của một Date và cho ra một chuỗi cho biết tham số Date là ngày thứ mấy trong tuần. ta thực hiện gọi hàm trên như sau:

DayofWeek = THU(#1/198#)

(DayofWeek = "Thứ Tư" ngày 1/1/98 là ngày thứ tư)

3. Sự kiện

3.1. Giới thiệu

Sự kiện là các tình huống xảy ra trên các đối tượng khi chương trình chạy. Visual Basic cho phép ta có thể lập trình để xử lý, đáp ứng sự kiện xảy ra một cách tức thời.

Các đối tượng khác nhau sẽ có những sự kiện khác nhau. Trong quá trình lập trình cần quan tâm đến các sự kiện có thể xảy ra và tìm cách đáp ứng chúng.

3.2. Các sự kiện của đối tượng

- Sự kiện **Active**: sự kiện xảy ra khi Form bắt đầu trở thành cửa sổ hoạt động.

  Áp dụng cho: Form, MDI Form.

- Sự kiện **Click**: sự kiện xảy ra khi ta Click trong Form hoặc trong Control.

Áp dụng cho: CheckBox, CommandButton, ComboBox, DirListBox, FileListBox, Form, Frame, Image, Label, ListBox, PictureBox, TextBox.

- Sự kiện **DblClick**: sự kiện xảy ra khi ta Double Click trong Form hoặc trong Control.

Áp dụng cho: ComboBox, FileListBox, Form, Frame, Image, Label, ListBox, OLE, OptionButton, PictureBox, TextBox.

- Sự kiện **DeActive**: sự kiện xảy ra khi Form biến mất khỏi màn hình (Form được đóng lại và không phải là ẩn Form bởi phương thức Hide).

  Áp dụng cho: Form, MDI Form.

- Sự kiện **DragDrop**: sự kiện xảy ra khi thực hiện thao tác "rê" và thả một Control trên Form.

Áp dụng cho: CheckBox, ComboBox, CommandButton, DirListBox, DriveListBox, FileListBox, Form, MDI Form, Frame, Hscrollbar, Vscrollbar, Image, Label, ListBox, OptionButton, PictureBox, TextBox.

- Sự kiện **DragOver**: sự kiện xảy ra khi thực hiện thao tác "rê" và thả một Control trên một Control khác.

Áp dụng cho: CheckBox, ComboBox, CommandButton, DirListBox, DriveListBox, FileListBox, Form, MDI Form, Frame, Hscrollbar, Vscrollbar, Image, Label, ListBox, OptionButton, PictureBox, TextBox.

- Sự kiện **GotFocus**: sự kiện xảy ra khi Form hoặc Control có tiêu điểm hoạt động (có con trỏ).

Áp dụng cho: CheckBox, ComboBox, CommandButton, DirListBox, DriveListBox, FileListBox, Form, Hscrollbar, Vscrollbar, ListBox, OptionButton, PictureBox, TextBox.

- Sự kiện **Initialize**: sự kiện xảy ra trong khi ta tạo một Form mới bằng hàm CreateObject.

Áp dụng cho: Form, MDI Form.

- Sự kiện **KeyDown**: sự kiện xảy ra khi ta nhấn phím trên Form hoặc trên Control (nếu nhấn không nhả thì sự kiện KeyDown lập lại nhiều lần).

Áp dụng cho: CheckBox, ComboBox, CommandButton, DirListBox, DriveListBox, FileListBox, Form, Hscrollbar, Vscrollbar, ListBox, OptionButton, PictureBox, TextBox.

- Sự kiện **KeyPress**: sự kiện xảy ra khi ta nhấn và nhả phím trên Form hoặc trên Control.

Áp dụng cho: CheckBox, ComboBox, CommandButton, DirListBox, DriveListBox, FileListBox, Form, Hscrollbar, Vscrollbar, ListBox, OptionButton, PictureBox, TextBox.

- Sự kiện **KeyUp**: sự kiện xảy ra khi nhả một phím vừa nhấn.

Áp dụng cho: CheckBox, ComboBox, CommandButton, DirListBox, DriveListBox, FileListBox, Form, Hscrollbar, Vscrollbar, ListBox, OptionButton, PictureBox, TextBox.

- Sự kiện **LinkClose**: sự kiện xảy ra khi quá trình DDE (Dynamic Data Exchange) kết thúc.

Áp dụng cho: Form, MDI Form, Label, PictureBox, TextBox.

