

TRƯỜNG ĐẠI HỌC KINH TẾ QUỐC DÂN
KHOA TIN HỌC KINH TẾ
TS. ĐẶNG QUẾ VINH

LẬP TRÌNH VISUAL BASIC 6.0 CƠ BẢN



NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

TRƯỜNG ĐẠI HỌC KINH TẾ QUỐC DÂN
KHOA TIN HỌC KINH TẾ
TS. ĐẶNG QUẾ VINH

LẬP TRÌNH VISUAL BASIC 6.0 CƠ BẢN



NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

LỜI NÓI ĐẦU

Visual Basic là một trong những ngôn ngữ lập trình thuộc dòng Visual Studio của MicroSoft. Đây là một ngôn ngữ lập trình đa năng, được dùng để phát triển phần mềm giải nhiều loại bài toán khác nhau. Visual Basic rất mạnh trong lập trình cơ sở dữ liệu. Ngôn ngữ lập trình được sử dụng để lập trình trong Access và trong Excel chính là Visual Basic. Visual Basic là ngôn ngữ lập trình dễ học, dễ hiểu và dễ sử dụng. Vì vậy số người sử dụng Visual Basic ngày càng nhiều hơn và Visual Basic đang được giảng dạy trong nhiều trường đại học và cao đẳng tin học ở nước ta.

Cuốn sách này viết cho những người bắt đầu học lập chương trình bằng ngôn ngữ Visual Basic. Quá trình lập trình bằng Visual Basic được trình bày từ đầu. Độc giả không cần phải có kinh nghiệm và kiến thức về lập trình từ trước. Các khái niệm và kỹ thuật lập trình cơ bản, như hằng, biến, biểu thức... đều được trình bày và giải thích đầy đủ. Các ví dụ trong sách được chọn lọc và thử nghiệm cẩn thận.

Nội dung cuốn sách bao gồm 14 chương.

Chương 1 giới thiệu về quá trình ra đời và phát triển của Visual Basic.

Chương 2 trình bày cách thiết kế và thực hiện một chương trình đơn giản trong Visual Basic.

Chương 3 trình bày các thuộc tính chung nhất của các đối tượng của Visual Basic.

Chương 4 nghiên cứu các kiểu dữ liệu của Visual Basic.

Các chương 5, 6, 7 trình bày các kỹ thuật lập trình cơ bản, như lập trình nhập - xuất dữ liệu, lập trình lựa chọn, lập trình có chu trình.

Chương 8 trình bày cách khai báo và sử dụng mảng.

Chương 9 nghiên cứu cách viết và sử dụng các hàm tự định nghĩa (UDF).

Chương 10 dành cho chủ đề in ấn trong Visual Basic.

Chương 11 nghiên cứu cách tạo lập và xử lý tệp.

Chương 12 nghiên cứu các đối tượng dùng để thiết kế giao diện.

Chương 13 nghiên cứu các đối tượng đồ họa.

Chương 14 trình bày kỹ thuật sử dụng đa biểu mẫu.

Các chương trình ví dụ trong cuốn sách này sẽ dùng phiên bản Visual Basic 6.0 để viết.

Đây là cuốn sách về kỹ thuật lập trình bằng Visual Basic 6.0 cơ bản, do đó một số kỹ thuật lập trình mở rộng, đặc biệt là lập trình cơ sở dữ liệu, không được đề cập đến ở đây.

Hy vọng cuốn sách sẽ là tài liệu tham khảo tốt cho sinh viên, giáo viên khoa tin học của các trường đại học, các trường cao đẳng và cho những người bắt đầu học lập chương trình bằng ngôn ngữ lập trình Visual Basic.

Vì trình độ và khả năng của tác giả có hạn, nên cuốn sách không tránh khỏi những sai sót, nhầm lẫn. Kính mong bạn đọc lượng thứ. Tác giả rất hoan nghênh những ý kiến đóng góp của bạn đọc.

Tác giả

GIỚI THIỆU VISUAL BASIC

Chương này trình bày nguồn gốc ra đời và phát triển của Visual Basic và những đặc điểm của ngôn ngữ lập trình này. Đồng thời chương cũng giới thiệu cách cài đặt Visual Basic, cách khởi động Visual Basic và môi trường phát triển tổng hợp của Visual Basic 6.0.

1.1. TỪ BASIC ĐẾN VISUAL BASIC

Ngôn ngữ BASIC (Viết tắt của Beginner's All- purpose Symbolic Instruction Code) là ngôn ngữ lập trình bậc cao, do John Kemeny và Thomat Kurtz sáng tạo ra vào năm 1964 tại trường đại học Dartmouth, Newhampshire nước Mỹ. BASIC sử dụng các từ tiếng Anh để viết các câu lệnh. BASIC là một ngôn ngữ lập trình bậc cao, vạn năng, đại chúng, dễ học, rất thích hợp với những người không chuyên tin học và mới bắt đầu nghiên cứu tin học. Mọi hệ máy vi tính đều được cài đặt ngôn ngữ BASIC, coi như một phần mềm bắt buộc đi theo máy.

Từ ngày ra đời đến nay BASIC đã luôn luôn được phát triển và đa dạng hoá. Mỗi hãng sản xuất máy vi tính bao giờ cũng xây dựng và cài đặt ngôn ngữ BASIC riêng cho máy của hãng đó. Ngôn ngữ BASIC được ta nghiên cứu trong giáo trình này là ngôn ngữ Microsoft BASIC, do hãng MICROSOFT tạo ra. Ngôn ngữ BASIC của Microsoft cũng không ngừng được hoàn thiện và phát triển, nên có những phiên bản khác nhau, như BASICA, QBASIC...

BASIC là ngôn ngữ lập trình làm việc trong môi trường DOS. Trong môi trường DOS mọi giao tiếp giữa người sử dụng và máy tính điện tử đều được thực hiện theo kiểu văn bản. Trong môi trường giao tiếp này thì cả người và máy tính đều dùng các câu văn bản để giao tiếp với nhau. Chương trình BASIC dùng để giải một bài toán cũng là một văn bản, bao gồm các câu lệnh mà máy tính cần thực hiện để giải bài toán. Chương trình dựa trên nền văn bản thường được gọi là chương trình thủ tục.

Trong môi trường WINDOWS có kiểu giao tiếp người - máy mới, trực quan, thông qua các hình ảnh và các hình vẽ, gọi là GUI (Graphic User Interface: giao diện với người sử dụng theo kiểu đồ họa). Trong môi trường GUI thì cả người và máy tính đều có thể dùng các hình tượng, các hình ảnh để giao tiếp với nhau. Lập trình trong môi trường GUI phức tạp hơn và đòi hỏi các công cụ lập trình cao cấp hơn. Ngôn ngữ lập trình Visual Basic (VB) ra đời nhằm cung cấp cho lập trình viên công cụ tuyệt vời để lập trình trong môi trường WINDOWS.

Ngôn ngữ lập trình Visual Basic có các đặc điểm chính sau đây:

- Kế thừa mọi ưu điểm của ngôn ngữ BASIC, nên quen thuộc với các lập trình viên.
- Cho phép lập trình visual trong môi trường windows.
- Về cơ bản Visual Basic là ngôn ngữ lập trình hướng biến cố.
- Cho phép lập trình hướng đối tượng và lập trình Internet
- Tạo rất nhiều thuận lợi cho việc lập trình nhờ sử dụng các công cụ điều khiển có sẵn trong VB. Nhờ vậy mà việc lập trình trong Visual Basic dễ dàng hơn và đỡ tốn công sức hơn.

- Là công cụ tuyệt vời để lập trình cơ sở dữ liệu.

- Cho phép tạo ra các chương trình WINDOWS độc lập dưới dạng tệp khả thi (*.EXE). Người sử dụng (NSD) có thể chạy (thực hiện) chương trình đó trong môi trường WINDOWS mà không cần có Visual Basic trong hệ thống của họ.

- Visual Basic là đối thủ cạnh tranh đáng gờm của ngôn ngữ lập trình C++.

Năm 1991 phiên bản đầu tiên của ngôn ngữ Visual Basic ra đời. Đó là phiên bản Visual Basic 1.0. Từ đó đến nay Visual Basic không ngừng được phát triển và hoàn thiện. Cho đến nay đã có các phiên bản sau đây của Visual Basic:

Visual Basic 2.0 ra đời năm 1992.

Visual Basic 3.0 ra đời năm 1993.

Visual Basic 4.0 ra đời năm 1995.

Visual Basic 5.0 ra đời năm 1997.

Visual Basic 6.0 ra đời năm 1998.

Visual Basic là một phần của bộ công cụ phát triển phần mềm Visual Studio. Visual Basic 6.0 có 3 bản khác nhau sau đây:

Learning Edition: là phiên bản có quy mô nhỏ, bao gồm môi trường cơ bản để phát triển ứng dụng.

Profesional Edition: Ngoài các chức năng có trong Learning Edition, bản này còn có các chức năng cho phép phát triển các ứng dụng Internet và cơ sở dữ liệu.

Enterprise Edition: Đây là bản đầy đủ nhất của Visual Basic. Bản này bao gồm tất cả các chức năng của hai bản nêu trên và được bổ sung những chức năng mới để phát triển các ứng dụng kinh tế và thương mại rộng lớn tích hợp với SQL Server.

1.2. CÀI ĐẶT VISUAL BASIC

• Các yêu cầu đối với hệ thống

Để có thể chạy được Visual Basic hệ thống máy tính của bạn phải đáp ứng được các yêu cầu sau đây:

- Hệ thống đang sử dụng MS WINDOWS 95 trở lên.
- Bộ vi xử lý từ 486DX trở lên.
- Một ổ đĩa CD.
- Màn hình VGA hoặc màn hình độ phân giải cao được hỗ trợ bởi MS WINDOWS.
- 16 MB RAM trở lên.
- Thiết bị chuột.

• Các bước cài đặt Visual Basic 6.0

Để cài đặt phiên bản Visual Basic 6.0 Enterprise Edition trong bộ Microsoft Visual Studio 6.0 có thể tham khảo một cách cài đặt sau đây:

1. Khởi động WINDOWS
2. Đặt đĩa CD có Visual Basic 6.0 vào ổ đĩa CD.
3. Từ màn hình WINDOWS nháy đúp chuột vào biểu tượng My Computer.
4. Nháy đúp chuột và biểu tượng đĩa CD.
5. Nháy đúp chuột vào biểu tượng Setup.exe để thực hiện chương trình cài đặt.
6. Trả lời các câu hỏi của chương trình Setup để cài đặt các thành phần phụ, sau đó chọn ô Microsoft Visual Basic để cài đặt Microsoft Visual Basic. Máy tính sẽ tự động cài đặt Microsoft Visual Basic.

1.3. KHỞI ĐỘNG VISUAL BASIC

Có nhiều cách khởi động Visual Basic. Sau đây là hai cách thường được sử dụng nhất:

Cách 1: Nháy đúp chuột vào biểu tượng của Visual Basic trên màn hình. Đây là cách đơn giản nhất để khởi động VB. Tuy nhiên, nếu trên màn hình không có biểu tượng của VB thì không thể sử dụng cách này được.

Cách 2:

- Chọn Start;
- Chọn Programs;
- Chọn Microsoft Visual Studio 6.0;
- Chọn Microsoft Visual Basic 6.0;

Chú ý: Các thao tác khởi động Visual Basic theo cách 2 có thể được viết ngắn gọn hơn như sau (bằng cách bỏ từ chọn ở đầu mỗi dòng):

- Start;
- Programs;
- Microsoft Visual Studio 6.0;
- Microsoft Visual Basic 6.0;

Hoặc có thể được viết ngắn gọn hơn nữa, chỉ gồm một dòng như sau:

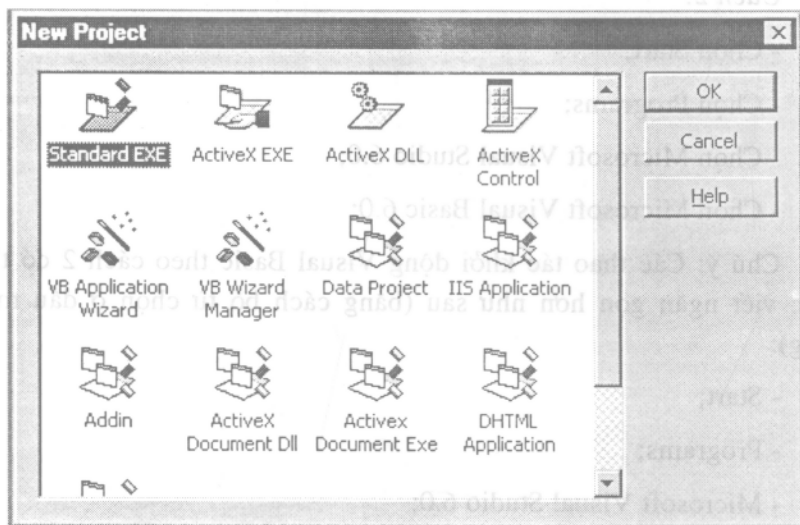
Start, Programs, Microsoft Visual Studio 6.0, Microsoft Visual Basic 6.0

Trong các phần sau, khi đã quen với các thao tác trong VB, ta sẽ thường xuyên sử dụng cách viết này để chỉ ra một dãy các thao tác cần

được thực hiện liên tiếp nhau để đạt được một mục đích nhất định.

Trong hai cách khởi động VB nêu trên thì cách 1 hay được sử dụng hơn.

Sau khi thực hiện các thao tác đó máy tính sẽ khởi động Visual Basic. Khi Visual Basic đã được khởi động thành công, trên màn hình xuất hiện hộp thoại tạo dự án mới (Dialog box for starting a new project) của Visual Basic. Màn hình lúc này có dạng sau đây:



Hình 1.1. Hộp thoại tạo dự án mới của Visual Basic

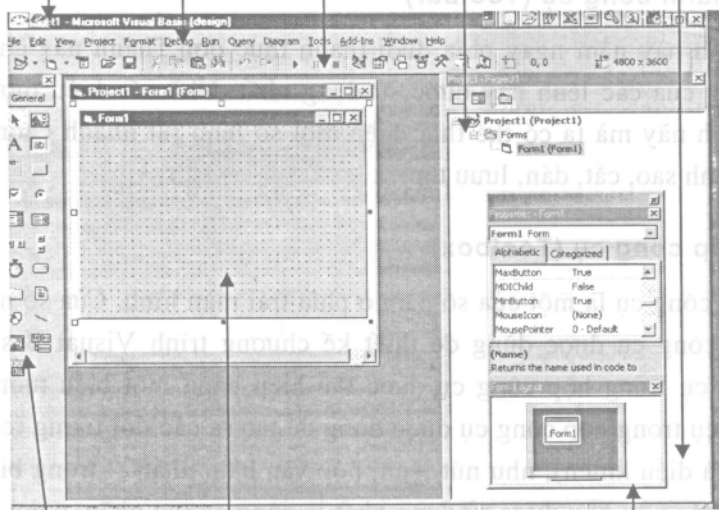
Ta thấy rằng trong hộp thoại này có nhiều biểu tượng. Biểu tượng đầu tiên có tên là Standar.exe. Biểu tượng này đã được Visual Basic chọn sẵn. Đây là biểu tượng cho phép ta khởi tạo một dự án mới trong Visual Basic. Lúc này nếu ta chọn OK thì màn hình giao diện chính của Visual Basic sẽ xuất hiện và ta có thể bắt đầu thiết kế một ứng dụng mới.

1.4. MÔI TRƯỜNG PHÁT TRIỂN TỔNG HỢP CỦA VB

Môi trường phát triển tổng hợp (IDE : Integrated Development Environment) là nơi ta tạo ra các chương trình Visual Basic. Môi trường phát triển tổng hợp của VB được trình bày trong màn hình giao diện chính của VB. Môi trường phát triển tổng hợp của VB gồm:

MÔI TRƯỜNG PHÁT TRIỂN TỔNG HỢP (IDE) CỦA VB

Titlebar Menubar Toolbar Project explorer Properties windows



Toolbox Form window

Form Layout window

Hình 1.2. Màn hình giao diện chính của Visual Basic

– Thanh tiêu đề (Titlebar)

Đây là thanh trên cùng của màn hình giao diện chính của Visual Basic. Thanh này chứa tên dự án đang đọc thiết kế.

– Thanh thực đơn (Menubar)

Đây là thanh thứ hai của màn hình giao diện chính của Visual Basic. Thanh này chứa thực đơn chính của VB. Ta thấy thực đơn chính của VB cũng giống như thực đơn chính của các phần mềm khác của Microsoft, ví dụ như Word hoặc Excel chẳng hạn.

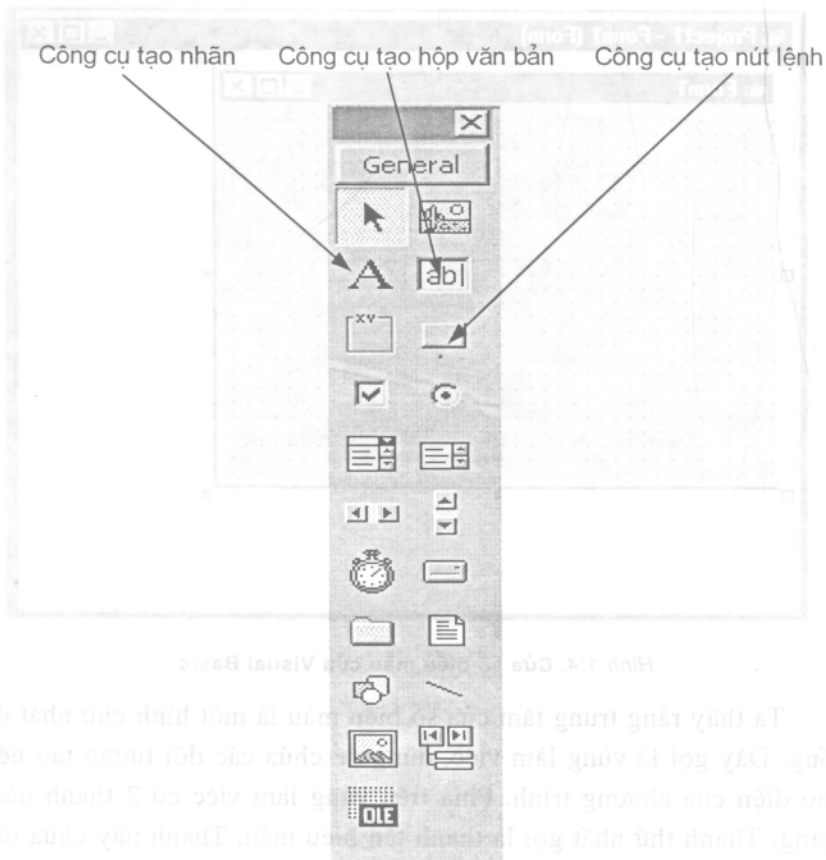
– Thanh công cụ (Toolbar)

Thanh này nằm ngay phía dưới thanh thực đơn. Thanh này chứa biểu tượng của các lệnh hay được sử dụng nhất. Nhờ các biểu tượng trong thanh này mà ta có thể thực hiện một số lệnh rất nhanh chóng, như các lệnh sao, cắt, dán, lưu tệp....

– Hộp công cụ (Toolbox)

Hộp công cụ là một cửa sổ nằm ở phía trái màn hình. Cửa sổ này chứa các công cụ được dùng để thiết kế chương trình Visual Basic. Mỗi công cụ trong hộp công cụ được thể hiện bằng một biểu tượng. Các công cụ trong hộp công cụ được dùng để tạo ra các đối tượng (còn được gọi là điều khiển), như nút lệnh, hộp văn bản, nhãn... trong biểu mẫu. Các công cụ hay được sử dụng nhất là công cụ tạo nhãn, công cụ tạo hộp văn bản và công cụ tạo nút lệnh⁽¹⁾.

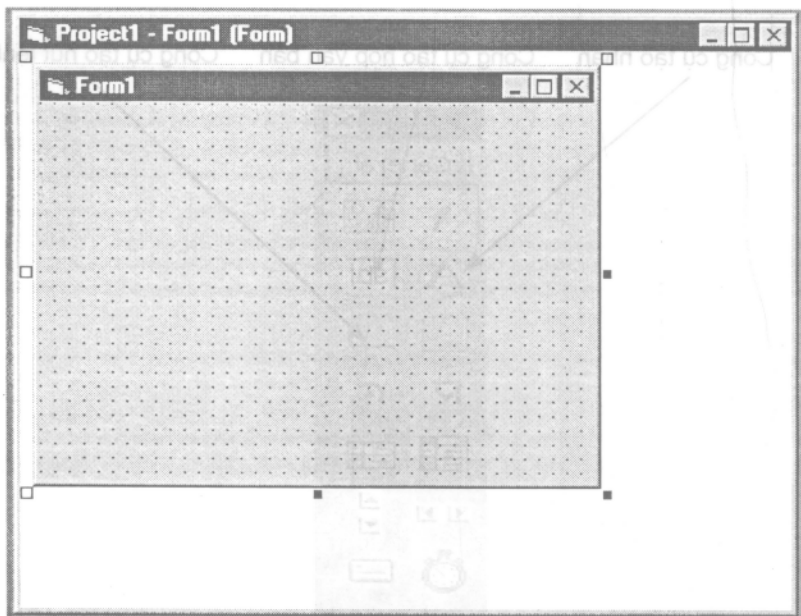
¹ Theo cách nói của lập trình hướng đối tượng thì mỗi công cụ trong hộp công cụ là một lớp. Lớp là một khuôn mẫu được dùng để tạo ra một hoặc nhiều đối tượng, còn được gọi là điều khiển. Visual Basic kết hợp các thuộc tính và phương thức của đối tượng vào một công cụ. Một số thuộc tính và phương thức có thể thấy được. Còn một số thuộc tính và phương thức bị che giấu đi.



Hình 1.3. Hộp công cụ của Visual Basic

– Cửa sổ biểu mẫu (Form Window)

Đây là cửa sổ trung tâm của màn hình giao diện chính của VB. Các công việc thiết kế giao diện của chương trình đều diễn ra trong cửa sổ này. Trong quá trình thiết kế chương trình lập trình viên sẽ thêm vào cửa sổ này nhiều đối tượng khác nhau để tạo nên giao diện của chương trình.

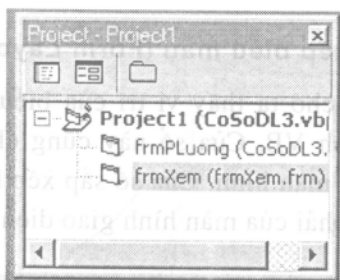


Hình 1.4. Cửa sổ biểu mẫu của Visual Basic

Ta thấy rằng trung tâm cửa sổ biểu mẫu là một hình chữ nhật để trống. Đây gọi là vùng làm việc, dùng để chứa các đối tượng tạo nên giao diện của chương trình. Phía trên vùng làm việc có 2 thanh nằm ngang. Thanh thứ nhất gọi là thanh tên biểu mẫu. Thanh này chứa tên của biểu mẫu (phía trước có tên dự án). Thanh thứ hai gọi là thanh tiêu đề. Thanh này chứa tiêu đề của biểu mẫu.

– Cửa sổ dự án (Project Explorer)

Cửa sổ này chứa những thông tin chung nhất về dự án, như tên dự án và tên của các biểu mẫu có trong dự án này. Thanh tiêu đề của cửa sổ này chứa tên dự án. Bên trong cửa sổ này có tên các biểu mẫu của dự án này. Cửa sổ dự án thường xuất hiện ở góc trên phía phải của màn hình giao diện chính của VB.



Hình 1.5. Cửa sổ dự án của Visual Basic

– Cửa sổ thuộc tính (Properties Window)

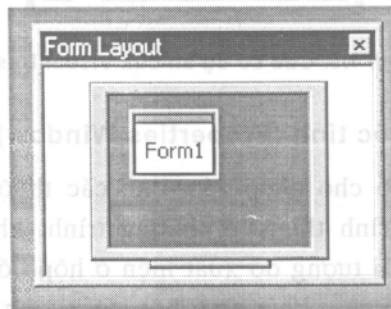
Đây là cửa sổ cho phép xác định các thuộc tính của các đối tượng. Trong quá trình thiết kế chương trình, khi ta chọn một đối tượng thì tên của đối tượng đó xuất hiện ở hộp đối tượng (có ở dòng thứ hai) của cửa sổ này. Phía dưới hộp đối tượng là tất cả các thuộc tính mà đối tượng này có. Cửa sổ thuộc tính thường xuất hiện ở phía phải của màn hình giao diện chính của VB.



Hình 1.6. Cửa sổ thuộc tính của Visual Basic

– Cửa sổ sắp xếp biểu mẫu (Form Layout Window)

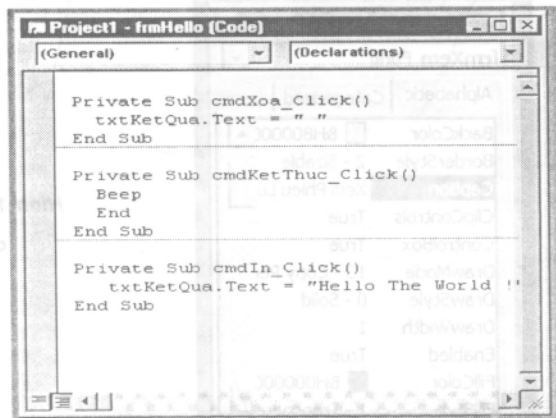
Cửa sổ này chỉ cho ta thấy vị trí của biểu mẫu trong thời gian thực hiện chương trình VB. Cửa sổ này cũng cho phép ta di chuyển biểu mẫu xung quanh màn hình. Cửa sổ sắp xếp biểu mẫu thường xuất hiện ở góc dưới phía phải của màn hình giao diện chính của VB.



Hình 1.7. Cửa sổ sắp xếp biểu mẫu của Visual Basic

– Cửa sổ lệnh (Code Window)

Cửa sổ này chứa phần mã lệnh của chương trình VB. Cửa sổ này không luôn luôn xuất hiện trên màn hình. Cửa sổ lệnh là nơi lập trình viên viết các thủ tục biến cố, các hàm của người sử dụng và các modul độc lập. Khi cửa sổ lệnh xuất hiện nó thường xuất hiện ở vị trí của cửa sổ biểu mẫu.



Hình 1.8. Cửa sổ mã lệnh của Visual Basic

Chú ý: vị trí của hộp công cụ và các cửa sổ của VB nêu ở các phần trên chỉ là tương đối. Người sử dụng có thể dễ dàng sử dụng chuột để di chuyển chúng đến vị trí cần thiết theo ý muốn của mình.

1.5. HAI BƯỚC LẬP TRÌNH TRONG VB

Khái niệm chương trình:

Theo nghĩa thông thường thì chương trình là một dãy các lệnh (hoặc câu lệnh) mà máy tính điện tử có thể hiểu được và thực hiện được để giải một bài toán.

Trong Visual Basic thì chương trình luôn luôn gồm 2 phần là phần giao diện và phần mã lệnh. Phần giao diện của chương trình luôn luôn thấy được trên màn hình, gồm các đối tượng như biểu mẫu, các nhãn, các hộp văn bản, các nút lệnh, các thanh cuộn... Phần giao diện của chương trình còn được gọi là màn hình giao diện. Phần mã lệnh bao gồm các hàm và các thủ tục liên quan đến các đối tượng có trong phần giao diện.

Phần mã lệnh của chương trình VB giống như các chương trình nguồn trong lập trình cấu trúc. Phần mã lệnh của chương trình VB nằm trong cửa sổ lệnh, không luôn luôn xuất hiện trên màn hình, mà chỉ xuất hiện trong những hoàn cảnh nhất định hoặc khi ta muốn điều đó. Vì đặc điểm của chương trình VB như thế nên quá trình lập trình trong VB luôn luôn gồm 2 bước:

- *Bước 1:* Thiết kế giao diện (Visual Programming): dùng các công cụ của VB để tạo ra màn hình giao diện của chương trình. Biểu mẫu là nơi thực hiện thiết kế giao diện của chương trình.

- *Bước 2:* Viết mã lệnh cho chương trình (Code Programming): sử dụng các cấu trúc lập trình của VB để viết các thủ tục biến cố cho các đối tượng, viết các hàm và các thủ tục độc lập của chương trình. Cửa sổ lệnh được sử dụng để viết mã lệnh cho chương trình.

1.6. CÁC ƯU, NHƯỢC ĐIỂM CỦA LẬP TRÌNH TRONG VISUAL BASIC

• Ưu điểm

- Các ứng dụng kiểu giao diện với người dùng bằng đồ họa dễ sử dụng, dễ học.
- Lập trình viên không cần viết mã để tạo ra các GUI
- Visual Basic có nhiều công cụ có sẵn để thực hiện nhiều công việc khác nhau một cách nhanh chóng và dễ dàng.

• Nhược điểm

- Các chương trình thường lớn nên thời gian thực hiện thường lâu.
- Đòi hỏi máy có cấu hình cao, như dung lượng bộ nhớ RAM lớn, tốc độ bộ vi xử lý nhanh, ổ đĩa cứng có dung lượng lớn...).

CÂU HỎI ÔN TẬP

1. Ngôn ngữ lập trình Visual Basic (VB) là gì? Hãy nêu quá trình ra đời và phát triển của VB.
2. Hãy trình bày các đặc điểm của VB.
3. Hộp công cụ là gì? Hãy nêu sự khác nhau giữa thanh công cụ và hộp công cụ.
4. Điều khiển là gì? Hãy nêu tên ba điều khiển hay được sử dụng nhất?
5. Môi trường phát triển của VB gồm những yếu tố nào? Hãy trình bày vắn tắt từng yếu tố đó.
6. Hãy trình bày các loại cửa sổ của IDE của VB.
7. Chương trình VB là gì? Hãy trình bày hai bước lập trình trong VB.
8. Hãy nêu các ưu điểm và nhược điểm của lập trình trong VB.

CHƯƠNG TRÌNH ĐẦU TIÊN

Chương này trình bày cách viết và thực hiện một chương trình Visual Basic đơn giản. Mục đích của chương này là đưa ra cho độc giả một cái nhìn khái quát, tổng thể về quá trình viết và thực hiện một chương trình trong Visual Basic. Nhiều khái niệm và đối tượng của Visual Basic được sử dụng trong chương này chỉ được trình bày và giải thích rất đại cương, ngắn gọn, vắn tắt hoặc chỉ sử dụng mà không giải thích gì cả. Điều này có thể gây những thắc mắc, khó hiểu cho bạn đọc. Xin các bạn cứ yên tâm sau khi nghiên cứu xong các chương tiếp theo của quyển sách này, những thắc mắc, khó hiểu đó sẽ dần dần được giải đáp.

2.1. TẠO MỘT THƯ MỤC CHỨA CHƯƠNG TRÌNH VISUAL BASIC

Để tiện cho việc lưu trữ và tìm kiếm các chương trình Visual Basic ta nên tạo ra một thư mục riêng cho các chương trình này. Giả sử thư mục này có tên là C:\BAITAPVB\.

Sau này, nếu không có gì giải thích thêm, thì ta luôn luôn ngầm định rằng các chương trình Visual Basic do ta tạo ra sẽ được lưu trữ trong thư mục này.

2.2. CÁC ĐỐI TƯỢNG ĐƯỢC SỬ DỤNG TRONG CHƯƠNG TRÌNH

Chương trình VB thường gồm nhiều đối tượng khác nhau. Trong chương 3 ta sẽ định nghĩa và nghiên cứu đầy đủ hơn về đối tượng.

Theo nghĩa hẹp được sử dụng ở đây thì đối tượng là những thành phần được sử dụng để thiết kế và thực hiện chương trình VB. Đối tượng có thể là biểu mẫu, nhãn, hộp văn bản, nút lệnh...

Mỗi đối tượng đều có các thuộc tính. Thuộc tính của đối tượng là những đặc điểm của đối tượng này để phân biệt đối tượng này với đối tượng khác. Ví dụ nút lệnh có các thuộc tính như tên, tiêu đề, chiều cao, căn rộng...

Mỗi đối tượng đều có các phương thức. Phương thức của đối tượng là những hành động đối tượng có thể thực hiện được hoặc những hành động cho phép thực hiện trên đối tượng. Ví dụ nút lệnh có các phương thức như nháy chuột (click) lên nút lệnh,

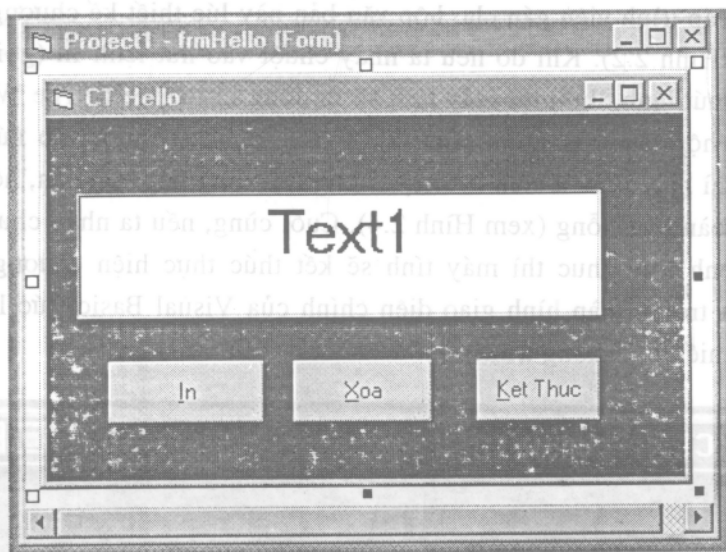
Trong chương này ta sẽ sử dụng 3 đối tượng của VB là biểu mẫu, hộp văn bản, nút lệnh để thiết kế và thực hiện một chương trình VB đơn giản. Đó là chương trình Hello.

2.3. GIỚI THIỆU CHƯƠNG TRÌNH "HELLO"

Chúng ta sẽ gọi chương trình đầu tiên này là chương trình Hello để thân thiện. Chương trình Hello không làm điều gì to tát, phức tạp. Nó chỉ làm mỗi một công việc đơn giản là in trên màn hình dòng kí tự "Hello The World!".

Như đã nói trong mục 1.5, quá trình lập chương trình trong Visual Basic gồm 2 bước, là bước thiết kế giao diện và bước viết mã lệnh. Trong bước thiết kế giao diện lập trình viên dùng các công cụ của Visual Basic để thiết kế màn hình giao diện của chương trình. Ta

cũng đã biết rằng biểu mẫu được sử dụng để thiết kế giao diện của chương trình. Sau khi thiết kế xong, màn hình giao diện của chương trình Hello có dạng sau đây:

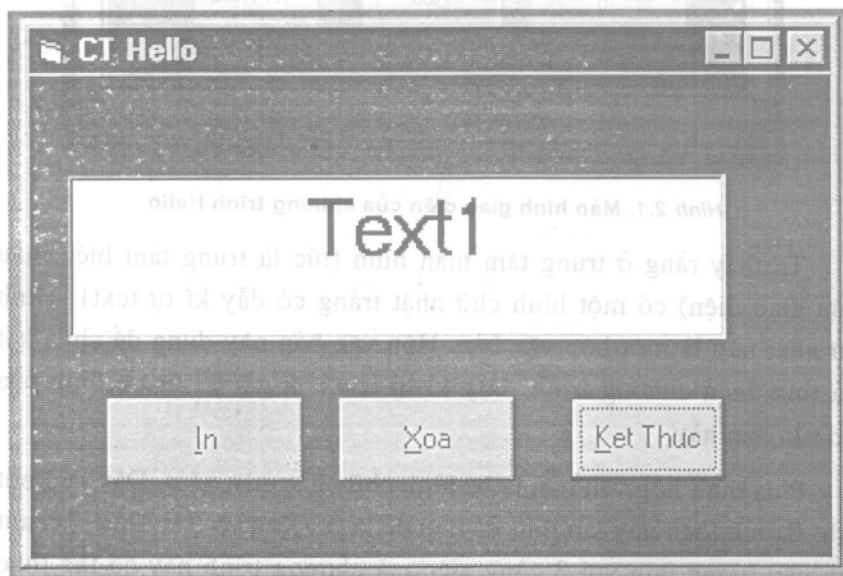


Hình 2.1. Màn hình giao diện của chương trình Hello

Ta thấy rằng ở trung tâm màn hình (tức là trung tâm biểu mẫu chứa giao diện) có một hình chữ nhật trắng có dãy ký tự text1. Hình chữ nhật này là một hộp văn bản. Hộp văn bản này dùng để chứa kết quả thực hiện chương trình. Dãy ký tự text1 là giá trị ngầm định của hộp văn bản này.

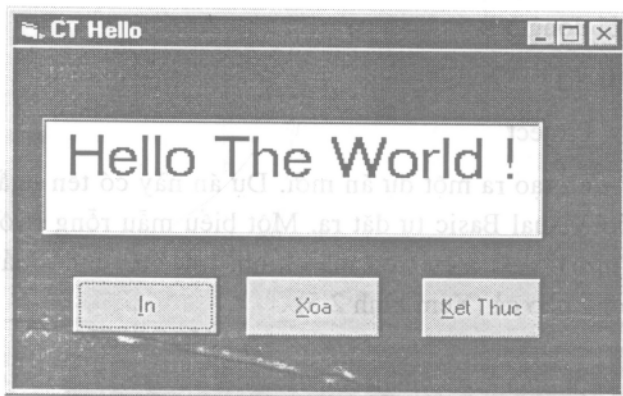
Phía dưới hộp văn bản là ba hình chữ nhật màu xám. Đây là 3 nút lệnh. Ba nút lệnh này lần lượt có tiêu đề là In, Xoa và Ket Thuc. Ba nút lệnh này tương ứng với 3 công việc mà chương trình này có thể thực hiện được là in kết quả (là in một dãy ký tự trong hộp văn bản chứa kết quả), xóa kết quả (là xóa dãy ký tự có trong hộp văn bản chứa kết quả) và kết thúc chương trình.

Khi chạy (tức là thực hiện) chương trình Hello biểu mẫu chứa giao diện của chương trình sẽ xuất hiện trên màn hình. Hộp văn bản lúc đầu chứa dãy ký tự text1, là giá trị ngầm định của hộp văn bản, được lập trình viên gán cho hộp văn bản này lúc thiết kế chương trình (xem Hình 2.2). Khi đó nếu ta nhấn chuột vào nút lệnh In (gọi tắt là dùng nút lệnh “In”) thì máy tính sẽ in dòng ký tự “Hello The World!” trong hộp văn bản (xem Hình 2.3). Nếu ta nhấn chuột vào nút lệnh Xoa thì máy tính sẽ xóa hết mọi thông tin trong hộp văn bản, hộp văn bản hoàn toàn rỗng (xem Hình 2.4). Cuối cùng, nếu ta nhấn chuột vào nút lệnh Ket Thuc thì máy tính sẽ kết thúc thực hiện chương trình Hello, trở về màn hình giao diện chính của Visual Basic (tức là màn hình thiết kế chương trình).

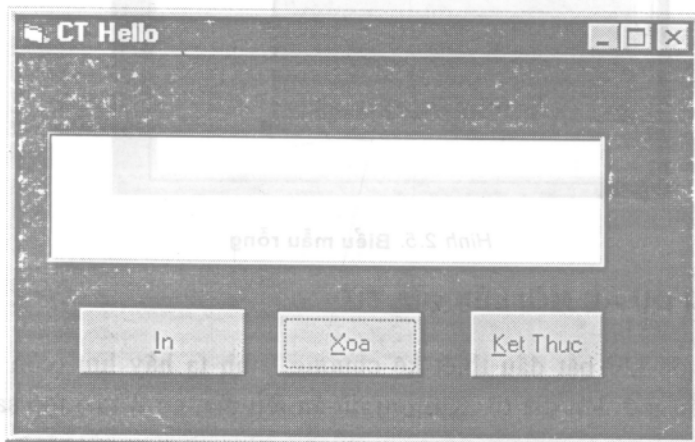


Hình 2.2. Màn hình của chương trình Hello

khí bắt đầu thực hiện chương trình



Hình 2.3. Màn hình của chương trình Hello khi dùng nút lệnh “In”



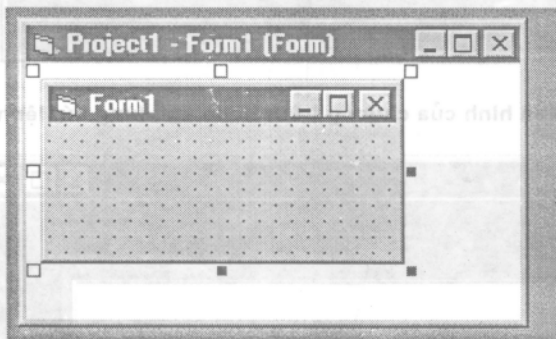
Hình 2.4. Màn hình của chương trình Hello khi dùng nút lệnh “Xóa”

2.4. TẠO MỘT DỰ ÁN MỚI CHO CHƯƠNG TRÌNH HELLO

Công việc đầu tiên của quá trình thiết kế một chương trình trong VB là tạo ra một dự án mới cho chương trình. Để tạo một dự án mới cần phải thực hiện các thao tác sau đây:

- Khởi động VB
- File
- New Project

Máy tính tạo ra một dự án mới. Dự án này có tên ngầm định là Project1, do Visual Basic tự đặt ra. Một biểu mẫu rỗng (trống, trắng) có tên là form1 xuất hiện trên màn hình. Lúc này biểu mẫu chưa có một đối tượng nào cả (Xem hình 2.5) .



Hình 2.5. Biểu mẫu rỗng

2.5. LƯU DỰ ÁN MỚI LÊN ĐĨA TỬ

Trước khi bắt đầu thiết kế chương trình ta hãy lưu (còn gọi là ghi) dự án mới lên đĩa từ. Khi lưu dự án lên đĩa, có 2 loại tệp sau đây được tạo ra:

- Tệp chứa thông tin về dự án, cần cho VB xây dựng dự án hiện hành. Tên tệp này có phần mở rộng là .vbp.
- Tệp chứa thông tin về biểu mẫu. Tên tệp này có phần mở rộng là .frm. Mỗi biểu mẫu có một tệp riêng. Dự án có bao nhiêu biểu mẫu thì sẽ có bấy nhiêu tệp kiểu này tương ứng với các biểu mẫu đó.

Để lưu tệp chứa dự án lên đĩa từ cần phải thực hiện các thao tác

sau đây:

- File
- Save Project as
- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (Hello)
- OK

Để lưu tệp chứa biểu mẫu lên đĩa từ cần phải thực hiện các thao tác sau đây:

- File
- Save Form as
- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (Hello)
- OK

Sau khi thực hiện các thao tác trên, các tệp của dự án mới đã được lưu lên đĩa từ. Bây giờ ta có thể chuyển sang bước thiết kế giao diện cho chương trình Hello.

2.6. THIẾT KẾ GIAO DIỆN CHO CHƯƠNG TRÌNH

Thiết kế giao diện là bước thứ nhất của quá trình thiết kế chương trình VB. Trong bước công việc này ta sẽ sử dụng các công cụ trong hộp công cụ để lần lượt tạo ra các đối tượng (xin nhắc lại: đối tượng còn được gọi là điều khiển) của chương trình. Sau đó ta sẽ xác định các thuộc tính cần thiết cho từng đối tượng.

2.6.1. Thay đổi các thuộc tính của biểu mẫu

Một chương trình VB ít nhất phải có một biểu mẫu. Biểu mẫu có

nhiều thuộc tính. Hai thuộc tính hay được sử dụng nhất là:

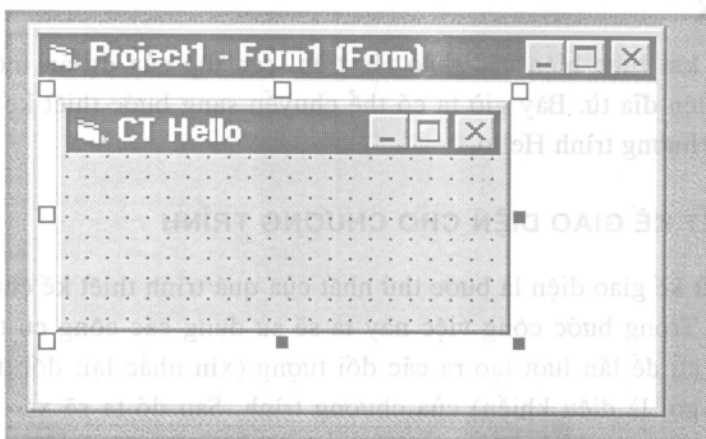
- Caption (tiêu đề): dùng để xác định tiêu đề cho biểu mẫu.
- Name (tên) : dùng để xác định tên cho biểu mẫu.

Như đã thấy ở phần trên, khi tạo ra một dự án, Visual Basic đã tạo ra một biểu mẫu có tên và tiêu đề đều là Form1. Bây giờ ta sẽ thay đổi hai thuộc tính này của biểu mẫu đó cho phù hợp với chương trình cụ thể của ta.

Để xác định thuộc tính Name cho biểu mẫu, cần phải làm như sau:

- Chọn biểu mẫu (bằng cách nhấp chuột vào biểu mẫu).
- Chọn Caption trong cửa sổ thuộc tính.
- Xác định giá trị “CT Hello” cho thuộc tính này (Xem hình 2.6).

Dãy ký tự “CT Hello” sẽ xuất hiện trong thanh tiêu đề của biểu mẫu.



Hình 2.6. Cửa sổ thuộc tính khi xác định thuộc tính Caption cho biểu mẫu

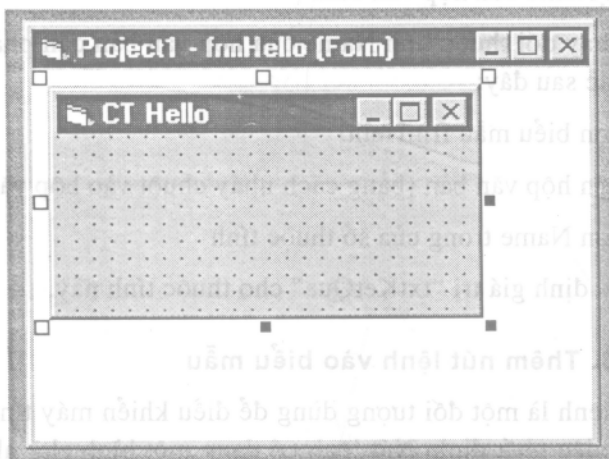
Để xác định thuộc tính Name cho biểu mẫu, cần phải làm như sau:

- Chọn biểu mẫu

- Chọn Name trong cửa sổ thuộc tính

- Xác định giá trị “frmHello” cho thuộc tính này (Xem hình 2.7)

Dãy ký tự “frmHello” sẽ xuất hiện trong thanh chứa tên của cửa sổ biểu mẫu.



Hình 2.7. Cửa sổ thuộc tính khi xác định thuộc tính Name cho biểu mẫu

2.6.2. Thêm hộp văn bản vào biểu mẫu

Chương trình Hello có một hộp văn bản. Hộp văn bản này dùng để chứa kết quả thực hiện chương trình.

2.6.2.1. Dùng công cụ tạo hộp văn bản để thêm hộp văn bản vào biểu mẫu

Để thêm hộp văn bản vào biểu mẫu cần phải thực hiện các thao tác sau đây:

- Nháy đúp chuột vào công cụ tạo hộp văn bản trong hộp công cụ. Một hộp văn bản sẽ xuất hiện trong biểu mẫu.

- Dùng chuột để đưa hộp văn bản đến vị trí cần thiết.
- Dùng chuột điều chỉnh kích thước hộp văn bản để hộp văn bản có được kích thước mong muốn.

2.6.2.2. Thay đổi thuộc tính của hộp văn bản

Hộp văn bản có nhiều thuộc tính. Trong mục này ta chỉ thay đổi thuộc tính Name của hộp văn bản.

Để thay đổi thuộc tính Name của hộp văn bản cần phải thực hiện các thao tác sau đây:

- Chọn biểu mẫu frmHello
- Chọn hộp văn bản (bằng cách nháy chuột vào hộp văn bản)
- Chọn Name trong cửa sổ thuộc tính
- Xác định giá trị “txtKetQua” cho thuộc tính này.

2.6.3. Thêm nút lệnh vào biểu mẫu

Nút lệnh là một đối tượng dùng để điều khiển máy tính thực hiện một công việc nhất định. Nút lệnh có dạng một hình chữ nhật, trên đó có ghi tiêu đề của nút lệnh.

Chương trình Hello có 3 nút lệnh. Như đã thấy trong mục 2.2, các nút lệnh dùng để điều khiển máy tính thực hiện những công việc xác định.

2.6.3.1. Dùng công cụ tạo nút lệnh để thêm nút lệnh vào biểu mẫu

Để thêm nút lệnh vào biểu mẫu cần phải thực hiện các thao tác sau đây:

- Nháy đúp chuột vào công cụ tạo nút lệnh trong hộp công cụ. Một nút lệnh sẽ xuất hiện trong biểu mẫu.
- Dùng chuột để đưa nút lệnh đến vị trí cần thiết.

- Dùng chuột điều chỉnh kích thước nút lệnh để nút lệnh có được kích thước mong muốn.

2.6.3.2. Thay đổi thuộc tính của nút lệnh

Nút lệnh có nhiều thuộc tính. Trong mục này ta chỉ thay đổi thuộc tính Name và thuộc tính Caption của nút lệnh.

Để thay đổi thuộc tính Name của nút lệnh ta phải thực hiện các thao tác sau đây:

- Chọn biểu mẫu frmHello
- Chọn nút lệnh (bằng cách nháy chuột vào nút lệnh).
- Chọn Name trong cửa sổ thuộc tính.
- Xác định giá trị cần thiết cho thuộc tính này.

Để thay đổi thuộc tính Caption của nút lệnh ta làm như sau:

- Chọn biểu mẫu frmHello
- Chọn nút lệnh.
- Chọn Caption trong cửa sổ thuộc tính.
- Xác định giá trị cần thiết cho thuộc tính này.

Sau đây là các thao tác cụ thể để tạo ra các nút lệnh của chương trình Hello

Tạo ra nút lệnh In

Để tạo ra nút lệnh In cần thực hiện các thao tác sau đây:

- Nháy đúp chuột vào công cụ tạo nút lệnh trong hộp công cụ. Một nút lệnh có tên là command1 sẽ xuất hiện trong biểu mẫu.

- Dùng chuột để đưa nút lệnh này đến vị trí phía dưới và thẳng bên trái hộp văn bản txtKetQua đã có trong biểu mẫu.

- Dùng chuột điều chỉnh kích thước nút lệnh này theo ý muốn.

Thay đổi thuộc tính của nút lệnh In

Để thay đổi thuộc tính Name của nút lệnh In ta phải thực hiện các thao tác sau đây:

- Chọn biểu mẫu frmHello
- Chọn nút lệnh command1.
- Chọn Name trong cửa sổ thuộc tính.
- Xác định giá trị “cmdIn” cho thuộc tính này.

Để thay đổi thuộc tính Caption của nút lệnh cmdIn ta thực hiện như sau:

- Chọn biểu mẫu frmHello
- Chọn nút lệnh cmdIn.
- Chọn Caption trong cửa sổ thuộc tính.
- Xác định giá trị “&In” cho thuộc tính này.

Chú ý: kí tự “&” đứng trước kí tự “I” trong giá trị của thuộc tính Caption làm cho phím “I” trở thành phím gõ tắt và kí tự “I” được gạch dưới khi xuất hiện trên nút lệnh cmdIn.

Tạo ra nút lệnh Xoa

Để tạo ra nút lệnh Xoa cần thực hiện các thao tác sau đây:

- Nháy đúp chuột vào công cụ tạo nút lệnh trong hộp công cụ. Một nút lệnh có tên là command1 sẽ xuất hiện trong biểu mẫu.

- Dùng chuột để đưa nút lệnh này đến vị trí phía phải nút lệnh In.
- Dùng chuột điều chỉnh kích thước nút lệnh này theo ý muốn.

Thay đổi thuộc tính của nút lệnh Xoa

Để thay đổi thuộc tính Name của nút lệnh Xoa ta làm như sau:

- Chọn biểu mẫu frmHello
- Chọn nút lệnh command1.
- Chọn Name trong cửa sổ thuộc tính.
- Xác định giá trị “cmdXoa” cho thuộc tính này.

Để thay đổi thuộc tính Caption của nút lệnh ta cần thực hiện các thao tác sau đây:

- Chọn biểu mẫu frmHello.
- Chọn nút lệnh cmdXoa.
- Chọn Caption trong cửa sổ thuộc tính.
- Xác định giá trị “&Xoa” cho thuộc tính này.

Chú ý: kí tự “&” đứng trước kí tự “X” trong giá trị của thuộc tính Caption làm cho phím “X” trở thành phím gõ tắt và kí tự “X” được gạch dưới trên nút lệnh cmdXoa.

Tạo ra nút lệnh Ket Thuc

Để tạo ra nút lệnh Ket Thuc cần thực hiện các thao tác sau đây:

- Chọn biểu mẫu frmHello
- Nháy đúp chuột vào công cụ tạo nút lệnh trong hộp công cụ. Một nút lệnh có tên là command1 sẽ xuất hiện trong biểu mẫu.
- Dùng chuột để đưa nút lệnh này đến vị trí phía phải nút lệnh Xoa.
- Dùng chuột điều chỉnh kích thước nút lệnh này theo ý muốn.

Thay đổi thuộc tính của nút lệnh Ket Thuc

Để thay đổi thuộc tính Name của nút lệnh Ket Thuc ta làm như sau:

- Chọn biểu mẫu frmHello

- Chọn nút lệnh command1.
- Chọn Name trong cửa sổ thuộc tính.
- Xác định giá trị “cmdKetThuc” cho thuộc tính này.

Để thay đổi thuộc tính Caption của nút lệnh này ta làm như sau:

- Chọn biểu mẫu frmHello
- Chọn nút lệnh cmdKetThuc.
- Chọn Caption trong cửa sổ thuộc tính.
- Xác định giá trị “&Ket Thuc” cho thuộc tính này.

Chú ý: kí tự “&” đứng trước kí tự “K” trong giá trị của thuộc tính Caption làm cho phím “K” trở thành phím gõ tắt và kí tự “K” được gạch dưới khi xuất hiện trên nút lệnh cmdKetThuc.

Đến đây ta đã kết thúc bước thiết kế giao diện cho chương trình Hello. Lúc này màn hình giao diện của chương trình có dạng như Hình 2.1. Sau đây là các đối tượng của chương trình Hello và các thuộc tính của các đối tượng đó.

Bảng 2.1. Các đối tượng của chương trình HELLO

Đối tượng	Thuộc tính	Giá trị
Biểu mẫu	Name	FrmHello
	Caption	CT Hello
Nút lệnh	Name	CmdKetThuc
	Caption	&Ket Thuc
Nút lệnh	Name	CmdXoa
	Caption	&Xoa
Nút lệnh	Name	CmdIn
	Caption	&In
Text box	Name	TxtKetQua
	Text	text1

2.7. VIẾT CÁC LỆNH CHO CHƯƠNG TRÌNH

Đây là bước thứ 2 của quá trình lập chương trình trong VB. Visual Basic là ngôn ngữ lập trình theo biến cố (có người gọi là lập trình theo sự kiện), nghĩa là các lệnh được thực hiện để đáp ứng lại một biến cố (event) nào đó. Biến cố có thể là việc ấn một phím, việc nhấp chuột vào một nút lệnh, việc tải một biểu mẫu vào bộ nhớ ... Mỗi đối tượng chỉ có thể đáp ứng được một số xác định các biến cố. Mỗi biến cố cần có một đoạn chương trình tương ứng. Đoạn chương trình này được gọi là một thủ tục-biến cố. Thủ tục biến cố sẽ được thực hiện khi xảy ra biến cố. Thủ tục - biến cố còn được gọi là thủ tục cho ngắn gọn.

Để tạo một thủ tục cho biến cố nhấp chuột vào nút lệnh ta làm như sau:

- Chọn biểu mẫu chứa nút lệnh.
- Nhấp đúp chuột vào nút lệnh (ví dụ nút lệnh cmdIn). Cửa sổ lệnh có mẫu thủ tục - biến cố có dạng sau đây sẽ xuất hiện (Xem hình 2.8).
- Viết các lệnh cần thiết vào chỗ dành cho chúng.
- Đóng cửa sổ lệnh

```
Private Sub cmdIn_Click()  
.....  
.....  
(Chỗ dành để viết các câu lệnh)  
.....  
.....  
End Sub
```

Hình 2.8. Cửa sổ lệnh khi viết thủ tục cho biến cố nhấp chuột vào một nút lệnh

Sau đây là các thao tác cụ thể để tạo ra các thủ tục cho biến cố nhấp chuột vào các nút lệnh của chương trình Hello.

Tạo thủ tục cho biến cố nhấp chuột vào nút lệnh cmdKetThuc

Để tạo thủ tục-biến cố cho nút lệnh cmdKetThuc ta làm như sau:

- Chọn biểu mẫu frmHello.

- Nháy đúp chuột vào nút lệnh cmdKetThuc. Mẫu thủ tục - biến cố cho biến cố nhấp chuột vào nút lệnh cmdKetThuc có dạng sau đây sẽ xuất hiện trong cửa sổ lệnh:

```
Private Sub cmdKetThuc_Click()  
End Sub
```

Thủ tục này sẽ được thực hiện khi người sử dụng nhấp chuột vào nút lệnh cmdKetThuc.

- Trong mục 2.2 ta đã biết rằng khi người sử dụng nhấp chuột vào nút lệnh cmdKetThuc thì máy tính sẽ kết thúc chương trình Hello. Vì vậy ta viết vào thủ tục này câu lệnh sau đây:

```
End
```

Sau khi làm như vậy thủ tục này sẽ có dạng sau đây:

```
Private Sub cmdKetThuc_Click()  
End  
End Sub
```

- Đóng cửa sổ lệnh (bằng cách nhấp chuột vào nút Close của cửa sổ này).

- Lưu dự án lên đĩa từ.

Chạy thử chương trình Hello

Bây giờ ta có thể chạy thử chương trình Hello. Để chạy thử chương trình ta làm như sau:

- Chọn Start trong thanh thực đơn của Visual Basic.
- Chọn Run (hoặc gõ phím F5).

Máy tính bắt đầu thực hiện chương trình Hello. Giao diện của chương trình Hello có dạng Hình 2.1 sẽ xuất hiện trên màn hình. Lúc này chương trình chưa thể thực hiện được một công việc nào cả. Nếu ta nháy chuột vào các nút lệnh cmdIn hoặc cmdXoa thì máy tính không thực hiện công việc nào cả. Sở dĩ như vậy là vì ta chưa viết thủ tục cho biến cố nháy chuột vào các nút lệnh này. Nếu ta nháy chuột vào nút lệnh cmdKetThuc thì máy tính sẽ thực hiện thủ tục cmdKetThuc_Click() và chương trình Hello sẽ kết thúc.

Tạo thủ tục cho biến cố nháy chuột vào nút lệnh cmdXoa

Để tạo thủ tục này ta làm như sau:

- Chọn biểu mẫu frmHello.

- Nháy đúp chuột vào nút lệnh cmdXoa. Mẫu thủ tục cho biến cố nháy chuột vào nút lệnh cmdXoa có dạng sau đây sẽ xuất hiện trong cửa sổ lệnh:

```
Private Sub cmdXoa_Click()  
End Sub
```

Thủ tục này sẽ được thực hiện khi người sử dụng nháy chuột vào nút lệnh cmdXoa.

- Trong mục 2.2 ta đã biết rằng khi người sử dụng nháy chuột vào nút lệnh cmdXoa thì máy tính sẽ xóa các thông tin có trong hộp văn bản txtKetQua. Đó là cơ sở để ta viết vào thủ tục này câu lệnh sau:

```
txtKetQua.Text = ""
```

Sau khi làm như vậy thủ tục này sẽ có dạng sau đây:

```
Private Sub cmdIn_Click()
```

```
txtKetQua.Text = ""  
End Sub
```

- Đóng cửa sổ lệnh.
- Lưu dự án lên đĩa từ.

Chạy thử chương trình Hello với nút lệnh cmdXoa

Bây giờ ta lại chạy thử chương trình Hello. Lần này ta sẽ sử dụng nút lệnh cmdXoa. Ta thực hiện các thao tác sau:

- Chọn Start trong thanh thực đơn của Visual Basic.
- Chọn Run

Giao diện của chương trình Hello sẽ xuất hiện trên màn hình.

- Nháy chuột vào nút lệnh cmdXoa. Máy tính sẽ xóa các kí tự có trong hộp văn bản txtKetQua. Hộp văn bản này trở nên hoàn toàn trắng (Xem hình 24).

- Nháy chuột vào nút lệnh cmKetThuc để kết thúc chương trình.

Tạo thủ tục cho biến cố nháy chuột vào nút lệnh cmdIn

Để tạo thủ tục này ta làm như sau:

- Chọn biểu mẫu frmHello.

- Nháy đúp chuột vào nút lệnh cmdIn. Mẫu thủ tục cho biến cố nháy chuột vào nút lệnh cmdIn có dạng sau đây sẽ xuất hiện trong cửa sổ lệnh:

```
Private Sub cmdIn_Click()  
End Sub
```

Thủ tục này sẽ được thực hiện khi người sử dụng nháy chuột vào nút lệnh cmdIn.

- Trong mục 2.2 ta đã biết rằng khi người sử dụng nháy chuột

vào nút lệnh cmdIn thì máy tính sẽ in dãy kí tự "Hello The World !" trong hộp văn bản txtKetQua. Vì vậy ta cần viết vào thủ tục này câu lệnh sau đây:

```
txtKetQua.Text = "Hello The World !"
```

Sau khi làm như vậy thủ tục này sẽ có dạng sau đây:

```
Private Sub cmdIn_Click()  
    txtKetQua.Text = "Hello The World !"  
End Sub
```

- Đóng cửa sổ lệnh.
- Lưu dự án lên đĩa từ.

Chạy thử chương trình Hello với nút lệnh cmdIn

Bây giờ ta lại chạy thử chương trình Hello. Lần này ta sẽ sử dụng nút lệnh cmdIn. Ta thực hiện các thao tác sau đây:

- Chọn Start trong thanh thực đơn của Visual Basic.
- Chọn Run

Giao diện của chương trình Hello sẽ xuất hiện trên màn hình.

- Nháy chuột vào nút lệnh cmdIn. Dãy kí tự "Hello The World !" sẽ xuất hiện trong hộp văn bản txtKetQua. Hộp văn bản này có dạng như trong Hình 2.3.

- Nháy chuột vào nút lệnh cmKetThuc để kết thúc chương trình.
- Lưu dự án lên đĩa từ.
- Đóng cửa sổ lệnh (bằng cách nháy chuột vào nút Close của cửa sổ này).

Như vậy là ta đã viết xong các thủ tục cho tất cả các nút lệnh của chương trình Hello. Trong cửa sổ lệnh các thủ tục đó có dạng sau đây:

```

Private Sub cmdXoa_Click()
    txtKetQua.Text = " "
End Sub

Private Sub cmdKetThuc_Click()
    End
End Sub

Private Sub cmdIn_Click()
    txtKetQua.Text = "Hello The
World !"
End Sub

```

Trong các thủ tục này ta đã sử dụng 3 câu lệnh của VB. Tác dụng của các câu lệnh đó được cho trong bảng dưới đây.

Bảng 2.2. Các câu lệnh đã được sử dụng:

Câu lệnh	Ý nghĩa
txtKetQua.Text = "Hello The World !"	Đây là câu lệnh gán. Câu lệnh này gán dãy kí tự "Hello The World !" cho thuộc tính Text của hộp văn bản txtKetQua. Điều này làm cho dãy kí tự "Hello The World !" sẽ được in (tức là sẽ xuất hiện) trong hộp văn bản txtKetQua.
txtDisplay.Text = " "	Đây là câu lệnh gán. Câu lệnh này gán dãy kí tự rỗng (tức là dãy kí tự không có một kí tự nào cả) cho thuộc tính Text của hộp văn bản có tên là txtKetQua. Lệnh này làm cho hộp văn bản txtKetQua hoàn toàn trắng, không còn chứa một kí tự nào cả. Điều đó tương đương với việc xoá hộp văn bản.
End	Lệnh này kết thúc chương trình.

2.8. THỰC HIỆN CHƯƠNG TRÌNH HELLO

Đến lúc này thì chương trình Hello của ta đã hoàn toàn sẵn sàng làm việc. Để thực hiện chương trình ta làm như sau:

- Run
- Start

Giao diện của chương trình Hello có dạng Hình 2.1 sẽ xuất hiện trên màn hình.

- Nháy chuột vào nút lệnh In để xem kết quả.
- Nháy chuột vào nút lệnh Xoa để xoá kết quả.
- Nháy chuột vào nút lệnh Ket Thuc để kết thúc chương trình. Máy tính sẽ kết thúc chương trình Hello và trở về màn hình giao diện chính của VB.

Chú ý: Có thể dùng tổ hợp phím ALT_I, Alt_X, Alt_K thay cho việc nháy chuột vào các nút lệnh tương ứng. Các phím I, X, K trong chương trình này được gọi là các phím gõ tắt (short cut) hoặc phím nóng (hot key).

2.9. TẠO RA CHƯƠNG TRÌNH KHẢ THI

Trong mục 2.8 ta đã thực hiện (tức là chạy) chương trình Hello. Trong trường hợp này ta chạy chương trình Hello trong môi trường Visual Basic. Tuy nhiên Visual Basic cho phép ta tạo ra chương trình khả thi cho chương trình Hello.

Để tạo ra chương trình khả thi cần phải thực hiện các thao tác sau:

- Chọn File
- Chọn Make HELLO.exe
- Xác định thư mục chứa tệp này là c:\baitapvb\

- Chấp nhận tên tệp này là HELLO
- Chọn OK

Sau khi làm như vậy máy tính sẽ tạo ra tệp HELLO.exe. Ta có thể chạy chương trình này trong môi trường WINDOWS mà không cần khởi động Visual Basic hoặc ở những máy tính không có Visual Basic.

CÂU HỎI ÔN TẬP

1. Tại sao cần tạo ra một thư mục riêng để chứa các chương trình Visual Basic?
2. Màn hình giao diện của chương trình Hello gồm những đối tượng nào? Các đối tượng đó được dùng để làm gì?
3. Thuộc tính của đối tượng là gì? Thuộc tính có ý nghĩa gì? Công cụ nào của Visual Basic được sử dụng để xác định các thuộc tính của đối tượng.
4. Phương thức của đối tượng là gì? Phương thức dùng để làm gì? Công cụ nào của Visual Basic được sử dụng để tạo ra các phương thức cho các đối tượng.
5. Hai bước công việc để tạo ra một thủ tục - biến cố cho một nút lệnh là gì? Hai bước công việc đó có đặc điểm gì?
6. Những câu lệnh nào đã được sử dụng trong chương trình Hello? Nêu ý nghĩa của từng câu lệnh đó.
7. Hãy sửa chữa chương trình Hello trong chương này để khi người sử dụng nhấp chuột vào nút lệnh cmdIn thì máy tính sẽ in câu chào tình cảm hơn như "Chào các bạn!" hoặc "Chào bạn Mimi xinh đẹp!".

ĐỐI TƯỢNG VÀ THUỘC TÍNH

Trong chương 2 ta đã trình bày ngắn gọn về đối tượng và thuộc tính của đối tượng. Ta cũng đã dùng một số đối tượng của VB để thiết kế và thực hiện chương trình VB đầu tiên. Chương này ta sẽ trình bày đầy đủ hơn về khái niệm đối tượng và ý nghĩa của các thuộc tính thông dụng nhất của các đối tượng. Sau đó sẽ nghiên cứu những đối tượng thông dụng như nhãn, hộp văn bản, nút lệnh....

3.1. KHÁI NIỆM ĐỐI TƯỢNG

Trong thiết kế và thực hiện các chương trình Visual Basic ta luôn luôn phải sử dụng các đối tượng. Vậy đối tượng là gì?

Theo nghĩa thông thường thì đối tượng (viết tắt là đối tượng) là bất cứ cái gì mà con người nhìn thấy và sờ được. Còn ở đây, trong lĩnh vực lập trình Visual Basic, ta dùng từ này theo nghĩa hẹp. Theo nghĩa này thì đối tượng là những cái mà ta sử dụng để thiết kế và thực hiện một chương trình Visual Basic. Đối tượng có thể là một biểu mẫu, một nút lệnh, một hộp văn bản, một nhãn ...

Mỗi đối tượng đều có các thuộc tính. Thuộc tính của đối tượng là những đặc điểm của đối tượng để mô tả đối tượng và phân biệt đối tượng này với đối tượng khác. Ví dụ đối tượng nút lệnh có các thuộc

tính như tên, kích thước, vị trí, tiêu đề... Mỗi đối tượng trong Visual Basic đều có một tập hợp thuộc tính đã quy định trước. Cửa sổ thuộc tính của Visual Basic được dùng để xác định thuộc tính cho các đối tượng.

Mỗi đối tượng đều có thể thực hiện một số hành động hoặc cho phép thực hiện một số hành động trên nó. Phương thức của đối tượng là hành động mà đối tượng có thể thực hiện được hoặc hành động được phép thực hiện trên nó. Mỗi đối tượng trong Visual Basic đều có một tập hợp các phương thức đã quy định trước. Các phương thức của đối tượng được thực hiện nhờ các thủ tục biến cố, các hàm hoặc các module độc lập. Cửa sổ lệnh được dùng để viết các module chương trình cho các phương thức của các đối tượng. Tuy nhiên, có nhiều phương thức của các đối tượng đã được lập trình sẵn trong Visual Basic.

Trong chương trước ta đã sử dụng các công cụ của hộp công cụ để thêm các đối tượng vào biểu mẫu của chương trình Visual Basic.

Chú ý: Trong Visual Basic hai thuật ngữ *đối tượng* (object) và *điều khiển* (control) thường được sử dụng thay thế cho nhau. Điều khiển là những đối tượng được đặt trong biểu mẫu. Mặc dù biểu mẫu cũng là một đối tượng, nhưng nó không phải là một điều khiển. Trong cuốn sách này thuật ngữ đối tượng được dùng để chỉ cả biểu mẫu lẫn các điều khiển (là các đối tượng có trong biểu mẫu).

3.2. MỘT SỐ THUỘC TÍNH THƯỜNG DÙNG CỦA CÁC ĐỐI TƯỢNG

Mỗi đối tượng có một tập hợp các thuộc tính riêng có của mình. Có một số thuộc tính chung, tồn tại ở nhiều loại đối tượng khác nhau. Sau đây sẽ trình bày chi tiết các thuộc tính như vậy.

3.2.1. Thuộc tính Name (Tên)

Thuộc tính này xác định tên cho đối tượng. Tên này được dùng để truy nhập đối tượng trong phần mã lệnh. Mỗi đối tượng đều có tên ngầm định, do VB đặt cho đối tượng khi đối tượng vừa được tạo ra. Ví dụ biểu mẫu có tên ngầm định là *Form1*, nút lệnh có tên ngầm định là *command1*...

Tên đối tượng được đặt theo các qui tắc sau đây:

- Tên có từ 1 đến 40 kí tự, là chữ cái, chữ số và dấu ngang dưới (_).

- Tên không được trùng với từ khoá.

- Tên phải bắt đầu bằng chữ cái và không được chứa dấu cách. Tên đối tượng có thể gồm chữ hoa, chữ thường, hoặc cả chữ hoa lẫn chữ thường. Visual Basic không phân biệt chữ hoa và chữ thường trong tên đối tượng.

- Tên đối tượng phải gợi nhớ. Ví dụ ta có thể đặt tên cho nút lệnh kết thúc chương trình là *Hoa*, nhưng tên *cmdKetThuc* thì tốt hơn vì nó gợi nhớ và hợp với qui ước đặt tên đối tượng của VB.

Ba kí tự đầu tiên của tên đối tượng được gọi là tiền tố (của tên). Visual Basic quy định cho mỗi loại đối tượng một tiền tố xác định. Sử dụng tiền tố của tên cho phép dễ dàng nhận ra loại đối tượng khi biết tên đối tượng. Sau đây là bảng tiền tố của tên các loại đối tượng.

Bảng 3.1. Các tiền tố của tên đối tượng

Tiền tố	Loại đối tượng
cbo	Hộp combo (Combo Box)
chk	Hộp kiểm (Check Box)
cmd	Nút lệnh (Command Button)
dir	Hộp danh sách thư mục (Directory List Box)

Tiền tố	Loại đối tượng
drv	Hộp danh sách ổ đĩa (Driver List Box)
fil	Hộp danh sách tệp (File List Box)
fra	Khung (Frame)
frm	Biểu mẫu (Form)
grd	Khung lưới (Grid)
hsb	Thanh cuộn ngang (Horizontal Scroll Bar)
img	Hình ảnh (Image)
lbl	Nhãn (Label)
lin	Đường thẳng (Line)
lst	Hộp danh sách (List Box)
mnu	Thực đơn (Menu)
opt	Nút tùy chọn (Option Button)
pic	Hộp chứa hình vẽ (Picture)
shp	Hình vẽ (Shape)
tmr	Bộ định thời gian (Timer)
txt	Hộp văn bản (Text Box)
vsb	Thanh cuộn dọc (Vertical Scroll Bar)

Chú ý: Quy định về tiền tố của tên đối tượng là không bắt buộc. Tuy nhiên chúng ta nên tuân theo quy định này.

3.2.2. Thuộc tính Caption (Tiêu đề)

Thuộc tính Caption xác định tiêu đề cho đối tượng. Tiêu đề là một dãy kí tự bất kỳ do người sử dụng xác định. Tiêu đề của đối tượng xuất hiện cùng với đối tượng trong biểu mẫu. Tiêu đề của đối tượng cho phép người sử dụng nhận ra đối tượng trong màn hình giao diện

của chương trình.

Trong tiêu đề, nếu có dấu & trước một kí tự thì kí tự này được gạch dưới và trở thành phím nóng. Ta đã biết điều này khi xác định thuộc tính Caption cho các đối tượng của chương trình Hello.

Trong chương trước, khi thiết kế chương trình Hello, ta đã xác định tiêu đề cho biểu mẫu, cho nút lệnh. Tuy nhiên không phải tất cả các biểu mẫu đều có thuộc tính này. Ví dụ hộp văn bản không có thuộc tính Caption.

3.2.3. Thuộc tính Appearance (hình dạng)

Thuộc tính này xác định khi nào đối tượng có dạng 3 chiều. Thuộc tính này có 2 giá trị là 0 và 1. Giá trị 0 của thuộc tính này làm cho đối tượng có dạng hình phẳng. Giá trị 1 của thuộc tính này làm cho đối tượng có dạng hình trong không gian 3 chiều.

3.2.4. Thuộc tính Enabled (khả ứng)

Thuộc tính này xác định khả năng đáp ứng các biến cố tác động trong thời gian thực hiện chương trình của đối tượng. Thuộc tính này có 2 giá trị là False và True. Nếu thuộc tính này có giá trị True thì đối tượng có thể đáp ứng các biến cố tác động trong thời gian thực hiện chương trình. Nếu thuộc tính này có giá trị False thì đối tượng không có khả năng đó.

3.2.5. Thuộc tính Font (Font chữ)

Thuộc tính này dùng để xác định font cho đối tượng. Các đối tượng chứa văn bản, như hộp văn bản, nhãn, nút lệnh ... đều có thuộc tính này.

Giá trị của thuộc tính này là tên các font chữ đang được sử dụng.

Ta có thể gán cho thuộc tính này tên của font chữ bất kỳ hiện có trong máy tính với một kích thước xác định.

3.2.6. Thuộc tính Height (chiều cao)

Để xác định kích thước các đối tượng Visual Basic dùng đơn vị đo là twip (twip - twentieth of a point : một phần hai mươi point). Một twip có độ dài bằng 1/1440 inch (1 inch = 2,54 cm) hoặc bằng 1/567 cm. Một point có độ dài bằng 1/72 inch.

Ngoài twip ra, Visual Basic còn cho phép lập trình viên sử dụng những đơn vị đo độ dài khác, như inch, cm, kí tự...

Thuộc tính Height xác định chiều cao của đối tượng. Chiều rộng của đối tượng được đo bằng twip. Cũng có thể dùng chuột để xác định chiều cao của đối tượng. Nhưng cách làm này không hoàn toàn chính xác. Chiều cao của đối tượng có thể bị thay đổi trong quá trình thực hiện chương trình.

3.2.7. Thuộc tính Width (chiều rộng)

Thuộc tính Width xác định chiều rộng của đối tượng. Chiều rộng của đối tượng được đo bằng twip. Cũng có thể dùng chuột để xác định chiều rộng của đối tượng. Nhưng cách làm này không hoàn toàn chính xác. Chiều rộng của đối tượng cũng có thể bị thay đổi trong quá trình thực hiện chương trình.

3.2.8. Thuộc tính Left (khoảng cách trái)

Thuộc tính này xác định khoảng cách tính bằng twip từ biên trái của đối tượng đến biên trái của đối tượng bên ngoài chứa đối tượng này.

3.2.9. Thuộc tính Top (khoảng cách trên)

Xác định khoảng cách tính bằng twip từ biên trên của đối tượng

đến biên trên của đối tượng ngoài chứa đối tượng này.

3.2.10. Thuộc tính BackColor (màu nền)

Xác định màu nền của đối tượng. Thuộc tính này có thể nhận màu bất kỳ trong hệ thống màu đang được sử dụng.

3.2.11. Thuộc tính ForeColor (màu chữ)

Xác định màu của chữ viết của đối tượng. Cũng giống như thuộc tính BackColor, thuộc tính này có thể nhận màu bất kỳ trong hệ thống màu đang được sử dụng.

3.2.12. Thuộc tính Visible (thấy được)

Xác định tính thấy được của đối tượng. Thuộc tính này có 2 giá trị là False và True. Nếu thuộc tính này có giá trị True thì đối tượng có thể nhìn thấy được trong thời gian thực hiện chương trình. Nếu thuộc tính này có giá trị False thì đối tượng không nhìn thấy được, tức là bị ẩn trong thời gian thực hiện chương trình.

Bảng 3.2. Tóm tắt một số thuộc tính dùng chung cho nhiều loại đối tượng

Thuộc tính	Ý nghĩa, tác dụng
Appearance	Xác định khi nào đối tượng có dạng 3 chiều.
BackColor	Xác định màu nền của đối tượng
BorderStyle	Xác định kiểu viền đối tượng
Caption	Xác định tiêu đề cho đối tượng.
Enabled	Xác định tính khả ứng của đối tượng.
Font	Xác định font chữ cho đối tượng.

Thuộc tính	Ý nghĩa, tác dụng
ForeColor	Xác định màu chữ của đối tượng
Height	Xác định chiều cao của đối tượng.
Left	Xác định khoảng cách trái.
Name	Xác định tên đối tượng.
Top	Xác định khoảng cách trên.
Visible	Xác định tính thấy được của đối tượng.
Width	Xác định chiều rộng của đối tượng.

Sau đây ta sẽ nghiên cứu kỹ hơn một số đối tượng thường dùng và các thuộc tính quan trọng nhất của các đối tượng đó.

3.3. BIỂU MẪU

3.3.1. Khái niệm biểu mẫu

Biểu mẫu (Form) là một đối tượng hay được sử dụng nhất trong chương trình VB. Biểu mẫu là nơi thiết kế giao diện của chương trình. Mỗi chương trình VB ít nhất phải có một biểu mẫu, và có thể có nhiều biểu mẫu. Có thể nói biểu mẫu là đối tượng trung tâm của chương trình VB. Biểu mẫu thường chứa nhiều đối tượng để tạo nên phần giao diện của chương trình. Khi tạo ra một dự án mới VB luôn luôn tự động tạo ra một biểu mẫu có tên là Form1.

3.3.2. Các thuộc tính của biểu mẫu

Biểu mẫu có đến hơn 50 thuộc tính. Những thuộc tính cơ bản nhất và hay được sử dụng nhất của biểu mẫu được cho trong bảng dưới đây:

Bảng 3.3. Các thuộc tính cơ bản của biểu mẫu

Thuộc tính	Ý nghĩa, tác dụng
Appearance	Xác định khi nào biểu mẫu có dạng 3 chiều.
BackColor	Xác định màu nền của biểu mẫu
BorderStyle	Xác định kiểu đường viền xung quanh biểu mẫu
Caption	Xác định tiêu đề cho biểu mẫu. Tiêu đề ngầm định của biểu mẫu là tên biểu mẫu.
Enabled	Xác định tính khả ứng của các đối tượng có trong biểu mẫu.
Font	Xác định font chữ cho biểu mẫu.
ForeColor	Xác định màu chữ của biểu mẫu
Height	Xác định chiều cao của biểu mẫu.
Left	Xác định khoảng cách từ biên trái của màn hình đến biên trái của biểu mẫu.
MDIChild	Xác định biểu mẫu này có thuộc kiểu cha con không. Nếu thuộc tính này là True thì biểu mẫu này là biểu mẫu MDI, tức là biểu mẫu con trong biểu mẫu cha. Nếu thuộc tính này là False (ngầm định) thì biểu mẫu này không phải là biểu mẫu MDI.
Name	Xác định tên biểu mẫu. Tên ngầm định của biểu mẫu là Form1.
ScaleMode	Xác định kiểu đơn vị đo được sử dụng trong biểu mẫu. Thuộc tính này có 8 giá trị. Giá trị ngầm định của thuộc tính này là 1, tương ứng với đơn vị đo là twip. Khi đó đơn vị đo cho các thuộc tính Scale khác sẽ là twip. Các giá trị của thuộc tính này được cho trong Bảng 2.4.
Top	Xác định khoảng cách từ biên trên của màn hình đến biên trên của biểu mẫu.
Visible	Xác định tính thấy được của biểu mẫu.
Width	Xác định chiều rộng của biểu mẫu.

Bảng 3.4. Các giá trị của thuộc tính ScaleMode

Giá trị	Ý nghĩa
0	Đơn vị đo do lập trình viên xác định
1	Twip (ngầm định)
2	Point
3	Pixel
4	Kí tự chuẩn (rộng 120 twip, cao 240 twip)
5	Inch
6	Mm
7	Cm

3.3.3. Tạo ra biểu mẫu

Như đã nói ở các phần trên, khi ta tạo ra một dự án mới thì VB tự động tạo ra một biểu mẫu có tên là Form1. Nếu dự án chỉ có một biểu mẫu thì không cần nói gì thêm. Tuy nhiên, nếu dự án có nhiều biểu mẫu, thì cần phải tạo ra thêm các biểu mẫu mới cho dự án này. Muốn tạo ra một biểu mẫu mới ta làm như sau:

- Chọn Project
- Chọn Add Form
- Xác định vị trí và kích thước của biểu mẫu.
- Xác định các thuộc tính khác cho biểu mẫu.

3.3.4. Xoá biểu mẫu

Trong quá trình thiết kế chương trình, vì lí do nào đấy mà ta muốn xoá một biểu mẫu. Để xoá một biểu mẫu cần phải thực hiện các thao tác sau đây:

- Chọn Project
- Chọn Remove *tên biểu mẫu*

Sau khi làm như vậy máy tính sẽ xoá biểu mẫu có tên sau từ Remove khỏi chương trình đang được thiết kế.

3.4. NHÃN

3.4.1. Khái niệm nhãn

Nhãn (label) là một đối tượng đơn giản nhất trong biểu mẫu. Nhãn được dùng chủ yếu để in một dãy kí tự mà NSD (người sử dụng) không thể thay đổi được trong thời gian thực hiện chương trình. Nhãn có dạng một hình chữ nhật bên trong có một dãy kí tự. Nhãn cũng được sử dụng để nhận các thông báo trợ giúp.

Trong hộp công cụ thì công cụ tạo nhãn là một hình chữ nhật có kí tự “A” (chữ A hoa). Khi ta nhấp đúp chuột vào công cụ này thì một nhãn mới sẽ được tạo ra trong biểu mẫu. Nhãn này có tiêu đề ngầm định là label1.

3.4.2. Các thuộc tính cơ bản của nhãn

Nhãn có đến hơn 30 thuộc tính. Đa số các thuộc tính của nhãn trùng với thuộc tính của biểu mẫu. Các thuộc tính cơ bản của nhãn được cho trong bảng sau đây.

Bảng 3.5. Các thuộc tính cơ bản của nhãn

Thuộc tính	Ý nghĩa, tác dụng
Alignment	Xác định kiểu canh lề cho văn bản có trong nhãn. Thuộc tính này có 3 giá trị là: 0 - canh trái, 1 - canh phải, 2 - canh giữa.
Appearance	Xác định khi nào nhãn có dạng 3 chiều.
BackColor	Xác định màu nền của nhãn
BorderStyle	Xác định kiểu đường viền xung quanh nhãn.
Caption	Xác định tiêu đề cho nhãn.

Thuộc tính	Ý nghĩa, tác dụng
Enabled	Xác định tính khả ứng của nhãn.
Font	Xác định font chữ cho nhãn.
ForeColor	Xác định màu chữ của nhãn.
Height	Xác định chiều cao của nhãn.
Left	Xác định khoảng cách trái của nhãn.
Name	Xác định tên nhãn.
Top	Xác định khoảng cách trên nhãn.
Visible	Xác định tính thấy được của nhãn.
Width	Xác định chiều rộng của nhãn.

3.4.3. Tạo ra nhãn trong biểu mẫu

Để tạo ra một nhãn cần phải:

- Nháy đúp chuột vào công cụ tạo nhãn trong hộp công cụ. Một nhãn xuất hiện trong biểu mẫu.
- Xác định vị trí và kích thước của nhãn.
- Xác định các thuộc tính của nhãn.

3.4.4. Xoá một nhãn có trong biểu mẫu

Để xoá một nhãn ta làm như sau:

- Chọn nhãn;
- Chọn Edit
- Chọn Delete

Nhãn đã được chọn sẽ bị xoá khỏi biểu mẫu.

3.5. HỘ VĂN BẢN

3.5.1. Khái niệm hộ văn bản

Hộ văn bản (HVB) là một đối tượng trong biểu mẫu, được dùng chủ yếu để nhập và xuất văn bản. Hộ văn bản có dạng một hình chữ nhật chứa một dòng văn bản do lập trình viên xác định khi xác định các thuộc tính của hộ văn bản. Mặc dù hộ văn bản có thể chứa các số, nhưng VB luôn luôn xem các dữ liệu có trong hộ văn bản là dòng văn bản, tức là một dãy kí tự.

3.5.2. Các thuộc tính cơ bản của hộ văn bản

Hộ văn bản có hơn 40 thuộc tính. Các thuộc tính cơ bản của hộ văn bản được cho trong bảng sau đây.

Bảng 3.6. Các thuộc tính cơ bản của hộ văn bản

Thuộc tính	Ý nghĩa, tác dụng
Alignment	Xác định kiểu canh lề cho văn bản có trong HVB. Thuộc tính này có 3 giá trị là: 0 - canh trái, 1 - canh phải, 2 - canh giữa.
Appearance	Xác định khi nào HVB có dạng 3 chiều.
BackColor	Xác định màu nền của HVB
BorderStyle	Xác định kiểu đường viền xung quanh HVB.
Enabled	Xác định tính khả ứng của HVB.
Font	Xác định font chữ cho HVB.
ForeColor	Xác định màu chữ của HVB.
Height	Xác định chiều cao của HVB.
Left	Xác định khoảng cách trái của HVB.
MaxLength	Xác định độ dài tối đa của văn bản có thể được nhập

Thuộc tính	Ý nghĩa, tác dụng
	vào HVB. Giá trị ngầm định của thuộc tính này là 0. Giá trị này có nghĩa là có thể nhập đến 32000 kí tự vào HVB.
MultiLine	Xác định tính đa dòng của HVB. Nếu thuộc tính này có giá trị True thì có thể xuất nhiều dòng văn bản trong HVB. Nếu thuộc tính này có giá trị False thì chỉ có thể 1 dòng văn bản trong HVB.
Name	Xác định tên HVB.
PasswordChar	Xác định kí tự mật khẩu (Password). Giả sử thuộc tính này có giá trị là "*", thì kí tự "*" sẽ được gọi là kí tự mật khẩu. Nếu thuộc tính này đã được xác định như thế, thì khi người sử dụng nhập văn bản vào HVB các kí tự tạo nên văn bản sẽ không xuất hiện trong hộp văn bản, mà mỗi kí tự đó được thay bằng một kí tự "*" để người khác không nhìn thấy được.
Text	Xác định văn bản sẽ xuất hiện trong HVB. Giá trị ngầm định của thuộc tính này thường là "Text1". Trong thời gian thực hiện chương trình, khi người sử dụng có thể nhập vào HVB các văn bản mới thì giá trị thuộc tính này cũng thay đổi. Khi gán cho thuộc tính này một dòng văn bản bằng câu lệnh gán, thì dòng văn bản đó sẽ được in trong HVB. Ta đã thấy điều này khi viết phần mã lệnh cho chương trình Hello trong chương 2.
Top	Xác định khoảng cách trên của HVB.
Visible	Xác định tính thấy được của HVB.
Width	Xác định chiều rộng của HVB.

3.5.3. Tạo ra hộp văn bản trong biểu mẫu

Để tạo ra một hộp văn bản cần phải:

- Nháy đúp chuột vào công cụ tạo HVB trong hộp công cụ. Một HVB mới xuất hiện trong biểu mẫu.

- Xác định vị trí và kích thước của hộp văn bản
- Xác định các thuộc tính của hộp văn bản

3.5.4. Xoá một hộp văn bản đã có trong biểu mẫu

Để xoá một HVB ta làm như sau:

- Chọn HVB;
- Chọn Edit
- Chọn Delete

Hộp văn bản đã được chọn sẽ bị xoá khỏi biểu mẫu.

3.6. NÚT LỆNH

3.6.1. Khái niệm nút lệnh

Nút lệnh là một đối tượng trong biểu mẫu, dùng để điều khiển máy tính điện tử thực hiện một công việc nào đấy. Trong biểu mẫu nút lệnh có dạng một hình chữ nhật.

3.6.2. Các thuộc tính của nút lệnh

Các thuộc tính cơ bản của nút lệnh được cho trong bảng sau đây.

Bảng 3.7. Các thuộc tính cơ bản của nút lệnh

Thuộc tính	Ý nghĩa, tác dụng
Appearance	Xác định khi nào nút lệnh có dạng 3 chiều.
BackColor	Xác định màu nền của nút lệnh
Caption	Xác định tiêu đề cho nút lệnh.
Default	Nếu thuộc tính này có giá trị True thì nút lệnh này sẽ đáp ứng biến cố gõ phím ENTER khi biểu mẫu được kích hoạt cho dù tiêu điểm đang ở một đối tượng khác. Nếu thuộc tính này có giá trị False (ngầm định) thì nút lệnh không có đặc điểm như vậy.
Enabled	Xác định tính khả ứng của nút lệnh.
Font	Xác định font chữ cho nút lệnh.
ForeColor	Xác định màu chữ của nút lệnh.
Height	Xác định chiều cao của nút lệnh.
Left	Xác định khoảng cách trái của nút lệnh.
Name	Xác định tên nút lệnh.
Top	Xác định khoảng cách trên của nút lệnh.
Visible	Xác định tính thấy được của nút lệnh.
Width	Xác định chiều rộng của nút lệnh.

3.6.3. Tạo ra nút lệnh trong biểu mẫu

- Nháy đúp chuột vào công cụ tạo nút lệnh trong hộp công cụ.
- Một nút lệnh có tiêu đề `command1` xuất hiện trong biểu mẫu.
- Xác định vị trí và kích thước của nút lệnh.
 - Xác định các thuộc tính cho nút lệnh (tiêu đề và tên).

3.6.4. Xoá nút lệnh khỏi biểu mẫu

Để xoá một nút lệnh ta làm như sau:

- Chọn nút lệnh cần xoá.
- Chọn Edit.
- Chọn Delete.

Nút lệnh đã được chọn sẽ bị xoá khỏi biểu mẫu.

Các nút lệnh có thể tiếp nhận nhiều biến cố. Biến cố hay được sử dụng nhất là nháy chuột (click) lên nút lệnh. Để nút lệnh có thể đáp ứng biến cố này phải viết một thủ tục thích hợp. Thủ tục này sẽ được thực hiện khi người sử dụng nháy chuột lên nút lệnh. Ta đã biết điều này khi thiết kế chương trình Hello trong chương trước.

Trên đây chúng ta đã nghiên cứu khái niệm, thuộc tính, cách tạo ra và cách xoá bốn loại đối tượng là biểu mẫu, nhãn, hộp văn bản và nút lệnh. Đây là những đối tượng hay được sử dụng nhất để thiết kế phần giao diện của chương trình VB. Visual Basic còn nhiều đối tượng khác nữa. Các đối tượng đó cùng với các thuộc tính của chúng sẽ được nghiên cứu trong các chương sau khi các đối tượng này được sử dụng.

CÂU HỎI ÔN TẬP

1. Đối tượng là gì?
2. Thuộc tính của đối tượng là gì?
3. Phương thức của đối tượng là gì?
4. Biểu mẫu là gì? biểu mẫu có vai trò gì trong thiết kế chương trình VB?
5. Hãy trình bày những thuộc tính cơ bản của biểu mẫu.
6. Nhãn là gì? Hãy nêu vai trò của nhãn chương trình VB?
7. Hãy trình bày những thuộc tính cơ bản của nhãn.
8. Hộp văn bản là gì? Hộp văn bản dùng để làm gì?

9. Hãy trình bày những thuộc tính cơ bản của hộp văn bản. Thuộc tính nào là thuộc tính riêng có của hộp văn bản?

10. Nút lệnh là gì? Nút lệnh dùng để làm gì?

11. Hãy trình bày những thuộc tính cơ bản của nút lệnh. Thuộc tính nào là thuộc tính riêng có của nút lệnh?

12. Nút lệnh có thể đáp ứng biến cố nào?

NHỮNG KIỂU DỮ LIỆU CƠ SỞ CỦA VISUAL BASIC

Trong chương trước ta đã nghiên cứu các đối tượng của VB. Đó là các yếu tố để thiết kế phần giao diện của chương trình Visual Basic. Trong chương này ta sẽ nghiên cứu các kiểu dữ liệu cơ sở của Visual Basic. Các dữ liệu này sẽ được sử dụng trong các câu lệnh để thiết kế phần mã lệnh của chương trình Visual Basic.

4.1. KHÁI NIỆM KIỂU DỮ LIỆU

Các dữ liệu mà máy tính điện tử có thể tiếp nhận và xử lý được có nhiều đặc điểm khác nhau. Chúng khác nhau về nội dung, miền giá trị, hình thức thể hiện... Xã hội con người càng phát triển thì khối lượng dữ liệu ngày càng lớn và những điểm khác nhau nêu trên cũng ngày càng nhiều hơn.

Căn cứ vào đặc điểm của dữ liệu mà người ta chia dữ liệu thành các kiểu dữ liệu (data type) khác nhau. Một kiểu dữ liệu được định nghĩa với hai điểm chính là:

- Một tập hợp giá trị của kiểu dữ liệu này;
- Một số các phép tính có thể thực hiện trên tập hợp các giá trị đó;

Trong máy tính điện tử dữ liệu được chứa trong (chiếm) một số byte của bộ nhớ. Các kiểu dữ liệu khác nhau sẽ chiếm số byte khác nhau.

Trong Visual Basic cho phép nhận và xử lý các kiểu dữ liệu cơ sở sau đây:

Bảng 4.1. Các kiểu dữ liệu cơ sở của Visual Basic

Tên kiểu dữ liệu	Ý nghĩa	Miền giá trị
Currency	Kiểu tiền tệ	-\$922,337,203,685,477.5808 -> +\$922,337,203,685,477.5808
Byte	Kiểu byte (số nguyên ngắn)	0 -> 255
Integer	Kiểu số nguyên	-32768 -> 32767
Long	Kiểu số nguyên dài	-2147483648 -> 2147483647
Single	Kiểu số thực	Có đến 7 chữ số (-3.402823E+38 -> 3.402823E+38)
Double	Kiểu số thực kép	Có đến 16 chữ số (-1.79769313486232E+308 -> 1.79769313486232E+308)
Variant	Kiểu tổng hợp	Có thể chứa giá trị kiểu bất kỳ
Boolean	Kiểu logic	True hoặc False
String	Kiểu chuỗi (dãy kí tự)	Có thể chứa từ 0 -> 65400 kí tự
Date	Kiểu thời gian	Có thể chứa thông tin về thời gian từ 1 tháng 1 năm 100 đến 31 tháng 12 năm 9999.

Bảng này chứa 10 kiểu dữ liệu. Sáu kiểu dữ liệu đầu tiên, từ kiểu tiền tệ đến kiểu số thực kép, có đặc điểm chung là giá trị của chúng là các số. Thuật ngữ “dữ liệu số” được dùng để chỉ tất cả các kiểu dữ liệu này.

Trong bảng trên có tên và miền giá trị của các kiểu dữ liệu. Các phép tính của các kiểu dữ liệu này sẽ được trình bày trong phần 4.4. dưới đây.

4.2. CÁC HẰNG TRONG VB

Hằng là yếu tố rất hay được sử dụng trong chương trình Visual Basic. Hằng được kết hợp với các biến, các hàm và các dấu phép toán để tạo nên các biểu thức.

4.2.1. Khái niệm hằng

Hằng (const) là những đại lượng cố định, không thay đổi. Muốn sử dụng hằng ở đâu ta phải viết chính bản thân hằng ở đó.

Trong VB hằng được chia thành hằng ký tự, hằng số và hằng logic.

4.2.2. Hằng chuỗi ký tự

Hằng chuỗi ký tự (gọi tắt là hằng chuỗi) là một dãy có từ 1 đến 255 ký tự được đặt trong một cặp dấu nháy kép.

Ví dụ: hằng chuỗi ký tự có thể có dạng:

"Ha Noi"

" Tong so bang : "

" Chao em ! "

Hằng ký tự có thể gồm ký tự bất kỳ dùng trong Visual Basic (ngoài dấu nháy kép).

Có một số hằng kí tự có sẵn trong VB. Ví dụ hằng *vbCrLf* gây ra việc chuyển xuống đầu dòng mới khi in. Hằng *vbNullString* cho một chuỗi rỗng (là chuỗi không có một kí tự nào cả).

Chú ý: hằng chuỗi có thể không có một kí tự nào cả. Đó là hằng chuỗi rỗng. Hằng này được thể hiện bằng hai dấu nháy kép đi liền nhau ("") hoặc bằng hằng chuỗi *vbNullString*.

4.2.3. Hằng số

Hằng số là các số được viết theo các qui tắc của VB dưới các dạng khác nhau.

Hằng số nguyên (gọi tắt là hằng nguyên) là một số nguyên (nằm trong một khoảng xác định) như ta vẫn viết hàng ngày. Số có thể có dấu. Số không dấu là số dương.

Ví dụ về các hằng số nguyên:

1

-5

1975

Hằng số thực có 2 dạng là hằng số thực dấu phẩy tĩnh và hằng số thực dấu phẩy động.

Hằng số thực dấu phẩy tĩnh là những số thực có chứa dấu chấm thập phân để ngăn chia phân nguyên và phần lẻ.

Ví dụ về các hằng số thực dấu phẩy tĩnh:

3.1416

-2.50

0.45

Hằng số thực dấu phẩy động là những số thực được viết dưới dạng gồm có hai phần, phần định trị và phần bậc. Giữa hai phần đó có

ký tự E hoặc D ngăn cách.

Ví dụ về các hằng số thực dấu phẩy động:

Số viết thường	Số viết dưới dạng dấu phẩy động.
3,14	0.314E+1
135	0.135E+3
-0,05	-0.5E-1

Các hằng số thực có thể có độ chính xác đơn hoặc độ chính xác kép (gấp đôi). Các số có độ chính xác đơn được lưu trữ trong máy tới 7 chữ số, được in tới 7 chữ số. Các số có độ chính xác kép được lưu trữ tới 17 chữ số, được in tới 16 chữ số.

Khi viết các số dấu phẩy động, ký tự E được dùng để viết các số chính xác đơn, ký tự D được dùng để viết các số chính xác kép.

Cần nhớ rằng các hằng và biến thực độ chính xác kép chiếm nhiều bộ nhớ hơn, và việc tính toán trên chúng đòi hỏi nhiều thời gian hơn.

4.2.4. Hằng logic

Có hai hằng logic là **True** (đúng) và **False** (sai). Các hằng này được sử dụng để xây dựng các biểu thức logic.

4.2.5. Hằng kiểu thời gian

Các hằng thời gian trong Visual Basic có dạng:

#Jan 1, 2004# hoặc "Jan 1, 2004"

#January 1, 2004# hoặc "January 1, 2004"

#1/1/ 2004# hoặc "1/1/ 2004"

Các hằng kiểu thời gian trên đây đều chỉ ngày 1 tháng 1 năm 2004. Như vậy chúng ta thấy rằng Visual Basic tương đối "thoảng" trong việc chấp nhận các cách viết hằng kiểu thời gian.

4.2.6. Khai báo hằng

Trong phần trên ta đã nói: “Muốn sử dụng hằng ở đâu ta phải viết chính bản thân hằng ở đó”. Tuy nhiên ta cũng có thể sử dụng tên hằng thay cho chính bản thân hằng. Muốn vậy ta phải khai báo hằng. Để khai báo hằng ta dùng câu lệnh khai báo có dạng sau đây:

`Const tên_hằng = giá trị`

Ví dụ: có thể khai báo hằng PI như sau:

`Const PI = 3.1416`

Sau khi khai báo như vậy thì trong phần mã lệnh của chương trình mỗi khi cần dùng giá trị 3.1416 ở đâu ta chỉ việc viết PI ở đó. Như vậy ta thấy rằng sử dụng tên hằng thuận tiện hơn sử dụng bản thân hằng vì tên hằng ngắn hơn, dễ viết hơn và gọi nhớ hơn.

4.3. BIẾN

Biến được sử dụng rất rộng rãi và có vai trò cực kỳ quan trọng trong lập trình. Suy cho cùng thì mọi chương trình đều phải làm 3 công việc sau đây:

- Cho phép người sử dụng nhập giá trị của một số biến, gọi là biến nhập hoặc biến ban đầu;
- Sử dụng giá trị các biến đó để thực hiện một quá trình tính toán xác định để tạo ra giá trị một số biến khác gọi là biến xuất hoặc biến kết quả;
- In (tức là xuất) giá trị các biến này dưới dạng xác định để cung cấp cho người sử dụng.

Biến là yếu tố hay được ta sử dụng nhất khi viết phần mã lệnh của chương trình Visual Basic. Vậy biến là gì?

4.3.1. Khái niệm biến

Biến (variable) là những đại lượng thay đổi. Muốn sử dụng biến ở đâu ta phải viết tên biến ở đó. Trong quá trình thực hiện chương trình ta có thể thay đổi giá trị của các biến.

4.3.2. Cách đặt tên biến

Tên biến trong VB là một dãy ký tự, gồm chữ cái, chữ số và dấu ngang dưới (_) thoả mãn các điều kiện sau đây:

- Tên biến phải bắt đầu bằng chữ cái và không được chứa dấu cách (space).

- Tên biến có thể dài đến 256 ký tự.

- Có thể dùng cả chữ thường lẫn chữ hoa để đặt tên biến. VB không phân biệt chữ thường và chữ hoa trong tên biến.

- Tên biến không được trùng với các từ khoá của Visual Basic (là những từ tiếng Anh, có ý nghĩa cố định trong ngôn ngữ Visual Basic).

- Tên biến phải gọi nhớ.

Ví dụ:

Các tên biến sau đây là đúng:

ALFA

X1

TIENLUONG, TienLuong (có giá trị như nhau)

DONGIA, DON_GIA (Hai biến khác nhau)

Các dãy ký tự sau đây không phải là tên biến:

1X : không bắt đầu bằng chữ cái.

TIEN* : Chứa dấu *, không cho phép có trong tên biến.

DON GIA : Chứa dấu cách.

4.3.3. Kiểu của biến

Mỗi biến đều phải thuộc về một kiểu dữ liệu nhất định. Kiểu của biến được xác định khi khai báo biến hoặc đặt tên biến. Nếu kiểu của biến không được chỉ ra thì VB luôn luôn cho rằng đó là biến kiểu tổng hợp (Variant).

Có thể xác định một số kiểu của biến bằng kí tự cuối cùng của tên biến theo bảng sau đây.

Bảng 4.2. Xác định kiểu của biến bằng kí tự cuối cùng của tên biến

Kiểu của biến	Ý nghĩa	Kí tự cuối cùng của tên biến
Currency	Tiền tệ	@
String	Dãy kí tự	\$
Integer	Số nguyên	%
Long Integer	Số nguyên dài	&
Single Precision	Số thực	!
Double Precision	Số thực kép	#

4.3.4. Khai báo biến

Các biến được sử dụng trong chương trình VB có thể bắt buộc phải được khai báo (giống như trong Pascal) hoặc không cần phải khai báo (giống như trong FoxPro).

Nếu ở phần General của chương trình có câu lệnh:

Option Explicit

Thì tất cả các biến được sử dụng trong chương trình VB đều bắt

buộc phải được khai báo trước. Còn nếu ở đầu phần mã lệnh của chương trình không có câu lệnh này thì không cần phải khai báo các biến trước khi sử dụng chúng.

Chú ý : Nếu ở trong phần môi trường của VB đã chọn mục:

Require Variable declaration

Thì cũng cần phải khai báo các biến trước khi sử dụng chúng và Visual Basic tự động thêm câu lệnh ***Option Explicit*** vào phần General của mỗi chương trình .

Việc khai báo biến được thực hiện bằng câu lệnh ***Dim***. Câu lệnh này có qui cách sau đây:

`Dim <tên biến> [ As <tên kiểu dữ liệu> ]`

Ở đây:

- Tên biến là tên của biến đang được khai báo;
- Tên kiểu dữ liệu là một từ khoá dùng để xác định kiểu dữ liệu, như String, Integer, Single, double...

Theo câu lệnh này Visual Basic sẽ tạo ra biến có tên trong lệnh. Phần

`As <tên kiểu dữ liệu>`

được dùng để xác định kiểu của biến nếu cần. Đây là thành phần tùy chọn.

Chú ý: câu lệnh Dim chỉ tạo ra biến chứ không gán giá trị ban đầu cho biến. Nếu muốn gán giá trị ban đầu cho biến lập trình viên phải dùng câu lệnh gán để làm việc này.

Các ví dụ về câu lệnh khai báo dữ liệu:

Câu lệnh Dim

Ý nghĩa

Dim i as Integer	Khai báo biến i kiểu số nguyên
Dim n as Long	Khai báo biến n kiểu số nguyên dài
Dim x as Single	Khai báo biến x kiểu số thực
Dim y as Double	Khai báo biến y kiểu số thực kép
Dim x as Single, i as Integer	Khai báo biến x kiểu số thực, biến i kiểu số nguyên
Dim Ten\$, I%, X!	Khai báo 3 biến. biến Ten\$ kiểu chuỗi, biến I% kiểu số nguyên, biến X! kiểu số thực.

Các chú ý:

1. Câu lệnh:

```
Dim m,n as Integer
```

Khai báo 2 biến là m và n. Biến n có kiểu là Integer. Biến m có kiểu là Variant.

2. Giả sử có câu lệnh khai báo:

```
Dim s%, i%
```

thì viết s% và s có giá trị như nhau; và viết i% và i cũng có giá trị như nhau.

4.3.5. Miền hoạt động của các biến

Miền hoạt động của các biến còn được gọi là miền tác động hoặc phạm vi hoạt động của biến. Miền hoạt động của một biến là phần chương trình tại đó biến này có nghĩa (tức là tại đó có thể sử dụng được biến này).

Thông thường thì biến được khai báo trong thủ tục nào thì có

miền hoạt động là thủ tục đó. Các biến có đặc điểm này được gọi là biến cục bộ hoặc biến địa phương (Local Variable).

Có những biến được khai báo trong phần General của chương trình. Những biến này có thể được sử dụng trong nhiều thủ tục khác nhau. Các biến có đặc điểm này được gọi là biến toàn cục (Global Variable).

4.4. CÁC PHÉP TOÁN

Các phép toán là một trong hai yếu tố tạo thành một kiểu dữ liệu. Mỗi kiểu dữ liệu có các phép toán riêng của mình. Sau đây ta sẽ nghiên cứu các phép toán cho từng kiểu dữ liệu.

4.4.1. Phép toán đối với các chuỗi

Chỉ có một phép toán cho các chuỗi. Phép toán này được gọi là phép toán nối (hoặc ghép) 2 chuỗi. Phép toán này được ký hiệu bằng dấu "&" hoặc dấu "+". Tác dụng của phép toán này là máy tính tạo ra một chuỗi mới bằng cách nối (hoặc ghép) hai chuỗi đã cho với nhau.

Ví dụ về phép toán nối các chuỗi kí tự:

“Viet “ + “Nam” cho kết quả là “Viet Nam”

“Viet “ & “Nam” cho kết quả là “Viet Nam”

4.4.2. Các phép toán đối với dữ liệu kiểu số

Các phép toán trình bày trong mục này được áp dụng cho các kiểu dữ liệu tiền tệ, các kiểu số nguyên, các kiểu số thực. Tuy nhiên hai phép toán “\” (chia số nguyên) và mod (lấy số dư của phép chia nguyên) chỉ áp dụng cho các dữ liệu kiểu số nguyên.

Bảng 4.3. Các phép toán số học (dùng cho các kiểu số nguyên, số thực và tiền tệ)

Dấu phép toán	Ý nghĩa	Ví dụ
+	Cộng	$2 + 3$ cho kết quả là 5
-	Trừ	$5 - 2$ cho kết quả là 3
*	Nhân	$2 * 3$ cho kết quả là 6
/	Chia	$5 / 2$ cho kết quả là 2.5
\	Chia số nguyên	$5 \setminus 2$ cho kết quả là 2
mod	Lấy số dư của phép chia nguyên	$5 \bmod 4$ cho kết quả là 1
^	Nâng lên lũy thừa	$3 ^ 2$ cho kết quả là 9

4.4.3. Các phép toán logic

Các phép toán logic dùng cho dữ liệu kiểu logic. Tên và ý nghĩa các phép toán này được cho trong bảng sau đây:

Bảng 4.4. Các phép toán logic

Phép toán	Ý nghĩa
NOT	Phủ định logic
OR	Cộng logic
AND	Nhân logic
XOR	Hoặc loại trừ logic

Kết quả thực hiện phép toán logic luôn luôn là một giá trị logic. Quy tắc thực hiện các phép toán logic được cho trong bảng sau đây:

Bảng 4.5. Quy tắc thực hiện các phép toán logic

X	Y	NOT X	X AND Y	X OR Y	X XOR Y
False	False	True	False	False	False
False	True	True	False	False	True
True	False	False	False	False	True
True	True	False	True	True	False

Các phép toán logic được sử dụng để xây dựng các biểu thức logic.

4.4.4. Các phép tính quan hệ

Các phép tính quan hệ so sánh hai giá trị. Hai giá trị này phải cùng là chuỗi ký tự hoặc cùng là các số. Kết quả phép so sánh là TRUE nếu quan hệ được thoả mãn. Ngược lại, nếu quan hệ không được thoả mãn thì kết quả phép so sánh là FALSE. Các kết quả này thường được sử dụng để ra quyết định khi lập chương trình phân nhánh.