- Sự kiện **LinkError**: sự kiện xảy ra khi quá trình DDE xảy ra lỗi.

Áp dụng cho: Form, MDI Form, Label, PictureBox, TextBox.

- Sự kiện **LinkExecute**: sự kiện xảy ra khi một lệnh được gởi đến ứng dụng đích trong quá trình DDE.

Áp dụng cho: Form, MDI Form.

- Sự kiện **LinkOpen**: sự kiện xảy ra khi quá trình DDE được khởi động.

Áp dụng cho: Form, MDI Form, Label, PictureBox, TextBox.

- Sự kiện **Load**: sự kiện xảy ra khi Form đã được nạp và thể hiện trên màn hình.

Áp dụng cho: Form, MDI Form.

- Sự kiện **LostFocus**: sự kiện xảy ra khi Form hoặc Control vừa mất con trỏ (Form, Control khác nhận con trỏ).

Áp dụng cho: CheckBox, ComboBox, CommandButton, DirListBox, DriveListBox, FileListBox, Form, Hscrollbar, Vscrollbar, ListBox, OptionButton, PictureBox, TextBox.

- Sự kiện **Unload**: sự kiện xảy ra khi Form được gỡ bỏ khỏi màn hình. Ta có thể ngắt sự kiện này bằng cách gán tham số Cancel = -1.

  Áp dụng cho: Form, MDI Form.

- Sự kiện **Terminate**: sự kiện xảy ra khi mọi tham chiếu đến Form được giải phóng khỏi vùng nhớ. Sự kiện này xảy ra sau sự kiện Unload.

  Áp dụng cho: Form, MDI Form.

- Sự kiện **Resize**: sự kiện xảy ra khi Form, PictureBox hiện lần đầu tiên hoặc bị thay đổi kích thước.

  Áp dụng cho: Form, MDI Form, PictureBox.

- Sự kiện **Paint**: sự kiện xảy ra khi một phần trên Form hoặc toàn bộ Form hoặc PictureBox bị thay đổi nội dung.

  Áp dụng cho: Form, PictureBox.

- Sự kiện **QueryUnload**: sự kiện xảy ra trước khi ứng dụng kết thúc và cửa sổ Form chưa đóng. Sự kiện này xảy ra trước sự kiện Unload. Ta có thể ngắt sự kiện này bằng cách gán tham số Cancel một giá trị <>0.

  Áp dụng cho: Form, MDI Form.

- Sự kiện **MouseDown**: sự kiện xảy ra khi ta Click nút chuột bất kỳ trên Form hoặc Control.

Áp dụng cho: CheckBox, CommandButton, DirListBox, FileListBox, Form, Frame, Image, Label, ListBox, MDI Form, OptionButton, PictureBox, TextBox.

- Sự kiện **MouseUp**: sự kiện xảy ra khi ta nhả nút chuột đã nhấn.

Áp dụng cho: CheckBox, CommandButton, DirListBox, FileListBox, Form, Frame, Image, Label, ListBox, MDI Form, OptionButton, PictureBox, TextBox.

- Sự kiện **MouseMove**: sự kiện xảy ra khi ta rê trỏ chuột trên Form hoặc trên Control.

Áp dụng cho: CheckBox, CommandButton, DirListBox, FileListBox, Form, Frame, Image, Label, ListBox, MDI Form, OptionButton, PictureBox, TextBox.

- Sự kiện **Change**: sự kiện xảy ra khi ta thay đổi dữ liệu trên Control.

Áp dụng cho: ComboBox, DirListBox, DriveListBox, Hscrollbar, Vscrollbar, Label, PictureBox, TextBox.

- Sự kiện **ParttternChange**: sự kiện xảy ra khi thuộc tính Parttern của FileListBox bị thay đổi.

  Áp dụng cho: FileListBox.

- Sự kiện **PathChange**: sự kiện xảy ra khi thuộc tính Path của FileListBox bị thay đổi.

  Áp dụng cho: FileListBox.

- Sự kiện **Scroll**: sự kiện xảy ra khi người sử dụng thay đổi "con chạy" trên thanh cuộn.

  Áp dụng cho: Hscrollbar, Vscrollbar.

- Sự kiện **Timer**: sự kiện một khoảng thời gian (theo thuộc tính Interval) trôi qua.

  Áp dụng cho: Timer.

- Sự kiện **Update**: sự kiện xảy ra khi dữ liệu trong OLE bị thay đổi.

  Áp dụng cho: OLE.