Bảng 4.6. Các phép toán quan hệ

Tên quan hệ	Dấu phép tính
Bằng	=
Khác	< > hoặc ><
Nhỏ hơn	<
Lớn hơn	>
Lớn hơn hoặc bằng	>= hoặc =>
Nhỏ hơn hoặc bằng	<= hoặc =<

Chú ý:

- Nếu trong một biểu thức có cả các phép tính quan hệ lẫn các phép toán số học, thì các phép toán số học được ưu tiên thực hiện trước.

- Khi so sánh hai chuỗi ký tự, thì mã ASCII của các cặp ký tự tương ứng của hai chuỗi ký tự đó được so sánh với nhau.

4.5. HÀM CHUẨN

Trong quá trình giải bài toán lập trình viên thường phải tính toán giá trị một số hàm số, ví dụ $\sin$ của x , căn bậc hai của x ... Trong

Visual Basic có sẵn chương trình tính giá trị các hàm này. Mỗi chương trình này có một tên xác định. Các chương trình này được gọi là các hàm. Tên các chương trình này gọi là tên hàm. Muốn tính giá trị hàm ở đâu ta chỉ việc viết tại đó tên hàm và đối số của hàm trong cặp dấu ngoặc tròn (). Ví dụ: hàm tính căn bậc hai của x trong Visual Basic có tên là `sqr()`. Muốn tính căn bậc hai của x ở đâu ta chỉ việc viết `sqr(x)` ở đó.

Hệ thống hàm chuẩn của Visual Basic rất phong phú. Nó làm cho việc lập trình trong Visual Basic dễ dàng hơn rất nhiều và làm cho khả năng của Visual Basic cũng tăng lên rất nhiều.

Mỗi kiểu dữ liệu có một tập hợp hàm chuẩn của mình. Trong chương 8 ta sẽ nghiên cứu kỹ hơn về hàm. Sau đây là một số hàm toán học chuẩn thường gặp.

Bảng 4.7. Một số hàm toán học chuẩn thường gặp

Tên hàm	Ý nghĩa
<code>Abs(x)</code>	Trả về giá trị tuyệt đối của x
<code>Cos(x)</code>	Trả về cosin của x (tính bằng radian)
<code>Exp(x)</code>	Trả về e^x
<code>Log(x)</code>	Trả về logarit tự nhiên của x
<code>Sin(x)</code>	Trả về sin của x (tính bằng radian)
<code>Sqr(x)</code>	Trả về căn bậc 2 của x
<code>Tan(x)</code>	Trả về tang của x (tính bằng radian)

4.6. BIỂU THỨC

Biểu thức là yếu tố không thể thiếu được để viết nhiều câu lệnh của chương trình VB. Nhờ có biểu thức ta có thể sử dụng các hằng, các biến, các hàm đã cho để tạo ra một giá trị mới.

4.6.1. Khái niệm về biểu thức

Biểu thức là một dãy gồm các hằng, các biến, các hàm và các thuộc tính của đối tượng, được kết hợp với nhau bởi các dấu phép toán để tạo nên một giá trị duy nhất.

Trong trường hợp đơn giản nhất, biểu thức chỉ gồm một hằng, một biến một hàm hoặc một thuộc tính của một đối tượng.

Ví dụ: Biểu thức có thể có dạng.

$(\text{DiemToan} + \text{DiemLy} + \text{DiemHoa})/3$

“Ho va ten “ + TEN\$

“ Diem Trung Bình = “ + txtKetQua.Text

A1

15

$\text{sqr}(\text{Delta})$

$(a <> 0) \text{ and } (b <> 0)$

4.6.2. Quy tắc viết biểu thức

Khi viết biểu thức trong Visual Basic cần phải tuân theo các qui định sau đây:

- Phải sử dụng đúng các dấu phép toán của Visual Basic, như “*” cho phép nhân, “/” cho phép chia, “>=” cho phép lớn hơn hoặc bằng v.v...

- Toàn bộ biểu thức phải nằm trên cùng một dòng.

- Khi cần thiết có thể dùng các dấu ngoặc tròn (và chỉ các dấu ngoặc tròn) để chỉ ra thứ tự ưu tiên của việc thực hiện các tính khi tính giá trị biểu thức.

Các biểu thức chia thành biểu thức chuỗi kí tự, biểu thức số và

biểu thức logic. Biểu thức chuỗi kí tự là biểu thức cho giá trị là một chuỗi kí tự. Biểu thức số là biểu thức cho giá trị là một số. Biểu thức logic là biểu thức cho giá trị là hằng logic True hoặc False.

4.6.3. Thứ tự thực hiện các phép toán khi xác định giá trị biểu thức

Khi tính giá trị các biểu thức, các phép tính được thực hiện theo thứ tự ưu tiên sau đây :

1. Các lời gọi hàm được thực hiện đầu tiên.
2. Các phép tính số học.

Các phép tính số học có thứ tự ưu tiên như sau:

- a. Phép nâng lên lũy thừa (^)
- b. Phép *, /
- c. Phép \ (chia nguyên)
- d. Phép MOD
- e. Phép +, -

3. Các phép tính quan hệ.
4. Các phép toán logic.

Các phép tính logic có thứ tự ưu tiên như sau:

- a. NOT
- b. AND
- c. OR
- d. XOR

Đối với các phép tính có cùng mức ưu tiên, thì chúng được thực hiện từ trái sang phải. Các phép toán trong ngoặc được thực hiện trước các phép toán ngoài dấu ngoặc.

4.6.4. Kiểu của biểu thức

Biểu thức có thể chứa các hằng, các biến, các hàm thuộc các kiểu khác nhau. Vậy kiểu của biểu thức được xác định như thế nào?

Ta biết rằng biểu thức luôn luôn xác định một giá trị. Giá trị này phải thuộc về một kiểu dữ liệu nào đấy. Kiểu của biểu thức là kiểu của giá trị được xác định bằng biểu thức này.

Biểu thức số là biểu thức cho một giá trị số. Biểu thức chuỗi là biểu thức cho giá trị là một chuỗi kí tự. Biểu thức logic là biểu thức cho một giá trị là True hoặc False.

Chú ý: Giả sử có một biểu thức logic:

$$x > 0 \quad (1)$$

Biểu thức này cho giá trị True khi biến x có giá trị dương (> 0).

Biểu thức (1) cũng còn được gọi là một điều kiện. Điều kiện này được thoả mãn khi biến x có giá trị dương (> 0).

Vì vậy:

- Hai cách nói: “biểu thức logic $x > 0$ ” và “điều kiện $x > 0$ ” là hai cách nói tương đương, có thể được dùng thay thế cho nhau.

- Hai cách nói: “biểu thức logic $x > 0$ có giá trị True” và “điều kiện $x > 0$ được thoả mãn” là hai cách nói tương đương, có thể được dùng thay thế cho nhau.

Các ví dụ về biểu thức:

TienLuong + TienThuong - TienNo

SoLuong * DonGia

$b * b - 4 * a * c$

$(-b + \text{sqr}(\text{Delta})) / (2 * a)$

DiemTB >= 7.5 AND DiemTB < 9.0

Trong các chương sau chúng ta sẽ thấy rằng biểu thức là yếu tố được sử dụng rất rộng rãi khi viết phần mã lệnh của chương trình.

CÂU HỎI ÔN TẬP

1. Tại sao dữ liệu lại được chia thành các kiểu khác nhau? Việc làm này mang lại lợi ích gì?
2. Một kiểu dữ liệu bao gồm những yếu tố nào?
3. Visual Basic có những kiểu dữ liệu cơ sở nào?
4. Hãy trình bày đặc điểm của các kiểu dữ liệu cơ sở của Visual Basic.
5. Hằng là gì? Trong Visual Basic có những loại hằng nào?
6. Hãy nêu lợi ích của việc khai báo hằng và dùng tên hằng.
7. Biến là gì? Biến được dùng để làm gì?
8. Hãy nêu những điểm giống nhau và khác nhau giữa biến và hằng.
9. Hãy trình bày các cách xác định kiểu của biến.
10. Hãy trình bày khái niệm và tác dụng của biểu thức.
11. Hãy nêu các quy tắc viết các biểu thức. Cho một số ví dụ về biểu thức.
12. Hãy nêu thứ tự thực hiện các phép tính khi tính giá trị biểu thức.
13. Có các loại biểu thức nào trong Visual Basic?

LẬP CHƯƠNG TRÌNH ĐƠN GIẢN

Chương này tập trung vào bước thứ hai của quá trình thiết kế chương trình Visual Basic. Đó là viết phần mã lệnh cho chương trình. Chương này sẽ trình bày các câu lệnh của Visual Basic và kết hợp những câu lệnh này với các yếu tố đã nghiên cứu trong chương 3 và chương 4 để tạo ra những chương trình khác nhau. Chương này cũng nghiên cứu một số đối tượng khác của Visual Basic và sử dụng các đối tượng đó để viết các chương trình.

5.1. QUY ĐỊNH VỀ CÁCH VIẾT CÁC DÒNG LỆNH TRONG CHƯƠNG TRÌNH VB

Khi viết các dòng lệnh trong phần mã lệnh của chương trình Visual Basic phải tuân theo các quy tắc sau đây:

- Thường thường một câu lệnh được viết trên một dòng của chương trình. Một dòng của chương trình có thể chứa tối đa 1023 ký tự.
- Một dòng của chương trình có thể chứa nhiều câu lệnh. Khi đó các câu lệnh cách nhau dấu : (hai chấm).
- Nếu cần một phần câu lệnh có thể được viết ở dòng tiếp theo. Khi đó ở cuối dòng trên phải có hai ký tự dấu cách và dấu ngang dưới (" _") đi liền nhau.

5.2. CÁCH VIẾT DÒNG CHÚ THÍCH TRONG CHƯƠNG TRÌNH

Trong phần mã lệnh của chương trình Visual Basic ta có thể viết các dòng chú thích. Dòng chú thích được dùng để giải thích cho người đọc chương trình về mục đích của một đoạn chương trình, về ý nghĩa một câu lệnh, một biến, một biểu thức... Visual Basic không chú ý đến các dòng chú thích. Không bắt buộc phải viết các dòng chú thích. Tuy nhiên, trong các chương trình dài và phức tạp nên có các dòng chú thích. Chúng sẽ rất có ích cho bạn, đặc biệt là khi bạn cần sửa chữa chương trình sau một thời gian dài.

Có hai cách viết dòng chú thích trong chương trình.

Cách thứ nhất là dùng dấu nháy đơn ('). Visual Basic xem phần văn bản kể từ dấu nháy đơn đến hết dòng lệnh là dòng chú thích.

Cách thứ hai là dùng lệnh REM. Visual Basic xem toàn bộ dòng lệnh có lệnh REM ở đầu dòng là dòng chú thích.

Ví dụ: các dòng chú thích có thể được viết như sau:

```
Private Sub cmdIn_Click()  
    REM Đây là thủ tục in kết quả  
        txtKetQua.Text = "Hello The World !" ' Thay  
        doi nội dung HVB  
End Sub
```

Trong ví dụ trên thì toàn bộ dòng lệnh thứ nhất và một phần dòng lệnh thứ hai là dòng chú thích.

5.3. MỘT SỐ CÂU LỆNH CỦA VISUAL BASIC

Trong phần này ta sẽ nghiên cứu một số câu lệnh đơn giản và hay được sử dụng nhất của VB.

5.3.1. Câu lệnh gán

Câu lệnh gán là câu lệnh cơ bản, quan trọng nhất của mọi ngôn ngữ lập trình. Trong Visual Basic câu lệnh gán được dùng để gán giá trị một biểu thức cho một biến hoặc một thuộc tính của một đối tượng. Trong VB câu lệnh gán có dạng sau đây:

Tên_biến = biểu_thức

hoặc

Tên_đối_tượng.Thuộc_tính = biểu_thức

Tác dụng: Theo câu lệnh này máy tính giá trị biểu thức phía phải dấu bằng (=) và gán giá trị đó cho biến hoặc thuộc tính phía trái dấu bằng (=).

Ví dụ:

```
s = i% + t!
```

```
txtKetQua.Text = "Xin Chao Tat Ca!"
```

Tất cả các kiểu dữ liệu số (Integer, Long, Single, Double, Currency) có thể được gán cho nhau và cho biến Variant. Visual Basic làm tròn giá trị trước khi gán cho biến Integer.

5.3.2. Câu lệnh BEEP

Câu lệnh này có qui cách sau đây:

Beep

Tác dụng: Theo câu lệnh này máy tính phát ra một tiếng kêu "pip". Tiếng kêu này thường được dùng để để lưu ý người sử dụng về một điều gì đấy.

5.3.3. Câu lệnh END

Câu lệnh này có qui cách sau đây:

End

Theo câu lệnh này máy kết thúc thực hiện chương trình Visual Basic, trở về màn hình giao diện chính của Visual Basic.

5.4. CHUYỂN ĐỔI KIỂU DỮ LIỆU TRONG QUÁ TRÌNH TÍNH GIÁ TRỊ BIỂU THỨC

Visual Basic cho phép các dữ liệu trong một biểu thức có thể có kiểu khác nhau. Kiểu của kết quả sẽ được xác định theo yêu cầu sử dụng.

Ví dụ: giả sử có các câu lệnh gán sau đây:

```
i%=1
```

```
t!=3.33
```

```
s%= i% + t!
```

```
x! = i% + t!
```

thì s% có giá trị là 4, còn x! có giá trị là 4.33. Qua ví dụ trên ta thấy rằng:

- Các dữ liệu trong một biểu thức có thể có kiểu khác nhau;
- Kiểu của giá trị biểu thức được chuyển đổi cho phù hợp với kiểu của biến sẽ nhận giá trị đó bằng câu lệnh gán.

Có thể nói rằng câu lệnh gán của Visual Basic rất “thoáng”. Các câu lệnh gán sau đây sẽ cho thấy điều đó.

Câu lệnh	Tác dụng
Dim i As Integer	Khai báo biến i là biến kiểu số nguyên.
Dim x as Single	Khai báo biến x là biến kiểu số thực.

Câu lệnh	Tác dụng
<code>I = 1 + "2"</code>	Câu lệnh này hợp lệ. Chuỗi "2" được biến thành số nguyên 2, rồi số nguyên này được cộng với 1. Kết quả thực hiện phép cộng này là số 3. Số 3 sẽ được gán cho biến i. Như vậy, sau khi thực hiện câu lệnh này biến i có giá trị là 3.
<code>I = "1" + "2"</code>	Câu lệnh này hợp lệ. Đầu tiên các chuỗi "1" và "2" được cộng với nhau để tạo thành chuỗi "12". Sau đó chuỗi này được biến đổi thành số 12. Số 12 này được gán cho biến i. Vì vậy sau khi thực hiện câu lệnh này biến i có giá trị là 12.
<code>x = 5.5</code>	Câu lệnh này hợp lệ.
<code>x = "5.5"</code>	Câu lệnh này hợp lệ. Đầu tiên chuỗi "5.5" được biến đổi thành số thực 5.5. Số này được gán cho biến x. Sau khi thực hiện câu lệnh này biến x có giá trị là số thực 5.5.
<code>x = "5,5"</code>	Câu lệnh này hợp lệ. Đầu tiên chuỗi "5,5" được biến đổi thành số thực 55, vì Visual Basic bỏ qua dấu ",", không cho đó là dấu phẩy thập phân, dùng để ngăn chia phần nguyên và phần lẻ của số. Số 55 này được gán cho biến x. như vậy sau khi thực hiện câu lệnh này biến x có giá trị là số thực 55.
<code>x = "5A5"</code>	Câu lệnh này không hợp lệ. Sở dĩ như vậy là vì chuỗi "5A5" chứa kí tự "A", nên không thể được biến đổi thành một số. Gặp câu lệnh này máy sẽ in thông báo lỗi có dạng sau đây: Run-time error '13' Type mismatch
<code>I = i + "AB"</code>	Câu lệnh này không hợp lệ. Sở dĩ như vậy là vì chuỗi "AB" không thể được biến đổi thành một số nguyên. Gặp câu lệnh này máy sẽ in thông báo lỗi có dạng sau đây: Run-time error '13' Type mismatch

Trên đây ta thấy rằng Visual Basic đã tự động chuyển đổi kiểu dữ liệu khi tính giá trị biểu thức hoặc khi thực hiện câu lệnh gán. Tuy nhiên, khi cần ta có thể dùng các hàm chuyển đổi kiểu dữ liệu để làm công việc này. Dưới đây là hai hàm hay được sử dụng nhất.

Hàm **Val(a\$)**: Đổi giá trị của a\$ từ dạng dãy ký tự sang dạng số.

Hàm **Str(x)**: Đổi giá trị của x từ dạng số sang dạng dãy ký tự (x phải là biến kiểu dữ liệu số)

5.5. CHƯƠNG TRÌNH KHÔNG CÓ PHẦN NHẬP DỮ LIỆU

Các chương trình có trong phần này không có phần nhập dữ liệu. Để tính toán các chương trình trong phần này sử dụng các dữ liệu có trong hộp văn bản hoặc được tạo ra bằng các đối tượng khác có trong biểu mẫu.

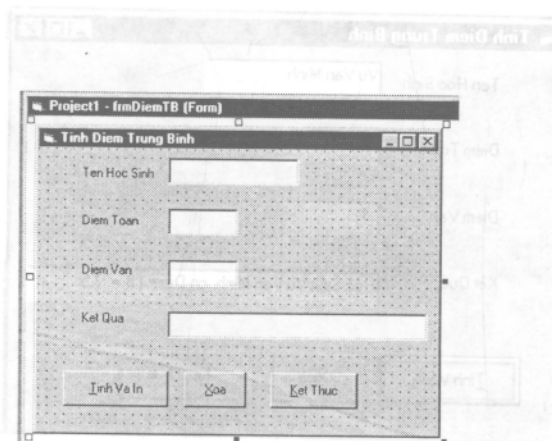
5.5.1. Sử dụng dữ liệu có trong hộp văn bản

Trong các chương trình lập theo cách này các dữ liệu ban đầu tham gia tính toán được người sử dụng ghi trực tiếp vào các hộp văn bản trong thời gian thực hiện chương trình. Các quá trình tính toán được thực hiện với các giá trị có trong các hộp văn bản đó.

Trong mục này ta sẽ nghiên cứu chương trình DiemTB (Điểm Trung Bình)

Giới thiệu chương trình DiemTB

Như tên của nó, chương trình DiemTB dùng để tính điểm trung bình. Màn hình giao diện của chương trình DiemTB có dạng sau đây:

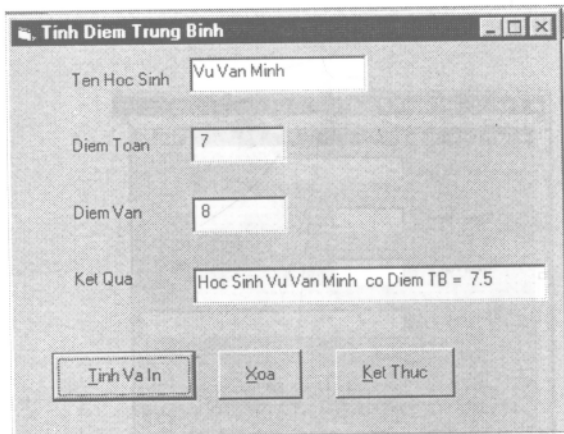


Hình 5.1. Màn hình giao diện của chương trình DiemTB

Màn hình giao diện của chương trình này có 4 nhãn và 4 hộp văn bản. Tiêu đề các nhãn giải thích nội dung dữ liệu có trong hộp văn bản phía phải. Hộp thứ nhất chứa tên học sinh. Hộp thứ hai chứa điểm toán. Hộp thứ ba chứa điểm văn. Chương trình **DiemTB** sẽ tính điểm trung bình 2 môn học này và in điểm trung bình đó kèm theo tên sinh viên trong hộp văn bản thứ tư, là hộp văn bản dùng để chứa kết quả.

Chương trình DiemTB có 3 nút lệnh. Ba nút lệnh này lần lượt có tiêu đề là **T**inh Va In, **X**oa và **K**et Thuc. Tiêu đề ba nút lệnh này nói lên công việc mà chúng sẽ thực hiện.

Khi chạy chương trình DiemTB, sau khi người sử dụng đã nhập dữ liệu vào 3 hộp văn bản đầu tiên, nếu ta nhấp chuột vào nút lệnh “Tinh Va In” thì kết quả thực hiện chương trình sẽ xuất hiện trong hộp văn bản chứa kết quả. Lúc này màn hình giao diện của chương trình có dạng như trong hình 5.2.



Hình 5.2. Màn hình của chương trình DiemTB khi dùng nút lệnh “Tinh Va In”.

Ta sẽ lần lượt thực hiện các bước công việc sau đây để thiết kế chương trình DiemTB.

Bước 1. Tạo một dự án mới cho chương trình DiemTB

Để làm việc này cần phải thực hiện các thao tác sau đây:

- Khởi động VB
- File
- New Project

Một biểu mẫu rỗng xuất hiện trên màn hình.

Bước 2. Lưu dự án mới lên đĩa từ

Để lưu tệp chứa dự án lên đĩa từ ta làm như sau:

- File
- Save Project as

- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (DiemTB)
- OK

Để lưu tệp chứa biểu mẫu lên đĩa từ cần phải thực hiện các thao tác sau đây:

- File
- Save Form1 as
- Xác định thư mục lưu biểu mẫu (c:\baitapvb)
- Xác định tên tệp lưu biểu mẫu (DiemTB)
- OK

Bước 3. Thiết kế giao diện cho chương trình

Trong bước công việc này ta sẽ sử dụng các công cụ trong hộp công cụ để lần lượt tạo ra 4 nhãn, 4 hộp văn bản và 3 nút lệnh của chương trình. Trong chương 2, khi thiết kế chương trình Hello, ta đã nghiên cứu cách tạo ra hộp văn bản và nút lệnh. Việc tạo ra nhãn cũng hoàn toàn tương tự tạo ra nút lệnh. Trong mục này sẽ không trình bày lại các thao tác đó để tránh dài dòng.

Sau khi tạo ra các đối tượng ta phải xác định cho các đối tượng đó những thuộc tính thích hợp. Ta cũng đã biết cách xác định các thuộc tính cho các đối tượng khi thiết kế chương trình Hello trong chương 2.

Bảng 5.1. Các đối tượng của chương trình DiemTB

Số TT	Loại đối tượng	Thuộc tính	Giá trị
1	Biểu mẫu	Name Caption	FrmDiemTB Tinh Diem Trung Binh
2	Nhãn	Name Caption	LblTen Ten SV
3	Nhãn	Name Caption	LblDToan Diem Toan
4	Nhãn	Name Caption	LblDVan Diem Van
5	Nhãn	Name Caption	LblKetQua Ket Qua
6	Text box	Name Text	TxtTen ""
7	Text box	Name Text	TxtDToan ""
8	Text box	Name Text	TxtDVan ""
9	Text box	Name Text	TxtKetQua 0
10	Nút lệnh	Name Caption	CmdKetThuc &Ket Thuc
11	Nút lệnh	Name Caption	CmdXoa &Xoa
12	Nút lệnh	Name Caption	CmdTinhValn &Tinh Va In

Bước 4. Viết các thủ tục - biến cố cho các nút lệnh

Chương trình DiemTB có 3 thủ tục - biến cố sau đây:

```
Private Sub cmdXoa_Click()
    txtTen.Text = " "
    txtDToan.Text = " "
    txtDVan.Text = " "
```



```

        txtKetQua.Text = " "
End Sub

Private Sub cmdKetThuc_Click()
    Beep
    End
End Sub

Private Sub cmdTinhVaIn_Click()
    Dim dtb!
    dtb! = (Val(TxtDToan.Text) + Val(txtDVan.Text)) / 2
    txtKetQua.Text = " Hoc Sinh " + txtTen.Text + " co
    Diem TB = " + Str(dtb!)
End Sub

```

Giải thích chương trình

Ta đã biết tất cả các câu lệnh trong các thủ tục - biến cố này. Tuy nhiên, hãy chú ý đến câu lệnh gán thứ hai của thủ tục - biến cố cuối cùng:

```

txtKetQua.Text = "Hoc Sinh " + txtTen.Text + " co
Diem TB = " + Str(dtb!)

```

Theo câu lệnh này, đầu tiên Visual Basic chuyển đổi giá trị của biến số thực dtb! sang dạng chuỗi kí tự. Sau đó chuỗi vừa thu được sẽ được ghép với 3 chuỗi còn lại để tạo ra kết quả. Sau đó kết quả này được gán cho thuộc tính text của hộp văn bản txtKetQua.

Chú ý: nếu không dùng hàm str(dtb!) để chuyển đổi giá trị của biến số thực dtb! sang dạng chuỗi kí tự thì sẽ gây ra lỗi “Type mismatch”.

5.5.2. Sử dụng thanh cuộn

Chương trình trong mục này sử dụng các thanh cuộn để tạo ra các số nguyên sẽ tham gia tính toán. Các số nguyên này sẽ được đưa vào các hộp văn bản để hiện trên màn hình.

• *Khái niệm thanh cuộn*

Thanh cuộn là một đối tượng trong biểu mẫu được dùng để tạo ra các số nguyên nằm trong một khoảng xác định.

Visual Basic có hai loại thanh cuộn là thanh cuộn ngang (Horizontal Scrollbar), thanh cuộn dọc (Vertical Scrollbar).

• *Tạo ra thanh cuộn trong biểu mẫu*

Để tạo ra thanh cuộn trong biểu mẫu ta làm như sau:

- Nháy đúp vào công cụ tạo thanh cuộn trong hộp công cụ ;
- Đưa thanh cuộn đến vị trí cần thiết;
- Xác định giá trị cho các thuộc tính của thanh cuộn:

• *Các thuộc tính của thanh cuộn*

Các thuộc tính của thanh cuộn được cho trong bảng sau đây:

Bảng 5.5. Các thuộc tính của thanh cuộn

Thuộc tính	Ý nghĩa, tác dụng
Enabled	Xác định tính khả ứng của các thanh cuộn .
Height	Xác định chiều cao của thanh cuộn.
LargeChange	Xác định số đơn vị sẽ thay đổi khi người sử dụng nháy chuột bên trong thanh cuộn (ngầm định là 1).
Left	Xác định khoảng cách trái của thanh cuộn.
Max	Xác định giá trị lớn nhất của thanh cuộn (1 -> 32767).

Thuộc tính	Ý nghĩa, tác dụng
	ngầm định là 1.
Min	Xác định giá trị nhỏ nhất của thanh cuộn (1 -> 32767), ngấm định là 1.
Name	Xác định tên thanh cuộn (Ba kí tự đầu là hsb cho thanh cuộn ngang, vsb cho thanh cuộn dọc).
SmallChange	Xác định số đơn vị sẽ thay đổi khi người sử dụng nháy chuột vào mũi tên ở cuối thanh cuộn.
Top	Xác định khoảng cách trên của thanh cuộn.
Value	Cho một số nguyên xác định bởi thanh cuộn, ngấm định là 1. Khi người sử dụng nháy chuột vào bên trong thanh cuộn hoặc nháy chuột vào mũi tên ở cuối thanh cuộn hoặc dùng chuột thay đổi vị trí con chạy trong thanh cuộn thì giá trị này sẽ thay đổi.
Visible	Xác định tính thấy được của thanh cuộn.
Width	Xác định chiều rộng của thanh cuộn.

Thanh cuộn có thể đáp ứng biến cố là **change** (thay đổi). Biến cố này sẽ xảy ra khi người sử dụng thay đổi giá trị của thanh cuộn bằng một trong các cách:

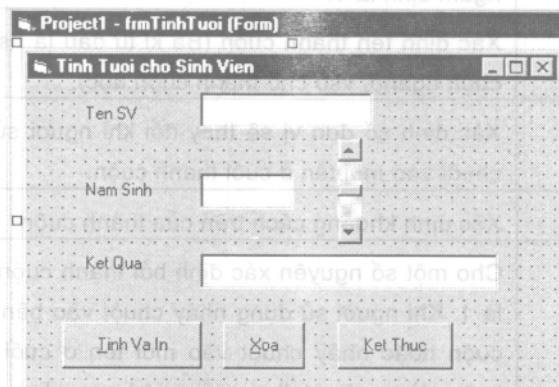
- Nháy chuột vào bên trong thanh cuộn;
- Nháy chuột vào mũi tên ở cuối thanh cuộn;
- Dùng chuột thay đổi vị trí con chạy trong thanh cuộn.

Các hành động này của người sử dụng làm thay đổi giá trị của thanh cuộn.

Sau đây ta nghiên cứu một chương trình có sử dụng thanh cuộn. Chương trình này có tên là **TinhTuoi**.

Giới thiệu chương trình *TinhTuoi*

Như tên của nó, chương trình *TinhTuoi* dùng để tính tuổi. Màn hình giao diện của chương trình *TinhTuoi* có dạng sau đây:



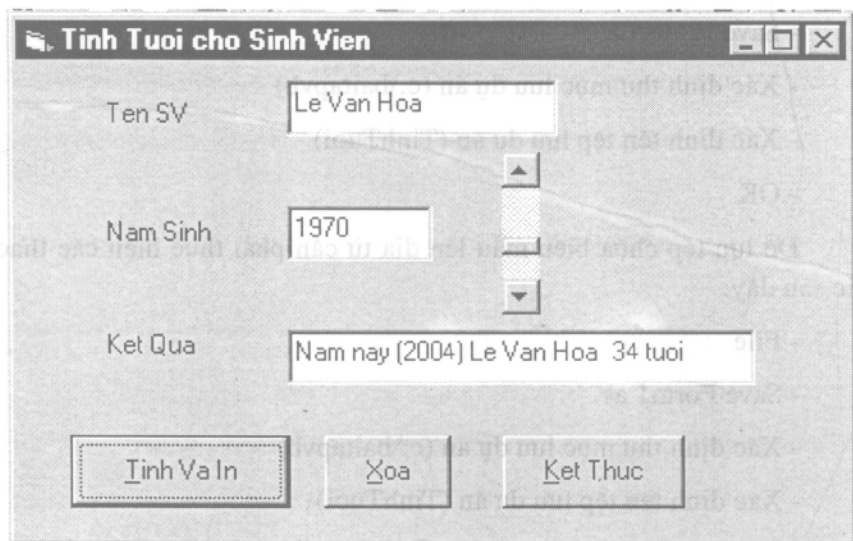
Hình 5.3. Màn hình giao diện của chương trình *TinhTuoi*

Màn hình giao diện của chương trình này có 3 nhãn và 3 hộp văn bản. Tiêu đề các nhãn giải thích nội dung dữ liệu có trong hộp văn bản phía phải. Hộp thứ nhất chứa tên sinh viên. Hộp thứ hai chứa năm sinh. Hộp văn bản thứ ba chứa kết quả. Chương trình *TinhTuoi* sẽ tính tuổi của sinh viên và in tuổi đó kèm theo tên sinh viên trong hộp văn bản thứ ba.

Chương trình *TinhTuoi* có một thanh cuộn dọc. Thanh cuộn này nằm ở phía bên phải của biểu mẫu, dùng để xác định năm sinh của sinh viên. Khi người sử dụng thay đổi giá trị của thanh cuộn thì giá trị của thanh cuộn sẽ xuất hiện trong hộp văn bản thứ hai. Người sử dụng không phải ghi năm sinh của sinh viên vào hộp văn bản này.

Chương trình *TinhTuoi* có 3 nút lệnh. Ba nút lệnh này lần lượt có tiêu đề là Tinh Va In, Xoa và Ket Thuc. Tiêu đề ba nút lệnh này nói lên công việc mà chúng sẽ thực hiện.

Khi chạy chương trình TinhTuoi, sau khi người sử dụng đã nhập tên sinh viên vào hộp văn bản đầu tiên và thay đổi giá trị của thanh cuộn, nếu ta nhấn chuột vào nút lệnh “Tinh Va In” thì kết quả thực hiện chương trình sẽ xuất hiện trong hộp văn bản chứa kết quả. Lúc này màn hình giao diện của chương trình có dạng như trong hình 5.4.



Hình 5.4. Màn hình của chương trình TinhTuoi khi dùng nút lệnh “Tinh Va In”.

Ta sẽ lần lượt thực hiện các bước công việc sau đây để thiết kế chương trình TinhTuoi.

Bước 1. Tạo một dự án mới cho chương trình TinhTuoi

Để làm việc này cần phải thực hiện các thao tác sau đây:

- Khởi động VB
- File
- New Project

Một biểu mẫu rỗng xuất hiện trên màn hình.

Bước 2. Lưu dự án mới lên đĩa từ

Để lưu tệp chứa dự án lên đĩa từ ta làm như sau:

- File
- Save Project as
- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (TinhTuoi)
- OK

Để lưu tệp chứa biểu mẫu lên đĩa từ cần phải thực hiện các thao tác sau đây:

- File
- Save Form1 as
- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (TinhTuoi)
- OK

Bước 3. Thiết kế giao diện cho chương trình

Trong bước công việc này ta sẽ sử dụng các công cụ trong hộp công cụ để lần lượt tạo ra 3 nhãn, 3 hộp văn bản và 3 nút lệnh của chương trình. Khi thiết kế các chương trình trong các phần trước ta đã biết cách tạo ra hộp văn bản, nút lệnh, và nhãn. Thao tác tạo ra đối tượng thanh cuộn cũng hoàn toàn tương tự.

Sau khi tạo ra các đối tượng ta phải xác định cho các đối tượng đó những thuộc tính thích hợp. Ta cũng đã biết cách thực hiện các công việc này.

Bảng 5.1. Các đối tượng của chương trình TinhTuoi

Số TT	Loại đối tượng	Thuộc tính	Giá trị
1	Biểu mẫu	Name Caption	FrmTinhTuoi Tinh Tuoi Cho Sinh Viên
2	Nhãn	Name Caption	LblTen Ten SV
3	Nhãn	Name Caption	LblNamSinh NamSinh
4	Nhãn	Name Caption	LblKetQua Ket Qua
5	Text box	Name Text	TxtTen ""
6	Text box	Name Text	TxtNamSinh ""
7	Text box	Name Text	TxtKetQua ""
8	Nút lệnh	Name Caption	CmdKetThuc &Ket Thuc
9	Nút lệnh	Name Caption	CmdXoa &Xoa
10	Nút lệnh	Name Caption	CmdTinhVaIn &Tinh Va In
11	Thanh cuộn	Name LargeChange Max Min SmallChange Value	VsbNamSinh 10 2004 1903 1 1950

Bước 4. Viết các thủ tục - biến cố cho các nút lệnh

Chương trình TinhTuoi có 4 thủ tục - biến cố sau đây:

```
Private Sub cmdXoa_Click()  
    txtTen.Text = " "  
    TxtNamSinh.Text = " "  
    txtKetQua.Text = " "  
End Sub  
  
Private Sub cmdKetThuc_Click()  
    Beep  
    End  
End Sub  
  
Private Sub cmdTinhVaIn_Click()  
    Dim Tuoi%  
    Tuoi% = 2004 - Val(TxtNamSinh.Text)  
    txtKetQua.Text = "Nam nay (2004) " + txtTen.Text  
    + Str(Tuoi%) + " tuoi "  
End Sub  
  
Private Sub VSbNamSinh_Change()  
    TxtNamSinh. Text = VSbNamSinh.Value  
    txtKetQua.Text = " "  
End Sub
```

Giải thích chương trình

Ba thủ tục - biến cố đầu tiên đã quen thuộc đối với chúng ta. Thủ tục - biến cố thứ tư Private Sub VSbNamSinh_Change() sẽ được thực hiện khi người sử dụng thay đổi giá trị của thanh cuộn. Khi đó máy

tính thực hiện:

- Gán giá trị của thanh cuộn cho thuộc tính Text của hộp văn bản txtNamSinh;

- Xóa kết quả có trong hộp văn bản chứa kết quả.

Hai câu lệnh gán trong thủ tục - biến cố VSbNamSinh_Change() thực hiện hai công việc này.

Chạy thử chương trình TinhTuoi

Hãy gõ phím F5 để thực hiện chương trình TinhTuoi. Nhập tên sinh viên vào hộp văn bản thứ nhất. Thay đổi giá trị của thanh cuộn để xác định năm sinh. Nháy chuột vào nút lệnh “Tinh Va In” để xem kết quả. Xem xong nháy chuột vào nút lệnh “Ket Thuc” để kết thúc chương trình.

5.6. LẬP TRÌNH CÓ PHẦN NHẬP - XUẤT DỮ LIỆU

Chương trình trong mục này cho phép người sử dụng nhập dữ liệu ban đầu từ bàn phím. Các kết quả tính toán được sẽ được in trên màn hình của biểu mẫu. Chương trình Trong phần này sẽ không sử dụng các hộp văn bản để chứa các dữ liệu ban đầu và dữ liệu kết quả.

4.6.1. Hàm nhập dữ liệu

Việc nhập dữ liệu do người sử dụng gõ vào từ bàn phím được thực hiện bằng hàm nhập dữ liệu InputBox(). Hàm này có qui cách sau đây:

InputBox(LoiNhac\$, Tieude\$, NgamDinh\$)

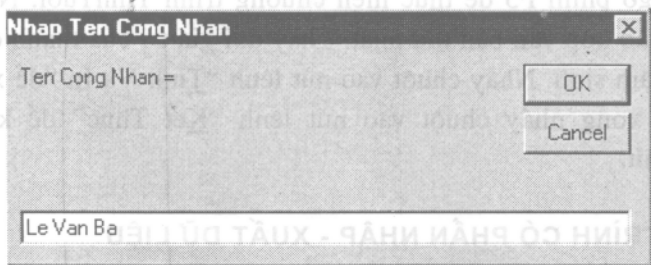
Ở đây:

LoiNhac\$ là dãy kí tự được dùng để hướng dẫn người sử dụng nhập dữ liệu.

TieuDe\$ là dãy kí tự được dùng làm tiêu đề của hộp nhập dữ liệu. Hộp này xuất hiện mỗi khi hàm nhập dữ liệu được thực hiện. Đây là thành phần tùy chọn.

NgamDinh\$ là dãy kí tự được dùng làm giá trị ngầm định của dữ liệu được nhập. Đây là thành phần tùy chọn.

Tác dụng: Hàm này làm xuất hiện một hộp nhập dữ liệu để người sử dụng nhập một dãy kí tự từ bàn phím (Xem hình 5.5) . Khi người sử dụng nhập xong (bằng cách gõ phím ENTER hoặc chọn OK) hàm này sẽ trả lại dãy kí tự vừa nhập.



Hình 5.5. Hộp nhập dữ liệu

Chú ý: Hàm InputBox() luôn luôn trả lại một dãy kí tự cho dù người sử dụng chỉ nhập vào các số. Muốn dùng các số (dưới dạng kiểu dãy kí tự) này để tính toán phải dùng hàm Val() để đổi chúng sang kiểu số.

5.6.2. Câu lệnh PRINT

Câu lệnh in được dùng để in dữ liệu trong biểu mẫu. Câu lệnh này có qui cách sau đây:

Print < biểu thức 1> [, <biểu thức 2>]...

Theo lệnh này máy tính giá trị các biểu thức và in các giá trị đó trên biểu mẫu. Nếu 2 biểu thức cách nhau dấu phẩy, các giá trị được in cách nhau một dấu cách. Nếu 2 biểu thức cách nhau dấu chấm phẩy

(;), thì giá trị của biểu thức sau sẽ được in ở vùng tiếp theo. Một vùng gồm 14 kí tự đi liền nhau trên một dòng của biểu mẫu.

Chú ý: Khi soạn thảo chương trình có thể gõ dấu hỏi “?” thay cho gõ từ “Print”. Visual Basic sẽ tự động thay dấu “?” bằng từ “Print”.

5.3.5. Câu lệnh CLS

Câu lệnh này có qui cách sau đây:

CLS

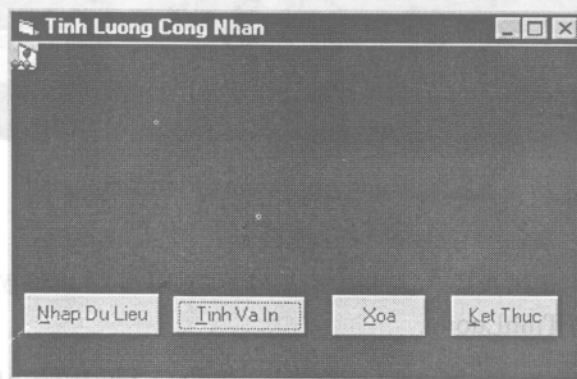
Theo câu lệnh này máy xóa các dữ liệu được in bằng câu lệnh Print trong biểu mẫu.

Trong phần này ta sẽ thiết kế chương trình **TinhLuong**.

Giới thiệu chương trình TinhLuong

Như tên của nó, chương trình TinhLuong dùng để tính lương cho công nhân. Bài toán tính lương công nhân có nội dung sau đây: cho biết Tên, tiền lương, tiền thưởng, tiền nợ của một công nhân. Hãy viết chương trình tính và in số tiền công nhân này được lĩnh.

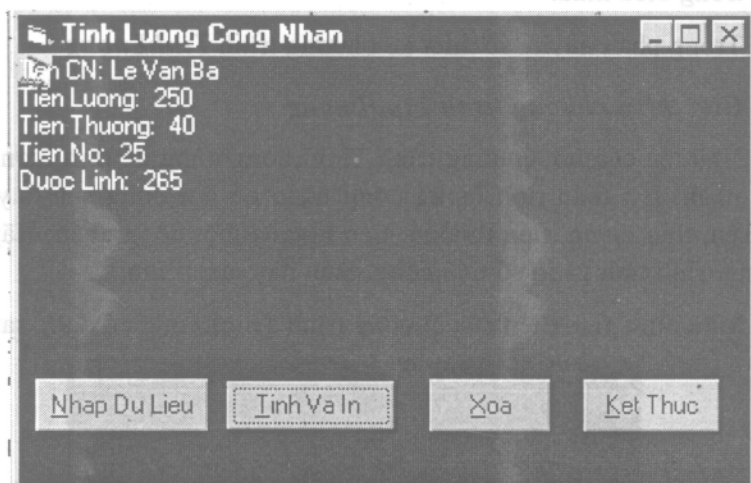
Màn hình giao diện của chương trình TinhLuong có dạng sau:



Hình 5.6. Màn hình giao diện của chương trình TinhLuong

Màn hình giao diện của chương trình này không có nhãn, không có hộp văn bản, mà chỉ có bốn nút lệnh. Bốn nút lệnh này lần lượt có tiêu đề là Nhap Du Lieu, Tinh Va In, Xoa và Ket Thuc. Tiêu đề bốn nút lệnh này nói lên công việc mà chúng sẽ thực hiện.

Trong thời gian chạy chương trình TinhLuong, khi sử dụng nút lệnh Nhap Du Lieu thì các hộp nhập dữ liệu sẽ xuất hiện để người sử dụng có thể nhập tên, tiền lương, tiền thưởng, tiền nợ của công nhân. Các dữ liệu đó sẽ được gán cho các biến. Nếu ta nhấn chuột vào nút lệnh “Tinh Va In” thì máy sẽ tính và in kết quả trên màn hình của biểu mẫu. Lúc này màn hình của chương trình có dạng sau:



Hình 5.7. Màn hình của chương trình TinhLuong
khi dùng nút lệnh “Tinh Va In”

Ta sẽ lần lượt thực hiện các bước công việc sau đây để thiết kế chương trình TinhLuong.

Bước 1. Tạo một dự án mới cho chương trình TinhLuong

Để làm việc này cần phải thực hiện các thao tác sau đây:

- Khởi động VB
- File
- New Project

Một biểu mẫu rỗng xuất hiện trên màn hình.

Bước 2. Lưu dự án mới lên đĩa từ

Để lưu tệp chứa dự án lên đĩa từ ta làm như sau:

- File
- Save Project as
- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (TinhLuong)
- OK

Để lưu tệp chứa biểu mẫu lên đĩa từ cần phải thực hiện các thao tác sau đây:

- File
- Save Form1 as
- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (TinhLuong)
- OK

Bước 3. Thiết kế giao diện cho chương trình

Trong bước công việc này ta sẽ sử dụng các công cụ trong hộp công cụ để lần lượt tạo ra 4 nút lệnh. Sau đó phải xác định cho các thuộc tính thích hợp cho các nút lệnh đó. Ta đã biết cách thực hiện các công việc này.

Bảng 5.1. Các đối tượng của chương trình TinhLuong

Số TT	Loại đối tượng	Thuộc tính	Giá trị
1	Biểu mẫu	Name Caption	FrmTinhLuong Tinh Luong Cong Nhan
2	Nút lệnh	Name Caption	CmdNhap &Nhap Du Lieu
3	Nút lệnh	Name Caption	CmdTinhValn &Tinh Va In
4	Nút lệnh	Name Caption	CmdKetThuc &Ket Thuc
5	Nút lệnh	Name Caption	CmdXoa &Xoa

Bước 4. Viết các thủ tục - biến cố cho các nút lệnh

Chương trình TinhLuong có 4 thủ tục - biến cố sau đây:

Dim ten\$, TienLuong!, TienThuong!, TienNo!

Private Sub cmdNhap_Click()

ten\$ = InputBox("Ten Cong Nhan = ")

TienLuong! = Val(InputBox("Tien Luong = "))

TienThuong! = Val(InputBox("Tien Thuong = "))

TienNo! = Val(InputBox("Tien No = "))

End Sub

Private Sub cmdXoa_Click()

Cls

End Sub

Private Sub cmdKetThuc_Click()

```

    Beep
End
End Sub

Private Sub cmdTinhVaIn_Click()
    Dim DuocLinh!
    DuocLinh! = TienLuong + TienThuong! - TienNo!
    Print "Ten CN: "; ten$
    Print "Tien Luong: "; TienLuong!
    Print "Tien Thuong: "; TienThuong!
    Print "Tien No: "; TienNo!
    Print "Duoc Linh: "; DuocLinh!
End Sub

```

Giải thích chương trình

Dòng lệnh đầu tiên của chương trình này khai báo 4 biến là `ten$`, `TienLuong!`, `TienThuong!`, `TienNo!`. Các biến này được khai báo ở đầu chương trình, nên là những biến toàn cục. Vì vậy các biến này có thể được sử dụng trong tất cả các thủ tục - biến cố của chương trình này.

Thủ tục - biến cố `cmdNhap_Click()` dùng để nhập dữ liệu ban đầu từ bàn phím. Vì ta muốn nhập giá trị của 4 biến, nên ta phải sử dụng hàm `InputBox()` bốn lần. Các giá trị của hàm này được gán cho các biến tương ứng bằng các câu lệnh gán cố trong thủ tục - biến cố này.

Trong thủ tục - biến cố `cmdTinhVaIn_Click()` có câu lệnh khai báo biến `DuocLinh!`. Biến này được khai báo trong một thủ tục - biến cố nên nó là biến cục bộ. Biến `DuocLinh!` chỉ có ý nghĩa trong

thủ tục - biến cố này. Thủ tục - biến cố `cmdTinhVaIn_Click()` sẽ được thực hiện khi người sử dụng sử dụng nút lệnh Tinh Va In. Khi đó máy tính thực hiện các công việc sau đây:

- Tính giá trị của biến `DuocLinh!` ;
- In trên biểu mẫu 4 dòng. Đây là kết quả thực hiện chương trình này.

Thủ tục - biến cố `cmdXoa_Click()` của chương trình này hơi khác thủ tục - biến cố này trong các chương trình trước. Trong thủ tục - biến cố này không có các câu lệnh gán chuỗi rỗng cho thuộc tính `Text` của các hộp văn bản, mà chỉ có một câu lệnh `Cls`. Câu lệnh `Cls` này xoá các dữ liệu đã được in bằng câu lệnh `Prin` trong biểu mẫu.

Thủ tục - biến cố `cmdKetThuc_Click()` của chương trình này hoàn toàn giống thủ tục - biến cố này của các chương trình trước.

Chạy thử chương trình TinhLuong

Hãy gõ phím `F5` để thực hiện chương trình `TinhLuong`. Sử dụng nút lệnh “Nhap Du Lieu” để nhập các dữ liệu ban đầu. Nháy chuột vào nút lệnh “Tinh Va In “ để xem kết quả. Dùng nút lệnh “Xoa” để xoá các dữ liệu đã được in trong biểu mẫu. Nháy chuột vào nút lệnh “Ket Thuc” để kết thúc chương trình.

Nhận xét: màn hình giao diện của chương trình `TinhLuong` này có nhiều điểm khác so với màn hình giao diện của các chương trình trước. Chương trình `TinhLuong` không sử dụng các nhãn để giải thích dữ liệu và không dùng các hộp văn bản để chứa dữ liệu. Màn hình giao diện kiểu này rất hay được sử dụng trong các chương trình ở các chương tiếp theo. Vì vậy mong bạn đọc đặc biệt lưu tâm đến kiểu màn hình giao diện này.

CÂU HỎI ÔN TẬP VÀ BÀI TẬP

Lý thuyết

1. Hãy trình bày các quy định về cách viết câu lệnh trong chương trình Visual Basic.

2. Hãy trình bày các cách viết dòng chú thích trong chương trình Visual Basic.

3. Hãy trình bày các quy tắc chuyển đổi kiểu dữ liệu khi thực hiện câu lệnh gán và khi tính giá trị biểu thức.

4. Có được phép cộng số với chuỗi 4 kí tự không? Nếu có thì tại sao và nếu không thì tại sao? Trường hợp nào được phép, trường hợp nào không được phép?

5. Thanh cuộn là gì? Thanh cuộn được dùng để làm gì?

Bài tập

1. Viết chương trình (có sử dụng thanh cuộn) để tính và in diện tích hình chữ nhật khi biết độ dài 2 cạnh.

2. Cho biết tên và điểm toán, lý và hoá của một sinh viên. Hãy viết chương trình (không sử dụng các nhãn và các hộp văn bản) để tính và in điểm TB cho sinh viên đó.

3. Hãy viết chương trình “Hộp văn bản kỳ ảo” có các đặc điểm sau đây: chương trình có một hộp văn bản và 2 thanh cuộn. Khi người sử dụng thay đổi giá trị của các thanh cuộn thì diện tích của hộp văn bản cũng thay đổi theo (to ra hoặc nhỏ đi).

CÁC CẤU TRÚC LỰA CHỌN

Trong quá trình giải nhiều bài toán thường xảy ra trường hợp là có nhiều cách tính toán để tiếp tục giải bài toán. Chọn cách tính toán nào trong các cách tính toán đó phụ thuộc vào việc một điều kiện nào đấy có được thỏa mãn hay không. Quá trình tính toán có đặc điểm như vậy được gọi là quá trình tính toán lựa chọn hoặc quá trình tính toán phân nhánh.

Chương này nghiên cứu cách lập chương trình giải các bài toán có quá trình tính toán lựa chọn hoặc phân nhánh trong Visual Basic. Các cụm từ “chương trình phân nhánh” và “chương trình lựa chọn” có nghĩa như nhau và được dùng thay thế cho nhau.

6.1. KHÁI NIỆM CHƯƠNG TRÌNH PHÂN NHÁNH (LỰA CHỌN)

Chương trình giải bài toán có chứa quá trình tính toán phân nhánh được gọi là chương trình phân nhánh.

Có thể định nghĩa chương trình phân nhánh theo cách khác: Chương trình phân nhánh là những chương trình mà ở một chỗ nào đó chương trình cần chọn một cách tính toán trong một số cách tính toán có thể có tùy thuộc vào việc một điều kiện nào đấy có được thỏa mãn hay không.

Quá trình tính học bổng trên cơ sở điểm trung bình (ĐTB) theo

công thức dưới đây là quá trình tính toán phân nhánh.

$$\text{Học bổng} = \begin{cases} 0 & \text{nếu ĐTB} < 7.5 \\ 80 & \text{nếu } 7.5 \leq \text{ĐTB} < 9.0 \\ 120 & \text{nếu ĐTB} \geq 9.0 \end{cases}$$

Chương trình tính học bổng theo công thức này là chương trình phân nhánh.

Visual Basic dùng 2 câu lệnh chuyên dùng để lập các chương trình phân nhánh là câu lệnh IF và câu lệnh SELECT. Những câu lệnh này được gọi là câu lệnh điều kiện. Sau đây ta sẽ nghiên cứu các câu lệnh này và sử dụng chúng để viết các chương trình phân nhánh.

6.2. CÂU LỆNH IF

Câu lệnh này dùng để kiểm tra một điều kiện và điều khiển thực hiện một khối lệnh này hay một khối lệnh khác tùy thuộc vào việc điều kiện có được thoả mãn hay không. Khối lệnh là một hoặc một số câu lệnh đi liền nhau. Khối lệnh có thể chiếm một hoặc nhiều dòng lệnh.

6.2.1. Câu lệnh If đơn giản

Câu lệnh này có qui cách như sau:

IF *điều kiện* THEN

Khối lệnh

END IF

Tác dụng: Theo lệnh này máy sẽ kiểm tra *điều kiện*. Nếu *điều kiện* được thoả mãn (cho giá trị True) máy sẽ thực hiện *Khối lệnh*, là các câu lệnh nằm giữa THEN và END IF. Ngược lại, nếu *điều kiện* không được thoả mãn (cho giá trị False) thì máy sẽ bỏ qua toàn bộ lệnh IF.

Ta thấy rằng câu lệnh IF đơn giản chỉ có một mệnh đề. Đó là mệnh đề IF. Câu lệnh IF dạng này cho phép thực hiện hoặc không thực hiện một khối lệnh.

Ví dụ: Cho 2 số a, b - thủ tục biến cố tìm và in số lớn nhất trong 2 số a, b có dạng như sau:

```
Dim a!, b!  
Private Sub cmdSoMax_Click()  
    Dim Max!  
    Max = a  
    If    Max < b Then  
        Max = b  
    End If  
    Cls  
    ? " So Lon Nhat = ", Max  
End Sub
```

Trong thủ tục - biến cố này ta đã sử dụng một câu lệnh IF đơn giản để thay đổi giá trị của biến Max nếu cần. Nhờ vậy mà Max luôn luôn chứa số lớn nhất trong 2 số a, b.

6.2.2. Câu lệnh If đầy đủ

Câu lệnh này có qui cách sau đây:

IF *điều kiện* THEN

Khối lệnh 1

ELSE

Khối lệnh 2

END IF

Tác dụng: Theo lệnh này máy sẽ kiểm tra *điều kiện*. Nếu *điều*

kiện được thoả mãn (cho giá trị True) máy sẽ thực hiện *Khối lệnh 1*, là các lệnh nằm giữa THEN và ELSE. Ngược lại, nếu *điều kiện* không được thoả mãn (cho giá trị False) thì máy sẽ thực hiện *Khối lệnh 2*, là các câu lệnh nằm giữa ELSE và END IF.

Ta thấy rằng câu lệnh IF đầy đủ có hai mệnh đề. Mệnh đề thứ nhất là mệnh đề IF. Mệnh đề thứ hai là mệnh đề ELSE. Câu lệnh IF dạng này cho phép thực hiện một trong 2 khối lệnh tùy thuộc vào việc điều kiện trong lệnh có được thoả mãn hay không.

Ví dụ: Thủ tục biến cố tìm số lớn nhất trong 2 số a, b có thể được viết theo cách khác như sau:

```
Dim a!, b!  
Private Sub cmdSoMax_Click()  
    Dim Max!  
    If a > b Then  
        MAX = a  
    Else  
        MAX = b  
    End If  
    Cls  
    ? " So LN = ", Max  
End Sub
```

Trong thủ tục - biến cố này ta đã sử dụng 1 câu lệnh IF đầy đủ để tính số lớn nhất trong 2 số a, b. Hai thủ tục - biến cố trên cho kết quả giống nhau.