- Sự kiện **LinkNotify**: sự kiện xảy ra khi chương trình ứng dụng, Form, Control thực hiện thay đổi dữ liệu trong quá trình DDE nếu thuộc tính LinkMode của Control được quy định là 3.

  Áp dụng cho: Label, PictureBox, TextBox.

4. Truyền tham số

Một đơn vị chương trình con dù là hàm hay thủ tục cũng thường cần được truyền vào những giá trị cần thiết để thực hiện. Việc truyền các giá trị cần thiết khi gọi thực hiện một chương trình con như vậy gọi là truyền tham số. Giống như các ngôn ngữ lập trình khác, truyền tham số trong VB cũng có hai loại là:

Truyền tham trị

Truyền tham biến

4.1. Truyền tham trị

Trong cách truyền tham trị, chỉ có bản sao của tham số thực được truyền cho tham số hình thức. Khi ấy mọi thay đổi giá trị của tham số hình thức thực chất chỉ ảnh hưởng đến bản sao được truyền chứ không thay đổi giá trị của tham số thực. Để truyền tham số theo dạng trị chúng ta phải dùng từ khoá *ByVal* trước khai báo các tham số hình thức tương ứng. Ví dụ hàm So_nto() dưới đây sẽ nhận vào một số nguyên thông qua tham số hình thức m, kiểm tra xem m có phải là số nguyên tố hay không và trả về giá trị *True* hay *False* tương ứng. Tham số hình thức m được khai báo *ByVal* nên việc truyền tham số khi sử dụng hàm So_nto() sẽ theo dạng truyền tham trị.

```
Function Songuyento(ByVal m As Long) As Boolean
```

```
  Dim i As Integer, n As Integer
  If m < 0 Then m = -m    'Kiem tra neu m am
'Kiem tra xem m>0 co phai la so nguyen to
  If m = 1 Then
    Songuyento = False
  Else
    n = m \ 2
    For i = 2 To n
      If (m Mod i = 0) Then Exit For
    Next
    If i <= n Then
      Songuyento = False
    Else
      Songuyento = True
    End If
End Function
```

4.2. Truyền tham biến

Để truyền tham số theo dạng tham biến chúng ta phải dùng từ khoá *ByRef* trước những khao báo tham số hình thức cần thiết. Mặc nhiên các tham số trong VB được truyền theo dạng tham biến, chính vì vậy các tham số hình thức không được khai báo với từ khoá *ByRef* hay *ByVal* sẽ được truyền theo dạng tham biến.

Khi truyền tham biến, mọi thao tác trên tham số hình thức đều tác động trực tiếp lên tham số thực. Có nghĩa là khi gọi thực hiện một ctrìh con có truyền tham biến thì các thay đổi gtrị trên tham số hình thức sẽ làm thay đổi gtrị của tham số thực. Ví dụ với hàm Songuyento() trên đây, nếu được khai báo là:

```
Function Songuyento(ByRef m As Long) As Boolean
  Dim i As Integer, n As Integer
  If m < 0 Then m = -m    'Kiem tra neu m am
'Kiem tra xem m>0 co phai la so nguyen to
```

```
  If m = 1 Then
    Songuyento = False
  Else
    n = m \ 2
    For i = 2 To n
      If (m Mod i = 0) Then Exit For
    Next
    If i <= n Then
      Songuyento = False
    Else
      Songuyento = True
    End If
End Function
```

Khi đó, chúng ta có thể kiểm tra một giá trị k có phải số nguyên tố hay không như sau:

```
  Dim k As Long, kt As Boolean, thongbao As String
  k = -6
  kt = Songuyento(k)
  If kt = True Then
    thongbao = str(k) & " la so nguyen to"
  Else
    thongbao = str(k) & " khong la so nguyen to"
  End If
  MsgBox thongbao
```

Kết quả thực hiện của các dòng lệnh trên đây sẽ là "6 khong la so nguyen to" thay vì "-6 khong la so nguyen to". Kết quả này không hiển thị đúng giá trị k lúc đầu. Đó là vì giá trị ***tham số thực k*** đã bị thay đổi trong quá trình thực hiện hàm Songuyento().

4.3. Tham số tuỳ chọn

Trong thủ tục hay hàm, chúng ta có thể khai báo một tham số là tuỳ chọn hay bắt buộc. Khi một tham số là tuỳ chọn, chúng ta có thể *truyền* hay *không truyền* giá trị cho nó khi gọi thực hiện thủ tục hay hàm.