Các chú ý:

- Mỗi lệnh IF phải có một END IF tương ứng.
- Mỗi một trong các thành phần sau đây:

IF *điều kiện* THEN

ELSE

END IF

phải chiếm riêng một dòng lệnh. Nếu không sẽ gây lỗi.

Ví dụ: câu lệnh IF sau đây là hợp lệ:

```
If    a < b Then
```

```
    max = b
```

```
Else
```

```
    max = a
```

```
End If
```

Còn câu lệnh IF sau đây:

```
If a < b Then max = b Else max = a End If
```

là không hợp lệ. Gặp câu lệnh này Visual Basic đưa ra thông báo lỗi sau đây:

“Compile error “

“ End If without block If “

6.2.3. Câu lệnh If lồng nhau

Một câu lệnh IF có chứa một câu lệnh IF khác được gọi là câu lệnh IF lồng nhau. Câu lệnh IF đầu tiên được gọi là câu lệnh IF ngoài. Câu lệnh IF thứ hai được gọi là câu lệnh IF trong. Khi đó mệnh đề ELSE (nếu có) sẽ tương ứng với mệnh đề IF gần nhất chưa có mệnh đề ELSE. Có thể xảy ra các trường hợp sau đây:

Trường hợp 1: câu lệnh IF trong nằm ở mệnh đề IF của câu lệnh IF ngoài. Trong trường hợp này câu lệnh IF lồng nhau có dạng:

```

IF điều kiện 1 THEN
    IF điều kiện 2 THEN
        Khối lệnh 1
    END IF
ELSE
    Khối lệnh 2
END IF

```

Trong trường hợp này mệnh đề ELSE tương ứng với mệnh đề IF điều kiện 1. Khối lệnh 1 sẽ được thực hiện nếu cả điều kiện 1 lẫn điều kiện 2 được thoả mãn (có giá trị True). Khối lệnh 2 sẽ được thực hiện nếu điều kiện 1 không được thoả mãn (có giá trị False).

Trường hợp 2: câu lệnh IF trong nằm ở mệnh đề ELSE của câu lệnh IF ngoài. Trong trường hợp này câu lệnh IF lồng nhau có dạng:

```

IF điều kiện 1 THEN
    Khối lệnh 1
ELSE
    IF điều kiện 2 THEN
        Khối lệnh 2
    END IF
END IF

```

Trong trường hợp này mệnh đề ELSE cũng tương ứng với mệnh đề IF điều kiện 1. Khối lệnh 1 sẽ được thực hiện nếu điều kiện 1 được thoả mãn. Khối lệnh 2 sẽ được thực hiện nếu điều kiện 1 không được thoả mãn nhưng điều kiện 2 lại được thoả mãn.

Trường hợp 3: Câu lệnh IF trong nằm ở cả 2 mệnh đề ELSE của

câu lệnh IF ngoài. Trong trường hợp này câu lệnh IF lồng nhau có dạng:

```
IF điều kiện 1 THEN
    IF điều kiện 2 THEN
        Khối lệnh 1
    END IF
ELSE
    IF điều kiện 3 Then
        Khối lệnh 2
    END IF
END IF
```

Trong trường hợp này mệnh đề ELSE cũng vẫn tương ứng với mệnh đề IF điều kiện 1. Khối lệnh 1 sẽ được thực hiện nếu điều kiện 1 và điều kiện 2 được thoả mãn. Khối lệnh 2 sẽ được thực hiện nếu điều kiện 1 không được thoả mãn và điều kiện 2 được thoả mãn.

Trong các ví dụ trên thì các câu lệnh IF trong đều là các câu lệnh IF đơn giản. Tuy nhiên các câu lệnh IF này cũng có thể là các câu lệnh IF đầy đủ và có thể chứa các câu lệnh IF bên trong nó. Nghĩa là các câu lệnh IF này cũng có thể là các câu lệnh IF lồng nhau. Cứ như vậy các câu lệnh IF sẽ có cấu trúc rất phức tạp.

Ví dụ: Sau đây là các câu lệnh IF lồng nhau dùng để tính học bổng theo công thức đã cho trong mục 6.1. trên đây.

Câu lệnh IF lồng nhau thứ nhất:

```
If DiemTrungBinh! < 7.5 Then
    HocBong% = 0
Else
```



```

If DiemTrungBinh! < 9 Then
    HocBong%= 120
Else
    HocBong%= 180
End If
End if

```

Trong câu lệnh IF lồng nhau này thì câu lệnh IF trong nằm ở mệnh đề ELSE của câu lệnh IF ngoài.

Câu lệnh IF lồng nhau thứ hai:

```

If DiemtrungBinh! >= 7.5 Then
    If DiemtrungBinh! >= 9 Then
        HocBong% = 180
    Else
        HocBong% = 120
    End If
Else
    HocBong% = 0
End If

```

Trong câu lệnh IF lồng nhau này thì câu lệnh IF trong nằm ở mệnh đề IF của câu lệnh IF ngoài.

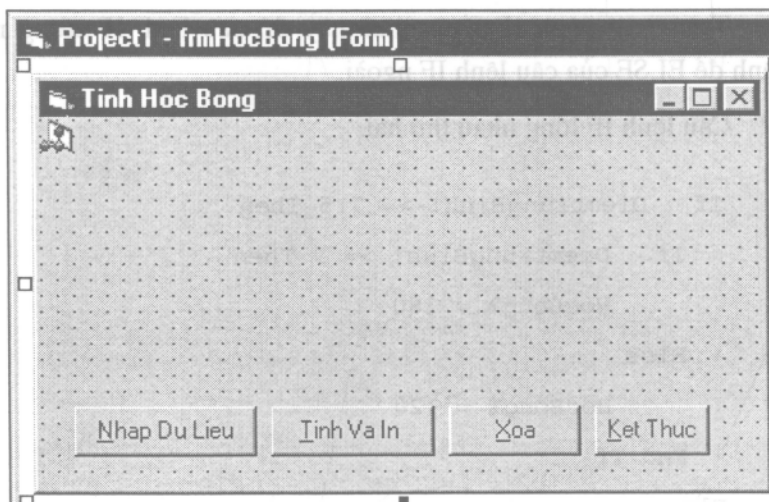
Hai câu lệnh IF nêu trên cho kết quả giống nhau.

Trong phần này ta sẽ thiết kế cụ thể chương trình *HocBong* (Học Bổng).

Giới thiệu chương trình HocBong

Chương trình HocBong dùng để tính học bổng cho sinh viên. Bài toán tính học bổng cho sinh viên có nội dung sau đây: cho biết Tên, điểm trung bình của một sinh viên. Hãy viết chương trình tính và in học bổng cho sinh viên này theo công thức đã nêu trong phần 6.1. ở trên.

Màn hình giao diện của chương trình HocBong có dạng sau đây:

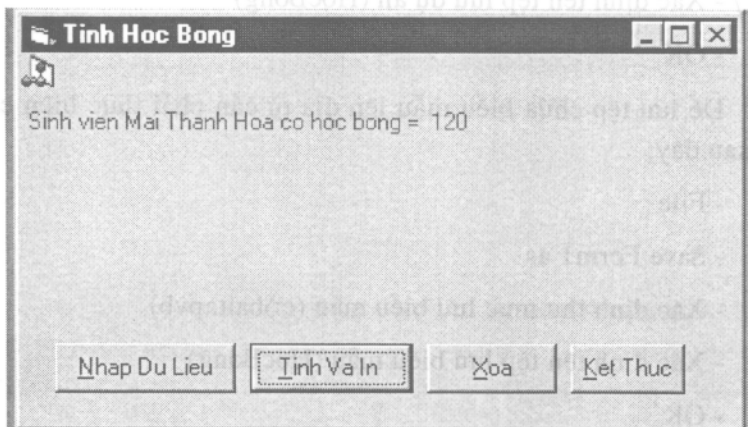


Hình 6.1. Màn hình giao diện của chương trình HocBong

Màn hình giao diện của chương trình này chỉ có một biểu mẫu và bốn nút lệnh. Bốn nút lệnh này lần lượt có tiêu đề là Nhập Du Lieu, Tính Va In, Xoa và Ket Thuc. Tiêu đề bốn nút lệnh này nói lên công việc mà chúng sẽ thực hiện.

Màn hình giao diện gồm có một biểu mẫu và 4 nút lệnh như màn hình giao diện của chương trình HocBong sẽ được sử dụng rất nhiều lần sau này. Ta sẽ gọi màn hình giao diện này là màn hình giao diện mẫu.

Trong thời gian chạy chương trình HocBong, khi sử dụng nút lệnh Nhap Du Lieu thì các hộp nhập dữ liệu sẽ xuất hiện để người sử dụng có thể nhập tên, điểm trung bình của sinh viên. Các dữ liệu đó sẽ được gán cho các biến tương ứng. Nếu ta nhấp chuột vào nút lệnh “Tinh Va In” thì máy sẽ tính và in học bổng của sinh viên trên màn hình của biểu mẫu. Lúc này màn hình của chương trình có dạng như trong hình 6.2.



Hình 6.2. Màn hình của chương trình HocBong khi dùng nút lệnh “Tinh Va In”.

Ta sẽ lần lượt thực hiện các bước công việc sau đây để thiết kế chương trình chương trình HocBong.

Bước 1. Tạo một dự án mới cho chương trình HocBong

Để làm việc này cần phải thực hiện các thao tác sau đây:

- Khởi động VB
- File
- New Project

Một biểu mẫu rỗng xuất hiện trên màn hình.

Bước 2. Lưu dự án mới lên đĩa từ

Để lưu tệp chứa dự án lên đĩa từ ta làm như sau:

- File
- Save Project as
- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (HocBong)
- OK

Để lưu tệp chứa biểu mẫu lên đĩa từ cần phải thực hiện các thao tác sau đây:

- File
- Save Form1 as
- Xác định thư mục lưu biểu mẫu (c:\baitapvb)
- Xác định tên tệp lưu biểu mẫu (HocBong)
- OK

Bước 3. Thiết kế giao diện cho chương trình

Trong bước công việc này ta sẽ sử dụng các công cụ trong hộp công cụ để lần lượt tạo ra 4 nút lệnh. Sau đó phải xác định cho các thuộc tính thích hợp cho các nút lệnh đó. Ta đã biết cách thực hiện các công việc này.

Bảng 6.1. Các đối tượng của chương trình HocBong

Số TT	Loại đối tượng	Thuộc tính	Giá trị
1	Biểu mẫu	Name	FrmHocBong
		Caption	Tình Hoc Bong
2	Nút lệnh	Name	CmdNhap
		Caption	&Nhap Du Lieu

Số TT	Loại đối tượng	Thuộc tính	Giá trị
3	Nút lệnh	Name Caption	CmdTinhVaIn &Tinh Va In
4	Nút lệnh	Name Caption	CmdKetThuc &Ket Thuc
5	Nút lệnh	Name Caption	CmdXoa &Xoa

Bước 4. Viết các thủ tục - biến cố cho các nút lệnh

Phần mã lệnh của chương trình HocBong có dạng sau đây:

```
Dim ten$, DiemTrungBinh!

Private Sub cmdNhap_Click()
    ten$ = InputBox(" Ten Sinh Vien = ")
    DiemTrungBinh! = Val(InputBox(" Diem Trung Binh = "))
End Sub

Private Sub cmdXoa_Click()
    Cls
End Sub

Private Sub cmdKetThuc_Click()
    Beep
    End
End Sub

Private Sub cmdTinhVaIn_Click()
Dim HocBong%
    If DiemTrungBinh! >= 7.5 Then
        If DiemTrungBinh! >= 9 Then
```

```

        HocBong% = 180
    Else
        HocBong% = 120
    End If
Else
    HocBong% = 0
End If
Cls
Print: Print
Print " Sinh vien "; ten$; " co hoc bong = "; HocBong%
End Sub

```

Giải thích chương trình

Dòng lệnh đầu tiên của chương trình này khai báo 2 biến là `ten$`, `DiemTrungBinh!`. Các biến này được khai báo ở đầu chương trình, nên là những biến toàn cục⁽¹⁾. Vì vậy các biến này có thể được sử dụng trong tất cả các thủ tục - biến cố của chương trình này.

Thủ tục - biến cố `cmdNhap_Click()` dùng để nhập dữ liệu ban đầu từ bàn phím. Vì ta muốn nhập giá trị của 2 biến `Ten$` và `DiemTrungBinh!`, nên ta phải sử dụng hàm `InputBox()` 2 lần. Các giá trị của hàm `InputBox()` được gán cho các biến tương ứng bằng các câu lệnh gán có trong thủ tục - biến cố này.

Trong thủ tục - biến cố `cmdTinhVaIn_Click()` có câu lệnh khai báo biến `HocBong%`. Biến này là biến cục bộ. Thủ tục - biến cố `cmdTinhVaIn_Click()` sẽ được thực hiện khi ta sử dụng sử dụng nút

⁽¹⁾ Khái niệm “biến toàn cục” ở đây có nghĩa là các biến này có thể được sử dụng trong tất cả các thủ tục - biến cố liên quan đến biểu mẫu của chương trình này. Khái niệm này được định nghĩa đầy đủ hơn trong chương “Hàm và thủ tục” (Chương 9).

lệnh Tinh Va In. Khi đó máy tính thực hiện các công việc sau đây:

- Tính giá trị của biến HocBong% căn cứ vào giá trị của biến DiemTrungBinh!. Công việc này được thực hiện bằng một câu lệnh IF lồng nhau. Ta đã nghiên cứu câu lệnh IF lồng nhau này trong một ví dụ ở mục trên.

- In trên biểu mẫu dòng kết quả, gồm tên sinh viên và học bổng của sinh viên này.

Các thủ tục - biến cố cmdXoa_Click() và cmdKetThuc_Click() của chương trình này đã quen thuộc với ta từ trước.

Chạy thử chương trình HocBong

Hãy gõ phím F5 để thực hiện chương trình HocBong. Sử dụng nút lệnh “Nhap Du Lieu” để nhập tên sinh viên và điểm trung bình. Dùng nút lệnh “Tinh Va In “ để xem kết quả. Dùng nút lệnh “Xoa” để xoá các dữ liệu đã được in trong biểu mẫu. Khi đã xem xong thì nháy chuột vào nút lệnh “Ket Thuc” để kết thúc chương trình.

6.3. CÂU LỆNH SELECT

Lệnh IF cho phép lập chương trình phân thành hai nhánh, tức là thực hiện một trong hai khối lệnh. Muốn lập chương trình phân thành nhiều nhánh ta có thể dùng nhiều lệnh IF lồng nhau. Tuy nhiên trong những trường hợp như vậy có thể dùng câu lệnh SELECT để cho chương trình sáng sủa, dễ viết dễ đọc hơn.

6.3.1. Câu lệnh SELECT đơn giản

Câu lệnh SELECT đơn giản có qui cách như sau:

```

SELECTE   CASE Biểu thức kiểm tra
           CASE Danh sách biểu thức 1
             Khối lệnh 1
           [CASE Danh sách biểu thức 2]
             Khối lệnh 2
           .....
           [CASE Danh sách biểu thức m
             Khối lệnh m]
END SELECT
[CASE ELSE
           Khối lệnh m+1]

```

Dòng đầu tiên của câu lệnh này được gọi là mệnh đề SELECT. Dòng cuối cùng của câu lệnh này được gọi là mệnh đề END. Mỗi một trong các phần còn lại có dạng:

```

CASE Danh sách biểu thức
  Khối lệnh

```

Được gọi là một mệnh đề CASE. Như vậy ta thấy rằng câu lệnh SELECT có m mệnh đề CASE.

Ở đây biểu thức kiểm tra (BTKT) là một biểu thức bất kỳ cho giá trị là một số hoặc một chuỗi kí tự. Biểu thức kiểm tra luôn luôn phải có mặt trong câu lệnh SELECT.

Danh sách biểu thức 1 (danh sách biểu thức 2...) gồm các biểu thức có giá trị thuộc kiểu số hoặc kiểu chuỗi kí tự, phù hợp với kiểu giá trị của biểu thức kiểm tra. Chúng có thể là một giá trị số, một miền giá trị có dạng A TO B hoặc một giá trị có thể so sánh được với giá trị biểu thức kiểm tra. Nếu muốn so sánh giá trị biểu thức đứng sau CASE với giá trị biểu thức kiểm tra thì sau từ khóa CASE phải có từ khóa IS trước dấu phép toán quan hệ. Mỗi khối lệnh gồm một hoặc một số câu lệnh.

Nếu một trong các trường hợp sau đây xảy ra với một mệnh đề CASE:

- Giá trị của BTKT bằng giá trị một biểu thức có trong danh sách biểu thức của mệnh đề CASE này;

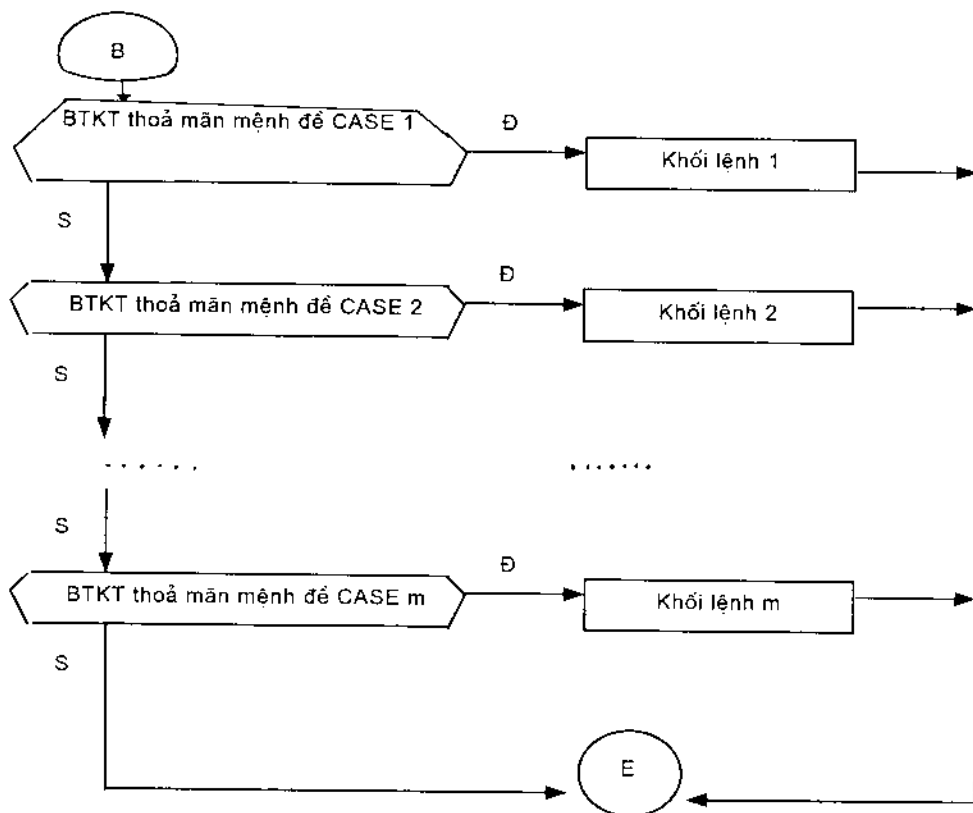
- Giá trị của BTKT thuộc về miền giá trị có trong mệnh đề CASE này;

- Giá trị của BTKT thoả mãn quan hệ có trong mệnh đề CASE này;

thì ta nói rằng BTKT thoả mãn mệnh đề CASE này.

Tác dụng: Theo lệnh SELECT CASE đầu tiên máy tính giá trị biểu thức kiểm tra. Sau đó máy kiểm tra xem biểu thức kiểm tra thoả mãn mệnh đề CASE nào. Nếu biểu thức kiểm tra thoả mãn mệnh đề CASE thứ i , thì máy thực hiện khối lệnh i ($i = 1, 2, \dots, m$) rồi chuyển đến thực hiện câu lệnh tiếp theo câu lệnh SELECT, là câu lệnh đứng sau END SELECT. Nếu biểu thức kiểm tra không thoả mãn một mệnh đề CASE nào cả, thì máy bỏ qua toàn bộ lệnh SELECT.

Quá trình thực hiện câu lệnh SELECT đơn giản được chỉ ra bằng sơ đồ khối trong hình dưới đây.



Hình 6.3. Sơ đồ khối thực hiện câu lệnh **SELECT CASE** đơn giản

Ví dụ về câu lệnh **SELECT CASE** đơn giản

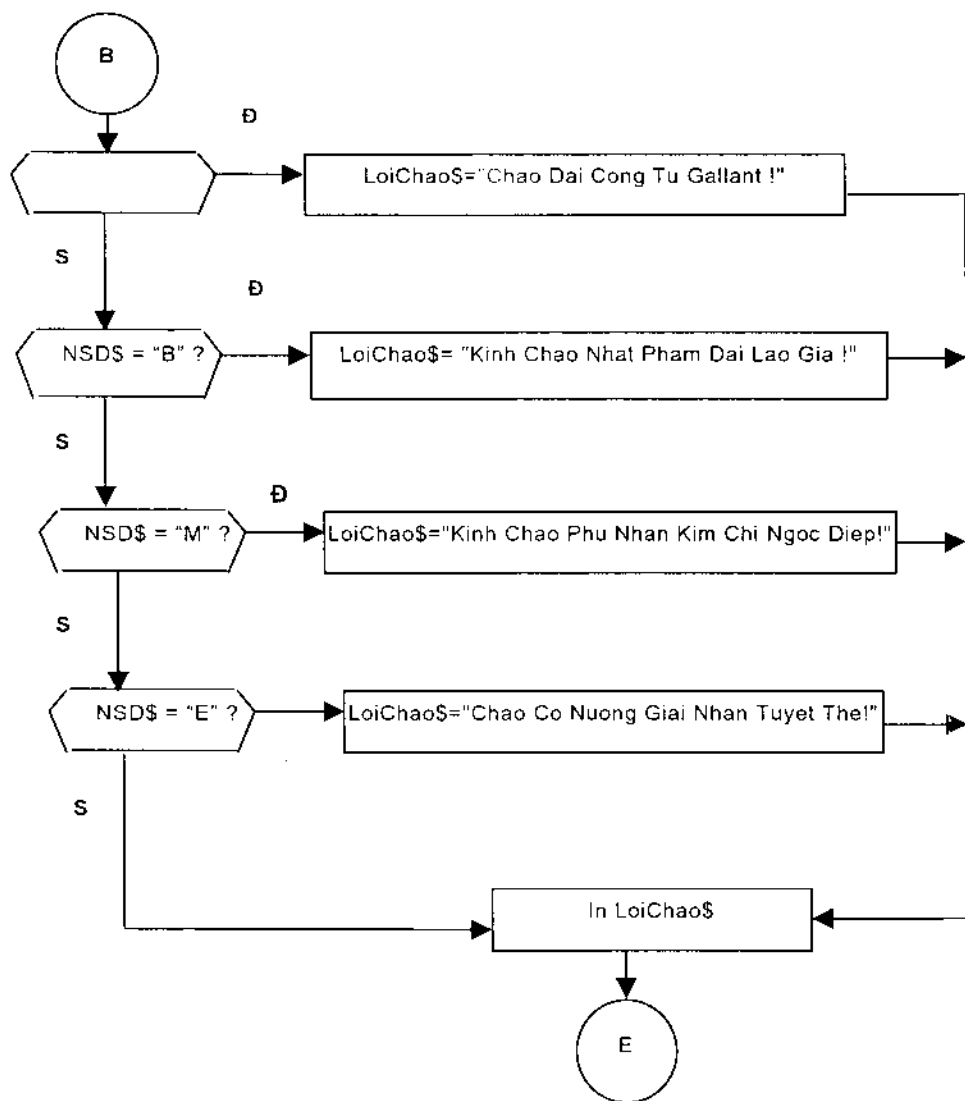
Trong ví dụ dưới đây câu lệnh **SELECT CASE** sẽ tạo ra một “lời chào thông minh”. Sở dĩ được gọi là “lời chào thông minh” vì nó phù hợp với người được chào (là người đang sử dụng máy tính) và làm vui lòng người này.

Giả sử đang ở trong một gia đình hạt nhân, gồm có bố (B), mẹ (M), anh trai (A) và em gái (E). Trước tiên cần biết ai đang sử dụng máy tính. Căn cứ vào đó ta sẽ dùng câu lệnh **SELECT CASE** sẽ tạo ra

một lời chào thích hợp. Đoạn chương trình dưới đây sẽ thực hiện công việc này.

```
Private Sub cmdViDu61_click()  
    Dim LoiChao$  
    Dim NSD$ `Nguoi Su Dung  
    LoiChao$ = " "  
    NSD$ = InputBox("Xin cho biet Ai đang dung may?(B,  
M, A, E) ")  
    Select Case NSD$  
        Case "A", "a"  
            LoiChao$ = "Chao Dai Cong Tu Gallant !"  
        Case "B", "b"  
            LoiChao$ = "Kinh Chao Nhat Pham Dai Lao Gia !"  
        Case "M", "m"  
            LoiChao$ = "Kinh Chao Phu Nhan Kim Chi Ngoc Diep!"  
        Case "E", "e"  
            LoiChao$ = "Chao Quy Co Nuong Giai Nhan Tuyet  
The !"  
    End Select  
    Print LoiChao$  
End Sub
```

Thứ tự thực hiện câu lệnh SELECT CASE khi tạo ra một lời chào thích hợp được trình bày trong sơ đồ ở hình 6.4.



Hình 6.4. Sơ đồ thực hiện câu lệnh SELECT CASE để tạo ra "lời chào thông minh"

6.3.2. Câu lệnh SELECT đầy đủ

Câu lệnh SELECT đầy đủ có qui cách như sau:

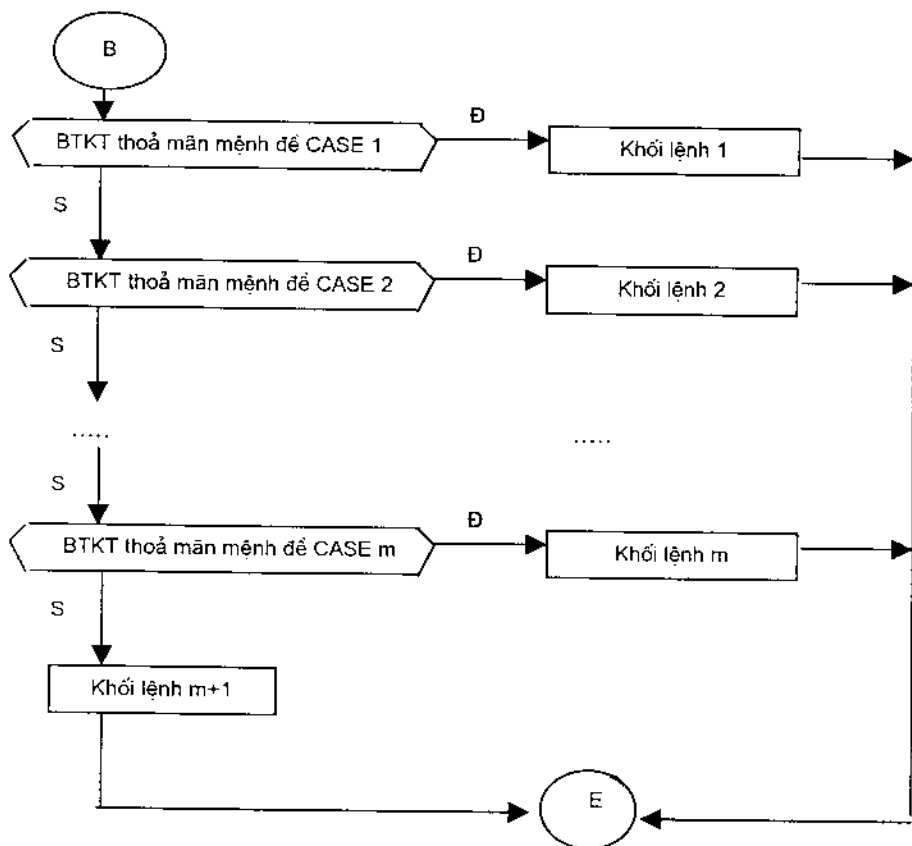
```
SELECT CASE Biểu thức kiểm tra
      CASE   Danh sách biểu thức 1
            Khối lệnh 1
      [CASE   Danh sách biểu thức 2]
            Khối lệnh 2
      .....
      [CASE   Danh sách biểu thức m
            Khối lệnh m]
      CASE ELSE
            Khối lệnh m+1
END SELECT
```

Ta thấy rằng tương tự câu lệnh SELECT đơn giản. Điểm khác biệt ở đây là câu lệnh SELECT đầy đủ có mệnh đề CASE ELSE. Trong mệnh đề CASE ELSE có khối lệnh m+1.

Câu lệnh SELECT đầy đủ được thực hiện tương tự câu lệnh SELECT đơn giản. Chỉ có một điểm khác biệt là: trong trường hợp biểu thức kiểm tra không thoả mãn mệnh đề CASE nào cả thì khối lệnh m+1 trong mệnh đề ELSE sẽ được thực hiện.

Quá trình thực hiện câu lệnh SELECT đầy đủ được chỉ ra bằng sơ đồ khối trong hình 6.5 dưới đây.

Ví dụ thứ hai.



Hình 6.5. Sơ đồ khối thực hiện câu lệnh SELECT CASE đầy đủ

Trong ví dụ dưới đây câu lệnh SELECT CASE được sử dụng để đoán chữ số do người sử dụng nhập vào từ bàn phím. Nếu người sử dụng nhập một chữ số từ 0 đến 9 thì máy tính sẽ in tên chữ số đó. Trong trường hợp người sử dụng nhập một kí tự không phải là chữ số thì máy tính sẽ in thông báo “Đây không là chữ số!”. Thủ tục - biến cố dưới đây sẽ thực hiện công việc này.

```

Private Sub cmdViDu62_Click()
    Dim ChuSo$, Ten$
    ChuSo$ = InputBox("Hay nhap 1 chu so (0 -> 9) ")
    Select Case ChuSo$
        Case "0"
            Ten$ = " chu so Khong."
        Case "1"
            Ten$ = " chu so Mot."
        Case "2"
            Ten$ = " chu so Hai."
        Case "3"
            Ten$ = " chu so Ba."
        Case "4"
            Ten$ = " chu so Bon."
        Case "5"
            Ten$ = " chu so Nam."
        Case "6"
            Ten$ = " chu so Sau."
        Case "7"
            Ten$ = " chu so Bay."
        Case "8"
            Ten$ = " chu so Tam."
        Case "9"
            Ten$ = " chu so Chin."
        Case Else
            Ten$ = " ki tu khong phai la chu so."
    End Select
    Print ChuSo$, " la " + Ten$
End Sub

```

Khi thực hiện thủ tục - biến cố này:

- Nếu người sử dụng nhập chữ số 0, thì kết quả có dạng: " 0 là chu so Khong. "

- Nếu người sử dụng nhập chữ số 1, thì kết quả có dạng: " 1 là chu so Mot. "

.....

- Nếu người sử dụng nhập chữ số 9, thì kết quả có dạng: " 9 là chu so Chín. "

- Nếu người sử dụng nhập kí tự "A", thì kết quả có dạng: " A là kí tu khong phải là chu so. ". Trong trường hợp này lệnh gán trong mệnh đề ELSE của câu lệnh SELECT được thực hiện.

6.3.3. Chương trình có sử dụng câu lệnh SELECT CASE

Hãy sử dụng lệnh SELECT CASE để viết chương trình giải bài toán phân loại sinh viên căn cứ vào điểm trung bình theo công thức sau đây:

$$\text{Loại} = \begin{cases} \text{Xuất sắc nếu} & \text{DTB} \Rightarrow 9 \\ \text{Giỏi nếu} & 8 \leq \text{DTB} < 9 \\ \text{Khá nếu} & 6,5 \leq \text{DTB} < 8 \\ \text{T bình nếu} & 5 \leq \text{DTB} < 6,5 \\ \text{Yếu nếu} & \text{DTB} < 5 \end{cases}$$

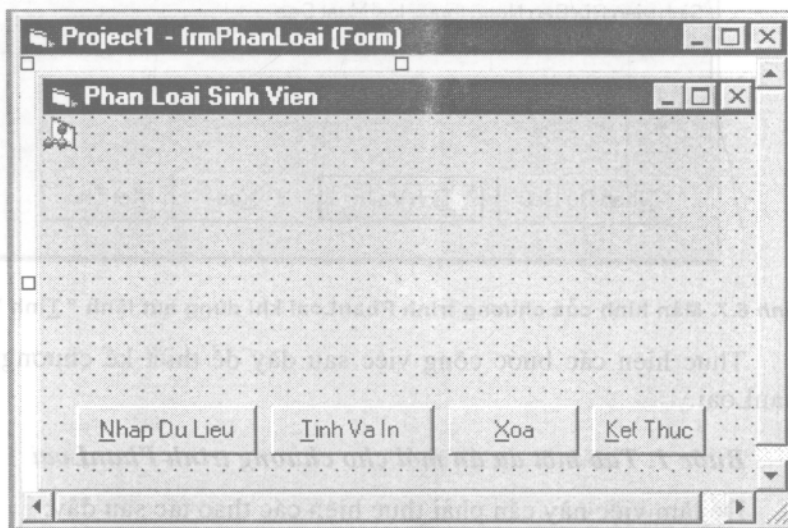
Ta gọi chương trình giải bài toán này là chương trình PhanLoai (Phân Loại).

Gới thiệu chương trình PhanLoai

Chương trình PhanLoai dùng để phân loại sinh viên. Bài toán này có nội dung như sau đây: cho biết tên và điểm trung bình của một sinh

viên. Hãy viết chương trình để xác định loại cho sinh viên này căn cứ vào điểm trung bình theo công thức đã cho ở trên.

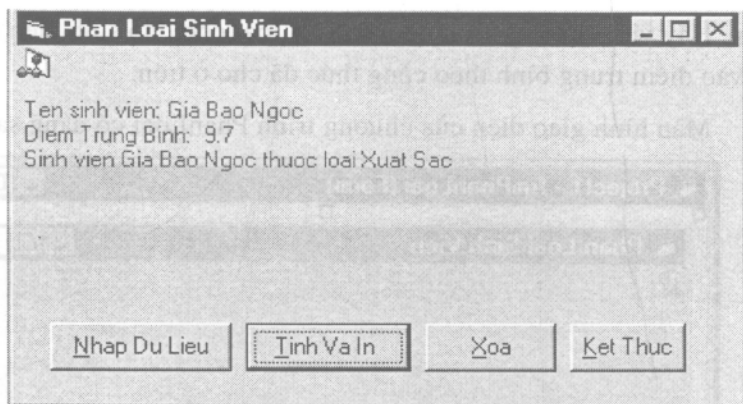
Màn hình giao diện của chương trình PhanLoai có dạng sau đây:



Hình 6.6. Màn hình giao diện của chương trình PhanLoai

Màn hình giao diện của chương trình này có 4 nút lệnh là Nhập Du Lieu, Tinh Va In, Xoa và Ket Thuc (hoàn toàn giống như màn hình giao diện của chương trình HocBong).

Nút lệnh Nhập Du Lieu dùng để nhập tên và điểm trung bình của sinh viên. Nút lệnh "Tinh Va In" dùng để điều khiển máy tính phân loại sinh viên và in kết quả trên màn hình của biểu mẫu. Màn hình của chương trình khi sử dụng nút lệnh "Tinh Va In" có dạng như trong hình 6.4.



Hình 6.7. Màn hình của chương trình PhanLoai khi dùng nút lệnh “Tinh Va In”

Thực hiện các bước công việc sau đây để thiết kế chương trình PhanLoai.

Bước 1. Tạo một dự án mới cho chương trình PhanLoai

Để làm việc này cần phải thực hiện các thao tác sau đây:

- Khởi động VB

- File

- New Project

Một biểu mẫu rỗng xuất hiện trên màn hình.

Bước 2. Lưu dự án mới lên đĩa từ

Để lưu tệp chứa dự án lên đĩa từ ta làm như sau:

- File

- Save Project as

- Xác định thư mục lưu dự án (c:\baitapvb)

- Xác định tên tệp lưu dự án (PhanLoai)

- OK

Để lưu tệp chứa biểu mẫu lên đĩa từ cần phải thực hiện các thao tác sau đây:

- File
- Save Form1 as
- Xác định thư mục lưu biểu mẫu (c:\baitapvb)
- Xác định tên tệp lưu biểu mẫu (PhanLoai)
- OK

Bước 3. Thiết kế giao diện cho chương trình

Trong bước công việc này ta sẽ sử dụng các công cụ trong hộp công cụ để lần lượt tạo ra 4 nút lệnh. Sau đó phải xác định cho các thuộc tính thích hợp cho các nút lệnh đó. Ta đã biết cách thực hiện các công việc này.

Bảng 6.2. Các đối tượng của chương trình PhanLoai

Số TT	Loại đối tượng	Thuộc tính	Giá trị
1	Biểu mẫu	Name	FrmPhanLoai
		Caption	Phan Loai Sinh Viên
2	Nút lệnh	Name	CmdNhap
		Caption	&Nhap Du Lieu
3	Nút lệnh	Name	CmdTinhVaIn
		Caption	&Tinh Va In
4	Nút lệnh	Name	CmdKetThuc
		Caption	&Ket Thuc
5	Nút lệnh	Name	CmdXoa
		Caption	&Xoa

Bước 4. Viết các thủ tục - biến cố cho các nút lệnh

Phần mã lệnh của chương trình PhanLoai có dạng sau đây:

```
Dim ten$, DiemTrungBinh!
```

```
Private Sub cmdNhap_Click()
```

```
    ten$ = InputBox(" Ten Sinh Vien = ")
```

```
    DiemTrungBinh! = Val(InputBox(" Diem Trung Binh = "))
```

```
End Sub
```

```
Private Sub cmdXoa_Click()
```

```
    Cls
```

```
End Sub
```

```
Private Sub cmdKetThuc_Click()
```

```
    Beep
```

```
    End
```

```
End Sub
```

```
Private Sub cmdTinhVaIn_Click()
```

```
    Dim Loai$
```

```
    Select Case DiemTrungBinh!
```

```
        Case Is >= 9
```

```
            Loai$ = "Xuat Sac"
```

```
        Case Is >= 8
```

```
            Loai$ = "Gioi"
```

```
        Case Is >= 6.5
```

```
            Loai$ = "Kha"
```

```
        Case Is >= 5
```

```
            Loai$ = "Trung Binh"
```

```

Case Else
    Loai$ = "Yeu"
End Select
Cls
Print " Ten sinh vien: "; Ten$
Print " Diem Trung Binh: "; DiemTrungBinh!
Print " Sinh vien "; Ten$; " thuoc loai "; Loai$
End Sub

```

Giải thích chương trình

Các thủ tục - biến cố cmdXoa_Click() và cmdKetThuc_Click() của chương trình này đã quen thuộc với ta từ trước. Sau đây sẽ chỉ giải thích thủ tục - biến cố cmdTinhVaIn_Click().

Trong thủ tục - biến cố cmdTinhVaIn_Click() có câu lệnh khai báo biến Loai%. Biến này là biến cục bộ. Thủ tục - biến cố cmdTinhVaIn_Click() sẽ được thực hiện khi ta chọn vào nút lệnh Tinh Va In. Khi đó máy tính thực hiện các công việc sau đây:

- Tính giá trị của biến Loai% căn cứ vào giá trị của biến DiemTrungBinh!. Công việc này được thực hiện bằng một câu lệnh SELEC CASE.

- In trên biểu mẫu 3 dòng kết quả. Dòng thứ nhất có tên sinh viên. Dòng thứ hai có điểm trung bình. Dòng thứ ba gồm tên sinh viên và loại của sinh viên này theo cách phân loại đã cho trong đầu bài..

Chạy thử chương trình PhanLoai

Hãy gõ phím F5 để thực hiện chương trình PhanLoai. Sử dụng nút lệnh "Nhap Du Lieu" để nhập tên sinh viên và điểm trung bình.

Dùng nút lệnh “Tính Va In “ để xem kết quả phân loại sinh viên. Dùng nút lệnh “Xoa” để xoá các dữ liệu đã được in trong biểu mẫu. Khi đã xem xong thì vào nút lệnh “Ket Thuc” để kết thúc chương trình.

Chú ý: Để có kết quả đúng, trước khi sử dụng nút lệnh “Tính Va In” cần sử dụng nút lệnh “Nhap Du Lieu” để nhập các dữ liệu cần cho chương trình.

6.4. ĐỐI TƯỢNG NÚT TỰY CHỌN

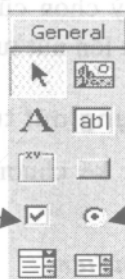
Trong các phần trên ta đã sử dụng các câu lệnh phân nhánh để thiết kế các chương trình lựa chọn. Các chương trình đó có đặc điểm là việc lựa chọn do máy tính thực hiện trong thời gian thực hiện chương trình căn cứ vào giá trị của một biến. Trong phần này ta sẽ sử dụng nút radio để thiết kế các chương trình lựa chọn. Các chương trình kiểu này có đặc điểm là việc lựa chọn do người sử dụng thực hiện trong thời gian thực hiện chương trình bằng cách chọn một nút tựy chọn có trong biểu mẫu.

6.4.1. Khái niệm nút tựy chọn

Nút tựy chọn (còn được gọi là nút radio) là loại đối tượng cho phép người sử dụng lựa chọn trong thời gian thực hiện chương trình. Các nút tựy chọn luôn luôn được sử dụng cùng nhau và làm việc theo kiểu loại trừ lẫn nhau. Các nút tựy chọn của một chương trình tạo thành một danh sách các nút tựy chọn. Trong thời gian thực hiện chương trình, ở mỗi thời điểm người sử dụng chỉ được phép chọn một (và chỉ một) nút tựy chọn trong danh sách này.

Muốn tạo ra nút tựy chọn ta phải sử dụng công cụ tạo ra nút tựy chọn. Biểu tượng của công cụ tạo ra nút tựy chọn được chỉ ra trong hình 6.8.

Thao tác tạo ra nút radio hoàn toàn tương tự thao tác tạo ra nút lệnh.



Hình 6.8. Biểu tượng của công cụ tạo nút tùy chọn và hộp kiểm trong hộp công cụ.

6.4.2. Các thuộc tính của nút tùy chọn

Bảng 6.3. Các thuộc tính của nút radio

Thuộc tính	Ý nghĩa, tác dụng
Appearance	Xác định khi nào nút tùy chọn có dạng 3 chiều.
BackColor	Xác định màu nền của nút tùy chọn.
Caption	Xác định tiêu đề cho nút tùy chọn.
Enabled	Xác định tính khả ứng của nút tùy chọn.
Font	Xác định font chữ cho nút tùy chọn.
ForeColor	Xác định màu chữ của nút tùy chọn.
Height	Xác định chiều cao của nút tùy chọn.
Left	Xác định khoảng cách trái của nút tùy chọn.
Name	Xác định tên nút tùy chọn, ba ký tự đầu tiên là option1.
Top	Xác định khoảng cách trên của nút tùy chọn.
Value	Có giá trị True hoặc False (ngầm định). Giá trị True của thuộc tính này làm cho nút tùy chọn được chọn ngay khi bắt đầu thực hiện chương trình. Khi người sử dụng chọn một nút tùy chọn thì thuộc tính Value của nút tùy chọn này sẽ có giá trị True, còn thuộc tính Value của tất cả các nút tùy chọn khác sẽ có giá trị False.
Visible	Xác định tính thấy được của nút tùy chọn.
Width	Xác định chiều rộng của nút tùy chọn.

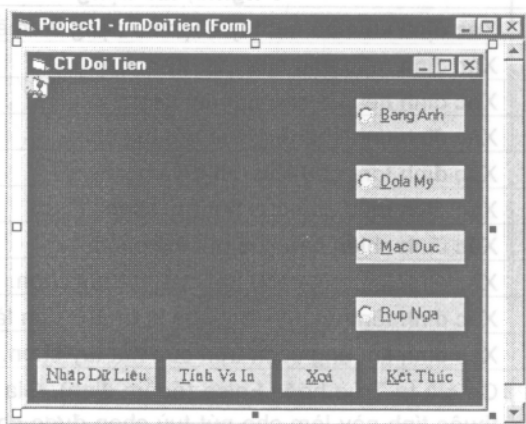
Giống như nút lệnh, nút tùy chọn cũng đáp ứng được thủ tục - biến cố người sử dụng nhấn chuột lên nút tùy chọn.

6.4.3. Chương trình sử dụng nút tùy chọn

Trong phần này ta sẽ thiết kế chương trình có tên là DoiTien (DoiTien).

Giới thiệu chương trình DoiTien

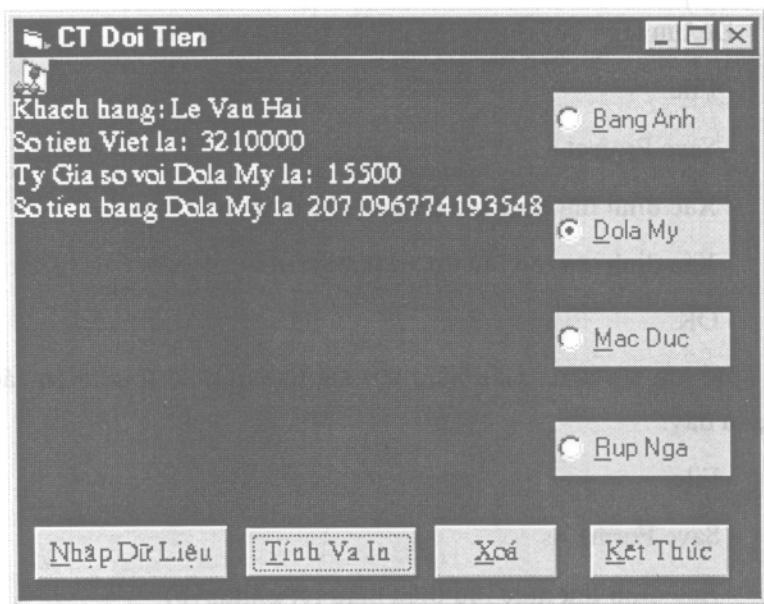
Chương trình DoiTien dùng để tính số tiền ngoại tệ khi đổi tiền. Bài toán này có nội dung sau đây: cho biết tên khách hàng đổi tiền và số tiền Việt Nam mà người này muốn đổi. Chương trình DoiTien sẽ tính số tiền ngoại tệ căn cứ vào số tiền Việt Nam và loại ngoại tệ do người sử dụng lựa chọn và in kết quả tính toán trên biểu mẫu. Màn hình giao diện của chương trình DoiTien có dạng sau đây:



Hình 6.9. Màn hình giao diện của chương trình DoiTien

Màn hình giao diện của chương trình này có 4 nút lệnh là là Nhập Dữ Liệu, Tính Và In, Xóa và Kết Thúc (hoàn toàn giống như màn hình giao diện của chương trình HocBong) và 4 nút tùy chọn.

Nút lệnh Nhập Dữ Liệu dùng để nhập tên khách hàng và số tiền Việt Nam. Nút lệnh “Tính Và In” dùng để điều khiển máy tính số tiền ngoại tệ và in kết quả trên màn hình của biểu mẫu. Màn hình của chương trình khi sử dụng nút lệnh “Tính Và In” có dạng như trong hình 6.10.



Hình 6.10. Màn hình của chương trình DoiTien
khi dùng nút lệnh “Tính Và In”.

Thực hiện các bước công việc sau đây để thiết kế chương trình DoiTien.

Bước 1. Tạo một dự án mới cho chương trình DoiTien

Để làm việc này cần phải thực hiện các thao tác sau đây:

- Khởi động VB

- File
- New Project

Một biểu mẫu rỗng xuất hiện trên màn hình.

Bước 2. Lưu dự án mới lên đĩa từ

Để lưu tệp chứa dự án lên đĩa từ ta làm như sau:

- File
- Save Project as
- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (DoiTien)
- OK

Để lưu tệp chứa biểu mẫu lên đĩa từ cần phải thực hiện các thao tác sau đây:

- File
- Save Form1 as
- Xác định thư mục lưu biểu mẫu (c:\baitapvb)
- Xác định tên tệp lưu biểu mẫu (DoiTien)
- OK

Bước 3. Thiết kế giao diện cho chương trình

Trong bước công việc này ta sẽ sử dụng các công cụ trong hộp công cụ để lần lượt tạo ra 4 nút lệnh và 4 nút tùy chọn. Sau đó phải xác định các thuộc tính thích hợp cho các nút lệnh và các nút tùy chọn đó. Ta đã biết cách thực hiện các công việc này.

Bảng 6.4. Các đối tượng của chương trình DoiTien

Số TT	Loại đối tượng	Thuộc tính	Giá trị
1	Biểu mẫu	Name Caption	FrmDoiTien CT Doi Tien
2	Nút lệnh	Name Caption Font	CmdNhap &Nhập Dữ Liệu .VnTime
3	Nút lệnh	Name Caption Font	CmdTinhValn &Tinh Và In .VnTime
4	Nút lệnh	Name Caption Font	CmdKetThuc &Kết Thúc .VnTime
5	Nút lệnh	Name Caption Font	CmdXoa &Xoá .VnTime
6	Nút tùy chọn	Name Caption	OptRup "&Rup Nga"
7	Nút tùy chọn	Name Caption	OptBang "&Bang Anh"
8	Nút tùy chọn	Name Caption	OptDola "&Dola My"
9	Nút tùy chọn	Name Caption	OptMac "&Mac Duc"

Chú ý: Trong chương trình này ta xác định giá trị “.VnTime” cho thuộc tính Font của tất cả các nút lệnh. Vì vậy tiêu đề của các nút lệnh có đầy đủ dấu tiếng Việt nên dễ đọc và trông đẹp mắt.

Bước 4. Viết các thủ tục - biến cố cho các nút lệnh

Phần mã lệnh của chương trình DoiTien có dạng sau đây:

```
Dim ten$, TienViet, TienNgoaiTe, Tygia, TenNgoaiTe

Private Sub cmdNhap_Click()
    ten$ = InputBox("Ten khách hàng = ", "Nhập ten khách  
hàng ", "Le Van Hai")
    TienViet = 1000 * Val(InputBox("Số tiền cần đổi =  
(000) "))
End Sub

Private Sub cmdXoa_Click()
    Cls
End Sub

Private Sub cmdKetThuc_Click()
    Beep
    End
End Sub

Private Sub cmdTinhVaIn_Click()
    If OptBang.Value Then
        TenNgoaiTe = "Bang Anh"
        Tygia = 28000
    End If
    If OptDola.Value Then
        TenNgoaiTe = "Dola My"
        Tygia = 15800
    End If
```

```

If OptMac.Value Then
    TenNgoaiTe = "Mac Duc"
    Tygia = 10000
End If
If OptRup.Value Then
    TenNgoaiTe = "Rup Nga"
    Tygia = 500
End If
TienNgoaiTe = TienViet / Tygia
Print
Print "Khach hang: "; ten$
Print "So tien Viet la: "; TienViet
Print "Ty Gia so voi " + TenNgoaiTe + " la: "; Tygia
Print "So tien bang " + TenNgoaiTe + " la ";
TienNgoaiTe
End Sub

```

Giải thích chương trình

Câu lệnh Dim khai báo các biến cần cho chương trình. Chú ý rằng trong câu lệnh này chỉ có kiểu của biến ten\$ là được xác định (kiểu string). Kiểu của các biến còn lại không được xác định. Điều đó có nghĩa là các biến này có kiểu là Variant (kiểu tổng hợp).

Khi người sử dụng vào nút lệnh “Tinh Va In” (sau khi đã nhập dữ liệu bằng cách sử dụng nút lệnh “Nhập dữ liệu” và chọn một trong các nút tùy chọn) thì thủ tục - biến cố cmdTinhVaIn_Click() sẽ được thực hiện và máy tính sẽ thực hiện các công việc sau đây:

- Thực hiện các câu lệnh IF để tính giá trị các biến TenNgoaiTe và Tygia. Mặc dù ở đây có 4 câu lệnh IF, nhưng mỗi lần thực hiện thủ

tục - biến cố “Tính Va In” chỉ có một trong các câu lệnh IF đó được thực hiện vì chỉ có một trong các nút tùy chọn có thuộc tính Value là True. Tuy nhiên, nếu người sử dụng không chọn một nút tùy chọn trước khi thực hiện thủ tục - biến cố “Tính Va In” , thì sẽ không có câu lệnh IF nào được thực hiện. Trong trường hợp này giá trị của biến Tygia không được xác định. Điều này sẽ gây ra lỗi.

- Tính giá trị của biến TienNgoaiTe căn cứ vào giá trị của biến TienViet và TyGia. Công việc này được thực hiện bằng một câu gán.

- In trên biểu mẫu 4 dòng kết quả. Dòng thứ nhất có tên khách hàng. Dòng thứ hai có số tiền Việt mà khách hàng muốn đổi. Dòng thứ ba có tỷ giá đồng tiền Việt Nam so với đồng ngoại tệ đó. Dòng thứ tư có số tiền tính bằng đồng ngoại tệ này.

Chú ý rằng thủ tục - biến cố cmdTinhVaIn_Click() chỉ làm việc chính xác sau khi người sử dụng đã nhập dữ liệu và chọn một trong các nút tùy chọn. Nếu không như vậy việc thực hiện thủ tục - biến cố này sẽ gây lỗi.

Chạy thử chương trình DoiTien

Hãy gõ phím F5 để thực hiện chương trình DoiTien. Sử dụng nút lệnh “Nhập Dữ Liệu” để nhập tên khách hàng và số tiền Việt Nam. Dùng chuột chọn một trong các nút tùy chọn để chọn loại ngoại tệ cần dùng. Dùng nút lệnh “Tính Va In “ để tính và in các thông tin kết quả. Dùng nút lệnh “Xóa” để xóa kết quả đã được in trong biểu mẫu. Có thể chọn một loại ngoại tệ khác (tức là chọn nút tùy chọn khác) rồi dùng nút lệnh “Tính Va In “ một lần nữa để xem kết quả mới. Khi đã xem xong thì nháy chuột vào nút lệnh “Kết Thúc” để kết thúc chương trình.

6.5. ĐỐI TƯỢNG HỘP KIỂM

Trong các phần trên ta đã sử dụng các nút tùy chọn để thiết kế các chương trình lựa chọn. Các nút tùy chọn có đặc điểm là ở mỗi thời

điểm người sử dụng chỉ có thể chọn một trong các các nút toàn cục có trong chương trình.

Đối tượng hộp kiểm cũng cho phép người sử dụng chọn trong thời gian thực hiện chương trình. Tuy nhiên, khác với nút tùy chọn, hộp kiểm cho phép người sử dụng chọn một số hộp kiểm trong các hộp kiểm có trong chương trình.

6.5.1. Khái niệm hộp kiểm

Hộp kiểm (Check box) là loại đối tượng cho phép người sử dụng chọn trong thời gian thực hiện chương trình. Các hộp kiểm của một chương trình tạo thành một danh sách hộp kiểm. Các hộp kiểm làm việc độc lập với nhau. Trong thời gian thực hiện chương trình, tại mỗi thời điểm người sử dụng có thể chọn một hoặc một số hộp kiểm trong danh sách hộp kiểm. Các hộp kiểm không loại trừ nhau như những nút tùy chọn.

Muốn tạo ra hộp kiểm ta phải sử dụng công cụ tạo hộp kiểm. Biểu tượng của công cụ tạo hộp kiểm được chỉ ra trong Hình 6.8 ở phần trên

Thao tác tạo ra hộp kiểm hoàn toàn tương tự thao tác tạo ra nút tùy chọn.

6.5.2. Các thuộc tính của hộp kiểm

Bảng 6.3. Các thuộc tính của hộp kiểm

Thuộc tính	Ý nghĩa, tác dụng
Appearance	Xác định khi nào hộp kiểm có dạng 3 chiều.
BackColor	Xác định màu nền của hộp kiểm
Caption	Xác định tiêu đề cho hộp kiểm.
Enabled	Xác định tính khả ứng của hộp kiểm.