Đoạn chương trình ví dụ tính giá trị phân số sau sẽ minh hoạ việc dùng tham số tuỳ chọn trong VB:

```
Function Phanso(p As Integer, Optional q As Integer) As Single
  If q = 0 Then
    Phanso = p
  Else
    Phanso = p / q
  End If
End Function
'Tinh tong hai phan so
Dim a As Single, b As Single, tong As Single
a = Phanso(3)
b = Phanso(2, 3)
tong = a + b
MsgBox tong
```

Khi không được truyền giá trị, tham số tuỳ chọn sẽ có giá trị mặc nhiên của kiểu dữ liệu khai báo. Khi ấy, những tham số tuỳ chọn có kiểu Variant sẽ có giá trị là rỗng. Trong trường hợp này, chúng ta có thể dùng hàm IsMissing() để xác định xem một tham số tuỳ chọn kiểu Variant có được truyền giá trị hay không.

Ngoài ra, để tránh giá trị rỗng đối với những tham số tuỳ chọn, chúng ta có thể mô tả giá trị mặc nhiên của tham số tuỳ chọn như trong hàm tính giá trị phân số được định nghĩa lại dưới đây.

```
Function Phanso(p As Integer, Optional q As Integer = 1) As Single
  Phanso = p / q
End Function
```

Chương 6
Thiết Kế BIỂU MẪU DÙNG CÁC ĐIỀU KHIỂN

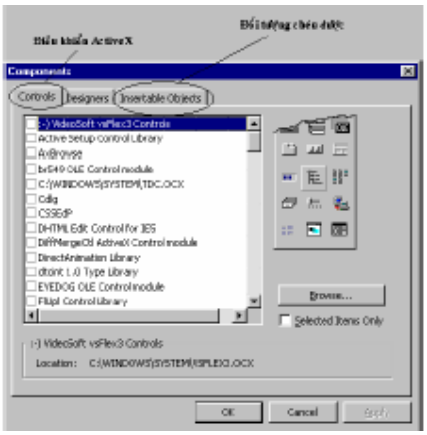
1. Phân loại điều khiển

Có 3 nhóm điều khiển trong Visual Basic:

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

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

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



2. Sử dụng các điều khiển

2.1. Listbox

2.1.1 Khái niệm

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

Biểu tượng (Shortcut) trên hộp công cụ 

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

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

sẽ tự động xuất hiện trên điều khiển. Dưới đây là một ví dụ về danh sách các lựa chọn đơn cột.

2.1.2 Thuộc tính

- **Name**: Đây là tên của danh sách lựa chọn, được sử dụng như một định danh. o **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? o 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?

2.1.3 Phương thức

- **AddItem**: Thêm một phần tử vào List Box. Cú pháp: **<Name>.AddIem(Item As String, [***Index***])**

```
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)

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 ()

mmand1.Value = True

End Sub

2.2. Combobox

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 (Shortcut) trên hộp công cụ 

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

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

- *Drop-down Combo Box*: 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.

2.3. Checkbox

2.3.1 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 

2.3.2 Thuộc tính

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

2.3.3 Sự kiện

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

2.4. Option Button

2.4.1 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.

2.4.2 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.

2.4.3 Sự kiện

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

2.5. Timer

2.5.1 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 vào 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.

2.5.2 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.

2.5.3 Sự kiện

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

2.5.4 Sử dụng điều khiển Timer

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

```
Private Sub Timer1_Timer()

If Label1.Caption <> CStr(Time) Then Label1.Caption =
Time

End If

End Sub
```

Thuộc tính Interval được thiết lập là 500 (tức 0.5 giây). Điều khiển Timer còn hữu ích trong việc tính toán thời gian cho một công việc nào đó, đến một thời điểm nào đó thì ta sẽ khởi tạo một công việc mới hoặc ngưng một công việc không còn cần nữa.

2.6. Hscroll

2.6.1 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.

2.6.2 Thuộc tính

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

2.6.3 Sự kiện

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

```
Private Sub HScroll1_Change()

Text1.FontSize = HScroll1.Value

End Sub
```

2.7. Vscroll

Biểu tượng (Shortcut) trên hộp công cụ 

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

2.8. Picture Box

2.8.1 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 

2.8.2 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.

2.8.3 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.

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.

2.9. Image

2.9.1 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 

2.9.2 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.

2.9.3 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.

2.10. Shape

Biểu tượng Shortcut trên hộp công cụ 

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

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

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