Thuộc tính	Ý nghĩa, tác dụng
Font	Xác định font chữ cho hộp kiểm.
ForeColor	Xác định màu chữ của hộp kiểm.
Height	Xác định chiều cao của hộp kiểm.
Left	Xác định khoảng cách trái của hộp kiểm.
Name	Xác định tên hộp kiểm, ba kí tự đầu tiên là chk.
Top	Xác định khoảng cách trên của hộp kiểm.
Value	Có giá trị True hoặc False (ngầm định). Giá trị True của thuộc tính này làm cho hộp kiểm được chọn ngay khi bắt đầu thực hiện chương trình. Khi người sử dụng chọn hộp kiểm thì thuộc tính Value của hộp kiểm này sẽ thay đổi (từ False sang True hoặc ngược lại). Điều này không ảnh hưởng đến thuộc tính Value của các hộp kiểm khác có trong chương trình.
Visible	Xác định tính thấy được của hộp kiểm.
Width	Xác định chiều rộng của hộp kiểm.

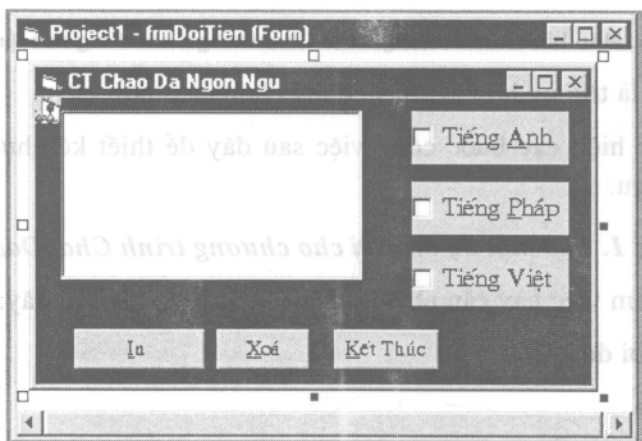
Giống như nút lệnh, hộp kiểm cũng đáp ứng được thủ tục - biến cố người sử dụng nháy chuột lên hộp kiểm (tức là chọn hộp kiểm). Cần chú ý đặc điểm sau đây khi chọn hộp kiểm. Giả sử hộp kiểm đang ở trạng thái không được chọn (thuộc tính Value bằng False). Lúc này nếu người sử dụng chọn hộp kiểm (tức là nháy chuột vào hộp kiểm) thì hộp kiểm chuyển sang trạng thái được chọn, thuộc tính Value sẽ chuyển thành True. Bây giờ, nếu người sử dụng chọn hộp kiểm lần thứ hai, thì hộp kiểm lại chuyển sang trạng thái không được chọn, và thuộc tính Value sẽ chuyển thành False. Việc người sử dụng chọn một hộp kiểm không ảnh hưởng đến các hộp kiểm khác.

6.5.3. Chương trình sử dụng hộp kiểm

Trong phần này ta sẽ thiết kế chương trình có tên là **ChaoDaNgu** (Chào Đa Ngữ, tức là chào bằng nhiều ngôn ngữ), là chương trình in lời chào bằng nhiều ngôn ngữ khác nhau.

Giới thiệu chương trình ChaoDaNgu

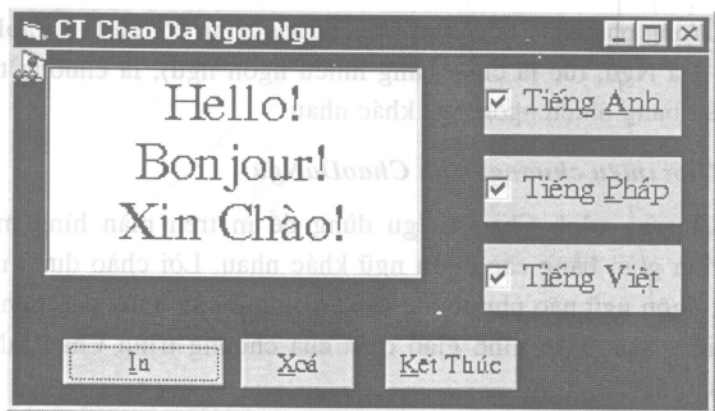
Chương trình ChaoDaNgu dùng để in trên màn hình một câu chào đơn giản bằng các ngôn ngữ khác nhau. Lời chào được in bằng những ngôn ngữ nào phụ thuộc vào việc người sử dụng đã chọn những hộp kiểm nào. Màn hình giao diện của chương trình ChaoDaNgu có dạng sau đây:



Hình 6.11. Màn hình giao diện của chương trình ChaoDaNgu

Màn hình giao diện của chương trình này có 1 hộp văn bản, 3 nút lệnh là In, Xóa và Kết Thúc (giống như màn hình giao diện của chương trình Hello) và 3 hộp kiểm.

Nút lệnh “In” dùng để điều khiển máy in lời chào bằng các ngôn ngữ khác nhau. Màn hình của chương trình khi sử dụng nút lệnh “In” có dạng như trong hình 6.12.



Hình 6.12. Màn hình của chương trình ChaoDaNgu khi dùng nút lệnh “In”.

Đây là trường hợp người sử dụng chọn cả 3 hộp kiểm.

Thực hiện các bước công việc sau đây để thiết kế chương trình ChaoDaNgu.

Bước 1. Tạo một dự án mới cho chương trình ChaoDaNgu

Để làm việc này cần phải thực hiện các thao tác sau đây:

- Khởi động VB

- File

- New Project

Một biểu mẫu rỗng xuất hiện trên màn hình.

Bước 2. Lưu dự án mới lên đĩa từ

Để lưu tệp chứa dự án lên đĩa từ ta làm như sau:

- File

- Save Project as

- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (ChaoDaNgu)
- OK

Để lưu tệp chứa biểu mẫu lên đĩa từ cần phải thực hiện các thao tác sau đây:

- File
- Save Form1 as
- Xác định thư mục lưu biểu mẫu (c:\baitapvb)
- Xác định tên tệp lưu biểu mẫu (ChaoDaNgu)
- OK

Bước 3. Thiết kế giao diện cho chương trình

Trong bước công việc này ta sẽ sử dụng các công cụ trong hộp công cụ để lần lượt tạo ra 1 hộp văn bản, 3 nút lệnh và 4 hộp kiểm. Sau đó phải xác định cho các thuộc tính thích hợp cho các nút lệnh và các hộp kiểm đó. Ta đã biết cách thực hiện các công việc này.

Bảng 6.6. Các đối tượng của chương trình ChaoDaNgu

Số TT	Loại đối tượng	Thuộc tính	Giá trị
1	Biểu mẫu	Name	FrmChaoDaNgu
		Caption	CT Chao Da Ngon Ngu
2	Hộp văn bản	Name	TxtKetQua
		font	.VnTime, size 20
		MultiLine	True
3	Nút lệnh	Name	CmdIn
		Caption	&In
		Font	.VnTime

Số TT	Loại đối tượng	Thuộc tính	Giá trị
4	Nút lệnh	Name Caption Font	CmdKetThuc &Kết Thúc .VnTime
5	Nút lệnh	Name Caption Font	CmdXoa &Xoá .VnTime
6	Hộp kiểm	Name Caption Value	ChkAnh "Tiếng &Anh" Value
7	Hộp kiểm	Name Caption Value	chkPhap "Tiếng &Pháp" Value
8	Hộp kiểm	Name Caption Value	ChkViet "Tiếng &Việt" Value

Chú ý: -Trong chương trình này ta xác định giá trị “.VnTime” cho thuộc tính Font của tất cả các nút lệnh và các hộp kiểm. Vì vậy tiêu đề của các nút lệnh và các hộp kiểm có đầy đủ dấu tiếng Việt.

- Thuộc tính MultiLine của hộp văn bản của chương trình này có giá trị True. Vì vậy có thể in nhiều dòng kết quả trong hộp văn bản này.

Bước 4. Viết các thủ tục - biến cố cho các nút lệnh

Phần mã lệnh của chương trình ChaoDaNgu có dạng sau đây:

```
Private Sub cmdIn_Click()  
    txtKetQua.Text = ""
```

```

If ChkAnh.Value Then
    txtKetQua.Text = txtKetQua.Text + "Hello!" +
        vbCrLf
End If

If ChkPhap.Value Then
    txtKetQua.Text = txtKetQua.Text + "Bonjour!" +
        vbCrLf
End If

If ChkViet.Value Then
    txtKetQua.Text = txtKetQua.Text + "Xin Chào!"
End If
End Sub

Private Sub cmdXoa_Click()
    txtKetQua.Text = ""
End Sub

Private Sub cmdKetThuc_Click()
    Beep
End
End Sub

```

Giải thích chương trình

Chương trình này không sử dụng một biến nào cả nên không có câu lệnh Dim.

Trong thời gian thực hiện chương trình, khi người sử dụng nhấn chuột vào nút lệnh “In” thì thủ tục - biến cố cmdIn_Click() sẽ được thực hiện và máy tính sẽ thực hiện các công việc sau đây:

- Gán cho `txtKetqua.Text` (thuộc tính `Text` của hộp văn bản `txtKetQua`) một chuỗi rỗng bằng câu lệnh gán thứ nhất.

- Lần lượt thực hiện các câu lệnh IF. Trong thủ tục - biến cố này có 3 câu lệnh IF, tương ứng với 3 hộp kiểm. Điều kiện trong các câu lệnh IF này là trạng thái được chọn của các hộp kiểm. Nếu một hộp kiểm được chọn thì điều kiện trong câu lệnh IF tương ứng với hộp kiểm này được thoả mãn, và câu lệnh gán có trong câu lệnh IF đó sẽ được thực hiện.

Ví dụ: nếu hộp kiểm “Tiếng Anh” đã được chọn thì thuộc tính `Value` của hộp kiểm này sẽ có giá trị `True` nên điều kiện trong câu lệnh IF sau đây được thoả mãn:

```
If ChkAnh.Value Then  
    txtKetQua.Text = txtKetQua.Text + "Hello!" + vbCrLf  
End If
```

Và câu lệnh gán:

```
txtKetQua.Text = txtKetQua.Text + "Hello!" + vbCrLf
```

sẽ được thực hiện. Câu lệnh gán này thêm câu chào tiếng Anh “Hello!” và hằng chuỗi kí tự `vbCrLf` (chuyển xuống đầu dòng mới) vào thuộc tính `Text` của hộp văn bản `txtKetQua`. Điều đó có nghĩa là câu chào bằng tiếng Anh sẽ xuất hiện trong hộp văn bản chứa kết quả. Hằng chuỗi kí tự `vbCrLf` theo sau câu chào bằng tiếng Anh đảm bảo cho câu chào tiếp theo (nếu có) sẽ được in ở đầu dòng tiếp theo của hộp văn bản chứa kết quả. Cần phải lưu ý rằng thuộc tính `MultiLine` của hộp văn bản này đã được xác định là `True`.

Điều tương tự cũng xảy ra đối với hai câu lệnh IF còn lại.

Chú ý: Thủ tục - biến cố `cmdIn_Click()` vẫn làm việc chính xác ngay cả khi người sử dụng không chọn một hộp kiểm nào cả. Trong trường hợp này trong hộp văn bản chứa kết quả sẽ không có một câu

chào nào cả vì không có câu lệnh IF nào được thực hiện.

Chạy thử chương trình ChaoDaNgu

Hãy gõ phím F5 để thực hiện chương trình ChaoDaNgu Dùng chuột chọn một trong các hộp kiểm để chọn loại câu chào. Dùng nút lệnh “In “ để xem các thông tin kết quả. Dùng nút lệnh “Xóa” để xóa kết quả đã được in trong hộp văn bản. Có thể chọn một hoặc một số hộp kiểm rồi dùng nút lệnh “In” một lần nữa để xem kết quả mới. Khi đã xem xong thì nháy chuột vào nút lệnh “Kết Thúc” để kết thúc chương trình.

CÂU HỎI ÔN TẬP VÀ BÀI TẬP

Lý thuyết

1. Quá trình tính toán phân nhánh là gì? Tại sao quá trình tính toán phân nhánh còn được gọi là quá trình tính toán lựa chọn?
2. Hãy nêu ý nghĩa của kỹ thuật lập chương trình lựa chọn.
3. Hãy trình bày các qui cách của câu lệnh IF. Cho ví dụ.
4. Hãy trình bày qui cách của câu lệnh SELECT. Cho ví dụ.
5. Nút tùy chọn là gì? Hãy trình bày các thuộc tính của nút tùy chọn.
6. Chương trình sử dụng nút tùy chọn có đặc điểm gì?
7. Hộp kiểm chọn là gì? Hãy trình bày các thuộc tính của hộp kiểm.
8. Chương trình sử dụng hộp kiểm có đặc điểm gì? Hãy so sánh với chương trình sử dụng nút tùy chọn.

Bài tập

1. Hãy viết dạng chương trình giải phương trình bậc hai $ax^2 + bx$

$+ c = 0$ khi cho biết các hệ số a, b, c .

2. Cho biết tên và năm sinh của 3 người. Hãy viết đoạn chương trình tìm và in tên người trẻ nhất.

3. Cho biết tên, giới tính và thu nhập của 3 công nhân. Hãy viết đoạn chương trình tìm và in tên nữ công nhân có thu nhập thấp nhất.

4. Hãy viết lại các thủ tục - biến cố cho chương trình DoiTien bằng cách sử dụng biến cố click cho các nút tùy chọn.

5. Hãy viết lại các thủ tục - biến cố cho chương trình ChaoDaNgu bằng cách sử dụng biến cố click cho các hộp kiểm.

CÁC CẤU TRÚC LẶP

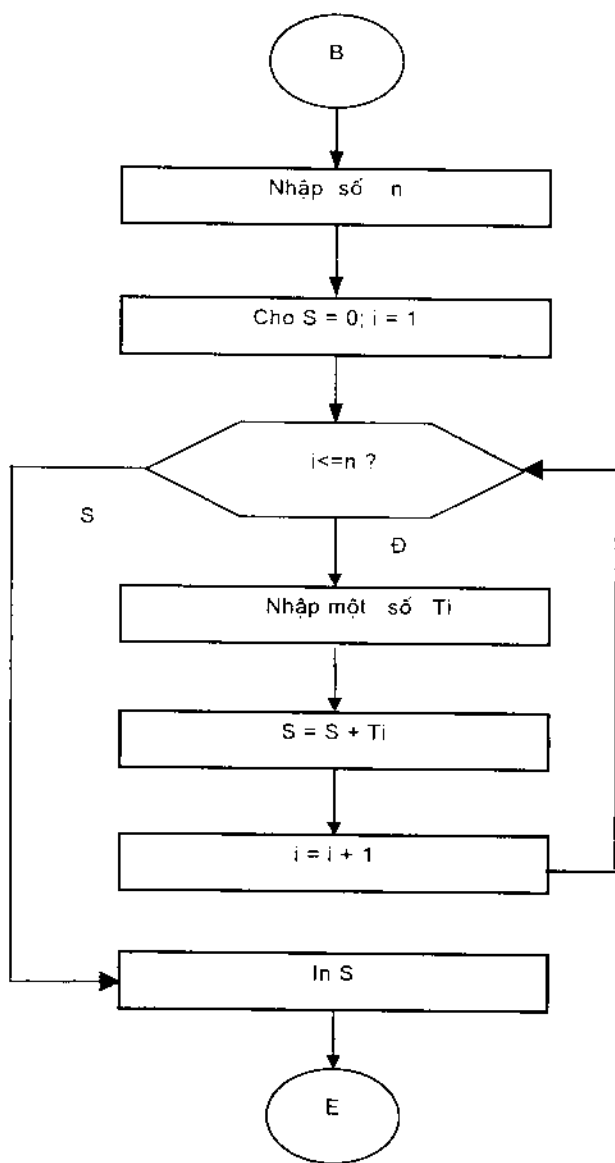
Chương này sẽ trình bày kỹ thuật lập một loại chương trình được sử dụng rất phổ biến. Đó là chương trình lặp, hoặc chương trình có chu trình. Các chương trình có chu trình sẽ được viết theo nhiều cách khác nhau, bằng cách sử dụng nhiều câu lệnh khác nhau.

7.1. KHÁI NIỆM CHƯƠNG TRÌNH LẶP

Quá trình tính toán có chu trình là quá trình tính toán có một phần công việc tính toán được thực hiện lặp lại nhiều lần. Phần công việc tính toán được thực hiện lặp lại nhiều lần này được gọi là vòng lặp hoặc chu trình. Số lần lặp lại của chu trình có thể biết trước hoặc không biết trước. Trong quá trình giải các bài toán thực tế rất hay gặp các quá trình tính toán có đặc điểm này.

Chương trình lặp (còn được gọi là chương trình có chu trình) là chương trình có một phần (một số các lệnh) được thực hiện lặp lại nhiều lần. Phần các lệnh được thực hiện lặp lại nhiều lần đó được gọi là vòng lặp hoặc chu trình của chương trình.

Ví dụ: Cho $t_1, t_2, \dots, t_n$ là tiền lương của n công nhân. Quá trình tính tổng tiền lương (S) của n người công nhân này là quá trình tính toán lặp. Quá trình tính toán này được thực hiện theo thuật toán sau đây:



Hình 7.1. Thuật toán tính tổng n số

Nhìn thuật toán này ta thấy các bước công việc thứ 4, 5, 6, 7 được thực hiện lặp lại n lần. Như vậy đây là một quá trình tính toán lặp.

Visual Basic, cũng như các ngôn ngữ lập trình hiện đại khác, có những lệnh chuyên dùng để viết các chương trình lặp. Những lệnh này có tên là lệnh lặp hoặc lệnh chu trình. Đó là các lệnh FOR, DO, WHILE. Các lệnh chu trình cho phép thực hiện lặp lại nhiều lần một phần của chương trình. Dưới đây ta sẽ lần lượt nghiên cứu các lệnh lặp của Visual Basic và dùng những lệnh đó để viết các chương trình lặp.

7.2. CÂU LỆNH CHU TRÌNH SỐ HỌC (CÂU LỆNH FOR)

Câu lệnh này dùng để viết các chương trình có chu trình có số lần lặp lại biết trước. Câu lệnh này được gọi là câu lệnh chu trình số học hoặc câu lệnh FOR. Câu lệnh này có qui cách như sau:

FOR biến = n_1 TO n_2 [STEP n_3]

Khối lệnh (thân chu trình)

NEXT [biến số]

ở đây :

- Biến đứng sau từ khoá **For** phải là biến số. Nó dùng để đếm số lần lặp lại chu trình, nên nó được gọi là biến điều khiển chu trình.

- n_1 , n_2 , n_3 là các biểu thức số học: n_1 xác định giá trị ban đầu cho biến điều khiển chu trình; n_2 xác định giá trị cuối cùng của biến điều khiển chu trình; n_3 là số gia của biến điều khiển chu trình. Số gia này có thể âm, dương. Phần tùy chọn [Step n_3] có thể bị bỏ qua (tức là không được sử dụng). Trong trường hợp này thì ngầm định rằng $n_3 = 1$.

- Sau từ khoá **Next** có thể có tên biến. Đó là tên biến điều khiển chu trình. Tên biến này cho ta biết từ khoá này liên quan đến chu trình nào. Tuy nhiên không bắt buộc phải viết tên biến sau từ khoá **Next**.

- Khối lệnh nằm giữa **For** và **Next** được gọi là thân chu trình.

Trong trường hợp $n3 > 0$ thì câu lệnh chu trình **For** được thực hiện theo trình tự sau đây:

(1). Biến điều khiển chu trình gán giá trị ban đầu là $n1$.

(2). Kiểm tra điều kiện kết thúc chu trình (biến $\leq n2$). Nếu điều kiện này được thoả mãn thì máy sẽ thực hiện bước 3. Ngược lại, nếu điều kiện này không được thoả mãn thì chuyển đến thực hiện câu lệnh tiếp theo câu lệnh **For**.

(3). Thực hiện thân chu trình, hết câu lệnh này đến câu lệnh khác theo thứ tự xuất hiện của chúng trong thân chu trình.

(4). Cộng thêm $n3$ vào biến điều khiển chu trình.

(5). Quay lại thực hiện bước (2)

Chú ý:

-Trong trường hợp $n3 < 0$, thì điều kiện kết thúc chu trình sẽ là (biến $\geq n2$).

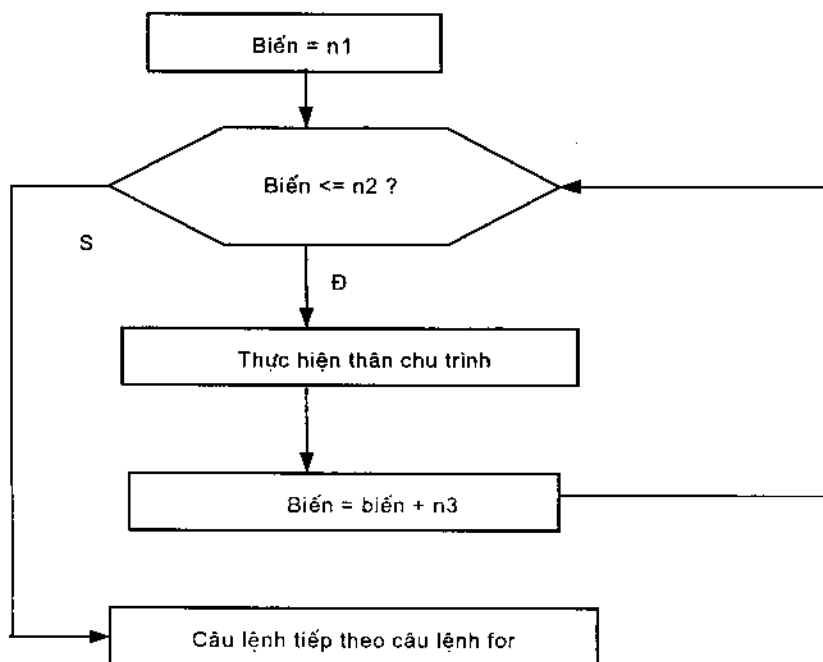
- Nếu không sử dụng **STEP** $n3$, thì ngầm định $n3 = 1$.

- Nếu $n3 = 0$ thì xảy ra việc lặp vô hạn.

- Nếu biến điều khiển chu trình là biến nguyên thì chu trình được thực hiện nhanh hơn.

- Trong thân chu trình có thể sử dụng biến điều khiển chu trình để tính toán, nhưng không được thay đổi giá trị biến này. Nếu không số lần lặp lại chu trình sẽ không chính xác.

Quá trình thực hiện câu lệnh **For** được thể hiện qua thuật toán trong Hình 7.2 sau đây:



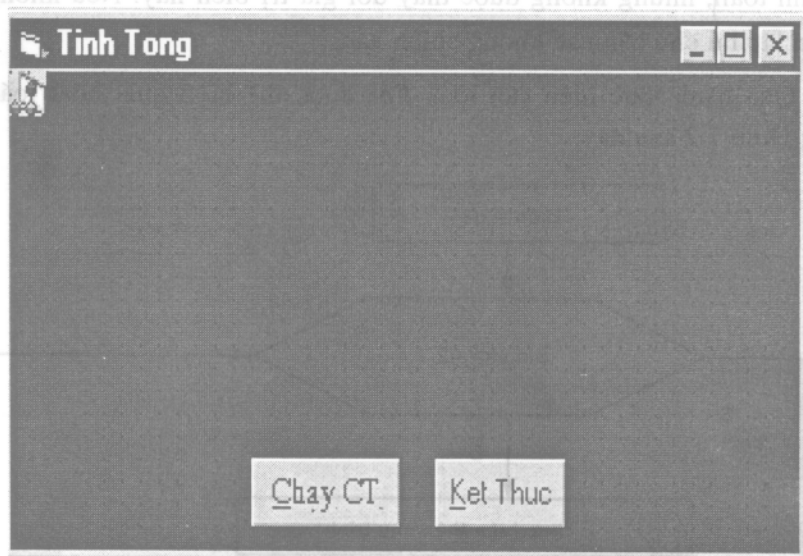
Hình 7.2. Thuật toán thực hiện lệnh **for**

Sau đây là một ví dụ về chương trình có chu trình số học. Chương trình này sử dụng lệnh **for**. Chương trình này có tên là **TinhTong**.

Giới thiệu chương trình TinhTong

Chương trình **TinhTong** dùng để tính và in tổng tài sản của n công nhân khi cho biết tài sản của mỗi công nhân.

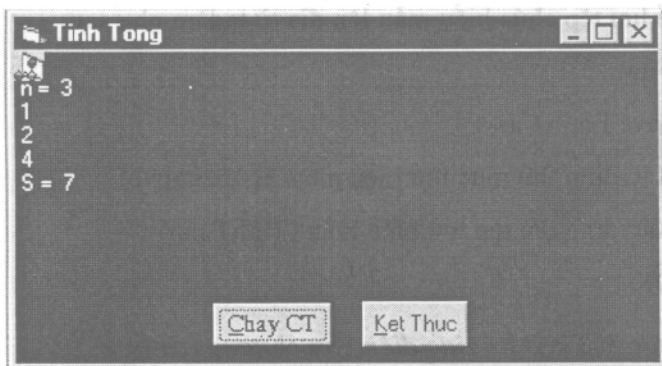
Màn hình giao diện của chương trình TinhTong có dạng sau đây:



Hình 7.3. Màn hình giao diện của chương trình TinhTong

Màn hình giao diện của chương trình này rất đơn giản, chỉ có 1 biểu mẫu và 2 nút lệnh là “Chay CT” (Chạy Chương Trình) và “Ket Thuc”. Màn hình giao diện này được gọi là màn hình giao diện đơn giản, được dùng để viết chương trình cho các ví dụ trong chương này và các chương sau.

Nút lệnh “Chay CT” dùng để điều khiển máy thực hiện chương trình. Còn nút lệnh “Ket Thuc”, như đã biết, dùng để kết thúc chương trình. Khi dùng nút lệnh “Chay CT” thì máy sẽ thực hiện thủ tục - biến cố của nút lệnh này để giải bài toán. Màn hình giao diện của chương trình này khi chạy chương trình có dạng sau đây:



Hình 7.4. Màn hình của chương trình TinhTong khi dùng nút lệnh “Chạy CT”

Thực hiện các bước công việc sau đây để thiết kế chương trình TinhTong.

Bước 1. Tạo một dự án mới cho chương trình TinhTong

Để làm việc này cần phải thực hiện các thao tác sau đây:

- Khởi động VB
- File
- New Project

Một biểu mẫu rỗng xuất hiện trên màn hình.

Bước 2. Lưu dự án mới lên đĩa từ

Để lưu tệp chứa dự án lên đĩa từ ta làm như sau:

- File
- Save Project as
- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (TinhTong)
- OK

Để lưu tệp chứa biểu mẫu lên đĩa từ ta làm như sau:

- File
- Save Form1 as
- Xác định thư mục lưu biểu mẫu (c:\baitapvb)
- Xác định tên tệp lưu biểu mẫu (TinhTong)
- OK

Bước 3. Thiết kế giao diện cho chương trình

Trong bước công việc này ta sử dụng các công cụ trong hộp công cụ để thiết kế giao diện của chương trình. Các đối tượng của chương trình này là:

Bảng 7.1. Các đối tượng của chương trình TinhTong

Số TT	Loại đối tượng	Thuộc tính	Giá trị
1	Biểu mẫu	Name Caption	FrmTinhTong CT Tinh Tong
3	Nút lệnh	Name Caption Font	CmdChayCT &Chạy CT .VnTime
4	Nút lệnh	Name Caption	CmdKetThuc &Ket Thuc

Bước 4. Viết các thủ tục - biến cố cho các nút lệnh

Phần mã lệnh của chương trình TinhTong có dạng sau đây:

```
Private Sub cmdChayct_Click()  
    Dim TaiSan, i, n, s  
    Cls
```



```

Print:
s = 0
n = Val(InputBox("Nhap so cong nhan "))

For i = 1 To n
    TaiSan = Val(InputBox("Nhap Tai san cua mot cong
nhan "))
    s = s + TaiSan
    Print TaiSan
Next
Print " S = "; s
End Sub

Private Sub cmdKetThuc_Click()
    Beep
    End
End Sub

```

Giải thích chương trình

Chương trình này có 4 biến cục bộ là TaiSan, i, n, s! . Biến n dùng để chứa số công nhân, tức là số lần lặp lại chu trình. Biến s dùng để chứa tổng cần tìm. Biến TaiSan dùng để chứa tài sản của từng công nhân, sẽ lần lượt được nhập vào từ bàn phím và được in trên màn hình trong thời gian thực hiện chương trình. Biến i là biến điều khiển chu trình.

Khi người sử dụng nhấp chuột lên nút lệnh “Chay CT” thì thủ tục - biến cố cmdChayct_Click() bắt đầu được thực hiện. Đầu tiên biến

s được gán giá trị 0 (để chuẩn bị cho việc tính tổng). Sau đó máy yêu cầu người sử dụng nhập số công nhân (số n). Số n cũng là số lần lặp lại chu trình.

Thân chương trình của câu lệnh For gồm có 3 câu lệnh. Câu lệnh thứ nhất:

```
TaiSan = Val(InputBox("Nhap Tai san cua mot cong  
nhan " ))
```

yêu cầu người sử dụng nhập tài sản của một người. Tài sản này sẽ được chứa trong biến TaiSan. Câu lệnh thứ hai là câu lệnh gán:

```
s = s + TaiSan
```

sẽ cộng giá trị của biến TaiSan vào tổng s. Câu lệnh thứ ba:

```
Print TaiSan
```

Sẽ in giá trị biến TaiSan trên màn hình biểu mẫu. Thân chu trình được thực hiện lặp lại n lần, cho phép ta nhập và cộng tài sản của n công nhân vào tổng s. Giá trị biến s, là tổng cần tìm, được in ở dòng cuối cùng của màn hình biểu mẫu.

Hình 7.4. chứa kết quả thực hiện chương trình này với n=3, các tài sản được nhập vào lần lượt là 1, 2 và 4. Tổng s có giá trị là 7.

Chạy thử chương trình TinhTong

Hãy gõ phím F5 để thực hiện chương trình TinhTong. Cho n và TaiSan các giá trị khác nhau để xem kết quả tính toán và kiểm tra độ chính xác của chương trình..

7.3. LỆNH CHU TRÌNH LOGIC DO WHILE ... LOOP

Lệnh này được gọi là lệnh chu trình logic. Lệnh này dùng để viết

các chương trình có chu trình có số lần lặp lại không biết trước. Cũng như câu lệnh **for... Next**, câu lệnh **Do While ... Loop** là câu lệnh chu trình “ kiểm tra trước “, nghĩa là điều kiện kết thúc chu trình được kiểm tra trước khi thực hiện thân chu trình. Việc lặp lại chu trình được thực hiện khi mà biểu thức kiểm tra có giá trị True.

Lệnh này có quy cách như sau:

DO WHILE <biểu thức kiểm tra>

 Khối lệnh (thân chu trình)

LOOP

Biểu thức kiểm tra (viết tắt là BTKT) là một biểu thức logic. Biểu thức này cho một giá trị logic. Giá trị đó có thể là True hoặc False.

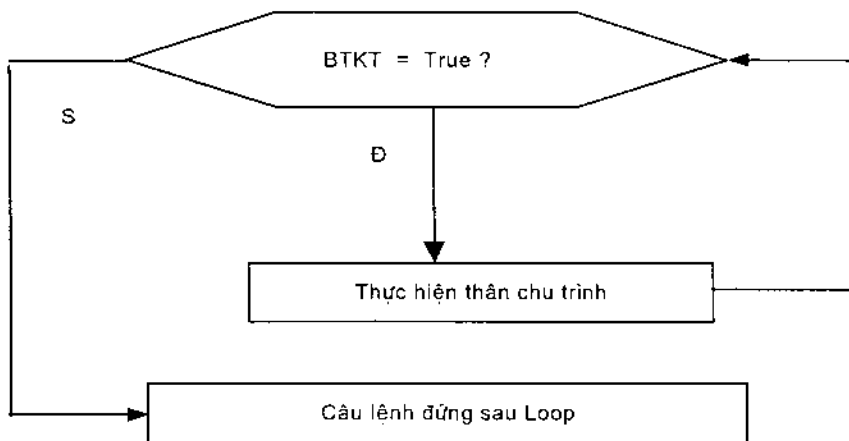
Câu lệnh này được thực hiện như sau:

(1). Tính giá trị biểu thức kiểm tra. Nếu giá trị này là True thì thực hiện bước 2. Ngược lại, nếu giá trị này là False thì kết thúc chu trình, chuyển đến thực hiện câu lệnh tiếp theo câu lệnh **Do While ... Loop**. Đó là câu lệnh đứng sau từ khoá **Loop**.

(2). Thực hiện thân chu trình, hết câu lệnh này đến câu lệnh khác theo thứ tự xuất hiện của chúng trong thân chu trình.

(3). Quay lại thực hiện bước (1)

Câu lệnh này được thực hiện theo thuật toán sau đây:



Hình 7.5. Thuật toán thực hiện lệnh Do While

Sau đây là một chương trình có sử dụng lệnh Do while... Loop. Chương trình này có tên là **TinhLaiSuat**.

Giới thiệu chương trình TinhLaiSuat

Bài toán tính lãi suất: Một khách hàng đem gửi tiết kiệm một số tiền, gọi là số tiền gửi (STGui). Khách hàng này muốn rằng sau n năm sẽ rút ra được một số tiền, gọi là số tiền rút (STRut), lớn hơn số tiền gửi. Hãy viết chương trình tính lãi suất nhỏ nhất để khách hàng đó đạt được mong muốn của mình. Chương trình TinhLaiSuat dùng để giải bài toán này.

Màn hình giao diện của chương trình TinhLaiSuat có dạng màn hình giao diện đơn giản mà ta đã biết, nên không cần trình bày lại nữa.

Thực hiện các bước công việc sau đây để thiết kế chương trình TinhLaiSuat.

Bước 1. Tạo một dự án mới cho chương trình TinhLaiSuat

Để làm việc này cần phải thực hiện các thao tác sau đây:

- Khởi động VB
- File
- New Project

Một biểu mẫu rỗng xuất hiện trên màn hình.

Bước 2. Lưu dự án mới lên đĩa từ

Để lưu tệp chứa dự án lên đĩa từ ta làm như sau:

- File
- Save Project as
- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (TinhLaiSuat)
- OK

Để lưu tệp chứa biểu mẫu lên đĩa từ ta làm như sau:

- File
- Save Form1 as
- Xác định thư mục lưu biểu mẫu (c:\baitapvb)
- Xác định tên tệp lưu biểu mẫu (TinhLaiSuat)
- OK

Bước 3. Thiết kế giao diện cho chương trình

Trong bước công việc này ta sử dụng các công cụ trong hộp công cụ để thiết kế giao diện của chương trình TinhLaiSuat. Các đối tượng của chương trình này là:

Bảng 7.2. Các đối tượng của chương trình TinhLaiSuat

Số TT	Loại đối tượng	Thuộc tính	Giá trị
1	Biểu mẫu	Name Caption	FrmTinhLaiSuat Tinh Lai Suat
3	Nút lệnh	Name Caption Font	CmdChayCT &Chạy CT .VnTime
4	Nút lệnh	Name Caption	CmdKetThuc &Kết Thúc

Bước 4. Viết các thủ tục - biến cố cho các nút lệnh

Phần mã lệnh của chương trình TinhLaiSuat có dạng sau đây:

```
Private Sub cmdChayct_Click()
    Dim STGui, STRut, LaiSuat, STTGian, n
    Cls
    Print:
    STGui = Val(InputBox("ST Gui = ", "Nhập ST Gui", "250"))
    STRut = Val(InputBox("ST Rut = ", "Nhập ST Rut", "350"))
    n = Val(InputBox("Số Năm = ", "Nhập Số năm", "5"))
    Print "Số Tiền Gui = ", STGui
    Print "Số Tiền Rut = ", STRut
    Print "Số Năm = ", n
    LaiSuat = 0
    STTGian = STGui * (1 + LaiSuat) ^ n
```

```

Do While STTGian < STRut
    LaiSuat = LaiSuat + 0.001
    STTGian = STGui * (1 + LaiSuat) ^ n
Loop
Print "Lai Suat nho nhat = ", LaiSuat
End Sub

Private Sub cmdKetThuc_Click()
    Beep
End
End Sub

```

Giải thích chương trình

Chương trình này có 5 biến cục bộ là STGui, STRut, LaiSuat, STTGian và n.

Biến n dùng để chứa số năm khách hàng gửi tiền tiết kiệm. Biến STGui dùng để chứa số tiền khách hàng đã gửi. Biến STRut dùng để chứa số tiền khách hàng muốn có khi rút tiền. Giá trị 3 biến này được người sử dụng nhập vào từ bàn phím. Biến LaiSuat dùng để chứa lãi suất sẽ tính được. Biến STTGian (Số tiền Trung Gian) là một biến trung gian. Biến này chứa số tiền tương ứng với mỗi mức lãi suất khác nhau. STTGian được tính theo công thức sau đây:

$$STTGian = STGui * (1 + LaiSuat) ^ n$$

Nhìn công thức này ta thấy rằng STTGian tỷ lệ thuận với lãi suất. Vì vậy lãi suất càng lớn thì STTGian càng lớn.

Lãi suất cần tìm được tính bằng cách tăng dần giá trị biến LaiSuat. Đầu tiên biến LaiSuat được gán giá trị 0. Sau đó, cứ mỗi lần

lập lại chu trình biến này được tăng thêm 0.001 (một phần nghìn). Quá trình lập lại chu trình sẽ kết thúc khi biểu thức:

`STTGian < STRut` cho giá trị False.

Khi người sử dụng nháy chuột lên nút lệnh “Chạy CT” thì thủ tục - biến cố `cmdChayct_Click()` bắt đầu được thực hiện.

Đầu tiên biến `LaiSuat` được gán giá trị 0. Sau đó máy tính `STTGian` theo công thức:

$$STTGian = STGui * (1 + LaiSuat)^n$$

Lúc này `LaiSuat = 0` nên $(1+LaiSuat) = 1$, do đó $(1+LaiSuat)^n$ cũng bằng 1. Vì vậy lúc này `STTGian = STGui`. Sau đó bắt đầu thực hiện câu lệnh chu trình để tính `LaiSuat` theo các bước sau đây:

(1). Tính giá trị biểu thức kiểm tra “`STTGian < STRut`”. Nếu giá trị biểu thức này là True thì thực hiện bước 2. Ngược lại, nếu giá trị biểu thức này là False thì chuyển đến thực hiện câu lệnh đứng sau từ khoá ***Loop***. Đó là câu lệnh:

`Print "Lai Suat nho nhat = ", LaiSuat`

Câu lệnh này in lãi suất nhỏ nhất cần tìm. Sau đó máy kết thúc thủ tục - biến cố này.

(2). Thực hiện thân chu trình, gồm 2 câu lệnh sau đây:

`LaiSuat = LaiSuat + 0.001` ` tăng `LaiSuat` thêm 1 phần nghìn

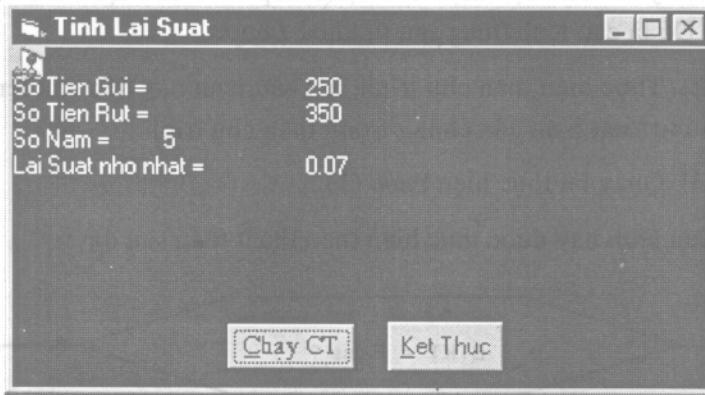
`STTGian = STGui * (1 + LaiSuat) ^ n` ` tính lại `STRut`

(3). Quay lại thực hiện bước (1)

Chạy thử chương trình TinhLaiSuat

Hãy gõ phím F5 để thực hiện chương trình `TinhLaiSuat`. Cho `STGui`, `STRut` và `n` các giá trị khác nhau để xem kết quả tính toán và

kiểm tra độ chính xác của chương trình. Hình 7.6. là kết quả thực hiện chương trình *TinhLaiSuat* với các giá trị của các biến đã cho trong chương trình.



Hình 7.6. Màn hình giao diện của chương trình *TinhLaiSuat* khi thực hiện chương trình

7.4. LỆNH CHU TRÌNH LOGIC DO UNTIL ... LOOP

Lệnh này cũng là lệnh chu trình logic. Cũng như câu lệnh *Do While ... Loop*, câu lệnh *Do Until ... Loop* cũng là câu lệnh chu trình “kiểm tra trước”, nghĩa là điều kiện kết thúc chu trình được kiểm tra trước khi thực hiện thân chu trình. Việc thực hiện chu trình được lặp lại khi biểu thức kiểm tra có giá trị False).

Lệnh này có quy cách như sau:

DO UNTIL <biểu thức kiểm tra>

Khối lệnh (thân chu trình)

LOOP

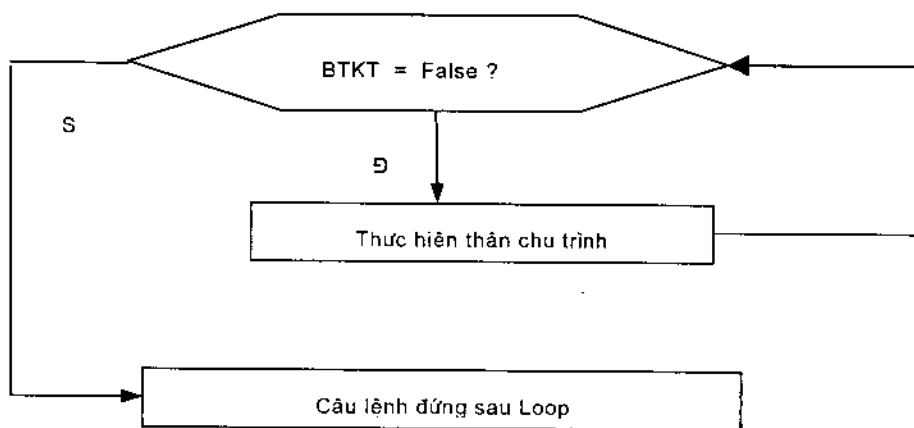
Câu lệnh này được thực hiện như sau:

(1). Tính giá trị biểu thức kiểm tra. Nếu giá trị này là False thì thực hiện bước (2). Ngược lại, nếu giá trị này là True thì kết thúc chu trình, chuyển đến thực hiện câu lệnh tiếp theo câu lệnh *Do Until ... Loop*. Đó là câu lệnh đứng sau từ khoá *Loop*.

(2). Thực hiện thân chu trình, hết câu lệnh này đến câu lệnh khác theo thứ tự xuất hiện của chúng trong thân chu trình.

(3). Quay lại thực hiện bước (1)

Câu lệnh này được thực hiện theo thuật toán sau đây:



Hình 7.7. Thuật toán thực hiện lệnh *Do Until*

Trong mục trên ta đã sử dụng câu lệnh *Do While... Loop* để viết thủ tục - biến cố `cmdChayct-click()` của chương trình *TinhLaiSuat*. Ta cũng có thể sử dụng lệnh *Do Until... Loop* để làm công việc đó.

Sau đây là thủ tục - biến cố `cmdChayct-click()` của chương trình *TinhLaiSuat* có sử dụng câu lệnh *Do Until... Loop*.

```

Private Sub cmdChayct_Click()
    Dim STGui, STRut, LaiSuat, STTGian, n
    Cls
    Print:
    STGui = Val(InputBox("ST Gui = ", "Nhap ST Gui",
"250"))
    STRut = Val(InputBox("ST Rut = ", "Nhap ST Rut",
"350"))
    n = Val(InputBox("So Nam = ", "Nhap So nam", "5"))
    Print "So Tien Gui = ", STGui
    Print "So Tien Rut = ", STRut
    Print "So Nam = ", n
    LaiSuat = 0
    STTGian = STGui * (1 + LaiSuat) ^ n
    Do Until STTGian >= STRut
        LaiSuat = LaiSuat + 0.001
        STTGian = STGui * (1 + LaiSuat) ^ n
    Loop
    Print "Lai Suat nho nhat = ", LaiSuat
End Sub

```

Thủ tục - biến cố này cho kết quả hoàn toàn giống thủ tục - biến cố cmdChayct_Click() có trong mục 7.3.

7.5. LỆNH CHU TRÌNH LOGIC DO ... LOOP WHILE

Lệnh này cũng là lệnh chu trình logic. Khác với câu lệnh *Do While ... Loop*, câu lệnh *Do ... Loop While* là câu lệnh chu trình “kiểm tra sau”, nghĩa là điều kiện kết thúc chu trình được kiểm tra sau khi thực hiện thân chu trình. Việc lặp lại chu trình được thực hiện khi

mà biểu thức kiểm tra có giá trị True.

Lệnh này có quy cách như sau:

DO

Khối lệnh (thân chu trình)

LOOP WHILE <biểu thức kiểm tra>

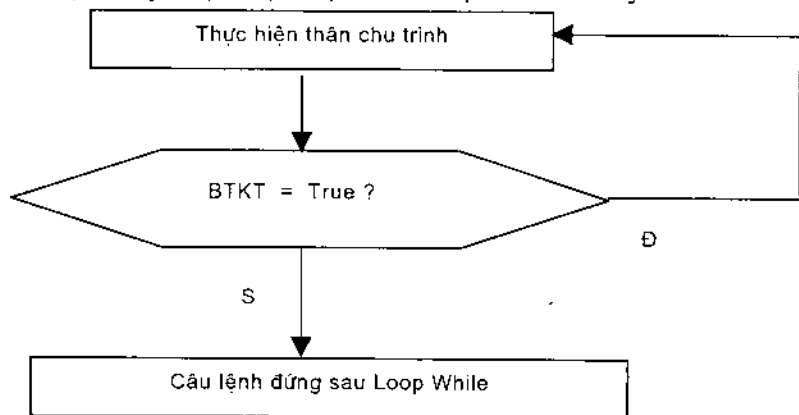
Câu lệnh này được thực hiện như sau:

(1). Thực hiện thân chu trình.

(2). Tính giá trị biểu thức kiểm tra. Nếu giá trị này là True thì quay lại thực hiện bước (1). Ngược lại, nếu giá trị này là False thì chuyển đến thực hiện bước (3).

(3). Kết thúc chu trình, chuyển đến thực hiện câu lệnh tiếp theo câu lệnh *Do ... Loop While*. Đó là câu lệnh đứng sau các từ khoá *Loop While*.

Câu lệnh này được thực hiện theo thuật toán sau đây:



Hình 7.8. Thuật toán thực hiện lệnh Do... Loop While

Như vậy, chúng ta thấy rằng theo câu lệnh thì thân chương trình được thực hiện ít nhất một lần.

Sau đây là một chương trình có sử dụng lệnh *Do... Loop while*. Chương trình này có tên là *TinhSoNam* (Tính Số Năm).

Giới thiệu chương trình TinhSoNam

Bài toán tính số năm chờ đợi có nội dung như sau: Anh X có một số tiền nhàn rỗi. Anh đem gửi tiết kiệm số tiền này. Anh X hi vọng rằng đến một lúc nào đó trong tương lai anh sẽ có đủ tiền để mua một chiếc ô tô. Cho biết số tiền anh X gửi (STGui), số tiền anh X muốn có trong tương lai (STTuongLai) và lãi suất tiết kiệm (LaiSuat). Hãy viết chương trình tính số năm ít nhất mà anh X phải chờ đợi kể từ năm gửi tiền để có được số tiền anh muốn có. Giả sử lãi suất tiết kiệm không thay đổi, anh X không rút bớt tiền và cũng không gửi thêm tiền trong suốt thời gian chờ đợi.

Chương trình *TinhSoNam* dùng để giải bài toán này.

Màn hình giao diện của chương trình *TinhSoNam* có dạng màn hình giao diện đơn giản mà ta đã biết trong các phần trên.

Thực hiện các bước công việc sau đây để thiết kế chương trình *TinhSoNam*.

Bước 1. Tạo một dự án mới cho chương trình TinhSoNam

Để làm việc này cần phải thực hiện các thao tác sau đây:

- Khởi động VB
- File
- New Project

Một biểu mẫu rỗng xuất hiện trên màn hình.

Bước 2. Lưu dự án mới lên đĩa từ

Để lưu tệp chứa dự án lên đĩa từ ta làm như sau:

- File
- Save Project as
- Xác định thư mục lưu dự án (c:\baitapvb)
- Xác định tên tệp lưu dự án (TinhSoNam)
- OK

Để lưu tệp chứa biểu mẫu lên đĩa từ ta làm như sau:

- File
- Save Form1 as
- Xác định thư mục lưu biểu mẫu (c:\baitapvb)
- Xác định tên tệp lưu biểu mẫu (TinhSoNam)
- OK

Bước 3. Thiết kế giao diện cho chương trình

Trong bước công việc này ta sử dụng các công cụ trong hộp công cụ để thiết kế giao diện của chương trình TinhSoNam. Các đối tượng của chương trình này là:

Bảng 7.3. Các đối tượng của chương trình TinhSoNam

Số TT	Loại đối tượng	Thuộc tính	Giá trị
1	Biểu mẫu	Name	FrmTinhSoNam
		Caption	Tinh So Nam
3	Nút lệnh	Name	CmdChayCT
		Caption	&Chạy CT
4	Nút lệnh	Name	CmdKetThuc
		Caption	&Kết Thúc

Bước 4. Viết các thủ tục - biến cố cho các nút lệnh

Phần mã lệnh của chương trình TinhSoNam có dạng sau đây:

```
Private Sub cmdChayct_Click()  
    Dim STGui, STTuongLai, LaiSuat, STTGian, n  
    Cls  
    Print:  
    STGui = Val(InputBox("ST Gui = ", "Nhập ST Gui",  
"250"))  
    STTuongLai = Val(InputBox("ST Tuong Lai = ", _  
"Nhập ST Tuong Lai", "350"))  
    LaiSuat = Val(InputBox("Lai Suat = ", _  
"Nhập Lai Suat ", "0.04"))  
    Print "So Tien Gui = ", STGui  
    Print "So Tien Tuong Lai = ", STTuongLai  
    Print "Lai Suat = ", LaiSuat  
    n = 0  
    Do  
        n = n + 1  
        STTGian = STGui * (1 + LaiSuat) ^ n  
    Loop While STTGian < STTuongLai  
    Print "So Nam Cho Doi = ", n  
End Sub  
  
Private Sub cmdKetThuc_Click()  
    Beep  
    End  
End Sub
```

Giải thích chương trình

Chương trình này có 5 biến là STGui, STTuongLai, LaiSuat, STTGian và n.

Biến STGui dùng để chứa số tiền khách hàng đã gửi. Biến STTuongLai dùng để chứa số tiền khách hàng muốn có khi rút tiền. Biến LaiSuat dùng để chứa lãi suất tiết kiệm. Giá trị 3 biến này được người sử dụng nhập vào từ bàn phím. Biến n dùng để chứa số năm khách hàng phải chờ đợi. Giá trị của biến n sẽ được tính khi thực hiện chương trình này. Biến STTGian (Số tiền Trung Gian) là một biến trung gian. Biến này chứa số tiền tương ứng với mỗi giá trị khác nhau của số năm chờ đợi. Ta đã biết cách tính STTGian trong chương trình TinhLaiSuat. STTGian được tính theo công thức sau đây:

$$STTGian = STGui * (1 + LaiSuat) ^ n$$

Nhìn công thức này ta thấy rằng STTGian tỷ lệ thuận với n, là số năm. Vì vậy n càng lớn thì STTGian càng lớn.

Số năm chờ đợi được tính bằng cách tăng dần giá trị biến n. Đầu tiên biến n được gán giá trị 0. Sau đó, cứ mỗi lần lặp lại chu trình biến này được tăng thêm 1. Quá trình lặp lại chu trình được thực hiện khi biểu thức:

$$STTGian < STTuongLai \quad \text{cho giá trị True.}$$

Khi người sử dụng nháy chuột lên nút lệnh “Chạy CT” thì thủ tục - biến cố cmdChayct_Click() bắt đầu được thực hiện.

Đầu tiên biến n được gán giá trị 0. Sau đó bắt đầu thực hiện câu lệnh chu trình để tính LaiSuat theo các bước sau đây:

(1). Thực hiện thân chu trình, gồm 2 câu lệnh sau đây:

$n = n + 1$ tăng n thêm 1

$STTGian = STGui * (1 + LaiSuat) ^ n$ tính lại STTGian

(2). Tính giá trị biểu thức kiểm tra " $STTGian < STTuongLai$ ". Nếu giá trị biểu thức này là True thì quay lại thực hiện bước (1). Ngược lại, nếu giá trị biểu thức này là False thì chuyển đến thực hiện bước (3).

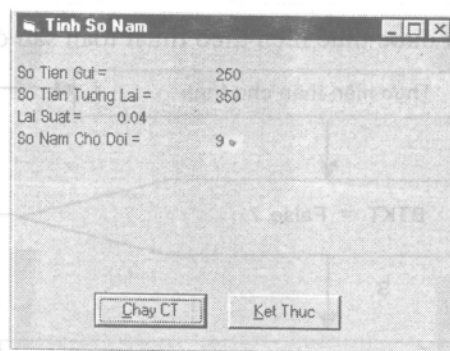
(3). Thực hiện câu lệnh tiếp theo câu lệnh Loop While. Đó là câu lệnh:

Print "Số Năm Cho Dãi = ", n

Câu lệnh này in giá trị biến n . Đó là số năm khách hàng phải chờ đợi để có đủ số tiền tương lai.

Chạy thử chương trình TinhSoNam

Hãy gõ phím F5 để thực hiện chương trình TinhSoNam. Cho các biến STGui, STTuongLai, LaiSuat các giá trị khác nhau để xem kết quả tính toán và kiểm tra độ chính xác của chương trình. Hình 7.9 là kết quả thực hiện chương trình TinhSoNam với các giá trị của các biến đã cho trong chương trình.



Hình 7.9.

7.6. LỆNH CHU TRÌNH LOGIC DO ... LOOP UNTIL

Cũng giống như câu lệnh *Do ... Loop While*, câu lệnh *Do ... Loop Until* là câu lệnh chu trình logic “kiểm tra sau”, nghĩa là điều kiện kết thúc chu trình được kiểm tra sau khi thực hiện thân chu trình. Tuy nhiên, đối với câu lệnh này thì việc lặp lại chu trình được thực hiện khi mà biểu thức kiểm tra có giá trị False.

Lệnh này có quy cách như sau:

DO

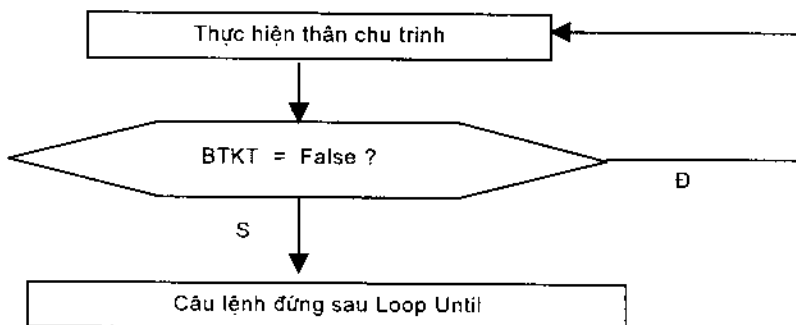
 Khối lệnh (thân chu trình)

LOOP UNTIL <biểu thức kiểm tra>

Câu lệnh này được thực hiện như sau:

- (1). Thực hiện thân chu trình.
- (2). Tính giá trị biểu thức kiểm tra. Nếu giá trị này là False thì quay lại thực hiện bước (1). Ngược lại, nếu giá trị này là True thì chuyển đến thực hiện bước (3).
- (3). Kết thúc chu trình, chuyển đến thực hiện câu lệnh tiếp theo câu lệnh *Do ... Loop Until*. Đó là câu lệnh đứng sau các từ khoá *Loop Until*.

Câu lệnh này được thực hiện theo thuật toán sau đây:



Hình 7.10. Thuật toán thực hiện lệnh Do... Loop While

Trong phần 7.5 ta đã sử dụng câu lệnh **Do ... Loop while** để viết thủ tục - biến cố cmdChayct_Click() cho chương trình **TinhSoNam**. Sau đây ta sẽ dùng câu lệnh **Do ... Loop Until** để viết thủ tục - biến cố cmdChayct_Click() cho chương trình này.

```
Private Sub cmdChayct_Click()  
    Dim STGui, STTuongLai, LaiSuat, STTGian, n  
    Cls  
    Print:  
    STGui = Val(InputBox("ST Gui = ", "Nhập ST Gui", "250"))  
    STTuongLai = Val(InputBox("ST Tuong Lai = ", _  
        "Nhập ST Tuong Lai", "350"))  
    LaiSuat = Val(InputBox("Lai Suat = ", _  
        "Nhập Lai Suat ", "0.04"))  
    Print "So Tien Gui = ", STGui  
    Print "So Tien Tuong Lai = ", STTuongLai  
    Print "Lai Suat = ", LaiSuat  
    n = 0  
    Do  
        n = n + 1  
        STTGian = STGui * (1 + LaiSuat) ^ n  
    Loop Until STTGian >= STTuongLai  
    Print "So Nam Cho Doi = ", n  
End Sub
```

Thủ tục - biến cố này cho kết quả hoàn toàn giống như kết quả

của thủ tục - biến cố `cmdChayct_Click()` trong phần 7.5.

Chúng ta thấy rằng Visual Basic có nhiều câu lệnh chu trình. Tùy từng bài toán cụ thể mà lập trình viên sẽ chọn câu lệnh chu trình thích hợp để viết chương trình của mình.

7.6. CÂU LỆNH EXIT

Khi nghiên cứu các câu lệnh chu trình ta thấy rằng chu trình sẽ kết thúc khi biểu thức kiểm tra có giá trị là False hoặc True tùy từng câu lệnh. Kết thúc chu trình như vậy được gọi là kết thúc chu trình bình thường.

Tuy nhiên, có nhiều trường hợp ta muốn kết thúc chu trình sớm, không đợi đến khi biểu thức kiểm tra có giá trị là False hoặc True. Kết thúc chu trình theo kiểu này được gọi là kết thúc chu trình bất thường.

Câu lệnh Exit được dùng để kết thúc chu trình bất thường. Câu lệnh này có qui cách sau đây:

Exit For hoặc

Exit Do

Câu lệnh **Exit For** dùng cho câu lệnh chu trình số học **For ... Next**.

Câu lệnh **Exit Do** dùng cho các câu lệnh chu trình logic **Do**

Câu lệnh này cho phép thoát khỏi thân chu trình (chấm dứt thực hiện thân chu trình) sớm, không đợi đến khi điều kiện kết thúc chu trình được thoả mãn.

Câu lệnh EXIT chỉ có ý nghĩa khi nằm trong thân chu trình. Gặp câu lệnh này máy lập tức chuyển đến thực hiện câu lệnh tiếp theo câu lệnh chu trình, tức là việc thực hiện thân chu trình bị chấm dứt ngay, không phụ thuộc giá trị của biểu thức kiểm tra là True hoặc False.

Sau đây là một ví dụ có sử dụng câu lệnh EXIT.

Giả sử có n người mua vé xổ số. Cho biết tên và số hiệu vé xổ số của từng người. Cho biết số hiệu vé trúng giải độc đắc. Hãy viết chương trình tìm và in tên người trúng giải độc đắc.

Chương trình phải lần lượt so sánh số hiệu vé của từng người với số hiệu vé trúng độc đắc. Tuy nhiên, khi đã tìm được người trúng giải độc đắc thì không cần tiếp tục thực hiện chương trình nữa vì chỉ có một vé trúng giải độc đắc mà thôi.

Thủ tục - biến cố cmdChayct_Click() của chương trình giải bài toán này như sau:

```
Private Sub cmdChayct_Click()  
Dim HoVaTen$, SoHieuVe$, SoHieuVeDocDac$, i, n,  
TimDuoc%  
n = Val(InputBox("So nguoi mua ve so so = "))  
SoHieuVeDocDac$ = InputBox("So Hieu Ve Doc Dac = ")  
TimDuoc = 0  
For i = 1 To n  
    HoVaTen$ = InputBox("Ho Va Ten = ")  
    SoHieuVe$ = InputBox("So Hieu Ve = ")  
    If SoHieuVe$ = SoHieuVeDocDac$ Then  
        TimDuoc% = 1  
        Print "Ong (ba)", HoVaTen$, " trung giai Doc Dac!"  
        Exit For  
    End If  
Next i  
If TimDuoc = 0 Then  
    Print "Khong co nguoi trung giai Doc Dac."  
End If  
End Sub
```

Giải thích chương trình

Chương trình này có các biến là `HoVaTen$`, `SoHieuVe$`, `SoHieuVeDocDac$`, `i`, `n`, `TimDuoc%`. Biến `HoVaTen$` chứa họ và tên những người mua vé xổ số. Biến `SoHieuVe$` dùng để chứa số hiệu vé xổ số mà người này được mua. Biến `SoHieuVeDocDac$` dùng để chứa số hiệu vé xổ số trúng giải độc đắc. Biến `n` dùng để chứa số người mua vé xổ số. Giá trị 4 biến này được người sử dụng nhập vào từ bàn phím. Biến `TimDuoc%` dùng để ghi kết quả tìm người trúng giải độc đắc. Giá trị 0 của biến này có nghĩa là không tìm được người trúng giải độc đắc. Giá trị 1 của biến này có nghĩa là đã tìm được người trúng giải độc đắc.

Khi thủ tục - biến cố này được thực hiện, đầu tiên máy yêu cầu người sử dụng nhập số người mua vé xổ số và số hiệu vé xổ số trúng giải độc đắc. Sau đó câu lệnh chương trình số học ***For ... Next*** được thực hiện. Mỗi lần lặp lại chương trình máy làm 2 việc:

- Yêu cầu người sử dụng nhập tên một người mua vé xổ số và số hiệu vé xổ số của người này.

- Kiểm tra xem vé xổ số vừa được nhập có phải là vé xổ số trúng giải độc đắc không? Nếu câu trả lời là “Không” thì tiếp tục lặp lại chu trình. Nếu câu trả lời là “Có” thì thực hiện 3 câu lệnh sau đây:

```
TimDuoc% = 1
```

```
Print "Ong (ba)", HoVaTen$, " trung giai Doc Dac!"
```

```
Exit For
```

Câu lệnh thứ nhất gán giá trị 1 cho biến `TimDuoc%` để ghi nhớ

rằng đã tìm được người trúng giải độc đắc. Câu lệnh thứ hai in họ và tên người trúng giải độc đắc. Câu lệnh thứ ba :

Exit For

Gây ra việc kết thúc chu trình bất thường.

Sau khi kết thúc câu lệnh chu trình máy sẽ kiểm tra giá trị biến `TimDuoc%`. Nếu giá trị biến này bằng 0, tức là không tìm được người trúng giải độc đắc, thì máy in thông báo về điều đó.

7.7. CHU TRÌNH LỒNG NHAU

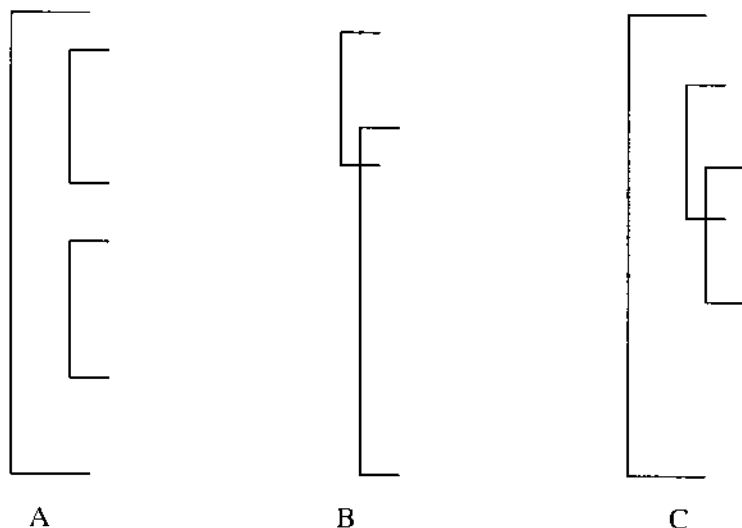
7.7.1. Khái niệm chu trình lồng nhau

Thân chu trình của câu lệnh chu trình có thể chứa câu lệnh bất kỳ. Như vậy trong thân chu trình của câu lệnh chu trình có thể có một câu lệnh chu trình khác. Trong trường hợp này ta nói rằng ta có chu trình lồng nhau. Như vậy chu trình lồng nhau là chu trình có chứa một chu trình khác. Khi đó chu trình đầu tiên được gọi là chu trình ngoài, chu trình thứ hai được gọi là chu trình trong. Một điều cần chú ý là, đến lượt mình, chu trình trong cũng có thể chứa câu lệnh chu trình. Tức là chu trình trong cũng có thể là chu trình lồng nhau. Cứ như vậy các chu trình lồng nhau có thể lồng nhau nhiều lần và có cấu trúc rất phức tạp.

7.7.2. Các qui tắc lập chương trình có chu trình lồng nhau

Khi lập các chương trình có chu trình lồng nhau cần tuân theo các qui tắc sau đây:

Quy tắc 1. Chu trình trong phải nằm gọn trong chu trình ngoài. Có nghĩa là các chu trình phải "bao nhau", chứ không được cắt nhau. Quy tắc này có thể được minh họa bằng các hình sau:



Hình 9. Hình ảnh các chu trình lồng nhau.

A. Các chu trình "bao nhau" : đúng.

B,C : Các chu trình cắt nhau, không "bao nhau" : sai.

Quy tắc 2. Được phép dùng lệnh GOTO để chuyển điều khiển từ chu trình trong ra chu trình ngoài. Đây là trường hợp kết thúc bất thường đối với chu trình trong. Nhưng không được phép dùng lệnh GOTO để chuyển điều khiển từ chu trình ngoài vào chu trình trong. Vì khi đó các biến điều khiển của chu trình trong không được gán các giá trị ban đầu. Chỉ được chuyển vào chu trình trong thông qua đầu của chu trình trong.

Quy tắc 3. Trong thân chu trình trong có thể sử dụng giá trị các biến điều khiển chu trình ngoài để tính toán, nhưng không được thay đổi các giá trị đó. Nếu không như vậy thì chu trình ngoài sẽ làm việc không chính xác.

7.7.3. Ví dụ chương trình có chu trình lồng nhau

Trong mục này ta sẽ nghiên cứu một ví dụ về chương trình có chu trình lồng nhau. Bài toán trong ví dụ này là bài toán tính giai thừa cho các số nguyên dương. Nội dung bài toán như sau:

Cho trước k số nguyên dương $n_1, n_2, \dots, n_k$. Hãy viết chương trình tính và in các số $n_1!, n_2!, \dots, n_k!$. biết rằng $n! = 1*2*\dots*n$.

Ta thấy rằng để tính giai thừa cho một số nguyên dương n ta cần một chu trình lặp lại n lần. Để tính giai thừa cho k số nguyên dương ta phải lặp lại chu trình nói trên k lần. Như vậy chương trình giải bài toán này là chương trình có chu trình lồng nhau. Thủ tục - biến cố cmdchayct_Click() của chương trình này có dạng sau đây:

```
Private Sub cmdchayct_Click()  
Dim k, GiaiThua&, i, j, n  
k = Val(InputBox("Nhap so k "))  
For i = 1 To k  
    n = Val(InputBox("Nhap 1 so nguyên dương "))  
    GiaiThua& = 1  
    For j = 1 To n  
        GiaiThua& = GiaiThua& * j  
    Next j  
    Print n; "! = "; GiaiThua&  
Next i  
End Sub
```

Giải thích chương trình

Chương trình này có các biến là Dim GiaiThua&, i, j, k, n.

Biến giaiThua& chứa giai thừa của các số nguyên dương, là các

số cần tìm. Biến k dùng để chứa số các số nguyên dương cho trước. Biến n dùng để lần lượt chứa các số nguyên dương đó. Các biến i, j là các biến điều khiển chương trình.

Khi thủ tục - biến cố này được thực hiện, đầu tiên máy yêu cầu người sử dụng nhập số k , là số các số nguyên dương cho trước. Sau đó chu trình ngoài, chu trình theo biến i , được thực hiện. Chu trình này được lặp lại k lần. Mỗi lần lặp lại chu trình này máy làm các việc sau đây:

- Yêu cầu người sử dụng nhập một số nguyên dương.

- Thực hiện câu lệnh chu trình theo biến j để tính giai thừa cho số nguyên này.

- In kết quả vừa tính được.

Ta thấy rằng thủ tục - biến cố này có chu trình lồng nhau. Chu trình ngoài là chu trình theo i . Chu trình trong là chu trình theo j . Cả chu trình ngoài lẫn chu trình trong đều là chu trình số học.

CÂU HỎI ÔN TẬP VÀ BÀI TẬP

Câu hỏi lý thuyết

1. Hãy trình bày khái niệm chương trình có chu trình.
2. Hãy phân biệt chu trình số học và chu trình logic.
3. Hãy trình bày qui cách và trình tự thực hiện các câu lệnh chu trình.
4. Hãy trình bày khái niệm chương trình có chu trình lồng nhau.
5. Hãy trình các quy tắc lập chương trình có chu trình lồng nhau.

Bài tập lập chương trình

1. Cho các số $a_1, a_2, \dots, a_n$. Hãy viết chương trình tìm số lớn nhất trong các số đó.

2. Cho biết tọa độ tâm và bán kính một hình tròn. Cho biết tọa độ các điểm $M_1, M_2, \dots, M_n$. Hãy viết chương trình cho máy tính điện tử đếm xem có bao nhiêu điểm đã cho rơi vào hình tròn đã cho.

3. Cho biết tọa độ 4 đỉnh của một hình chữ nhật có các cạnh song song với trục tọa độ. Cho biết tọa độ các điểm $M_1, M_2, \dots, M_n$. Hãy viết chương trình cho máy tính điện tử đếm xem có bao nhiêu điểm đã cho rơi vào hình chữ nhật đã cho.

4. Cho các số $a_1, a_2, \dots, a_n$. Hãy viết chương trình để:

a)- Tính và in tổng của chúng.

b)- Tính tổng các số dương và tổng các số âm có trong các số đó.

c)- Đếm số các số dương và số các số âm có trong các số đó.

d)- Đếm số các số bằng số nhỏ nhất có trong các số đó.

5. Giả sử đã cho n số thực $a_1, a_2, \dots, a_n$ (n chia hết cho 3). Hãy viết chương trình cho máy giải các phương trình bậc 2 $ax^2 + bx + c = 0$ với a, b, c là các nhóm 3 số liên tiếp của n số đã cho.

6. Viết một chương trình tính và in a^m với a là số thực, m là số nguyên dương.

7. Giả sử có n khách hàng gửi tiền tiết kiệm. Cho biết tên, số tiền đã gửi, năm gửi tiền và lãi suất tiết kiệm/năm của từng khách hàng. Hãy viết một chương trình tính số tiền hiện có của từng khách hàng (năm tính toán là năm 2002; giả sử khách hàng chỉ gửi tiền 1 lần và chưa rút tiền lần nào; lãi suất tiết kiệm không thay đổi).

8. Hãy viết chương trình cho máy tính:

$$e^x = 1 + x/1! + x^2/2! + x^3/3! + \dots + x^n/n!$$

9. Hãy viết chương trình cho máy tính $\cos(x)$ theo chuỗi Taylor:

$$\cos(x) = 1 - x^2/2! + x^4/4! - x^6/6! + \dots$$

KHAI BÁO VÀ SỬ DỤNG MẢNG

Trong các chương trước ta đã nghiên cứu các kiểu dữ liệu cơ sở của Visual Basic, như kiểu *Currency*, *Byte*, *Integer*, *Long*, *Single*, *double*... Từ các kiểu dữ liệu cơ sở đó người ta xây dựng nên các kiểu dữ liệu khác, gọi là dữ liệu có cấu trúc (hoặc dữ liệu phái sinh). Kiểu mảng là một kiểu dữ liệu có cấu trúc.

8.1. KHÁI NIỆM MẢNG

Trong các chương trình ta đã viết trước đây mỗi biến gắn liền với (hoặc thể hiện) một giá trị dữ liệu. Ta sẽ gọi những biến đó là biến đơn giản.

Để hiểu rõ mục đích sử dụng mảng ta nghiên cứu bài toán trong ví dụ 8.1.

Ví dụ 8.1: Thủ tục - biến cố sau đây nhập 4 số nguyên và in các số nguyên đó theo thứ tự ngược lại.

```
Private Sub cmdChayct_Click()  
    Dim sn1%, sn2%, sn3%, sn4%  
    sn1% = Val(InputBox("Nhập số nguyên thứ nhất"))  
    sn2% = Val(InputBox("Nhập số nguyên thứ hai"))  
    sn3% = Val(InputBox("Nhập số nguyên thứ ba"))
```

```

sn4% = Val(InputBox(" Nhập số nguyên thứ tư"))
cls
Print " Các số nguyên vừa nhập là : "
Print sn4%, sn3%, sn2%, sn1%
End Sub

```

Trong thủ tục - biến cố trên ta đã sử dụng 4 biến đơn giản là sn1, ... , sn4 để chứa 4 số nguyên được nhập vào từ bàn phím. Nếu chỉ sử dụng các biến đơn giản thì ta sẽ làm thế nào trong các trường hợp sau:

- a. Số các số nguyên cần xử lý rất lớn, lên đến hàng nghìn;
- b. Không biết cụ thể là cần xử lý bao nhiêu số nguyên;

Câu trả lời là: Đối với trường hợp a thì chương trình sẽ cực kỳ dài. Còn đối với trường hợp b thì không thể viết được chương trình.

Dữ liệu kiểu mảng được tạo ra để giải quyết những khó khăn này.

Mảng là một tập hợp gồm một số hữu hạn các phần tử có cùng một kiểu và có chung một tên. Mảng được tạo ra và tồn tại trong bộ nhớ trong của máy tính điện tử. Các phần tử của mảng được lưu trữ trong một dãy các ô nhớ liên tiếp nhau của bộ nhớ. Muốn truy nhập đến các phần tử của mảng ta phải dùng tên mảng kèm theo chỉ số. Tên mảng kèm theo chỉ số được gọi là biến có chỉ số.

Mảng có thể là mảng một chiều hoặc mảng nhiều chiều. Ta có thể khai báo mảng với số chiều là một, hai hay nhiều hơn nữa là tùy ý. Trong khuôn khổ của quyển sách này ta chỉ nghiên cứu mảng một chiều và mảng hai chiều.

Một véc tơ (gồm n phần tử cùng loại) là một mảng một chiều. Một ma trận (gồm m dòng và n cột các phần tử cùng loại) là một mảng hai chiều.

8.2. KHAI BÁO MẢNG

Trong chương trình VB, muốn sử dụng mảng thì phải khai báo mảng bằng lệnh DIM (hoặc lệnh PUBLIC).

Khai báo mảng một chiều

Lệnh khai báo mảng một chiều có qui cách như sau:

DIM tên_mảng(n) as TXĐK

Ở đây :

- Tên mảng là tên do người sử dụng đặt cho mảng. Tên mảng được đặt theo qui tắc đặt tên biến.

- *n* là một hằng số hoặc biểu thức gồm các hằng số, xác định số phần tử của mảng. Số phần tử của mảng còn được gọi là kích thước của mảng.

- Tham số TXĐK là một từ dùng để xác định kiểu dữ liệu. TXĐK có thể là một trong các từ khoá: *Currency, Byte, Integer, Long, Single, double, Variant, Boolean, String*. TXĐK xác định kiểu của các phần tử của mảng.

Câu lệnh khai báo này có hai tác dụng như sau:

- Tạo ra mảng một chiều có tên là tên_mảng có trong lệnh. Mảng này có *n+1* phần tử.

- Cho phép truy nhập các phần tử của mảng bằng các biến có chỉ số có dạng:

tên_mảng(0), tên_mảng(1), ..., tên_mảng(n)

Ở đây tên_mảng(*i*) là phần tử thứ *i* c mảng (*i* = 0, 1, ..., *n*).

Ví dụ có câu lệnh khai báo:

Dim A(10) as Integer

Trong ví dụ này tên mảng được khai báo là A, số phần tử của mảng (n) là 10, TXĐK là Integer.

Câu lệnh khai báo này có tác dụng như sau:

- Tạo ra mảng có tên là A gồm 11 phần tử (n+1), mỗi phần tử là một số nguyên. Các phần tử này được lưu trữ trong các ô liên tiếp của bộ nhớ trong của máy tính.

- Cho phép truy nhập các phần tử của mảng A bằng các biến có chỉ số: A(0), A(1), ..., A(i),, A(10). A(i) là phần tử thứ i của mảng. Trong trường hợp này các biến có chỉ số chỉ có một chỉ số.

Một số chú ý:

1. Câu lệnh khai báo mảng sau đây:

```
Dim A%(10)
```

Dùng kí tự cuối cùng của tên mảng để xác định kiểu của mảng. Câu lệnh đó tương đương với câu lệnh khai báo mảng:

```
Dim A(10) as Integer
```

mà ta đã nghiên cứu trên ở trên.

2. Trong câu lệnh khai báo mảng :

```
Dim a%(n)
```

Thì n phải là một hằng số hoặc một biểu thức gồm các hằng số. Nếu không như thế sẽ gây lỗi. Ví dụ có các câu lệnh sau đây:

```
n%=15
```

```
Dim b%(n%) `(*)
```

Trong câu lệnh khai báo mảng (*) thì n% là một biến, không phải là một hằng hoặc biểu thức gồm các hằng. Câu lệnh này không hợp lệ. Gặp câu lệnh này Visual Basic sẽ đưa ra thông báo lỗi sau đây:

Compile error:

Constant expression required

3. Khi có câu lệnh khai báo mảng:

```
Dim A(n) as Integer
```

thì miền giá trị cho phép của chỉ số là 0 -> n, tức là chỉ được dùng các biến có chỉ số A(i) với i=0, 1, ..., n. Nếu giá trị của chỉ số i vượt ra ngoài miền giá trị này thì sẽ gây lỗi. Ví dụ có các câu lệnh:

```
Dim a%(15), s%  
s%=0  
s%= s% + a%(16)  '(**)
```

Thì câu lệnh (**) là câu lệnh không hợp lệ vì giá trị của chỉ số (là 16) vượt ra ngoài miền giá trị cho phép. Gặp câu lệnh này Visual Basic đưa ra thông báo sau đây:

Run_time error '9':

Subscrip out of range

Sau đây là một ví dụ về khai báo và sử dụng mảng.

Ví dụ 8.2. Thủ tục - biến cố sau đây tính tổng n số nguyên được nhập từ bàn phím và in tổng đó.

```
Private Sub cmdChayct_Click()  
Dim s%, i%, n%  
Dim a(100) As Integer  
n% = Val(InputBox(" n = ", "Nhập số n ", "5"))  
For i = 0 To n%  
    a(i%) = Val(InputBox("Nhập một số nguyên "))  
Next  
s% = 0  
For i = 0 To n%
```

```

s% = s% + a(i%)
Next
Prin " S= ", s%
End Sub

```

Trong thủ tục - biến cố này có 2 câu lệnh chu trình For. Câu lệnh For thứ nhất dùng để nhập n số nguyên từ bàn phím. Các số nguyên đó được lưu trữ trong mảng a. Câu lệnh chương trình thứ hai dùng để tính tổng các số vừa nhập. Tổng này chứa trong biến s%.

Khi thực hiện thủ tục - biến cố này, sau khi tính và in xong tổng s%, các số nguyên vừa nhập vẫn tồn tại trong các biến có chỉ số a(0), a(1), ..., a(n). Ta có thể thực hiện các quá trình tính toán khác nhau trên các số đó.

Khai báo mảng hai chiều

Lệnh khai báo mảng hai chiều có qui cách như sau:

DIM tên_mảng(m, n) as TXĐK

Ở đây :

- Tên mảng là tên do người sử dụng đặt cho mảng. Tên mảng được đặt theo qui tắc đặt tên biến.

- *m* và *n* là một hằng số hoặc biểu thức gồm các hằng số; *m* xác định số dòng của mảng, *n* xác định số cột của mảng. Số phần tử của mảng còn được gọi là kích thước của mảng. Số phần tử của mảng 2 chiều bằng $m \times n$.

- TXĐK là một từ xác định kiểu dữ liệu (như Integer, String, Single...). TXĐK xác định kiểu của các phần tử của mảng.

Câu lệnh khai báo này có hai tác dụng như sau:

- Tạo ra mảng hai chiều (ma trận) có tên là tên_mảng đã cho

trong lệnh. Mảng hai chiều này gồm $m+1$ dòng, $n+1$ cột (tức là gồm $(m+1)*(n+1)$ phần tử).

- Cho phép truy nhập các phần tử của mảng bằng các biến có chỉ số gồm 2 chỉ số có dạng sau đây:

tên_mảng(0,0), tên_mảng(0,1), ..., tên_mảng(0, n)

tên_mảng(1,0), tên_mảng(1,1), ..., tên_mảng(1, n)

..... tên_mảng(i,j)

tên_mảng(m, 0), tên_mảng(m,1), ..., tên_mảng(m, n)

Ở đây tên_mảng(i,j) là phần tử nằm ở dòng thứ i cột thứ j của ma trận.

Ví dụ có câu lệnh khai báo:

Dim A(5, 10) as Integer

Câu lệnh này có các tác dụng sau đây:

- Tạo ra một mảng có tên là A, gồm 6 dòng và 11 cột. Mỗi phần tử của ma trận A là một số nguyên.

- Cho phép truy nhập các phần tử của mảng A bằng các biến có chỉ số:

A(0,0), A(0,1), ..., A(0, 10)

A(1,0), A(1,1), ..., A(1, 10)

..... A(i,j)

A(5, 0), A(5,1), ..., A(5, 110)

Phần tử A(i, j) là phần tử nằm ở dòng thứ i cột thứ j của ma trận A. Giá trị của i và j phải thỏa mãn điều kiện:

$0 \leq i \leq 5$

$0 \leq j \leq 10$

Chú ý: Câu lệnh khai báo:

```
Dim A%(5, 10)
```

Tương đương với câu lệnh khai báo:

```
Dim A(5, 10) as Integer
```

Sau đây là một ví dụ có khai báo mảng hai chiều.

Ví dụ 8.3. Thủ tục - biến cố sau đây tính và in tổng các phần tử của ma trận $a(m, n)$.

```
Private Sub cmdTinhVaIn_Click()  
Dim s%, i%, j%, n%, m%  
Dim a(100, 50) As Integer  
m% = Val(InputBox("Nhap so dong  "))  
n% = Val(InputBox("Nhap so cot  "))  
  
` Nhap cac phan tu cua ma tran  
For i % = 1 To m%  
    For j% = 1 To n%  
        a(i%, j%) = Val(InputBox("Nhap mot so nguyen  "))  
    Next  
Next  
  
` Tinh tong cac phan tu  
s% = 0  
For i % = 1 To m%  
    For j% = 1 To n%  
        s% = s% + a%(i%, j%)
```

```

Next
Next

Print "Tong = ", s%
End Sub

```

Trong thủ tục - biến cố này có 2 câu lệnh chu trình For lồng nhau. Câu lệnh For lồng nhau thứ nhất dùng để nhập các phần tử của ma trận a từ bàn phím. Câu lệnh For lồng nhau thứ hai dùng để tính tổng các phần tử của ma trận đó. Tổng này được chứa trong biến s%.

8.3. MỘT SỐ VÍ DỤ XỬ LÝ MẢNG

Ví dụ 8.4. Giả sử cho biết thu nhập của n công nhân. Thủ tục - biến cố sau đây tìm và in thu nhập nhỏ nhất trong các thu nhập đã cho.

```

Private Sub cmdchayct_Click()
Dim s, i, j, n, min
Dim ThuNhap(100)
n = Val(InputBox(" n = ", "Nhap so n ", "3"))
Cls

' Nhap cac PT cua mang
For i = 0 To n
    ThuNhap(i) = Val(InputBox("Thu Nhap 1 cong nhan = 
"))
Next

' Tim thu nhap nho nhat
min = ThuNhap(0)

```

```

For i = 1 To n
    If min > ThuNhap(i) Then
        min = ThuNhap(i)
    End If
Next
Print "Thu Nhap nho nhat  = ", min
End Sub

```

Trong thủ tục - biến cố này mảng ThuNhap(100) dùng để chứa thu nhập của n công nhân ($n \leq 100$). Biến min dùng để chứa thu nhập nhỏ nhất.

Trong thủ tục - biến cố này có 2 câu lệnh chu trình For. Câu lệnh For thứ nhất dùng để nhập từ bàn phím các phần tử của mảng ThuNhap(n). Câu lệnh chu trình For thứ hai dùng để xác định phần tử nhỏ nhất của mảng đó. Biến min dùng để chứa phần tử nhỏ nhất này. Đầu tiên biến min được gán giá trị ThuNhap(0). Sau đó biến min lần lượt được so sánh với các phần tử còn lại của mảng ThuNhap(n). Mỗi khi điều kiện:

```
min > ThuNhap(i)
```

được thoả mãn biến min sẽ được gán giá trị của ThuNhap(i) bằng câu lệnh gán sau đây:

```
min = ThuNhap(i)
```

Khi câu lệnh chu trình kết thúc biến min sẽ chứa phần tử nhỏ nhất của mảng ThuNhap(n). Giá trị biến min được in trên màn hình biểu mẫu bằng câu lệnh sau đây:

```
Print "Thu Nhap nho nhat  = ", min
```

Đây là kết quả thực hiện chương trình này.

Ví dụ 8.5. Cho mảng $a(n)$ gồm n số nguyên. Thủ tục - biến cố sau đây sắp xếp các phần tử của mảng $a(n)$ theo thứ tự tăng dần và in các phần tử đó theo thứ tự đã sắp xếp.

```
Private Sub cmdTinhVaIn_Click()  
Dim atg, i, j, n  
Dim a(100) As Integer  
  
' Nhập dữ liệu  
n = Val(InputBox("n = ", "Nhập số n ", "4"))  
For i = 0 To n  
    a(i) = Val(InputBox(" ai = ", "Nhập các số ai ", "0"))  
    Print a(i);  
Next  
  
' Sắp xếp tăng dần  
For i = 0 To n - 1  
    For j = i + 1 To n  
        If a(i) > a(j) Then  
            atg = a(i): a(i) = a(j): a(j) = atg  
        End If  
    Next j  
Next i  
  
' In kết quả  
For i = 0 To n  
    Print a(i);  
Next  
End Sub
```

Trọng tâm của thủ tục - biến cố này là đoạn chương trình sắp xếp tang dần. Có nhiều phương pháp sắp xếp dãy số theo thứ tự tăng dần. Ý tưởng của phương pháp được sử dụng ở đây như sau:

- Đầu tiên biến phần tử $a(0)$ thành phần tử nhỏ nhất trong n phần tử đã cho.

- Sau đó biến phần tử $a(1)$ thành phần tử nhỏ nhất trong $n-1$ phần tử còn lại.

-

- Sau đó biến phần tử $a(i)$ thành phần tử nhỏ nhất trong $n-i$ phần tử còn lại.

-

- Cuối cùng biến phần tử $a(n-2)$ thành phần tử nhỏ nhất trong 2 phần tử còn lại.

Để biến phần tử $a(i)$ thành phần tử nhỏ nhất trong $n - i$ phần tử còn lại ta làm như sau: lần lượt so sánh $a(i)$ với các phần tử đứng sau nó. Nếu $a(i)$ lớn hơn một phần tử $a(j)$ nào đó thì đổi chỗ 2 phần tử $a(i)$ và $a(j)$. Câu lệnh chu trình:

```
For j = i + 1 To n
```

```
  If a(i) > a(j) Then
```

```
    atg = a(i): a(i) = a(j): a(j) = atg
```

```
  End If
```

```
Next j
```

thực hiện việc biến phần tử $a(i)$ thành phần tử nhỏ nhất trong $n - i$ phần tử còn lại của mảng $a(n)$.

Ví dụ 8.6. Giả sử có m sinh viên, mỗi sinh viên thi n môn học. Cho biết tên từng sinh viên, điểm thi từng môn học của từng sinh viên. Hãy viết chương trình tính và in tên và điểm trung bình cho từng sinh viên (theo thứ tự tăng dần của điểm trung bình). Thủ tục - biến cố sau đây sẽ giải bài toán này.

```
Private Sub cmdChayct_Click()
Dim tensv$(100), Diem(100, 60), Diemtb(100)
Dim i, j, m, n, s, tg

' Nhap Du Lieu
m = Val(InputBox(" So sinh vien = ", " ", " 4"))
n = Val(InputBox(" So mon hoc = ", " ", " 3"))
For i = 0 To m - 1
    tensv$(i) = InputBox(" Ten sinh vien = ")
    For j = 0 To n - 1
        Diem(i, j) = Val(InputBox("Diem thi = "))
    Next j
Next i

' Tinh Diem TB
For i = 0 To m - 1
    s = 0
    For j = 0 To n - 1
        s = s + Diem(i, j)
    Next j
    Diemtb(i) = s / n
Next i
```

```

' Sáp xếp theo DiemTB
For i = 0 To m - 2
    For j = i + 1 To m - 1
        If Diemtb(i) > Diemtb(j) Then
            tg = tensv$(i)
            tensv$(i) = tensv$(j)
            tensv$(j) = tg
            tg = Diemtb(i)
            Diemtb(i) = Diemtb(j)
            Diemtb(j) = tg
        End If
    Next j
Next i

' In kết quả
Cls
For i = 0 To m - 1
    Print tensv$(i), Diemtb(i)
Next i
End Sub

```

Thủ tục - biến cố này có 3 mảng là `tensv$(100)`, `Diem(100, 60)` và `Diemtb(100)`. Mảng `tensv$(100)` dùng để chứa tên của `n` sinh viên. Mảng `Diem(100, 60)` dùng để chứa điểm thi từng môn học của từng sinh viên. Mảng `Diemtb(100)` dùng để chứa điểm trung bình của từng sinh viên. Các biến `m` (số sinh viên) và `n` (số môn học) phải thoả mãn điều kiện $m \leq 100$, $n \leq 60$.

Thủ tục - biến cố này có 4 đoạn chương trình riêng biệt. Đó là các đoạn chương trình:

* **Nhap Du Lieu:** Đoạn chương trình này dùng để nhập các dữ liệu ban đầu. Các dữ liệu đó gồm số sinh viên, số môn học, tên từng sinh viên và điểm thi từng môn học của từng sinh viên.

* **Tinh Diem TB:** Đoạn chương trình này dùng để tính điểm trung bình cho từng sinh viên. Đối với mỗi sinh viên, đầu tiên máy tính tổng điểm của sinh viên này. Sau đó tổng này được chia cho số môn học để tính điểm trung bình.

* **Sap xep theo DiemTB:** Đoạn chương trình này dùng để sắp xếp tên sinh viên và điểm trung bình theo thứ tự tăng dần của điểm trung bình.

* **In ket qua:** Đoạn chương trình này dùng để in tên và điểm trung bình của từng sinh viên. Đây là kết quả thực hiện chương trình.

CÂU HỎI ÔN TẬP VÀ BÀI TẬP

Lý thuyết

1. Hãy trình bày khái niệm dữ liệu mảng.
2. Hãy nêu qui cách và tác dụng của câu lệnh khai báo mảng một chiều.
3. Hãy trình bày cách phân bố các phần tử của mảng trong bộ nhớ.
4. Hãy trình bày khái niệm mảng 2 chiều.
5. Biến cố chỉ số là gì? Hãy trình bày cách truy nhập các phần tử của mảng.

Bài tập lập chương trình

1. Cho biết n số thực. Hãy viết chương trình cho máy tính và in giá trị trung bình của các số dương có trong các số đã cho.

2. Cho biết n số thực. Hãy viết chương trình cho máy tính và in khoảng biến thiên của các số đã cho.

3. Cho biết n số thực. Hãy viết chương trình cho máy in các khoảng tăng dần có trong các số đã cho.

4. Cho biết n số thực (n lẻ). Hãy viết chương trình cho máy sắp xếp các số đó sao cho số lớn nhất đứng giữa, số lớn thứ hai đứng bên trái, số lớn thứ 3 đứng bên phải...

5. Cho biết n số thực. Hãy viết chương trình cho máy tính và in giá trị mod của các số đã cho.

6. Cho biết tuổi và thu nhập của n cán bộ. Hãy viết chương trình xác định thu nhập trung bình của các cán bộ ở mỗi mức tuổi.

7. Giả sử có m sinh viên, mỗi sinh viên đã thi n môn học. Cho biết tên, điểm thi từng môn học của từng sinh viên. Hãy viết chương trình cho máy tính và in điểm trung bình của các sinh viên giỏi (tức là có điểm trung bình ≥ 8).

8. Giả sử có m sinh viên, mỗi sinh viên đã thi n môn học. Cho biết điểm thi từng môn học của từng sinh viên. Hãy viết chương trình cho máy xác định số môn học có điểm thi trung bình nhỏ hơn 5.

9. Cho biết n số thực. Một số được gọi là thừa nếu nó bằng số đứng ngay trước nó. Hãy viết chương trình xóa tất cả các số thừa.

10. Cho số thực x và một dãy gồm n số thực đã được sắp xếp theo thứ tự tăng dần. Hãy viết chương trình cho máy chèn số x vào dãy này cho sao cho dãy mới tạo ra cũng được sắp xếp theo thứ tự tăng dần.

HÀM VÀ THỦ TỤC

Hàm và thủ tục là những yếu tố rất quan trọng và được sử dụng rất phổ biến trong chương trình Visual Basic. Ngay trong chương trình đầu tiên, chương trình Hello, ta đã sử dụng các thủ tục - biến cố liên quan đến các nút lệnh. Chương này sẽ nghiên cứu các hàm chuẩn của Visual Basic và cách viết các hàm và thủ tục do người sử dụng tự định nghĩa.

9.1. LẬP TRÌNH CÓ CẤU TRÚC

Trong thực tế thường gặp những bài toán lớn. Việc viết, thử nghiệm, sửa chữa và bảo trì những chương trình như vậy rất khó khăn. Để cho công việc dễ dàng hơn người ta thường làm như sau: Trước tiên người ta chia bài toán này thành các bài toán nhỏ. Chương trình giải mỗi bài toán nhỏ sẽ được viết dưới dạng một chương trình con. Việc thử nghiệm, hiệu chỉnh, gỡ rối những chương trình con này dễ hơn. Mặt khác, việc viết và thử nghiệm các chương trình con có thể được thực hiện bởi nhiều người khác nhau, nên công việc được thực hiện nhanh hơn. Sau đó các chương trình con này được ghép lại để trở thành một chương trình thống nhất để giải bài toán lớn nói trên. Trong trường hợp này mỗi chương trình con có thể chỉ được sử dụng một lần. Phương pháp lập trình này được gọi là lập trình có cấu trúc hoặc lập trình thủ tục. Đây là lý do thứ nhất của việc viết và sử dụng chương trình con.

Khi thiết kế các chương trình trong các chương trước ta đã sử dụng phương pháp lập trình thủ tục. Mỗi chương trình thường gồm 4 công việc là nhập dữ liệu, tính và in (kết quả), xoá màn hình và kết thúc chương trình. Ta đã viết 4 thủ tục - biến cố riêng biệt để thực hiện 4 công việc đó. Mỗi thủ tục - biến cố này gắn liền với một nút lệnh. Thủ tục - biến cố sẽ được thực hiện khi người sử dụng nhấn chuột vào nút lệnh tương ứng. Ta có thể lần lượt viết và thử nghiệm từng thủ tục - biến cố một. Nhờ vậy công việc dễ dàng hơn và đỡ đơn điệu hơn.

Lý do thứ hai của việc viết và sử dụng chương trình con là chương trình con cho phép thực hiện nguyên tắc: "Viết một lần, dùng nhiều lần; một người viết, nhiều người dùng". Cụ thể như sau: Trong quá trình lập chương trình ta thường gặp trường hợp là có những đoạn chương trình được lặp đi lặp lại nhiều lần ở những chỗ khác nhau của chương trình. Để chương trình được ngắn gọn và tiết kiệm thời gian và bộ nhớ, người ta viết những đoạn chương trình đó một lần dưới dạng chương trình con. Khi cần chỉ việc viết lời gọi thực hiện chương trình con đó, chứ không phải viết lại chương trình con đó nữa. Hơn nữa, chương trình con này có thể được cung cấp cho người khác để sử dụng trong chương trình của họ. Cách sử dụng chương trình con kiểu này làm cho chương trình ngắn hơn, sáng sủa, dễ đọc, dễ hiểu hơn và cho phép tiết kiệm rất nhiều thời gian và công sức của các lập trình viên.

9.2. CÁC KHÁI NIỆM

Trong mục này sẽ trình bày các khái niệm liên quan đến hàm và thủ tục.

Chương trình con (subroutin hoặc subprogram) là một chương trình được viết theo những quy tắc đặc biệt để có thể được gọi thực hiện bằng một chương trình khác. Chương trình chứa lời gọi thực hiện chương trình con được gọi là chương trình chính.

Trong đa số các ngôn ngữ lập trình bậc cao chương trình con được chia thành hai loại là thủ tục và hàm. Như vậy khi nói chương trình con thì có nghĩa là muốn nói đến một thủ tục hoặc một hàm.

Hàm (function) là một chương trình con dùng để tính một giá trị nào đấy và trả lại giá trị đó thông qua tên hàm. Ví dụ hàm SIN cho giá trị sin của x thông qua $SIN(X)$. Trong chương trình nguồn $SIN(X)$ được gọi là lời gọi hàm. Nó đại diện cho (hoặc thể hiện) giá trị của hàm SIN ứng với X . Vì $SIN(X)$ là một giá trị nên nó có thể xuất hiện trong các biểu thức

Thủ tục (subroutin hoặc procedure) là một chương trình con dùng để thực hiện một công việc riêng biệt trên một tập hợp các tham số (hoặc không cần các tham số). Thủ tục sẽ được thực hiện mỗi khi gặp lời gọi thủ tục. Khác với hàm, thủ tục không trả lại giá trị thông qua tên thủ tục.

Hàm chuẩn (standard function) là những hàm đã được định nghĩa sẵn trong ngôn ngữ lập trình. Lập trình viên có thể sử dụng các hàm chuẩn mà không phải định nghĩa lại các hàm đó. Hầu hết các hàm toán học thông dụng đều là các hàm chuẩn. Tập hợp các hàm chuẩn tạo thành thư viện hàm chuẩn của ngôn ngữ lập trình. Thư viện hàm chuẩn là thành phần rất quan trọng của mỗi ngôn ngữ lập trình. Nó làm cho việc lập trình được dễ dàng hơn và làm cho khả năng của ngôn ngữ lập trình tăng lên rất nhiều.

Hàm tự định nghĩa (user-defined function) là những hàm không có sẵn trong thư viện hàm chuẩn của ngôn ngữ lập trình. Muốn sử dụng những hàm này người sử dụng phải tạo ra nó, tức là phải định nghĩa nó. Vì vậy những hàm này mới có tên là hàm do người sử dụng định nghĩa, tiếng Anh là user-defined function, viết tắt là UDF.

Thủ tục chuẩn (standard procedure) là những thủ tục đã được định nghĩa sẵn trong ngôn ngữ lập trình. Lập trình viên có thể sử dụng

các thủ tục này mà không phải định nghĩa lại các thủ tục đó. Các quá trình nhập-xuất dữ liệu thường được thực hiện nhờ các thủ tục nhập-xuất chuẩn của các ngôn ngữ lập trình.

Thủ tục tự định nghĩa (user - defined procedure) là những thủ tục không có sẵn trong ngôn ngữ lập trình. Muốn sử dụng những thủ tục này người sử dụng phải tạo ra nó, tức là phải định nghĩa nó.

Việc tạo ra và sử dụng các hàm tự định nghĩa và các thủ tục tự định nghĩa làm cho chương trình ngắn gọn hơn, sáng sủa hơn và cho phép tiết kiệm công sức của lập trình viên.

9.3. MỘT SỐ HÀM CHUẨN CỦA VB

Visual Basic có rất nhiều hàm chuẩn và thủ tục chuẩn. Sau đây là một số các hàm chuẩn thường dùng của Visual Basic.

9.3.1. Các hàm số học

Hàm	Ý nghĩa
ABS(x)	Cho giá trị tuyệt đối của x.
ATN(x)	Cho Arctang của x.
COS (x)	Cho cosin của góc x (x tính bằng radian)
EXP(x)	Nâng e lên lũy thừa x.
INT(x)	Cho số nguyên lớn nhất nhỏ hơn hoặc bằng x.
LOG (x)	Cho logarit tự nhiên của x.
RND()	Cho một số ngẫu nhiên t từ 0 đến nhỏ hơn 1 ($0 \leq t < 1$).
SIN(x)	Cho sin của góc x, x tính bằng radian.
SQR(x)	Cho căn bậc 2 của x.
TAN(x)	Cho tang của góc x, tính bằng radian.

9.3.2. Các hàm xử lý dãy ký tự

Hàm	Ý nghĩa
ASC(x\$)	Cho mã ASCII của ký tự đầu tiên trong x\$.
CHR\$(n)	Cho ký tự có mã ASCII là n, với n là một biểu thức số.
LCASE(x\$)	Làm cho x\$ chứa toàn chữ thường.
LEN(x\$)	Cho độ dài của x\$.
LEFT\$(x\$,n)	Cho n ký tự tận cùng bên trái của x\$.
MID\$(x\$, p, n)	Cho n ký tự kể từ ký tự thứ p của x\$. Các ký tự của x\$ được đánh số từ 1.
RIGHT\$(x\$,n)	Cho n ký tự tận cùng bên phải của x\$.
STRING\$(n, m)	Cho một dãy gồm n ký tự có mã ASCII là m.
STRING\$(n,x\$)	Cho một dãy lặp lại n lần ký tự đầu tiên của x\$.
UCASE(x\$)	Làm cho x\$ chứa toàn chữ hoa
VAL(x\$)	Cho giá trị số của x\$.

Chú ý: các hàm xử lý chuỗi trong Visual Basic luôn luôn coi ký tự đầu tiên của chuỗi là ký tự ở vị trí 1 (tức là ký tự thứ 1). Đây là điểm khác của Visual Basic so với các ngôn ngữ khác, ví dụ C++. Ngôn ngữ C++ luôn luôn coi ký tự đầu tiên của chuỗi là ký tự ở vị trí 0 (tức là ký tự thứ 0).

9.3.3. Một số hằng ký tự thường dùng

Kí tự	Hằng ký tự	Hàm tương ứng
Null Character	VbNullChar	Chr\$(0)
Carriage return	VbCr	Chr\$(13)
Line feed	VbLf	Chr\$(10)
Backspace	VbBack	Chr\$(8)
Tab	VbTab	Chr\$(9)
Vertical tab	VbVerticaltab	Chr\$(11)
Form feed	VbFormfeed	Chr\$(12)

9.3.4. Các hàm thời gian

Hàm	Ý nghĩa
Date()	Cho ngày, tháng, năm hiện hành
DateAdd(d)	Q Cách : DateAdd("Đơn vị TG", n, bt_Ngày) Cộng thêm n đơn vị thời gian vào bt_Ngày.
DateDiff()	Q Cách : DateDiff("Đơn vị TG", bt_Ngày1, bt_Ngày2) Cho số đơn vị TG giữa bt_Ngày1 và bt_Ngày2. Đơn vị TG được cho như sau: "h" là giờ, "d" là ngày, "m" là tháng, "n" là phút, "q" là 15 phút, "s" là giây, "w" là tuần "yyyy" là năm, .. Ví dụ: Print DateDiff("d", Now(), "1/1/01") Print DateDiff("w", Now(), "1/1/01")
Day(d)	Cho số chỉ ngày trong d (d là dữ liệu kiểu thời gian.) Ví dụ: hàm day("07/01/2002") cho số 1
Hour(t)	Cho số chỉ giờ trong t (t là dữ liệu chỉ giờ, phút, giây). Ví dụ hàm Hour(time()) cho số chỉ giờ của thời gian hiện thời. Còn hàm Hour("5:48:14 AM") cho số 5.
Minute(t)	Cho số chỉ phút trong t (t là dữ liệu chỉ giờ, phút, giây). Ví dụ hàm minute(time()) cho số chỉ phút của thời gian hiện thời. Còn hàm minute("5:48:14 AM") cho số 48.
Month(d)	Cho số chỉ tháng trong d (là dữ liệu kiểu thời gian.) Ví dụ: hàm month("07/01/2002") cho số 7
Now()	Cho giờ, phút, giây, ngày, tháng, năm hiện hành
Second(t)	Cho số giây trong t (t là dữ liệu chỉ giờ, phút, giây). Ví dụ hàm Second(time()) cho số chỉ giây của thời gian hiện thời. Còn hàm Second("5:48:14 AM") cho số 14.
Time()	Cho giờ, phút, giây hiện hành.
Year(d)	Cho số chỉ năm trong d (d là dữ liệu chỉ thời gian) Ví dụ: hàm Year("07/01/2002") cho số 2002

9.3.5. Các hàm tài chính thông dụng

Sau đây là tên các hàm tài chính thường dùng nhất của Visual Basic. Tác dụng và ý nghĩa các hàm này, nếu đọc giả quan tâm, xin đọc trong các tài liệu tham khảo hoặc sử dụng chức năng **Help** của Visual Basic.

DVB(), FV(), IPMT(), IR(), MIRR(), NPER(), NPV(), PMT(), PV(), RATE(), SLN(), SYD()

9.4. HÀM VÀ THỦ TỤC TỰ ĐỊNH NGHĨA

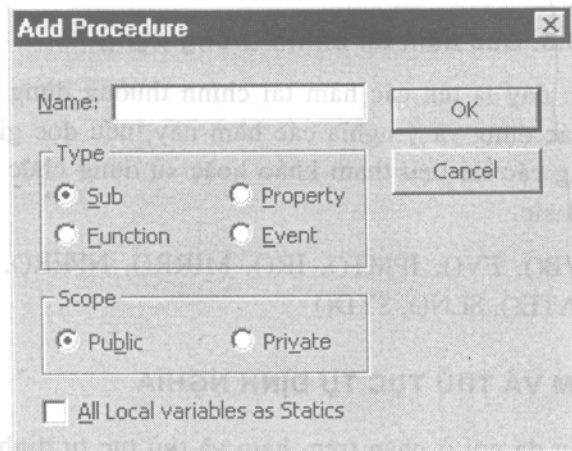
Như đã nói ở phần trên, hàm và thủ tục tự định nghĩa là những hàm và thủ tục không có sẵn trong ngôn ngữ lập trình. Đây là những hàm và thủ tục do lập trình viên tự tạo ra và sử dụng trong chương trình của mình. Phần này sẽ nghiên cứu cách định nghĩa và sử dụng các hàm và thủ tục tự định nghĩa.

Những hàm và thủ tục trình bày trong phần này là những hàm và thủ tục chỉ có ý nghĩa (tức là chỉ có thể được sử dụng) trong một biểu mẫu. Hàm (thủ tục) được định nghĩa trong biểu mẫu nào thì chỉ có ý nghĩa trong biểu mẫu đó.

9.4.1. Định nghĩa hàm và thủ tục

Để định nghĩa hàm hoặc thủ tục ta làm như sau:

- Chọn cửa sổ lệnh (Code)
- Chọn Tool
- Chọn Add Procedure. Hộp hội thoại Add Procedure xuất hiện (Xem hình 9.1)
- Xác định tên hàm hoặc thủ tục trong hộp Name
- Chọn Function hoặc Sub để xác định kiểu hàm hoặc thủ tục.



Hình 9.1. Hộp hội thoại Add Procedure

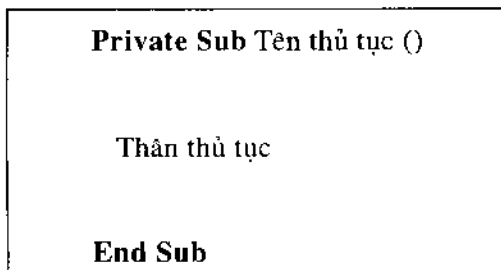
- Chọn Private để xác định rằng hàm hoặc thủ tục là cục bộ trong biểu mẫu.

- OK

Sau khi làm như vậy, một trong hai mẫu sau đây sẽ xuất hiện:

Private Function Tên hàm () Thân hàm End Function
--

Hình 9.2. Mẫu dùng để định nghĩa hàm



Hình 9.3. Mẫu dùng để định nghĩa tục

Dòng đầu tiên được gọi là đầu đề hàm (hoặc thủ tục). Đầu đề hàm bao gồm các từ khoá **Private Function**, tên hàm và danh sách tham số hình thức trong cặp dấu ngoặc đơn (nếu có). Đầu đề thủ tục hàm bao gồm các từ khoá **Private Sub**, tên thủ tục và danh sách tham số hình thức trong cặp dấu ngoặc đơn (nếu có).

Tên hàm và tên thủ tục được đặt theo quy tắc đặt tên biến.

Từ khoá **Private** chỉ ra rằng hàm hoặc thủ tục này là hàm hoặc thủ tục cục bộ, chỉ có nghĩa trong phạm vi biểu mẫu. Ngược lại, từ khoá **Public** chỉ ra rằng hàm hoặc thủ tục này là hàm hoặc thủ tục toàn cục, có nghĩa trong phạm vi toàn bộ dự án. Có thể sử dụng các hàm hoặc thủ tục toàn cục trong tất cả các biểu mẫu của dự án. Từ khoá **Global** cũng có nghĩa như từ khoá **Public**. Tuy nhiên, từ khoá **Global** được sử dụng trong các phiên bản trước của Visual Basic. Trong Visual Basic 6.0 từ khoá **Public** được dùng thay cho từ khoá **Global**.

Ta có thể chỉ ra kiểu của hàm (tức là kiểu của giá trị được trả lại thông qua tên hàm) giống như xác định kiểu của biến. Ví dụ các đầu đề hàm sau đây:

```
Private Function Tong% (a, b) ' (*)
Private Function Tong (a, b) as Integer
```

Hai đầu đề hàm này cùng xác định một hàm có tên là Tong, kiểu số nguyên, có 2 tham số hình thức là a và b.

Sau tên hàm (hoặc tên thủ tục) có thể có danh sách tham số hình thức được đặt trong cặp dấu ngoặc đơn. Hàm Tong% (a, b) có hai tham số hình thức là a và b. Hai tham số hình thức này có kiểu là kiểu tổng hợp. Nếu muốn ta có thể xác định kiểu của các tham số hình thức theo một trong hai cách sau đây:

```
Private Function Tong% (a%, b%)
```

```
Private Function Tong% (a as Integer, b as Integer)
```

Khi hàm Tong%() đã được định nghĩa như thế thì các tham số hình thức a và b có kiểu là kiểu số nguyên. Khi đó trong lời gọi hàm các tham số thực sự tương ứng (tức là các giá trị gửi cho hàm) phải có kiểu là kiểu số nguyên.

Thân hàm là một dãy các câu lệnh. Thân hàm sẽ được thực hiện khi gặp lời gọi hàm trong chương trình chính. Trong thân hàm ít nhất phải có một lệnh gán dùng để gán một giá trị cho tên hàm. Đây chính là giá trị được trả lại thông qua tên hàm. Giá trị này phụ thuộc vào giá trị các tham số hình thức.

Trong thân hàm có thể có những câu lệnh khai báo biến. Những biến được khai báo trong thân hàm là những biến cục bộ trong hàm này.

Dạng tổng quát của một hàm tự định nghĩa có dạng sau đây:

Private Function TênHàm (TSHT1, TSHT2...)

Khối lệnh

TênHàm = biểu thức

[Khối lệnh

TênHàm = biểu thức]

.....

End Function

TSHT1, TSHT2... là các tham số hình thức 1, tham số hình thức 2... Các tham số hình thức cách nhau bởi dấu phẩy(,).

Sau đây là ví dụ về một hàm tự định nghĩa.

```
Private Function Max! (a!, b!, c!)
```

```
    Dim KetQua!
```

```
    KetQua! = a!
```

```
    If KetQua! < b Then
```

```
        KetQua! = b!
```

```
    End If
```

```
    If KetQua! < c! Then
```

```
        KetQua! = c!
```

```
    End If
```

```
    Max! = KetQua!
```

```
End Function
```

Hàm này có tên là Max! . Hàm này dùng để tìm số lớn nhất trong 3 số thực a, b, c. Cách tìm số lớn nhất trong 3 số ta đã biết. Số lớn nhất này được chứa trong biến KetQua!. Câu lệnh gán:

```
Max! = KetQua!
```

Gán giá trị của biến KetQua! cho tên hàm là Max!

9.4.2. Sử dụng hàm và thủ tục

Để sử dụng hàm ta chỉ việc viết lời gọi hàm. Lời gọi hàm có dạng sau đây:

Tên hàm (TSTS1, TSTS2...)

TSTS1, TSTS2... là các tham số thực sự. Ta thấy rằng lời gọi hàm gồm tên hàm, sau đó là danh sách tham số thực sự (nếu có). Số lượng các tham số thực sự phải bằng số lượng các tham số hình thức. Kiểu của tham số thực sự phải trùng (phù hợp) với kiểu của tham số hình thức tương ứng. Lời gọi hàm được viết trong các biểu thức để tham gia vào quá trình tính giá trị biểu thức.

Ví dụ: Sau khi đã có hàm Max! như trên, trong chương trình chính ta có thể sử dụng hàm này như sau:

```
Private Sub cmdChayct_Click()  
    Dim x!, y!, z!  
    x = 2  
    y = 4  
    z = 1  
    Cls  
    Print "X, Y, Z = "; x; y; z; "Max = "; max(x, y, z)  
    x = 12  
    y = 6  
    z = 10  
    Print "X, Y, Z= "; x; y; z; "Max = "; max(x, y, z)  
End Sub
```


Khi thủ tục - biến cố `cmdChayct_Click()` được thực hiện trên màn hình có kết quả sau đây:

X, Y, Z = 2 4 1 Max = 4

Y, Y, Z = 12 6 10 Max = 12

Trong thủ tục - biến cố này lời gọi hàm có trong lệnh in. Nó có dạng :

`max(x, y, z)`

Lời gọi hàm này gồm có tên hàm (`max`) và các tham số thực sự (`x, y, z`), tham số nọ cách tham số kia một dấu phẩy (`,`). Các tham số thực sự đều có kiểu số thực, phù hợp với kiểu của các tham số hình thức tương ứng. Hàm này trả về giá trị lớn nhất của các biến (`x, y, z`). Lần thứ nhất hàm này trả về số 4. Lần thứ hai hàm này trả về số 12. Các giá trị này được in trên màn hình như ta đã thấy trong kết quả thực hiện chương trình.

Để thực hiện thủ tục ta phải viết lời gọi thủ tục. Lời gọi thủ tục có thể gồm câu lệnh CALL như sau:

`call tên_thủ_tục ([danh sách tham số thực sự])`

Hoặc không dùng câu lệnh CALL

`tên_thủ_tục [ danh sách tham số thực sự ]`

Nghĩa là nếu dùng câu lệnh CALL thì sau tên thủ tục phải có cặp dấu ngoặc đơn. Lời gọi thủ tục gồm tên thủ tục và danh sách các tham số thực sự (nếu cần). Danh sách tham số thực sự (nếu có) được đặt trong cặp dấu ngoặc tròn (nếu dùng lệnh CALL) và đi ngay sau tên thủ

tục. Số lượng các tham số thực sự phải bằng số lượng các tham số hình thức. Kiểu của tham số thực sự phải trùng (phù hợp) với kiểu của tham số hình thức tương ứng. Lời gọi thủ tục được viết trong chương trình chính như một câu lệnh độc lập.

Chú ý: Như đã thấy ở phần trên, lời gọi hàm được viết trong biểu thức. Còn lời gọi thủ tục được viết như một câu lệnh độc lập. Đây là một điểm khác nhau giữa thủ tục và hàm.

Sau đây là một ví dụ về cách viết và sử dụng thủ tục. Trong ví dụ này ta sẽ viết và sử dụng thủ tục có tên là **GiaiPTB2** (Giải Phương Trình Bậc 2). Thủ tục này dùng để giải một phương trình bậc hai.

Thủ tục giải phương trình bậc hai có dạng sau đây:

```
Private Sub GiaiPTB2(a!, b!, c!)
    Dim x1!, x2!, del!
    Print "a, b, c =", a, b, c
    If a = 0 Then
        Print " Không phải PT bậc 2. CT kết thúc!"
    Else
        del = b * b - 4 * a * c
        If del < 0 Then
            Print " PT Vô Nghiệm"
        Else
            x1 = (-b + Sqr(del)) / (2 * a)
            x2 = (-b - Sqr(del)) / (2 * a)
            Print "x1 = ", x1, "x2 = ", x2
        End If
    End If
End Sub
```

Đầu tiên thủ tục này in giá trị các hệ số a, b, c, sau đó giải phương trình bậc hai căn cứ vào các hệ số này. Nếu $a = 0$ máy sẽ in dòng thông báo:

" Không phải PT bậc 2. CT kết thúc!"

và kết thúc chương trình. Trong trường hợp $a \neq 0$ thì giá trị của biến Del (biệt số delta) sẽ được tính. Nếu $\text{del} < 0$ máy sẽ in dòng thông báo:

" PT Vô Nghiệm"

và kết thúc chương trình. Ngược lại, nếu $\text{del} \geq 0$ máy sẽ tính và in các nghiệm x1 và x2.

Để sử dụng thủ tục GiaiPTB2 ta phải viết thủ tục - biến cố sau đây:

```
Private Sub cmdChayct_Click()  
    Call GiaiPTB2(1, 2, 3)  
    Call GiaiPTB2(1, -2, 1)  
    Call GiaiPTB2(1, 2, -3)  
    Call GiaiPTB2(0, 2, -3)  
End Sub
```

Trong thủ tục - biến cố này có 4 lời gọi thủ tục GiaiPTB2 với các tham số thực sự khác nhau. Khi thủ tục - biến cố này được thực hiện trên màn hình biểu mẫu xuất hiện các kết quả sau đây:

a, b, c = 1 2 3

PT Vô nghiệm

a, b, c = 1 -2 1

x1 = 1 x2 = 1

a, b, c = 1 2 -3
x1 = 1 x2 = -3

a, b, c = 0 2 3

" Không phải PT bậc 2. CT kết thúc!"

Độc giả có thể tự mình kiểm tra lại độ chính xác của các kết quả này.

9.4.3. Phạm vi hoạt động của biến

Trong mục này ta sẽ dùng một khái niệm mới là đơn thể chương trình. Ta hãy xem đơn thể chương trình là gì?.

Đơn thể chương trình (modul) là tập hợp các thủ tục - biến cố, các thủ tục và các hàm liên quan đến một biểu mẫu hoặc không liên quan đến một biểu mẫu nào cả.

Tập hợp các thủ tục - biến cố, các thủ tục và các hàm liên quan đến một biểu mẫu được gọi là đơn thể chương trình biểu mẫu (có thể gọi là modul biểu mẫu). Tập hợp các thủ tục - biến cố, các thủ tục và các hàm không liên quan đến một biểu mẫu nào cả được gọi là đơn thể chương độc lập (modul độc lập).

Phạm vi của biến (còn gọi là miền hoạt động của biến) là phần chương trình mà ở đó có thể truy nhập đến biến (hoặc sử dụng biến).

Biến toàn cục: Là những biến được khai báo bằng từ khoá **public (global)** trong phần khai báo (General) của một đơn thể chương trình. Trong tất cả các thủ tục của ứng dụng có thể truy nhập các biến toàn cục. Có nghĩa là miền hoạt động của biến toàn cục toàn bộ các đơn thể chương trình của ứng dụng. Xin nhắc lại, từ khoá **global** được dùng trong các phiên bản cũ. Trong Visual Basic 6.0 từ

khoá này được thay bằng từ khoá **public**.

Biến module là những biến được khai báo bằng từ khoá **Dim** trong phần khai báo (General) của một đơn thể chương trình. Trong tất cả các thủ tục của đơn thể chương trình này có thể truy nhập các biến module. Có nghĩa là miền hoạt động của biến modul là toàn bộ đơn thể chương trình trong đó biến này được khai báo. Ta sẽ gọi các biến modul là biến chung cho dễ diễn đạt.

Biến cục bộ là những biến được khai báo bằng từ khoá **Dim** trong một thủ tục, một hàm hoặc một thủ tục - biến cố. Chỉ có thể truy nhập các biến cục bộ trong bản thân thủ tục hoặc hàm hoặc thủ tục - biến cố mà biến này được khai báo. Có nghĩa là miền hoạt động của biến cục bộ là thủ tục hoặc hàm hoặc thủ tục - biến cố mà trong đó nó được khai báo.

Chú ý:

- Các thủ tục - biến cố của một biểu mẫu có thể được thực hiện ở một biểu mẫu khác nếu thay từ khoá **Private** bằng từ **Public**;
- Các biến module (biến chung) có thể được sử dụng trong các đơn thể chương trình khác nếu được khai báo bằng từ khoá **public** thay cho từ khoá **Dim**.

9.4.4. Truyền giá trị cho tham số của chương trình con

Mục này sẽ nghiên cứu các phương pháp truyền giá trị cho các tham số hình thức của chương trình con từ chương trình chính.

Có hai cách truyền giá trị cho tham số hình thức: truyền giá trị và truyền địa chỉ. Tương ứng với với hai cách truyền giá trị đó là hai loại tham số hình thức: tham trị và tham biến.

Truyền giá trị cho tham số hình thức:

Theo phương pháp này thì các tham số hình thức phải là các tham số hình thức giá trị, gọi tắt là tham trị. Trong danh sách tham số hình thức phải có 2 từ khoá **By Val** đứng trước các tham trị.

Thủ tục sau đây có tham số hình thức là tham trị. Tên thủ tục NhanDoiTT có nghĩa là “Nhân Đôi Tham Trị”.

```
Public Sub NhanDoiTT(ByVal x, ByVal y)
x = x * 2
y = y * 2
Print "x = ", x, "y = ", y, "In trong CTC"
End Sub
```

Thủ tục này thực hiện một công việc đơn giản làm cho giá trị các tham số hình thức tăng lên 2 lần.

Để thực hiện thủ tục do ta viết thủ tục - biến cố sau đây:

```
Private Sub CmdChayCT_Click()

Dim a, b
a = 2
b = 4
Print "CTC co tham tri "
Print "Truoc khi gọi CTC a = ", a, " b = ", b
Call NhanDoiTT(a, b)
Print "Sau khi gọi CTC a = ", a, " b = ", b
End Sub
```

Thủ tục - biến cố cmdChayct_click() có lời gọi thủ tục NhanDoiTT(), nên thủ tục - biến cố cmdChayct_click() là chương

trình chính. Gặp câu lệnh:

```
Call NhanDoiTT(a, b)
```

Máy tạm rời khỏi chương trình chính để chuyển đến chương trình con (tức là thủ tục `NhanDoiTT()`). Khi đó các tham số hình thức được gán giá trị các tham số thực sự tương ứng. Mọi tính toán trong chương trình con được thực hiện trên các tham số hình thức, là các biến cục bộ trong chương trình con. Những thay đổi của các tham số hình thức không ảnh hưởng đến các tham số thực sự tương ứng. Có nghĩa là những thay đổi của các tham số hình thức trong chương trình con không được truyền cho các tham số thực sự ở chương trình chính.

Khi thủ tục - biến cố `cmdChayct_click()` được thực hiện trên màn hình có kết quả sau đây:

```
CTC co tham tri
Truoc khi goi CTC   a = 2   b = 4
x = 4   y = 8   In trong CTC
Sau khi goi CTC a = 2   b = 4
```

Chúng ta thấy rằng giá trị các tham số thực sự `a` và `b` không bị thay đổi trước và sau khi thực hiện thủ tục `NhanDoiTT(a, b)` mặc dù giá trị các tham số hình thức đã được tăng gấp 2 lần. Điều đó chứng tỏ những thay đổi của các tham số hình thức trong chương trình con không ảnh hưởng đến các tham số thực sự tương ứng ở chương trình chính.

Truyền địa chỉ cho tham số hình thức:

Theo phương pháp này thì các tham số hình thức phải là các tham số hình thức biến, gọi tắt là tham biến. Trong danh sách tham số hình thức không có 2 từ khoá **By** **Val** đứng trước các tham biến.

Thủ tục sau đây có tham số hình thức là tham biến. Tên thủ tục là NhanDoiTB(). Tên này nghĩa là “Nhân Đôi Tham Biến”.

```
Public Sub NhanDoiTB(x!, y!)  
    x = x * 2  
    y = y * 2  
    Print "x, y = ", x, y, "In trong CTC "  
End Sub
```

Ta thấy rằng trước các tham số hình thức x! và y! không có các từ khoá **By Val**. Vậy x! và y! là các tham biến.

Giả sử có thủ tục - biến cố:

```
Private Sub CmdChayCT_Click()  
    Dim a, b  
    a = 2  
    b = 4  
    Print "CTC co tham bien "  
    Print "Truoc khi gọi CTC a = ", a, " b = ", b  
    Call NhanDoiTB(a, b)  
    Print "Sau khi gọi CTC a = ", a, " b = ", b  
End Sub
```

Khi thực hiện thủ tục - biến cố này, gặp câu lệnh:

Call NhanDoiTB(a, b) thì máy tạm rời khỏi chương trình chính để chuyển đến thực hiện chương trình con (tức là thủ tục NhanDoiTB()). Khi đó các tham số hình thức được gán địa chỉ các tham số thực sự tương ứng. Như vậy, trong trường hợp này, các tham số hình thức đại diện cho địa chỉ của các tham số thực sự. Mọi tính toán trong

chương trình con được thực hiện trên các giá trị được xác định bởi các địa chỉ này. Do đó mọi thay đổi của các tham số hình thức trong chương trình con cũng chính là những thay đổi của các tham số thực sự trong chương trình chính. Nói khác đi, những thay đổi của các tham số hình thức sẽ được truyền cho các tham số thực sự tương ứng ở chương trình chính.

Khi thủ tục - biến cố `cmdChayct_click()` được thực hiện trên màn hình có kết quả sau đây:

```
CTC co tham bien
Truoc khi gọi CTC a = 2  b = 4
x = 4   y = 8   In trong CTC
Sau khi gọi CTC a = 4  b = 8
```

Khi thủ tục này được thực hiện thì giá trị của 2 tham số hình thức được nhân lên 2 lần. Những thay đổi của các tham số hình thức này sẽ được truyền cho các tham số thực sự tương ứng ở chương trình chính là a và b. Khi thoát khỏi chương trình con trở về chương trình chính thì biến a có giá trị là 4, biến b có giá trị là 8. Ta thấy giá trị các tham số a và b đã bị đổi, tức là những thay đổi của các tham số hình thức đã được truyền cho các tham số thực sự tương ứng trong chương trình chính.

9.5. MODULE ĐỘC LẬP

Module độc lập là những đơn thể chương trình độc lập, không thuộc về một biểu mẫu nào cả. Tập chứa module độc lập có phần mở rộng là .BAS. Khi module độc lập đã được thêm vào dự án thì có thể truy nhập đến các thủ tục hoặc các hàm có trong module đó từ tất cả các thủ tục của dự án. Để thêm module độc lập vào dự án ta làm như sau:

- Chọn Project
- Chọn Add Module
- Chọn New hoặc Existing
- Chọn OK

Việc tạo ra, sử dụng và loại bỏ các module độc lập không nằm trong phạm vi của cuốn sách này.

9.6. MỘT SỐ VÍ DỤ VỀ HÀM TỰ ĐỊNH NGHĨA

Ví dụ thứ nhất: Hàm tính $n!$ (n giai thừa). Xin nhắc lại rằng $n!$ được định nghĩa là tích n số tự nhiên đầu tiên. Tức là:

$$n! = 1 * 2 * \dots * n$$

Hàm tính $n!$ có dạng sau đây:

```
Private Function GiaiThua(k As Integer) As Long
    Dim kq As Long
    Dim i%
    kq = 1
    For i = 1 To k
        kq = kq * i
    Next
    GiaiThua = kq
End Function
```

Trong hàm này biến cục bộ **kq** dùng để chứa kết quả tính toán ($n!$). Sau khi tính xong kết quả, tên hàm được gán giá trị của **kq** nhờ câu lệnh gán:

GiaiThua = kq

Nhờ vậy mà $n!$ (là giá trị của biến **kq**) được trả lại thông qua tên hàm.

Thủ tục - biến cố sử dụng hàm này có dạng:

```
Private Sub CmdChayct_Click()  
    Dim n%  
    n = 3  
    Print n; "! = "; GiaiThua(n)  
    n = 5  
    Print n; "! = "; GiaiThua(n)  
End Sub
```

Khi thực hiện thủ tục - biến cố này trên màn hình xuất hiện kết quả sau đây:

$$3! = 6$$

$$5! = 120$$

Ví dụ thứ hai: hàm xác định xem số nguyên k có phải là số nguyên tố hay không. Hàm này có tên là SoNT (Số Nguyên Tố). Chúng ta biết rằng số nguyên tố là những số chỉ chia hết cho 1 và chính nó. Hàm xác định xem số nguyên k có phải là số nguyên tố hay không có dạng sau đây:

```
Private Function SoNT(k As Integer) As Boolean  
    Dim i%, dem%  
    dem = 0  
    SoNT = True  
    For i = 2 To Int(k / 2)  
        If k Mod i = 0 Then  
            dem = dem + 1
```

```

    End If
Next i
If dem > 0 Then
    SoNT = False
End If
End Function

```

Hàm này có kiểu là kiểu logic (Boolean). Vì vậy giá trị trả lại thông qua tên hàm này hoặc là True hoặc là False.

Thủ tục - biến cố sử dụng hàm này có dạng:

```

Private Sub cmdChayct_Click()
    Dim n%
    Cls
    n = Val(InputBox(" n = ", "Nhập số nguyên n ", "5"))
    'n = 5
    If SoNT(n) Then
        Print n, "là SNT"
    Else
        Print n, "không là SNT"
    End If
End Sub

```

Khi thực hiện thủ tục - biến cố cmdChayct_click(), máy tính sẽ yêu cầu ta nhập một số nguyên. Nếu ta nhập số 5, máy sẽ in dòng kết quả:

5 là SNT

Còn nếu ta nhập số 8, máy sẽ in dòng kết quả:

8 không là SNT

Độc giả có thể dễ dàng kiểm tra kết quả này.

Ví dụ thứ ba: hàm xác định số tiền tương lai. Hàm này có tên là TTLai () (Tiền Tương Lai). Hàm này dùng để tính số tiền mà khách hàng sẽ nhận được trong tương lai khi biết số tiền gửi, số năm gửi tiền và lãi suất tính theo năm. Hàm này có dạng sau đây:

```
Private Function TTLai(TGui#, SoNam%, LaiSuat!) As Double
    Dim kq#
    kq = TGui * (1 + LaiSuat) ^ SoNam
    TTLai = kq
End Function
```

Thủ tục - biến cố sử dụng hàm này có dạng:

```
Private Sub CmdChayct_Click()
    Dim STGui#, SoNam%, LSuat!
    STGui = 100
    SoNam% = 5
    LSuat = 0.012
    Print "So tien gui = ", STGui
    Print "So nam = ", SoNam
    Print "Lai suat = ", LSuat
    Print "So tien tuong lai = ", TTLai(STGui, SoNam, LSuat)
End Sub
```

Khi thực hiện thủ tục - biến cố này trên màn hình xuất hiện kết

quả sau đây:

So tien gui = 100

So nam = 5

Lai suat = 0.012

So tien tuong lai = 106.1457384

Độc giả hãy tự kiểm tra kết quả tính toán này.

CÂU HỎI ÔN TẬP VÀ BÀI TẬP

Lý thuyết

1. Hãy trình bày các khái niệm chương trình, chương trình con, hàm, thủ tục, hàm chuẩn, hàm tự định nghĩa, thủ tục chuẩn, thủ tục tự định nghĩa.

2. Hãy nêu sự khác nhau giữa hàm và thủ tục.

3. Hãy nêu qui cách viết đầu đề hàm.

4. Hãy nêu qui cách viết danh sách tham số hình thức.

5. Hãy trình bày qui cách viết lời gọi hàm và lời gọi thủ tục.

6. Hãy trình bày quá trình thực hiện hàm khi gặp lời gọi hàm.

7. Hãy trình bày quy tắc truyền giá trị cho hàm.

8. Hãy trình bày khái niệm miền hoạt động của biến.

9. Hãy trình bày sự khác nhau giữa tham biến và tham trị. Nêu ý nghĩa của sự khác nhau giữa tham biến và tham trị.

Bài tập

1. Giả sử đã cho biết các số thực ai, bi, ci ($i = 1, 2, \dots, n$). Hãy viết một hàm tìm giá trị trung bình của 3 số đã cho, rồi sử dụng hàm đó để

viết chương trình cho máy tính và in giá trị trung bình của mỗi nhóm ba số a_i, b_i, c_i ($i = 1, 2, \dots, n$).

2. Giả sử đã cho biết các số thực a_i, b_i, c_i ($i = 1, 2, \dots, n$). Hãy viết một hàm tìm giá trị lớn nhất của 3 số đã cho, rồi sử dụng hàm đó để viết chương trình cho máy tính và in giá trị lớn nhất của mỗi nhóm ba số a_i, b_i, c_i ($i = 1, 2, \dots, n$).

3. Cho một số thực $a > 1$. Gọi $S(n)$ là tổng: $1 + 1/2 + 1/3 + \dots + 1/n$. Hãy lập chương trình (có chứa hàm tính $S(n)$) cho máy tìm số tự nhiên m lớn nhất sao cho $S(m) < a$.

4. Cho một số tự nhiên n . Hãy viết một hàm xác định xem số nguyên k có là số nguyên tố hay không. Sau đó viết chương trình sử dụng hàm đó để đếm xem có bao nhiêu số nguyên tố nhỏ hơn số n đã cho.

5. Hãy viết một hàm tính căn bậc 2 của số thực x theo phương pháp lặp Newton như sau:

$$Y_n = (Y_{n-1} + x/Y_{n-1})/2;$$

Với $Y_0 = x/2$ và độ chính xác bằng 0.000001. Hãy so sánh hàm này với hàm chuẩn $\text{sqrt}(x)$.

XUẤT DỮ LIỆU TRONG VISUAL BASIC

Trong các chương trước ta đã biết hai cách xuất dữ liệu là in dữ liệu trong hộp văn bản và in dữ liệu trên màn hình biểu mẫu. Trong chương này ta nghiên cứu đầy đủ hơn vấn đề xuất dữ liệu của Visual Basic.

10.1. IN DỮ LIỆU TRÊN BIỂU MẪU

In dữ liệu trên màn hình biểu mẫu (gọi tắt là in trên biểu mẫu) là một trong những cách xuất dữ liệu cơ bản của Visual Basic. Ta đã trình bày ngắn gọn về lệnh PRINT trong các mục 5.6.2 của chương 5. Câu lệnh này đã được sử dụng nhiều lần để viết các chương trình trong các chương trước. Trong mục này ta sẽ nghiên cứu đầy đủ và chi tiết hơn về câu lệnh PRINT. Đặc biệt là cách sử dụng lệnh PRINT kết hợp với hàm FORMAT.

10.1.1. Xác định vị trí in bằng CurrentX và CurrentY

CurrentX và CurrentY xác định vị trí bắt đầu in dữ liệu trong biểu mẫu. CurrentX xác định toạ độ ngang (X), CurrentY xác định toạ độ dọc (Y). Khi vừa bắt đầu làm việc hoặc sau khi thực hiện câu lệnh xoá

CLS thì CurrentX và CurrentY được gán giá trị 0.

Đơn vị đo được xác định khi thiết kế biểu mẫu. Đơn vị đo có thể là:

0 - user; 1 - Twip; 2 - Point; 3 - Pixel; 4 - Character; 5 - Inch; 6 - Millimeter; 7 - Centimeter.

Ví dụ có các câu lệnh: (frmathu là tên biểu mẫu)

```
CurrentX = frmathu.Width / 2  
CurrentY = frmathu.Height / 2  
Print "Hello!"
```

Thì dãy ký tự "Hello!" được in ở giữa biểu mẫu.

10.1.2. Xác định Font và các thuộc tính của Font trong quá trình in

Khi tạo ra các đối tượng ta đã dùng cửa sổ thuộc tính để xác định các thuộc tính của Font cho từng đối tượng. Tuy nhiên, trong thời gian thực hiện chương trình, nếu cần ta cũng có thể xác định các thuộc tính của Font cho từng đối tượng. Muốn vậy ta dùng các câu lệnh sau đây:

```
Tên_Đối_Tượng.Font.Name="Arial"  
Tên_Đối_Tượng.Font.Name="Courier New"  
Tên_Đối_Tượng.Font.Size= 18  
Tên_Đối_Tượng.Font.Bold= True  
Tên_Đối_Tượng.Font.Italic= True  
Tên_Đối_Tượng.Font.Underline= True
```

Ví dụ: để xác định các thuộc tính của Font cho biểu mẫu của chương trình Hello (có tên là frmHello như ta đã biết), ta phải viết

các câu lệnh sau đây:

```
FrmHello.Font.Name="Courier New"
```

```
FrmHello.Font.Size= 18
```

```
frmHello.Font.Bold= True
```

Nếu dự án chỉ có một biểu mẫu thì trong các lệnh này có thể bỏ qua tên biểu mẫu.

10.1.3. Lệnh in dữ liệu trên biểu mẫu: PRINT

Lệnh PRINT là lệnh được dùng phổ biến nhất để xuất dữ liệu. Lệnh này cho phép xuất ra màn hình hoặc máy in hầu hết các kiểu dữ liệu của Visual Basic. Khi kết hợp với hàm Format() và một số hàm khác lệnh PRINT có khả năng to lớn trong việc xuất dữ liệu theo các dạng khác nhau.

Lệnh PRINT có qui cách như sau:

PRINT [danh sách biểu thức] [;]

"Danh sách biểu thức" gồm các biểu thức, biểu thức nọ cách biểu thức kia bởi dấu phẩy, dấu cách hoặc dấu chấm phẩy.

Tác dụng:

Nếu không có danh sách biểu thức thì một dòng trống sẽ xuất hiện trên màn hình. Nếu có danh sách biểu thức, thì giá trị của các biểu thức được xác định và được in trên màn hình theo thứ tự viết chúng, kể từ đầu dòng của màn hình. Nếu biểu thức nọ cách biểu thức kia bằng dấu phẩy (,), thì giá trị của các biểu thức được in ở đầu mỗi vùng in (mỗi vùng in gồm 14 vị trí). Nếu các biểu thức cách nhau bởi dấu chấm phẩy (;), thì các giá trị được in cạnh nhau.

Ví dụ: Các câu lệnh:

```
Dim Ten$, i, j
```

```
Ten$ = "Le Van Hung"
```

```
i = 2
```

```
j = 15
```

```
Print "Ket Qua 1 (Phay)= ", Ten$, i, j '(1)
```

```
Print "Ket Qua 2(Cham Phay)= "; Ten$; i; j '(2)
```

Sẽ in trên màn hình kết quả sau đây:

Le Van Hung 2 5 'Do lệnh (1) in

Le Van Hung 2 15 ' Do lệnh (2) in

Chú ý:

a) Việc in dữ liệu trên máy in sẽ được nghiên cứu trong một mục riêng ở cuối chương này.

b) Nếu danh sách biểu thức của lệnh in kết thúc bằng dấu chấm phẩy (;), thì lệnh in tiếp theo sẽ in các giá trị từ vị trí in hiện thời trên cùng một dòng.

10.1.4. In có qui cách

In có qui cách là quá trình in có kèm theo sự biến đổi hoặc hiệu chỉnh cần thiết đối với dữ liệu để kết quả in có dạng xác định, thoả mãn được yêu cầu của người sử dụng kết quả.

Để có thể in có qui cách ta dùng các hàm sau đây:

Hàm Tab(n%): Đưa điểm in đến cột n của dòng hiện hành, là dòng đang in (sử dụng kích thước trung bình của kí tự theo font hiện hành). Nếu toạ độ cột hiện hành lớn hơn n thì Tab(n%) sẽ chuyển đến cột đó ở

dòng sau. Nếu $n\% < 1$ thì máy sẽ chuyển đến cột đầu tiên (cột 0).

Hàm Spc(n%): Hàm này chèn $n\%$ dấu cách vào vị trí hiện hành của dòng. Mỗi dấu cách có kích thước trung bình của ký tự theo font hiện hành. Giá trị của $n\%$ phải không âm.

Hàm Format()

Hàm này dùng để định dạng dữ liệu, tức là điều khiển việc in các giá trị dữ liệu theo một dạng xác định. Hàm này có qui cách như sau:

FORMAT(biểu_thức, khuôn_mẫu) [;]

Ở đây:

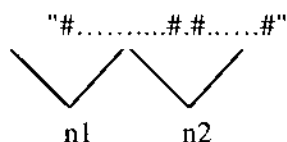
- Biểu thức là một biểu thức mà giá trị của nó sẽ được in.
- Khuôn mẫu là một hằng ký tự (hoặc một biểu thức ký tự). Chính khuôn mẫu sẽ qui định hình thức của các giá trị sẽ được in.

Visual Basic có nhiều loại khuôn mẫu khác nhau. Trong bài này ta nghiên cứu một số khuôn mẫu thường được sử dụng.

Khuôn mẫu dùng cho dữ liệu số

Có nhiều khuôn mẫu dùng cho dữ liệu số. Sau đây là 2 khuôn mẫu do lập trình viên tạo ra hay được sử dụng nhất.

Khuôn mẫu 1 có dạng:



Tác dụng: theo khuôn mẫu này các giá trị số được in ra gồm $n1+n2+1$ vị trí, trong đó $n1$ vị trí chứa phần nguyên, $n2$ vị trí chứa phần lẻ, giữa phần nguyên với phần lẻ có dấu chấm ngăn cách. Các số

không vô nghĩa (bên trái và bên phải) không được in. Nếu số cần in có nhiều hơn n1 chữ số ở phần nguyên thì tất cả các chữ số đó sẽ được in.

Ví dụ: lệnh

```
PRINT FORMAT(12.66, "##.##")
```

Cho kết quả: 12.66

Các chữ số 0 vô nghĩa bên trái sẽ được thay bằng dấu cách. Trước số âm có dấu trừ (-). Các số 0 vô nghĩa phía phải bị bỏ qua, tức là không được in.

Ví dụ: lệnh

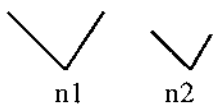
```
PRINT FORMAT(1.20, "##.##")
```

Cho kết quả 1.2

Chú ý: trong qui cách này, n1 hoặc n2 có thể bằng 0; có nghĩa là phần nguyên hoặc phần lẻ có thể sẽ không tồn tại.

Khuôn mẫu 2 có dạng:

"0.....0.0.....0"



n1 n2

Tác dụng: khuôn mẫu này tương tự qui cách 1, chỉ khác là các số 0 vô nghĩa phía trái và phía phải cũng được in.

Ví dụ : có các câu lệnh:

```
Dim x!
```

```
x! = 025.000
```

```
Print Format(x!, "####.##")
```

```
Print Format(x!, "0000.00")
```

```
Print Format(x!, "####.000")
```

Cho kết quả là:

25

0025.00

25.00

Các khuôn mẫu có sẵn:

Một số khuôn mẫu thường dùng đã được Visual Basic và đặt tên. Trong hàm Format ta có thể dùng tên khuôn mẫu thay cho việc viết bản thân khuôn mẫu đó. Ví dụ ta có thể dùng lệnh:

```
Print Format (SoTien, "Currency")
```

Thay cho câu lệnh:

```
Print Format (SoTien, "###,###.##")
```

Sau đây là bảng các khuôn mẫu có sẵn của Visual Basic và tác dụng của từng khuôn mẫu đó.

Bảng 10.1. Các khuôn mẫu có sẵn dành cho dữ liệu số

Tên khuôn mẫu	Tác dụng
General number	Cho một số không có dấu tách hàng nghìn
Currency	Cho một số có dấu tách hàng nghìn và có 2 chữ số phần lẻ
Fixed	Cho một số có ít nhất một chữ số phần nguyên và hai chữ số phần lẻ.
Standard	Cho một số có dấu tách hàng nghìn và có ít nhất một chữ số phần nguyên và hai chữ số phần lẻ.
Percent	In số dưới dạng phần trăm, tức là số được nhân với 100, bên phải có dấu %. Số luôn luôn có 2 chữ số phần lẻ.

Tên khuôn mẫu	Tác dụng
On/Off	In Off nếu là số 0, ngược lại thì in On
True/False	In False nếu là số 0, ngược lại thì in True
Yes/No	In No nếu là số 0, ngược lại thì in Yes

Bảng 10.2. Các khuôn mẫu có sẵn dành cho dữ liệu thời gian

Tên khuôn mẫu	Tác dụng
General date	Cho ngày và/hoặc giờ. Nếu không có phần lẻ sẽ cho ngày tháng. Nếu không có phần nguyên sẽ cho giờ. Nếu có cả phần nguyên lẫn phần lẻ sẽ được cả hai. Ví dụ <code>Format(now(),"General date")</code> cho 2/2/04 8:02:31 AM
Long Date	Cho thứ, ngày, tháng, năm đầy đủ (dài). Ví dụ <code>Format(now(),"Long date")</code> cho Monday, February 02, 2004
Medium Date	Cho ngày, tháng, năm dạng trung bình. Ví dụ <code>Format(now(),"Medium date")</code> cho 02-Feb-04
Short Date	Cho ngày, tháng, năm dạng ngắn. <code>Format(now(),"short date")</code> 2/2/04
Long Time	Cho thời gian dạng dài. Ví dụ <code>Format(now(),"Long time")</code> cho 8:08:08 AM
Medium Time	Cho thời gian dạng trung bình. Ví dụ <code>Format(now(),"Medium time")</code> cho 08:08 AM
Short Time	Cho thời gian dạng ngắn. Ví dụ <code>Format(now(),"short time")</code> cho 08:08

10.1.5. Ví dụ về chương trình in bảng

Giả sử một cửa hàng kinh doanh n hàng hoá. Cho biết tên, số lượng hàng hoá đã bán và đơn giá bán của từng loại hàng hoá. Hãy viết chương trình tính và in bảng có dạng sau đây:

BẢNG KÊ TIỀN BÁN HÀNG

Số TT	Tên hàng	Số lượng	Đơn giá	Số tiền
	Tổng cộng			

Chương trình này gọi là chương trình InBảng. Màn hình giao diện của chương trình này có một biểu mẫu và 4 nút lệnh. Bốn nút lệnh này lần lượt có tiêu đề là Nhap DL, Tính Va In, Xoa và Ket Thuc. Ta đã gọi gọi màn hình giao diện này là màn hình giao diện đơn giản và đã tạo ra nó nhiều lần trong các chương trước.

Sau đây là các thủ tục - biến cố của các nút lệnh của chương trình.

```
Option Explicit
```

```
Dim ten$(100), sl!(100), dg!(100)
```

```
Dim i%, n%
```

```
Private Sub cmdketThuc_Click()
```

```
End
```

```
End Sub
```

```
Private Sub CmdNhapDL_Click()
```

```
n = 2
```

```
ten$(1) = "Xe máy"
```



```

ten$(2) = "Tu lanh Sony"
sl!(1) = 12
sl!(2) = 234
dg!(1) = 222.21
dg!(2) = 135.53
End Sub

```

```

Private Sub cmdTinhVaIn_Click()
Dim vach$, tst!, st!, stt%
Dim c1, c2, c3, c4, c5, c6
c1 = 1
c2 = c1 + 6
c3 = c2 + 25
c4 = c3 + 14
c5 = c4 + 14
c6 = c5 + 14
vach$ = "-----"
"
Cls
Print Tab(c3); "Bang ke tien ban hang"
Print vach$; vach$
Print Tab(c1); "|STT";
Print Tab(c2); "| Ten hang";
Print Tab(c3); "| So Luong";
Print Tab(c4); "| Don Gia";
Print Tab(c5); "| So Tien";
Print Tab(c6); "| "

```

```

Print vach$; vach$
stt% = 1
For i = 1 To n
    st! = sl!(i) * dg!(i)
    Print Tab(c1); "| "; Format(stt%, "###");
    Print Tab(c2); "| "; ten$(i);
    Print Tab(c3); "| "; Spc(10 - Len(Str(Format(sl!(i),
"#####.00"))));
    Print Str(Format(sl!(i), "00000.00"));
    Print Tab(c4); "| "; Spc(10 - Len(Format(dg!(i),
"#####.00"))));
    Print Format(dg!(i), "#####.00");
    Print Tab(c5); "| "; Spc(10 - Len(Format(st!,
"#####.00"))));
    Print Format(st!, "#####.00");
    Print Tab(c6); "|"
    stt% = stt% + 1
    tst! = tst! + st!
Next
Print vach$; vach$
Print Tab(c1); "|";
Print Tab(c2); "| Tong Cong";
Print Tab(c3); "| ";
Print Tab(c4); "| ";
Print Tab(c5); "| "; Spc(10 - Len(Format(tst!,
"#####.00"))));

```

```

Print Format(tst!, "#####.00");
Print Tab(c6); "|"
Print vach$; vach$
End Sub

Private Sub CmdXoa_Click()

Cls

End Sub

```

Thủ tục - biến cố CmdNhapDL_Click() dùng để gán các giá trị ban đầu cho các biến. Thủ tục - biến cố cmdTinhVaIn_Click() dùng để tạo ra và in bảng kết quả. Khi thủ tục - biến cố này được thực hiện, sau khi sử dụng nút lệnh CmdNhapDL_Click() và cmdTinhVaIn_Click() thì màn hình của biểu mẫu có dạng sau đây:

STT	Ten hang	So Luong	Don Gia	So Tien
1	Xe may	12	222.21	2666.52
2	Tu lanh Sony	234	135.53	31714.02
	Tong Cong			34380.54

Hình 10.1. Màn hình kết quả của chương trình InBang

10.2. IN DỮ LIỆU TRÊN MÁY IN

Để in dữ liệu trên máy in ta phải sử dụng máy in. Trong Visual Basic máy in cũng được coi là một đối tượng. Từ khoá PRINTER là tên của đối tượng máy in. Không có công cụ điều khiển PRINTER trong hộp công cụ. Tất cả các công việc liên quan đến đối tượng PRINTER đều phải được thực hiện thông qua các câu lệnh của Visual Basic.

Mục này trình bày cách in các số và các chuỗi (dãy kí tự) trên máy in. Việc in các hình vẽ và hình ảnh trên máy in không nằm trong phạm vi của cuốn sách.

10.2.1. Cách Visual Basic gửi dữ liệu ra máy in

Windows có một chương trình quản lý toàn bộ các công việc in dữ liệu trên máy in. Ta sẽ gọi chương trình đó là bộ quản lý in. Khi chương trình ứng dụng có lệnh gửi thông tin ra máy in, thì bộ quản lý in sẽ chặn những dữ liệu đó lại. Hiếm khi có thể gửi trực tiếp dữ liệu ra máy in. Bộ quản lý in sẽ xác định cách trình bày dữ liệu của tất cả các chương trình Windows. Khi thấy có dữ liệu được gửi ra máy in, bộ quản lý in sẽ chặn các dữ liệu đó lại và thực hiện một số công việc biến đổi nếu cần trước khi gửi các dữ liệu này ra máy in. Bộ quản lý in của Windows có thể nhận biết được nhiều loại máy in khác nhau và sẽ biến đổi dữ liệu cho phù hợp với loại máy in được sử dụng.

10.2.2. Các câu lệnh liên quan đến máy in

Có nhiều câu lệnh có thể dùng với máy in. Sau đây ta nghiên cứu một số câu lệnh hay được sử dụng nhất.

• Câu lệnh PRINT

Câu lệnh này có qui cách sau đây:

PRINTER.PRINT [danh sách biểu thức] [;]

Câu lệnh này hoàn toàn tương tự câu lệnh PRINT đã trình bày trong mục 10.1.3. Chỉ có điều khác là theo câu lệnh này máy sẽ in giá trị các biểu thức có trong danh sách biểu thức trên máy in.

• Câu lệnh **Printer.EndDoc**

Quá trình in trên máy in chỉ thực sự bắt đầu khi bộ quản lý in đã nhận được tất cả các dữ liệu được gửi ra máy in hoặc khi gặp câu lệnh **Printer.EndDoc**. Vì vậy câu lệnh **Printer.EndDoc** được gọi là câu lệnh khởi động in.

Khi câu lệnh **Printer.Print** gửi dữ liệu ra máy in, bộ quản lý in sẽ thiết kế một hoặc nhiều trang dữ liệu, nhưng không gửi các dữ liệu đó ra máy in cho đến khi gặp câu lệnh **Printer.EndDoc**. Nếu không có câu lệnh **Printer.EndDoc** bộ quản lý in sẽ tập hợp tất cả các dữ liệu mà chương trình ứng dụng gửi ra máy in nhưng không in bất kỳ dữ liệu nào cho đến khi kết thúc chương trình ứng dụng.

Nếu bạn viết một chương trình in hoá đơn bán hàng, mà chương trình này được chạy suốt ngày, thì phải có câu lệnh **Printer.EndDoc** ở cuối thủ tục in hoá đơn để máy in từng hoá đơn mỗi khi thủ tục in hoá đơn được thực hiện.

• Câu lệnh **Printer.NewPage**

Thông thường bộ quản lý in của Windows đảm bảo rằng dữ liệu sẽ được in ở đầu trang mới khi kết thúc trang vật lý (ví dụ gồm 66 dòng). Vì vậy nếu dữ liệu cần in gồm 67 dòng, thì dòng thứ 67 sẽ được in ở đầu trang thứ hai.

Tuy nhiên, có những trường hợp ta muốn kết thúc trang sớm hơn, khi chưa hết trang vật lý. Câu lệnh **Printer.NewPage** cho phép ta làm việc đó. Gặp câu lệnh này máy sẽ in dòng dữ liệu tiếp theo ở đầu trang giấy mới. Nghĩa là câu lệnh **Printer.NewPage** gây ra việc đẩy giấy lên

một trang ở máy in.

Ví dụ về xuất dữ liệu ra máy in.

Giả sử có thủ tục - biến cố :

```
Private Sub cmdchayct_Click()  
Dim Ten$, DToan!, DLy!, DHoa!, DTBinh!  
Ten = "Le Van Ba"  
DToan = 8  
DLy = 9  
DHoa = 9  
DTBinh = (DToan + DLy + DHoa) / 3  
Printer.Print "Sinh Vien ", Ten, " Co Diem TB = ";  
Format(DTBinh, "##.##")  
Printer.EndDoc  
End Sub
```

Khi thủ tục - biến cố này được thực hiện dòng dữ liệu sau đây sẽ được in trên máy in:

Sinh Vien Le Van Ba Co Diem TB = 8.67.

10.3. IN DỮ LIỆU TRONG HỘP ẢNH

Trong phần 10.1. và 10.2. ta đã nghiên cứu cách dùng lệnh PRINT để in dữ liệu trên biểu mẫu và in dữ liệu trên máy in. Tuy nhiên còn có thể dùng câu lệnh PRINT để in dữ liệu trong các hộp ảnh.

10.3.1. Đối tượng hộp ảnh

Hộp ảnh (PictureBox) là một đối tượng của Visual Basic được dùng trong biểu mẫu để chứa các ảnh. Tuy nhiên, Visual Basic cho

phép dùng các câu lệnh PRINT, CLS ... để in dữ liệu trong hộp ảnh giống như đã làm đối với biểu mẫu.

Hộp ảnh có hơn 50 thuộc tính, 22 phương thức và đáp ứng được 19 biến cố. Đa số các thuộc tính của hộp ảnh ta đã biết. Một số thuộc tính đặc biệt của hộp ảnh là:

Bảng 10.3. Một số thuộc tính đặc biệt của hộp ảnh

Thuộc tính	Ý nghĩa
AutoRedrow	Nếu là True thì hộp ảnh sẽ xuất hiện lại dưới dạng đúng như trước khi nó bị che khuất.
AutoSize	Nếu là True thì hộp ảnh sẽ tự động điều chỉnh kích thước cho vừa với hình ảnh.
Picture	Xác định ảnh sẽ xuất hiện trong hộp ảnh

Thuộc tính Picture cho phép lập trình viên đặt hình ảnh vào hộp ảnh trong thời gian thiết kế chương trình. Để đặt hình ảnh vào hộp ảnh trong thời gian thực hiện chương trình người ta dùng câu lệnh sau đây:

`Tên_hộp_ảnh.Picture= LoadPicture(Tên_tệp)`

Nếu không có tên tệp trong phương thức LoadPicture(Tên_tệp), tức là câu lệnh gán lúc này có dạng:

`Tên_hộp_ảnh.Picture= LoadPicture()`

thì hình ảnh trong hộp ảnh sẽ bị xóa.

10.3.2. Sử dụng câu lệnh PRINT trong hộp ảnh

Việc in dữ liệu trong các hộp ảnh có những ưu điểm sau đây:

- Khi in dữ liệu trong biểu mẫu, dữ liệu có thể bị các đối tượng của biểu mẫu che khuất. Còn khi in dữ liệu trong hộp ảnh các dữ liệu

được in không bị đối tượng nào che khuất cả. Đây là ưu điểm nổi bật của in dữ liệu trong hộp ảnh so với in dữ liệu trong biểu mẫu.

- Các hộp ảnh đòi hỏi ít bộ nhớ hơn.

- Nếu cần có thể có nhiều hộp ảnh để chứa nhiều loại dữ liệu khác nhau trên cùng một biểu mẫu.

Muốn in dữ liệu trong hộp ảnh phải thêm đối tượng hộp ảnh vào biểu mẫu. Thao tác thêm đối tượng hộp ảnh vào biểu mẫu tương tự thao tác thêm hộp văn bản vào biểu mẫu. Để in dữ liệu trong hộp ảnh câu lệnh PRINT cần có dạng sau đây:

Tên_hộp_ảnh.PRINT [danh sách biểu thức] [:]

Theo câu lệnh này giá trị các biểu thức trong danh sách biểu thức sẽ được in trong hộp ảnh có tên đã chỉ ra.

Ví dụ: Giả sử trong biểu mẫu có một hộp ảnh tên là PicHop1 và có thủ tục - biến cố sau đây:

```
Private Sub cmdchayct_Click()  
Dim Ten$, DToan!, DLy!, DHoa!, DTBinh!  
Ten = "Le Van Ba"  
DToan = 8  
DLy = 9  
DHoa = 9  
DTBinh = (DToan + DLy + DHoa) / 3  
Print "Sinh Vien ", Ten  
Print "Co Diem TB = "; Format(DTBinh, "##.##"); "  
(Trong BM) "  
PicHop1.Print "Sinh Vien ", Ten  
PicHop1.Print "Co Diem TB = "; Format(DTBinh, "##.##");
```


" (Trong HA) "

End Sub

Trong thủ tục - biến cố này có 4 câu lệnh in (Print). Hai câu lệnh in đầu tiên in trên màn hình biểu mẫu hai dòng sau đây:

Sinh Viên Le Van Ba

Co Diem TB = 8.67 (Trong BM)

Còn 2 câu lệnh in sau cùng in trong hộp ảnh PicHopl hai dòng sau đây:

Sinh Viên Le Van Ba

Co Diem TB = 8.67 (Trong HA)

Dãy kí tự (Trong BM) có nghĩa là In Trong Biểu Mẫu. Còn dãy kí tự (Trong HA) có nghĩa là In Trong Hộp Ảnh.

CÂU HỎI ÔN TẬP

1. Hãy trình bày tác dụng của CurrentX và CurrentY.
2. In có qui cách là gì?
3. Qui cách là gì? Hãy trình bày tác dụng của hàm Format().
4. Hãy trình bày các qui cách dùng cho dữ liệu số.
5. Hãy trình bày các qui cách dùng cho dữ liệu kiểu ngày tháng.
6. Hãy trình bày các qui cách dùng cho dữ liệu thời gian.
7. Hãy trình bày cách Windows quản lý dữ liệu đưa ra máy in.
8. Hãy nêu qui cách và tác dụng của các lệnh dùng cho máy in.
9. Trình bày các đặc điểm của việc in dữ liệu trong hộp ảnh.

TẠO LẬP VÀ XỬ LÝ TỆP

Các chương trình mà ta đã viết trước đây có nhược điểm là các dữ liệu do ta nhập vào máy trong thời gian thực hiện chương trình sẽ bị xoá mất khi kết thúc chương trình. Sau này, nếu muốn sử dụng các dữ liệu đó, ta lại phải nhập chúng vào máy một lần nữa. Visual Basic, cũng như nhiều ngôn ngữ lập trình khác, có khả năng tạo lập và sử dụng tệp để khắc phục nhược điểm nêu trên.

11.1. KHÁI NIỆM TỆP

Tệp là một tập hợp các dữ liệu có cùng kiểu được nhóm lại với nhau, được ghi (lưu trữ) ở bộ nhớ ngoài của máy (ngày nay thường là đĩa từ) và được đặt một tên gọi là tên tệp. Tên tệp sẽ được sử dụng để mở và đóng tệp. Tệp tồn tại lâu dài trên đĩa từ, không bị xoá khi ta tắt máy.

Vì tệp không bị xoá khi tắt máy, nên những dữ liệu cần được lưu trữ lâu dài, được sử dụng nhiều lần thường được tổ chức theo những qui cách đặc biệt rồi ghi vào tệp. Khi cần dùng những dữ liệu này chỉ việc viết các câu lệnh mở tệp và đọc các dữ liệu đó từ tệp vào bộ nhớ, chứ không phải nhập các dữ liệu đó một lần nữa. Nhờ cách làm này mà tiết kiệm được rất nhiều thời gian, công sức và chi phí dành cho việc nhập dữ liệu. Đây là ưu điểm to lớn và lý do cơ bản của việc sử dụng tệp.

11.2. CÁC LOẠI TỆP TRONG VISUAL BASIC

Có ba loại tệp được dùng trong Visual Basic là tệp tuần tự, tệp ngẫu nhiên và tệp nhị phân.

Tệp tuần tự (sequential file) là tệp mà việc ghi dữ liệu vào tệp (hoặc đọc dữ liệu từ tệp) được thực hiện đối với từng phần nhỏ dữ liệu, từ đầu đến cuối. Có thể hình dung tệp tuần tự giống như một băng nhạc. Khi ta nghe băng nhạc đó ta phải nghe từ đầu. Nếu muốn nghe một bài hát nào đấy, ví dụ bài thứ 5, thì buộc phải nghe hết 4 bài đầu tiên hoặc tua qua các bài đó.

Tệp ngẫu nhiên (random file) là những tệp truy nhập trực tiếp. Ta có thể truy nhập trực tiếp đến mọi điểm của loại tệp này. Nếu tệp tuần tự giống như một băng nhạc, thì tệp ngẫu nhiên giống như một đĩa CD. Nếu ta yêu cầu máy chơi đĩa (CD player) chơi bài hát số 5, máy sẽ chuyển ngay đến một điểm xác định trên đĩa và chơi ngay bài hát này. Tương tự như vậy, ta có thể yêu cầu chương trình ứng dụng chuyển đến một điểm xác định trên tệp ngẫu nhiên và đọc hoặc ghi một phần của tệp tại điểm đó. Phần đó của tệp ngẫu nhiên được gọi là một bản ghi (record). Thường thường mỗi bản ghi chứa một số mục dữ liệu có liên quan với nhau. Mỗi mục dữ liệu này được gọi là một trường (field). Các trường của bản ghi có thể thuộc về các kiểu dữ liệu khác nhau. Các bản ghi trong tệp ngẫu nhiên có cấu trúc giống nhau. Có nghĩa là các bản ghi có số trường như nhau và kiểu cũng như kích thước của trường tương ứng của tất cả các bản ghi là giống nhau. Độ dài các trường của bản ghi được xác định chính xác trong chương trình ứng dụng và không thay đổi. Vì vậy độ dài của tất cả các bản ghi cũng bằng nhau. Chương trình ứng dụng biết được độ dài của bản ghi, do đó xác định được vị trí của mỗi bản ghi trong tệp. Giả sử độ dài của bản ghi là 60 byte, thì để tìm bản ghi thứ 5, chương trình ứng dụng chỉ việc nhảy qua 240 byte đầu tiên của tệp.

Tệp nhị phân (binary file) là loại tệp mà Visual Basic không giải thích bất cứ điều gì về nội dung của tệp. Để xử lý tệp nhị phân chương trình chỉ đọc từ tệp hoặc ghi vào tệp mỗi lần một byte dữ liệu.

Quá trình xử lý tệp thường gồm các công việc sau đây:

- Mở tệp
- Làm việc với tệp: nhập (tức là đọc) dữ liệu từ tệp, xuất (tức là ghi) dữ liệu ra tệp, tìm kiếm trong tệp ...)
- Đóng tệp

Visual Basic có những câu lệnh riêng để xử lý từng loại tệp nói trên. Trong chương này ta sẽ nghiên cứu cách xử lý tệp tuần tự và tệp ngẫu nhiên. Các vấn đề liên quan đến tệp nhị phân không nằm trong khuôn khổ cuốn sách này.

11.3. TỆP TUẦN TỰ

Phần này sẽ trình bày các câu lệnh của Visual Basic dành cho tệp tuần tự và sử dụng các câu lệnh đó để tạo lập và xử lý tệp tuần tự.

11.3.1. Mở tệp

Mở tệp là công việc đầu tiên của quá trình xử lý tệp. Cần phải mở tệp trước khi thực hiện các công việc khác đối với tệp này. Có thể mở tệp để đọc dữ liệu từ tệp (Input), để xuất dữ liệu ra tệp (Output) hoặc bổ sung dữ liệu vào tệp (Append). Câu lệnh mở tệp có dạng:

OPEN *tên_tệp* FOR *kiểu* AS [#] *số_nguyên*

Ở đây:

tên_tệp: là tên tệp cần mở (gồm cả ổ đĩa và đường dẫn nếu cần)

kiểu: là tham số xác định công việc được thực hiện với tệp này. Tham số này có các giá trị sau đây:

Bảng 11.1. Các giá trị của tham số kiểu trong câu lệnh OPEN

Giá trị	Ý nghĩa
Append	Mở tệp tuần tự để bổ sung dữ liệu vào cuối tệp này. Nếu tệp này không tồn tại thì máy sẽ tạo ra một tệp mới để ta có thể bổ sung dữ liệu vào cuối tệp này.
Input	Mở tệp tuần tự để nhập dữ liệu từ tệp này vào bộ nhớ. Nếu tệp này không tồn tại thì máy đưa ra thông báo lỗi.
Output	Mở tệp tuần tự để xuất dữ liệu từ bộ nhớ ra tệp này. Nếu đã tồn tại tệp cùng tên thì máy sẽ ghi tệp mới đè lên tệp đó (tức là xóa tệp đó).

[#] *số nguyên* là một số nguyên từ 1 đến 255 được gán cho tệp, được gọi là số hiệu tệp. Số này được sử dụng thay cho tên tệp trong các lệnh truy nhập đến tệp sau này. Dấu # trước số nguyên là tùy chọn (trong các ngôn ngữ BASIC khác là bắt buộc).

Lệnh mở tệp thực hiện một loạt công việc làm cho tệp sẵn sàng được xử lý. Lệnh mở tệp phải được thực hiện trước tất cả các lệnh khác muốn truy nhập tệp.

Ví dụ có câu lệnh:

Open "c:\sinhvien\thu.txt" For Output as 1

Câu lệnh này mở một tệp có tên là "thu.txt" để xuất dữ liệu. Tệp này nằm trong thư mục "sinhvien" của thư mục gốc của ổ đĩa C.

11.3.2. Đóng tệp

Sau khi sử dụng tệp cần phải đóng tệp trước khi kết thúc chương trình. Câu lệnh đóng tệp có dạng:

a) CLOSE *số_hiệu_tệp 1* [, *số_hiệu_tệp 2*] ...

b) CLOSE

Lệnh a) đóng các tệp có số hiệu tệp là số_hiệu_tệp 1 và số_hiệu_tệp 2 nếu có.

Lệnh b) đóng tất cả các tệp đang mở.

Câu lệnh đóng tệp ghi vào tệp những dữ liệu của tệp còn nằm ở bộ nhớ trong, giải phóng số hiệu tệp và vùng bộ nhớ đã dành cho tệp để có thể được dùng vào việc khác.

Ví dụ có các câu lệnh đóng tệp sau đây:

```
Close 1, 2, 3 \ (1)
```

```
Close \ (2)
```

Câu lệnh (1) đóng các tệp có số hiệu là 1, 2, 3. Câu lệnh (2) đóng tất cả các tệp đang mở.

11.3.3. Xóa tệp

Lệnh xóa tệp có dạng:

KILL *Tên_tệp*

Tên_tệp là tên tệp cần xóa (gồm cả ổ đĩa và đường dẫn nếu cần).

Lệnh này xóa khỏi đĩa từ tệp có tên trong lệnh. Ví dụ có câu lệnh sau đây:

```
Kill "c:\sinhvien\thu.txt"
```

Câu lệnh này xóa tệp có tên là "thu.txt" nằm trong thư mục "sinhvien" của thư mục gốc của ổ đĩa C.

11.3.4. Ghi dữ liệu vào tệp

Chúng ta có thể ghi dữ liệu vào tệp tuần tự bằng hai câu lệnh là WRITE # và PRINT #.

Câu lệnh WRITE

Câu lệnh này có dạng:

WRITE # số_hiệu_tệp [, ds biểu thức]

Lệnh này ghi dữ liệu kiểu bất kỳ ra tệp. Nếu không có danh sách biểu thức thì máy sẽ ghi 2 kí tự về đầu dòng (return mã 13) và dòng mới (mã 10) vào tệp.

Nếu danh sách biểu thức có nhiều biểu thức thì máy sẽ:

- Thêm dấu phẩy vào giữa giá trị 2 biểu thức liên nhau để ngăn cách chúng.

- Thêm kí tự “dấu nhảy kép” vào trước và sau giá trị dữ liệu kiểu chuỗi. Tức là các giá trị dữ liệu kiểu chuỗi được bao bọc bởi hai dấu nhảy kép khi ghi vào tệp.

- Luôn luôn ghi 2 kí tự về đầu dòng (return mã 13) và dòng mới (mã 10) vào cuối mỗi dòng.

Tệp tuần tự có dữ liệu được ghi theo kiểu này được gọi là tệp có dấu ngăn cách (delimited file).

Ví dụ: giả sử có các câu lệnh sau đây:

```
Tensv = "Dao Thi Hoa"
```

```
Lop = "TH"
```

```
Dtoan = 8
```

```
Write #1, Tensv, Lop, Dtoan
```

Sẽ ghi vào tệp có số hiệu là 1 giá trị các biến *Tensv*, *Lop*, *Dtoan* như sau:

```
"Dao Thi Hoa", "TH", 8↵
```

Chú ý:

- Nếu lệnh Write kết thúc bằng dấu chấm phẩy (;) thì lệnh Write tiếp theo sẽ ghi dữ liệu tiếp vào dòng hiện thời trên tệp.
- Có thể ghi vào tệp dữ liệu từ các biến, các hằng, các mảng cũng như các đối tượng điều khiển khác có trong biểu mẫu.
- Bổ sung dữ liệu vào tệp là ghi dữ liệu vào cuối tệp.

Câu lệnh PRINT #

Có thể dùng câu lệnh PRINT # để ghi dữ liệu vào tệp tuần tự. Câu lệnh này có dạng sau đây:

PRINT # số_hiệu_tệp [, ds biểu thức]

Câu lệnh này ghi giá trị các biểu thức có trong danh sách biểu thức vào tệp mà không có bất kỳ một thay đổi gì về qui cách các dữ liệu đó. Câu lệnh PRINT # tự động thêm kí tự xuống dòng vào cuối các dữ liệu được ghi vào tệp.

Ví dụ: giả sử có các câu lệnh sau đây:

```
Tensv = "Dao Thi Hoa"  
Lop = "TH"  
Dtoan = 8  
Print #1, Tensv, Lop, Dtoan
```

Sẽ ghi vào tệp có số hiệu là 1 giá trị các biến *Tensv*, *Lop*, *Dtoan* như sau:

```
Dao Thi Hoa TH 8
```

11.3.5. Đọc dữ liệu từ tệp tuần tự

Visual Basic có 2 câu lệnh và một hàm dùng để đọc dữ liệu từ tệp tuần tự. Đó là các câu lệnh INPUT # , LINE INPUT và hàm INPUT().

Câu lệnh INPUT #

Câu lệnh này dùng để đọc dữ liệu từ tệp có dấu ngăn cách. Tức là các dữ liệu được đọc bằng lệnh INPUT# phải được ghi vào tệp bằng lệnh WRITE#. Các dữ liệu đọc được sẽ được đặt vào các biến và các đối tượng điều khiển. Câu lệnh này có dạng:

INPUT # *số_hiệu_tệp* [, *ds biến*]

Lệnh này đọc dữ liệu từ tệp tuần tự có dấu ngăn cách các giá trị dữ liệu rồi gán các giá trị đó cho các biến có trong danh sách biến.

VD: Có lệnh Input #1, Tensv, Lop, Dtoan

Câu lệnh này đọc 3 giá trị từ tệp có số hiệu là 1 rồi gán các giá trị đó cho các biến Tensv, Lop và dtoan.

Câu lệnh LINE INPUT

Câu lệnh này đọc nguyên cả một dòng dữ liệu từ tệp tuần tự rồi gán dòng đó cho một biến kiểu String. Câu lệnh này có qui cách như sau:

LINE INPUT *số_hiệu_tệp* *tên_biến*

Chú ý: Các dữ liệu được đọc bằng lệnh LINE INPUT thường được ghi vào tệp bằng lệnh PRINT#.

Lệnh LINE INPUT đọc từ tệp các kí tự cho đến khi gặp carriage return (Chr(13)) hoặc hai kí tự carriage return–linefeed (Chr(13) + Chr(10)) đi liền nhau. Dãy hai kí tự carriage return–linefeed được bỏ qua, không được thêm vào dãy kí tự sẽ được gán cho biến có tên trong lệnh.

Có thể lặp lại câu lệnh LINE INPUT số lần cần thiết để đọc hết tất cả các dòng có trong tệp. Trong trường hợp này cần dùng hàm EOF(số hiệu tệp) làm điều kiện kết thúc quá trình đọc tệp.

Hàm EOF()

Hàm EOF(*số hiệu tệp*) cho giá trị False nếu lệnh INPUT# chưa đọc được điểm cuối tệp (EOF : End Of File). Hàm này cho giá trị True nếu lệnh INPUT# đọc được điểm cuối tệp, tức là đã đọc hết các dữ liệu có trong tệp.

Hàm INPUT()

Hàm này dùng để đọc một số kí tự từ tệp tuần tự. Nó đọc các kí tự dữ liệu và bỏ qua các dấu trở về đầu dòng.

Hàm này có qui cách sau đây:

INPUT (*L%*, *số_hiệu_tệp*)

Tham số *L%* là một giá trị nguyên, xác định độ dài của dữ liệu cần đọc (tính bằng byte). Tham số *số_hiệu_tệp* xác định tệp chứa dữ liệu được đọc.

Ví dụ: có câu lệnh sau đây:

```
Dong$ = INPUT (30, #2 )
```

Câu lệnh này đọc 30 byte (tức là 30 kí tự) dữ liệu từ tệp #2 rồi gán các dữ liệu đọc được cho các biến Dong\$.

Nếu ta muốn đọc toàn bộ tệp thì dùng hàm sau đây để xác định độ dài của tệp:

LOF(số hiệu tệp)

LOF là cách viết tắt của các từ Anh Length Of File (độ dài của tệp). Khi đó câu lệnh INPUT có dạng:

```
Dong$ = INPUT (LOF(2), #2 )
```

Câu lệnh này đọc toàn bộ dữ liệu từ tệp #2 rồi gán các dữ liệu đó cho biến Dong\$.

11.3.6. Ví dụ về tạo lập và xử lý tệp tuần tự

Chương trình trong ví dụ này có tên là TepTuanTu (Tệp Tuần Tự). Chương trình này tạo ra một tệp tuần tự. Tên tệp do người sử dụng đặt và được nhập vào từ bàn phím. Sau đó xem dữ liệu của tệp và có thể bổ sung các dòng dữ liệu vào tệp. Chương trình này có 5 nút lệnh. Tên và chức năng của các nút lệnh này là:

Tên nút lệnh	Chức năng
cmdTaoTep	Tạo tệp
cmdBoSung	Bổ sung dữ liệu vào tệp
cmdXem	Xem dữ liệu có trong tệp
cmdXoaMH	Xoá màn hình
cmdKetThuc	Kết Thúc chương trình

Phần mã lệnh của chương trình có dạng sau đây:

```
Option Explicit
Dim TenTep$, TraLoi$
Dim i%, j%, k%, Tensv$, Lop$, dt%

Private Sub cmdBoSung_Click()
    TenTep = InputBox("ten tep = ")
    Open "c:\baitaptc\" & TenTep & ".txt" For Append As 1
    TraLoi = "c"
    Do While TraLoi = "c"
        Tensv = InputBox("ten = ")
        Lop = InputBox("Lop =")
        dt = Val(InputBox("Diem Toan ="))
        Write #1, Tensv, Lop, dt
```

```

    TraLoi = InputBox("Tiep tuc ? (c/k)")
Loop
Close 1
End Sub

Private Sub cmdXem_Click()
TenTep = InputBox("ten tep = ")
Open "c:\baitaptc\" & TenTep & ".txt" For Input As 1
Do Until (EOF(1) = True)
    Input #1, Tensv, Lop, dt
    Print Tensv, Lop, dt
Loop
Close 1
End Sub

Private Sub cmdKetThuc_Click()
    End
End Sub

Private Sub cmdTaoTep_Click()
TenTep = InputBox("ten tep = ")
Open "c:\baitaptc\" & TenTep & ".txt" For Output As 1
TraLoi = "c"
Do While TraLoi = "c"
    Tensv = InputBox("ten = ")
    Lop = InputBox("Lop = ")
    dt = Val(InputBox("Diem Toan = "))

```

```

Write #1, Tensv, Lop, dt
TraLoi = InputBox("Tiep tục ? (c/k)")
Loop
Close 1
End Sub

Private Sub cmdXoaMH_Click()
   Cls
End Sub

```

Giải thích chương trình

Thủ tục - biến cố cmdTaoTep_Click() dùng để tạo ra tệp tuần tự có tên do người sử dụng đặt và nhập vào từ bàn phím. Sau khi tạo ra tệp, thủ tục - biến cố này cho phép người sử dụng lần lượt ghi các dòng dữ liệu vào tệp. Mỗi dòng dữ liệu gồm có 3 mục dữ liệu là tên sinh viên (Tensv), lớp (Lop), điểm toán (dt). Các dữ liệu này được nhập vào từ bàn phím và chứa trong các biến tương ứng là Tensv, Lop, và dt. Câu lệnh:

```
Write #1, Tensv, Lop, dt
```

Sẽ ghi các dữ liệu này vào tệp #1. Sau khi nhập dữ liệu vào các biến và ghi một dòng vào tệp, máy hỏi người sử dụng “Tiep tục (c/k)?”. Nếu câu trả lời là “c” thì việc nhập dữ liệu được tiếp tục lặp lại. Nếu câu trả lời là “k” thì kết thúc nhập dữ liệu và kết thúc thủ tục - biến cố này.

Thủ tục - biến cố cmdXem_Click() dùng để xem toàn bộ các dòng dữ liệu của tệp tuần tự. Khi thực hiện thủ tục - biến cố này, máy sẽ đọc từng dòng của tệp và in các dòng đọc được từ tệp trên màn hình biểu mẫu. Quá trình này sẽ kết thúc khi hàm EOF(#1) cho giá trị True, tức là khi đã đọc hết tệp.

Thủ tục - biến cố `cmdBoSung_Click()` cho phép bổ sung các dòng dữ liệu vào cuối tệp. Thủ tục - biến cố này tương tự thủ tục - biến cố `cmdTaoTep_Click()`.

11.4. TẬP NGẪU NHIÊN

Qua phần 11.2. chúng ta đã biết rằng tệp ngẫu nhiên bao gồm các bản ghi. Mỗi bản ghi chứa một số mục dữ liệu có liên quan với nhau, gọi là các trường. Các bản ghi có cấu trúc và độ dài giống nhau. Nhờ vậy mà chương trình ứng dụng có thể xác định được vị trí của một bản ghi bất kỳ trong tệp.

Muốn sử dụng bản ghi (để tạo ra tệp ngẫu nhiên) phải khai báo kiểu dữ liệu của bản ghi. Sau đó dùng kiểu này để khai báo các biến chứa dữ liệu của bản ghi.

11.4.1. Khai báo bản ghi

Visual Basic cho phép lập trình viên tự định nghĩa các kiểu dữ liệu riêng cho mình. Các bản ghi là dữ liệu có kiểu do lập trình viên tự định nghĩa, tiếng Anh là User Defined Type, viết tắt là UDT. Kiểu cấu trúc (struct type) là kiểu dữ liệu tự định nghĩa hay được dùng cho các bản ghi. Kiểu cấu trúc được định nghĩa theo mẫu sau đây:

```
TYPE tên_kiểu  
  Khai báo trường 1  
  Khai báo trường 2  
  .....  
  Khai báo trường n  
END TYPE
```

Chú ý:

- Phần khai báo kiểu cấu trúc này phải được viết trong đơn thể chương trình độc lập (modul độc lập).

- Mỗi trường của kiểu cấu trúc có thể có kiểu bất kỳ mà Visual Basic cho phép. Trường cũng có thể là mảng.

Sau khi đã được định nghĩa như vậy tên kiểu đứng sau TYPE có thể được sử dụng để khai báo các bản ghi.

Ví dụ: có thể khai báo kiểu cấu trúc có tên là `PhieuDiem` như sau:

```
TYPE PhieuDiem
HoVaTen As String * 25
Lop As String * 5
DToan As Integer
DLy As Integer
DHoa As Integer
DTBinh As Single
END TYPE
```

Xin nhắc lại là phần khai báo kiểu cấu trúc này phải được viết trong đơn thể chương trình độc lập (modul độc lập).

Sau khi khai báo như vậy, trong đơn thể biểu mẫu ta có thể khai báo biến dùng cho bản ghi có tên là `PDiem` như sau:

```
PDiem As PhieuDiem
```

Câu lệnh này khai báo một biến có tên là `PDiem`. Kiểu của biến này là `PhieuDiem`, là kiểu cấu trúc. Biến `PDiem` có tất cả các đặc điểm và các trường của `PhieuDiem`. Các trường của `PDiem` được truy nhập bằng cách viết sau đây:

PDiem.Tên_Trường

Ví dụ:

PDiem.Ten

PDiem.Lop

PDiem.Dtoan

PDiem.Dly

PDiem.Dhoa

PDiem.DTBinh

Tức là đặt tên bản ghi và một dấu chấm (.) trước tên trường.

Biến PDiem sẽ được sử dụng để tạo ra tệp ngẫu nhiên. Ta thấy rằng ở đây PhieuDiem có chức năng một từ định kiểu, dùng để xác định kiểu của dữ liệu trong câu lệnh khai báo dữ liệu.

Chú ý: Trong phần định nghĩa kiểu cấu trúc PhieuDiem có câu lệnh khai báo:

```
HoVaTen As String * 25
```

Câu lệnh này khai báo một trường có tên là HoVaTen, kiểu String và có độ dài là 25 byte. Điều này đảm bảo cho trường HoVaTen có độ dài cố định là 25 byte trong tất cả các bản ghi của tệp ngẫu nhiên. Nhờ vậy mà tất cả các bản ghi của tệp có độ dài bằng nhau.

11.4.2. Mở tệp ngẫu nhiên

Câu lệnh mở tệp ngẫu nhiên có dạng sau đây:

```
OPEN tên_tệp [ FOR RANDOM ] AS [#] số nguyên LEN =  
DoDaiBG
```

Theo ngầm định thì câu lệnh này dùng để mở tệp ngẫu nhiên, nên phần FOR RANDOM là tùy chọn. Tuy nhiên, nếu phần này bị bỏ qua thì nó cũng được Visual Basic tự động thêm vào câu lệnh này.

Phần **LEN** = **DoDaiBG** xác định độ dài của bản ghi tính bằng byte. Tham số **DoDaiBG** (độ dài bản ghi) là một giá trị nguyên. Nếu giá trị của **DoDaiBG** nhỏ hơn độ dài thực tế của bản ghi thì sẽ gây lỗi. Nếu giá trị của **DoDaiBG** lớn hơn độ dài thực tế của bản ghi thì các bản ghi vẫn được truy nhập bình thường, chỉ có điều sẽ gây lãng phí bộ nhớ. Để xác định chính xác độ dài bản ghi có thể sử dụng hàm **Len**(tên bản ghi).

Theo câu lệnh này máy sẽ mở tệp có tên trong lệnh. Nếu tệp này chưa có máy sẽ tạo ra nó.

Ví dụ: Có thể viết câu lệnh mở tệp như sau:

```
Open DiemThi For Random As # 1 Len = 40
Open DiemThi As # 1 Len = 40
Open DiemThi For Random As # 1 Len = Len(PDiem)
```

Trong các câu lệnh này thì tên tệp là *DiemThi*, độ dài bản ghi là 40. Giả sử tên bản ghi là *PDiem*, độ dài của bản ghi là 40, thì 3 câu lệnh này hoàn toàn tương đương nhau. Trong câu lệnh thứ ba có sử dụng hàm **Len(PDiem)** để xác định độ dài bản ghi.

11.4.3. Ghi dữ liệu vào tệp ngẫu nhiên

Câu lệnh ghi dữ liệu vào tệp ngẫu nhiên có dạng:

PUT # Số_hiệu_tệp, Vị_trí, Tên_bản_ghi

Ở đây **Vị_trí** là một giá trị nguyên, có giá trị nhỏ nhất là 1. Điều đó có nghĩa là vị trí của bản ghi trong tệp ngẫu nhiên được tính từ 1 trở đi.

Theo câu lệnh này máy ghi một bản ghi xác định bởi **Tên_bản_ghi** vào tệp xác định bởi **Số_hiệu_tệp** ở vị trí do **Vị_trí** xác định.

Ví dụ: có thể viết câu lệnh ghi dữ liệu vào tệp như sau:

```
Put #1, 1, PDiem
```

Câu lệnh này ghi vào vị trí đầu tiên (vị trí số 1) của tệp #1 bản ghi có tên là PDiem.

Lần đầu tiên tạo ra tệp ngẫu nhiên, muốn ghi vào tệp này nhiều bản ghi ta làm như sau: gán cho biến I (kiểu nguyên) giá trị 1, rồi thực hiện lặp lại 2 câu lệnh sau đây một số lần cần thiết:

```
Put #1, I, PDiem
```

```
I = I + 1
```

Cứ sau mỗi lần ghi một bản ghi vào tệp thì biến I được tăng thêm 1. Vì vậy các bản ghi sẽ được ghi vào các vị trí liên tiếp nhau trong tệp.

11.4.4. Đọc dữ liệu từ tệp ngẫu nhiên

Câu lệnh đọc dữ liệu từ tệp ngẫu nhiên có dạng:

```
GET # Số_hiệu_tệp, Vị_trí, Tên_bản_ghi
```

Theo câu lệnh này máy đọc một bản ghi từ tệp xác định bởi *Số_hiệu_tệp* ở vị trí do *Vị_trí* xác định. Biến có tên là *Tên_bản_ghi* sẽ nhận được nội dung bản ghi vừa đọc được bằng câu lệnh này.

Ví dụ: có thể viết câu lệnh đọc dữ liệu từ tệp như sau:

```
GET #1, 2, PDiem
```

Câu lệnh này đọc bản ghi ở vị trí thứ hai của tệp #1. Bản ghi vừa đọc được từ tệp sẽ được gán cho biến có tên là PDiem.

Để đọc lần lượt toàn bộ các bản ghi của tệp ngẫu nhiên ta làm như sau: gán cho biến I (kiểu nguyên) giá trị 1, rồi thực hiện lặp lại các câu lệnh sau đây:

Get #1, I, PDiem

Các câu lệnh sử dụng biến PDiem

I = I +1

Các câu lệnh này được thực hiện cho đến khi hàm EOF(#1) cho giá trị True.

Cứ sau mỗi lần đọc một bản ghi từ tệp thì biến I được tăng thêm 1. Vì vậy tất cả các bản ghi sẽ được đọc lần lượt từ đầu đến cuối tệp.

11.4.5. Ví dụ về tạo lập và xử lý tệp ngẫu nhiên

Chương trình trong ví dụ này có tên là TepNgauNhiem (Tệp Ngẫu Nhiên). Chương trình này 4 nút lệnh. Tên và chức năng của các nút lệnh này là:

Tên nút lệnh	Chức năng
cmdTaoTep	Tạo tệp
cmdXem	Xem dữ liệu có trong tệp
cmdXoaMH	Xoá màn hình
cmdKetThuc	Kết thúc chương trình

Phần mã lệnh của chương trình gồm 2 đơn thể chương trình. Đơn thể chương trình độc lập dùng để khai báo kiểu cấu trúc. Đơn thể chương trình biểu mẫu dành cho các thủ tục - biến cố của các nút lệnh của chương trình này. Phần mã lệnh của chương trình có dạng sau đây:

Đơn thể chương trình độc lập:

Option Explicit

Type PhieuDiem

Ten As String * 25

Lop As String * 5

```

DToan As Integer
DLy As Integer
DHoa As Integer
DTBinh As Single
End Type

```

Đơn thể chương trình biểu mẫu:

```

Dim TenTep$, TraLoi$
Dim PDiem As PhieuDiem
Dim Sobg As Long
Dim I As Long

Private Sub cmdXem_Click()
TenTep = InputBox("ten tep = ")
Open "c:\baitaptc\" & TenTep & ".rnd" As 1 Len =
Len(PDiem)
I = 1
Do Until (EOF(1) = True)
    Get #1, i, PDiem
    I = I + 1
Loop
sobg = I - 1
TraLoi = "c"
Do While TraLoi = "c"
    I = Val(InputBox("So hieu BG = (1-> " + Str(sobg) +
") = "))
    Get #1, I, PDiem
    Print PDiem.Ten, PDiem.Lop, PDiem.DToan, PDiem.DLy,

```

```

PDiem.DHoa
    TraLoi = InputBox("Tiep tuc ? (c/k)", " ", "c")
Loop
Close 1
End Sub

Private Sub cmdKetThuc_Click()
    End
End Sub

Private Sub cmdTaoTep_Click()
    TenTep = InputBox("Ten tep = ")
    Open "c:\baitaptc\" & TenTep & ".rnd" For Random As 1
    Len = 40
    TraLoi = "c"
    i = 1
    Do While TraLoi = "c"
        PDiem.Ten = InputBox("Ten SV = ")
        PDiem.Lop = InputBox("Lop =")
        PDiem.DToan = Val(InputBox("Diem Toan ="))
        PDiem.DLy = Val(InputBox("Diem Ly ="))
        PDiem.DHoa = Val(InputBox("Diem Hoa ="))
        TraLoi = InputBox("Tiep tuc (c/k) ?", " TT", "c")
        Put #1, i, PDiem
        i = i + 1
    Loop
Close 1
End Sub

```

```
Private Sub cmdXoaMH_Click()
```

```
Cls
```

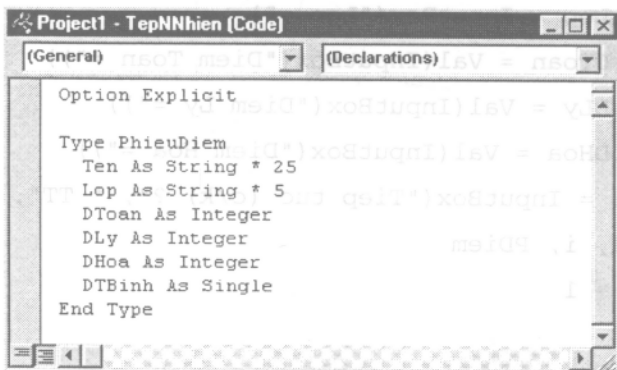
```
End Sub
```

Giải thích chương trình

Trong chương trình này ta phải tạo ra một đơn thể chương trình độc lập để khai báo kiểu dữ liệu cấu trúc. Để thêm đơn thể chương trình độc lập vào dự án ta thực hiện các thao tác sau đây:

- Chọn Project
- Chọn Add Module. Hộp Add Module xuất hiện.
- Chọn New
- Chọn Open

Sau khi thực hiện các thao tác đó, Visual Basic sẽ thêm một đơn thể chương trình độc lập vào dự án và cửa sổ mã lệnh của đơn thể này xuất hiện. Ta chỉ việc viết các dòng dùng để khai báo dữ liệu kiểu cấu trúc trong cửa sổ này. Cửa sổ mã lệnh của đơn thể chương trình độc lập của chương trình TepNNhien có dạng sau đây:



Hình 11.2. Cửa sổ mã lệnh của đơn thể chương trình độc lập của chương trình TepNNhien.

Khi lưu dự án lên đĩa từ đơn thể chương trình độc lập này được ghi trong một tệp riêng. Tên tệp này luôn luôn có phần mở rộng là “.BAS”. Như vậy chương trình TepNNhien sẽ có 3 tệp riêng biệt: một cho dự án, một tệp cho biểu mẫu và một tệp cho đơn thể chương trình độc lập.

Thủ tục - biến cố `cmdTaoTep_Click()` dùng để tạo tệp ngẫu nhiên có tên được người sử dụng nhập vào từ bàn phím. Biến `PDiem` có kiểu cấu trúc, dùng để chứa các bản ghi của tệp. Sau khi tạo ra tệp, thủ tục - biến cố này cho phép người sử dụng lần lượt nhập các bản ghi vào tệp. Sau khi nhập xong một bản ghi, máy hỏi “Tiep tục (c/k) ?”. Nếu câu trả lời là “c” thì việc nhập dữ liệu được tiếp tục. Nếu câu trả lời là “k” thì kết thúc nhập dữ liệu và kết thúc thủ tục - biến cố này.

Thủ tục - biến cố `cmdXem_Click()` dùng để xem từng bản ghi của tệp ngẫu nhiên. Khi thực hiện thủ tục - biến cố này, đầu tiên máy duyệt từ đầu đến cuối tệp để đếm số bản ghi của tệp. Số bản ghi của tệp được chứa trong biến `Sobg`. Sau đó quá trình xem dữ liệu bắt đầu. Quá trình này bắt đầu bằng việc nhập số hiệu bản ghi cần xem (từ 1 -> `Sobg`) vào biến `I`. Máy sẽ đọc từ tệp bản ghi thứ `I`, và gán các dữ liệu đọc được cho biến `PDiem`. Các trường của biến `PDiem` được in trên màn hình để cho người sử dụng xem. Sau đó máy hỏi người sử dụng “Tiep tục (c/k) ?”. Nếu câu trả lời là “c” thì việc xem dữ liệu được tiếp tục. Nếu câu trả lời là “k” thì kết thúc xem dữ liệu và kết thúc thủ tục - biến cố này.

CÂU HỎI ÔN TẬP

1. Tệp là gì? Tại sao cần sử dụng tệp?
2. Visual Basic có những loại tệp nào? Hãy nêu đặc điểm của từng loại tệp đó.
3. Quá trình xử lý tệp thường gồm những công việc gì?

4. Hãy trình bày các câu lệnh liên quan đến tệp tuần tự. Cho các ví dụ sử dụng các câu lệnh đó.

5. Hãy trình bày quy tắc khai báo bản ghi.

6. Hãy trình bày các câu lệnh liên quan đến tệp ngẫu nhiên. Cho các ví dụ sử dụng các câu lệnh đó.

7. Hãy viết thêm cho chương trình `TepTuanTu` một thủ tục - biến cố để xoá khỏi tệp những dòng có điểm toán nhỏ hơn 3 ($dt < 3$).

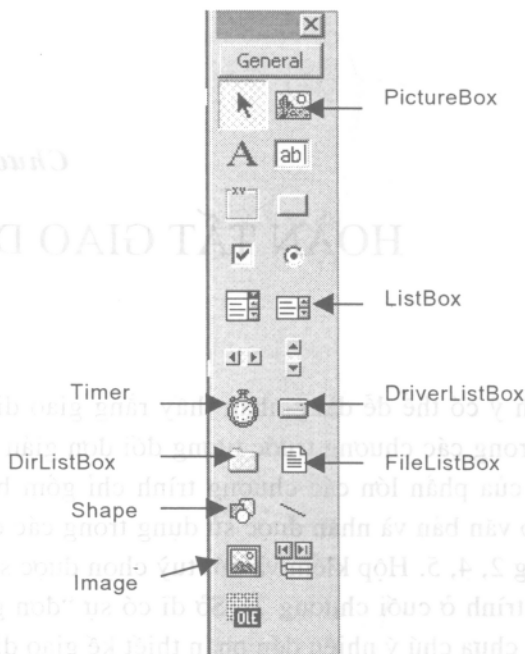
8. Hãy viết thêm cho chương trình `TepNNhien` một thủ tục - biến cố để sửa một bản ghi của tệp khi biết số hiệu bản ghi này.

HOÀN TẤT GIAO DIỆN

Các độc giả tinh ý có thể dễ dàng nhận thấy rằng giao diện của các chương trình có trong các chương trước tương đối đơn giản và đơn điệu. Phần giao diện của phần lớn các chương trình chỉ gồm ba hoặc bốn nút lệnh. Các hộp văn bản và nhãn được sử dụng trong các chương trình trong các chương 2, 4, 5. Hộp kiểm và nút tùy chọn được sử dụng trong một số chương trình ở cuối chương 7. Sở dĩ có sự “đơn giản và đơn điệu” đó là vì ta chưa chú ý nhiều đến phần thiết kế giao diện của chương trình.

Trong chương này ta sẽ nghiên cứu những đối tượng chuyên dùng để thiết kế giao diện của Visual Basic, như hệ soạn thảo thực đơn, các hộp hội thoại, câu lệnh thông báo và hàm thông báo. Sử dụng những đối tượng này làm cho giao diện của chương trình trở nên dễ sử dụng, đa dạng, đẹp mắt và hấp dẫn hơn.

Trong Hình 12.0. dưới đây có biểu tượng của các công cụ điều khiển các đối tượng được nghiên cứu trong chương này.



Hình 12.0. Hộp công cụ của Visual Basic

12.1. THỰC ĐƠN

Trong các chương trình trước đây, để giải quyết bài toán của mình, ta đã dùng các nút lệnh để điều khiển máy thực hiện các thủ tục - biến cố khác nhau. Trong Visual Basic có một cách khác để thực hiện công việc như vậy. Đó là sử dụng thực đơn để điều khiển chương trình.

12.1.1. Khái niệm thực đơn

Các độc giả chắc đã quen thuộc đối với hệ thống thực đơn của các phần mềm của MicroSoft, như thực đơn của MicroSoft Word hoặc MicroSoft Excel.... Mỗi trò chơi (Games) cũng có hệ thống thực đơn riêng của mình.

Thực đơn là một danh sách các chức năng mà một phần mềm có thể thực hiện được. Người sử dụng có thể dễ dàng chọn một trong các chức năng đó và điều khiển máy thực hiện chức năng này để giải bài toán của mình. Việc một chức năng của thực đơn có được chọn để sử dụng hay không tùy thuộc vào ý muốn của người sử dụng, nên mỗi chức năng của bảng thực đơn còn được gọi là một tùy chọn. Tuy nhiên, ở đây ta dùng từ “chức năng” để chỉ các thành phần của thực đơn.

Thực đơn được sử dụng rất rộng rãi trong các phần mềm của Microsoft. Hệ thống thực đơn của Visual Basic cũng tương tự như hệ thống thực đơn của các phần mềm khác của Microsoft, ví dụ Microsoft Word, Microsoft Excel....

Trong chương trình Visual Basic ta có thể tạo ra thực đơn ngang (tức là thực đơn theo hàng ngang), thực đơn dọc (thực đơn theo cột dọc) hoặc thực đơn hỗn hợp.

- Thực đơn ngang là loại thực đơn trong đó các chức năng của thực đơn xuất hiện chức năng nọ bên cạnh chức năng kia theo hàng ngang trên màn hình.

- Thực đơn dọc là loại thực đơn trong đó các chức năng của thực đơn xuất hiện chức năng nọ bên dưới chức năng kia theo cột dọc trên màn hình.

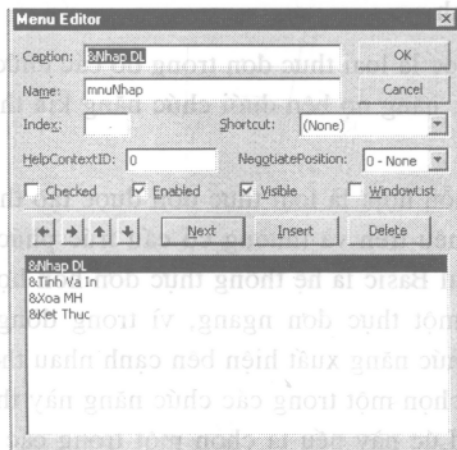
- Thực đơn hỗn hợp: là loại thực đơn được tạo thành do kết hợp hai loại thực đơn nêu trên và thường có cấu trúc phức tạp. Hệ thống thực đơn của Visual Basic là hệ thống thực đơn hỗn hợp. Trước hết ta thấy rằng đây là một thực đơn ngang, vì trong dòng thực đơn của Visual Basic các chức năng xuất hiện bên cạnh nhau theo hàng ngang. Tuy nhiên, nếu ta chọn một trong các chức năng này thì một thực đơn dọc lại xuất hiện. Lúc này nếu ta chọn một trong các chức năng của thực đơn dọc này, thì rất có thể một thực đơn dọc khác hoặc một hộp hội thoại sẽ xuất hiện.

Trong các phần dưới đây sẽ có các ví dụ về tạo lập và sử dụng các loại thực đơn vừa trình bày.

12.1.2. Tạo thực đơn

Visual Basic có sẵn một chương trình riêng dùng để soạn thảo thực đơn. Chương trình này có tên là hệ soạn thảo thực đơn (Menu Editor). Ta có thể dùng hệ soạn thảo thực đơn của Visual Basic để tạo ra thực đơn cho chương trình Visual Basic của mình. Để sử dụng hệ soạn thảo thực đơn cần phải thực hiện các thao tác sau đây:

- Chọn biểu mẫu
- Chọn Tool
- Chọn Menu Editor. Cửa sổ soạn thảo thực đơn sẽ xuất hiện (Hình 12.1).
- Lần lượt soạn thảo từng chức năng của thực đơn. Soạn thảo xong chọn OK.



Hình 12.1. Cửa sổ soạn thảo thực đơn của Visual Basic

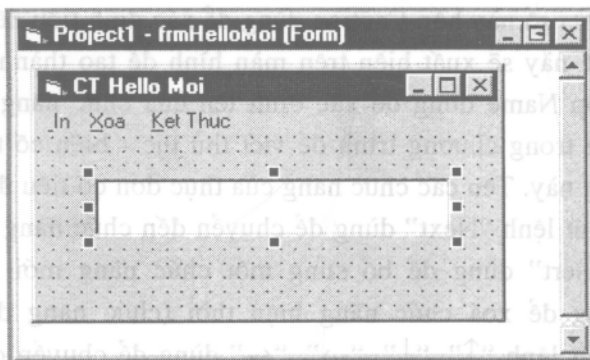
Trong cửa sổ này hộp Caption dùng để xác định tiêu đề của chức năng. Tiêu đề này sẽ xuất hiện trên màn hình để tạo thành hệ thống thực đơn. Hộp Name dùng để xác định tên của chức năng. Tên này được sử dụng trong chương trình để viết thủ tục - biến cố tương ứng với chức năng này. Tên các chức năng của thực đơn có tiêu đề là 3 chữ cái "mnu". Nút lệnh "Next" dùng để chuyển đến chức năng tiếp theo. Nút lệnh "Insert" dùng để bổ sung một chức năng mới. Nút lệnh "Delete" dùng để xóa chức năng hiện thời (chức năng đang được chọn). Các nút lệnh "↑", "↓", "→", "←" dùng để chuyển chức năng hiện thời lên trên, xuống dưới, sang phải hoặc sang trái.

Khi soạn thảo xong một chức năng cần chọn nút Next để tiếp tục soạn thảo chức năng tiếp theo. Khi đã soạn thảo xong tất cả các chức năng cần chọn OK. Khi đó cửa sổ soạn thảo thực đơn sẽ biến mất, còn hệ thống thực đơn vừa được tạo ra thì xuất hiện trên màn hình. Sau này, khi cần thiết, ta lại có thể dùng hệ soạn thảo thực đơn để bổ sung hoặc sửa chữa hệ thống thực đơn đã có trong chương trình.

Chú ý: Hệ soạn thảo thực đơn của Visual Basic chỉ cho phép tạo ra (tức là vẽ) các chức năng của thực đơn trên màn hình. Lập trình viên phải tự viết các thủ tục - biến cố tương ứng với các chức năng của thực đơn. Nếu không, khi các chức năng của thực đơn được sử dụng sẽ không có công việc nào được thực hiện cả.

Sau đây là một ví dụ về chương trình có sử dụng thực đơn. Chương trình này có tên là "HeLLoMoi" (Hello Mới). Chương trình này thực hiện công việc hoàn toàn giống chương trình Hello trong chương 2. Điểm khác là trong chương trình "HeLLoMoi" ta dùng thực đơn để thiết kế giao diện của chương trình.

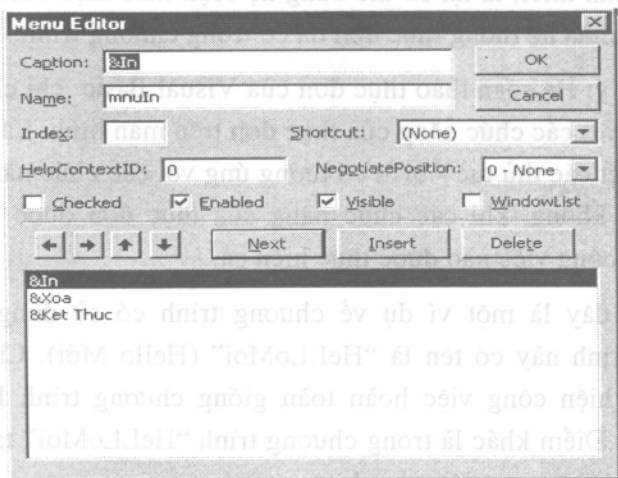
Màn hình giao diện của chương trình "HeLLoMoi" có dạng sau đây:



Hình 12.2. Màn hình giao diện của chương trình “HeLLoMoi”

Hệ thống thực đơn của chương trình HelloMoi là hệ thống thực đơn ngang.

Cửa sổ tạo lập thực đơn của chương trình “HeLLoMoi” có dạng sau đây:



Hình 12.3. Cửa sổ soạn thảo thực đơn của chương trình “HeLLoMoi”

Các chức năng của thực đơn có các thuộc tính sau đây:

**Bảng 12.1. Các chức năng của thực đơn
của chương trình “HeLLoMoi”**

Tiêu đề của chức năng	Tên chức năng
&In	MnuIn
&Xoa	MnuXoa
&Ket Thuc	MnuKetThuc

Các thủ tục - biến cố của của chương trình “HeLLoMoi” có dạng sau đây:

```
Private Sub mnuXoa_Click()
```

```
    txtKetQua.Text = " "
```

```
End Sub
```

```
Private Sub mnuKetThuc_Click()
```

```
    Beep
```

```
End
```

```
End Sub
```

```
Private Sub mnuIn_Click()
```

```
    txtKetQua.Text = "Hello The World !"
```

```
End Sub
```

Chạy thử chương trình “HeLLoMoi”.

Sau khi đã thiết kế chương trình như vậy, Hãy gõ phím F5 để chạy thử chương trình. Nếu nhấp chuột vào chức năng “In” (còn gọi là sử dụng chức năng “In”) thì thủ tục - biến cố `mnuIn_Click()` được thực hiện. Điều đó làm cho dòng kí tự “Hello The World !” xuất hiện trong hộp văn bản chứa kết quả. Nếu sử dụng chức năng “Xoa” thì thủ tục - biến cố `mnuXoa_Click()` sẽ được thực hiện. Điều đó làm cho nội

dung hộp văn bản chứa kết quả sẽ bị xoá. Còn nếu sử dụng chức năng “Ket Thuc” thì thủ tục - biến cố `mnKetThuc_Click()` sẽ được thực hiện. Câu lệnh End trong thủ tục - biến cố này sẽ kết thúc chương trình.

Chúng ta thấy rằng việc sử dụng các chức năng của thực đơn cũng có tác dụng tương tự như sử dụng các nút lệnh mà ta đã quen thuộc khi chạy các chương trình trong các chương trước.

12.2. HỘP HỘI THOẠI CHUNG

12.2.1. Hộp hội thoại chung là gì?

Hộp hội thoại chung là một đối tượng của Visual Basic được dùng để tạo ra các hộp hội thoại trong thời gian chạy chương trình để thực hiện các công việc sau đây:

- Mở tệp (Open)
- Lưu tệp lên đĩa từ với tên mới (Save As).
- Xác định màu (Color)
- Xác định font
- Xác định máy in (Printer)
- Xác định tệp Help

Từ một hộp hội thoại chung có thể tạo ra 6 hộp hội thoại khác nhau. Vì vậy hộp hội thoại này có tên là hộp hội thoại chung (Common Dialog Box). Có thể nói rằng hộp hội thoại chung là đối tượng rất có ích để thiết kế màn hình giao diện của chương trình. Một điều đáng chú ý là các cửa sổ hội thoại do hộp hội thoại chung tạo ra không xuất hiện trên màn hình của biểu mẫu lúc thiết kế chương trình, mà chỉ xuất hiện trên màn hình trong thời gian chạy chương trình. Các cửa sổ do hộp hội thoại chung tạo ra có dạng các cửa sổ chuẩn của MicroSoft Windows.

12.2.2. Bổ sung công cụ điều khiển hộp hội thoại chung vào hộp công cụ

Công cụ điều khiển hộp hội thoại chung thường không có sẵn trong hộp công cụ. Để bổ sung công cụ điều khiển này vào hộp công cụ ta làm như sau:

- Chọn Project
- Chọn Components
- Chọn Microsoft Common Dialog Control 6.0
- Chọn OK

Sau khi làm như vậy, công cụ điều khiển hộp hội thoại chung sẽ được bổ sung vào hộp công cụ. Để tạo ra một hộp hội thoại chung ta chỉ việc nháy đúp chuột vào công cụ điều khiển này. Khi đó một hộp hội thoại chung sẽ xuất hiện trong biểu mẫu. Công việc của ta lúc này là xác định các giá trị cho các thuộc tính của hộp hội thoại chung sao cho phù hợp với yêu cầu sử dụng.

12.2.3. Các thuộc tính cơ bản của hộp hội thoại chung

Hộp hội thoại chung có nhiều thuộc tính. Bảng sau đây liệt kê các thuộc tính chính của đối tượng này.

Bảng 12.2. Các thuộc tính cơ bản của hộp hội thoại chung

Thuộc tính	Ý nghĩa
DialogTitle	Tiêu đề hộp hội thoại chung
Filter	Lọc tệp
FileName	Tên tệp được chọn có cả đường dẫn
FileTitle	Tên tệp được chọn không có đường dẫn
InitDir	Thư mục lúc bắt đầu mở tệp
CancelError	Nếu là TRUE thì có thể bắt lỗi sinh ra trong thời gian thực hiện chương trình do người sử dụng chọn nút lệnh Cancel. Có thể bắt lỗi bằng cách dùng lệnh ON ERROR

12.2.4. Các phương thức của hộp hội thoại chung

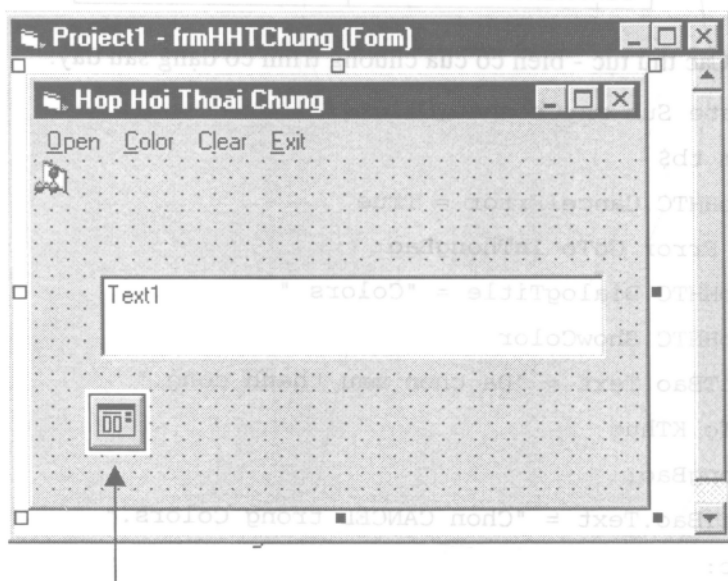
Hộp hội thoại chung (HHTC) có 6 phương thức, dùng để tạo ra 6 loại hộp hội thoại khác nhau. Các phương thức của hộp hội thoại chung được liệt kê trong bảng dưới đây:

Bảng 12.3. Các phương thức của hộp hội thoại chung

Phương thức	Ý nghĩa
ShowColor	Sử dụng chức năng Color của HHTC. Trong trường hợp này HHTC sẽ tạo ra một hộp hội thoại dùng để xác định màu sắc.
ShowFont	Sử dụng chức năng Font của HHTC. Trong trường hợp này HHTC sẽ tạo ra một hộp hội thoại dùng để xác Font.
ShowHelp	Sử dụng chức năng Help của HHTC. Trong trường hợp này HHTC sẽ tạo ra một hộp hội thoại dùng để xác định tệp chứa thông tin trợ giúp...
ShowOpen	Sử dụng chức năng Open của HHTC. Trong trường hợp này HHTC sẽ tạo ra một hộp hội thoại dùng để mở tệp. Trong hộp hội thoại này có hộp xác định tên tệp, hộp xác định đường dẫn, hộp xác định loại tệp có thể được mở, hộp liệt kê các tệp có trong thư mục đã xác định.
ShowPrinter	Sử dụng chức năng Printer của HHTC. Trong trường hợp này HHTC sẽ tạo ra một hộp hội thoại dùng để xác định máy in.
ShowSave	Sử dụng chức năng Save As của HHTC. Trong trường hợp này HHTC sẽ tạo ra một hộp dùng để lưu tệp lên đĩa từ với tên mới. Trong hộp hội thoại này có hộp để xác định tên tệp, hộp xác định đường dẫn, hộp liệt kê các tệp...

Sau đây là một ví dụ về chương trình có sử dụng hộp hội thoại chung. Chương trình này có tên là **HopHTChung** (Hộp Hội Thoại Chung).

Màn hình giao diện của chương trình **HopHTChung** gồm có một biểu mẫu, một hộp văn bản, một hộp hội thoại chung và một thực đơn có 4 chức năng là Open, Color, Clear và Exit (Xem Hình 12.4)



Đối tượng hộp hội thoại chung

Hình 12.4. Màn hình giao diện của chương trình "HopHTChung"

Hộp hội thoại chung của chương trình này có các thuộc tính sau:

**Bảng 12.4. Các thuộc tính của HHTC của chương trình
"HopHTChung"**

Thuộc tính	Giá trị
Name	CdbHHTC
FileName	C:\baitapct*.cpp
InitDir	C:\baitapct
CancelError	True

Các thủ tục - biến cố của chương trình có dạng sau đây:

```
Private Sub mnuChonMau_Click()
    Dim tb$
    CdbHHTC.CancelError = True
    On Error GoTo InThongBao
    CdbHHTC.DialogTitle = "Colors "
    CdbHHTC.ShowColor
    txtTBao.Text = "Da chon mau thanh cong."
    GoTo KThuc
InThongBao:
    txtTBao.Text = "Chon CANCEL trong Colors."
KThuc:
    Print "Ket Thuc chon mau."
End Sub

Private Sub mnuMoTep_Click()
    CdbHHTC.CancelError = True
    On Error GoTo InThongBao
    CdbHHTC.DialogTitle = "File Open "
    CdbHHTC.ShowOpen
```

```

txtTBao.Text = "Da mo tep " & CdbHHTC.FileName
GoTo KThuc
InThongBao:
    txtTBao.Text = "Chon CANCEL trong Open File."
KThuc: Print "Ket Thuc Mo Tep."
End Sub

Private Sub mnuXoa_Click()
    Cls
End Sub

Private Sub mnuKetThuc_Click()
    Beep
End
End Sub

```

Các chức năng của thực đơn có các thuộc tính sau đây:

**Bảng 12.5. Các chức năng của thực đơn
của chương trình "HopHTChung"**

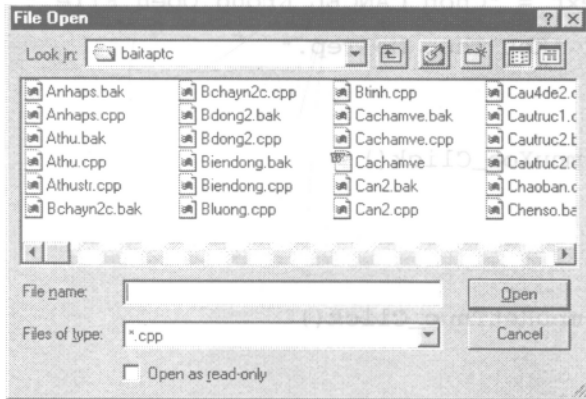
Tiêu đề của chức năng	Tên chức năng
&Open	mnuMoTep
&Color	mnuChonMau
C&lear	mnuXoa
&Exit	mnuKetThuc

Giải thích chương trình

Trong thời gian thực hiện chương trình, nếu người sử dụng nháy chuột vào chức năng Open, thì thủ tục - biến cố mnuMoTep_Click() sẽ được thực hiện. Trong thủ tục - biến cố này có câu lệnh:

CdbHHTC.ShowOpen

Nhờ có câu lệnh này mà hộp hội thoại chung sẽ tạo ra một cửa sổ hội thoại mở tệp như trong Hình 12.5.



Hình 12.5. Màn hình của CT “HopHTChung” khi dùng chức năng Open

Lúc này nếu ta chọn tên một tệp rồi nhấn chuột vào nút lệnh “Open” trong cửa sổ mở tệp, thì trong hộp văn bản của cửa sổ này sẽ xuất hiện dòng ký tự “Da mo tep ...”. Còn nếu ta chọn nút lệnh “Cancel” thì bẫy lỗi On Error GoTo InThongBao sẽ sập. Điều đó dẫn đến việc câu lệnh sau đây được thực hiện:

```
txtTBao.Text = "Chon CANCEL trong Open File."
```

Kết quả là trong hộp văn bản của cửa sổ mở tệp sẽ xuất hiện dòng ký tự sau đây:

```
"Chon CANCEL trong Open File."
```

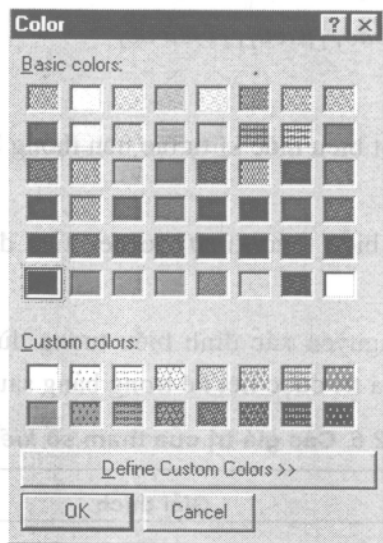
Sau đó máy in trên màn hình dòng ký tự:

```
"Ket Thuc Mo Tep."
```

Và thủ tục - biến cố này kết thúc.

Quá trình sử dụng chức năng “Color” cũng xảy ra tương tự. Chỉ

có điều trong trường hợp này cửa sổ hội thoại có dạng như trong Hình 12.6 dưới đây:



Hình 12.6. Màn hình của CT “HopHTChung” khi dùng chức năng Clear

12.3. LỆNH MSGBOX

Trong thời gian thực hiện chương trình thường có nhu cầu gửi cho người sử dụng một thông báo nào đấy hoặc cảnh báo cho người sử dụng về một điều gì đó. Ta có thể làm việc đó bằng cách in trực tiếp thông báo đó trên màn hình hoặc trong hộp văn bản. Nhưng làm như vậy có điều bất tiện là khó gây được sự chú ý của người sử dụng và làm thay đổi nội dung của màn hình hoặc hộp văn bản. Cách tốt nhất để làm việc này là dùng lệnh MsgBox.

Lệnh MsgBox tạo ra trên màn hình một cửa sổ nhỏ mà ta gọi là hộp thông báo. Hộp này chứa một thông báo mà chương trình muốn

gửi cho người sử dụng và một nút lệnh OK. Khi người sử dụng đã xem xong thông báo và nhấp chuột vào nút lệnh OK, hộp thông báo này sẽ biến mất. Lệnh này có qui cách sau đây:

MsgBox Thông_báo, [Kiểu], [Tiêu_đề]

Ở đây:

Thông_báo là một biểu thức kí tự tạo nên thông báo được gửi cho người sử dụng;

Tiêu_đề là một biểu thức kí tự tạo nên tiêu đề của hộp thông báo;

Kiểu là một số nguyên xác định biểu tượng dùng để cảnh báo. Tham số *kiểu* có các giá trị được liệt kê trong bảng sau đây:

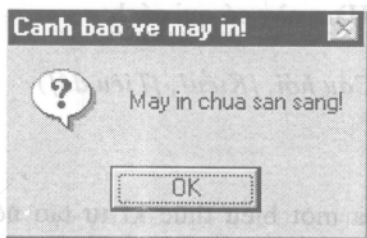
Bảng 12.6. Các giá trị của tham số kiểu

Giá trị	Giải thích
16	Hiện biểu tượng STOP
32	Hiện biểu tượng dấu hỏi (?)
48	Hiện biểu tượng dấu chấm than (!)
64	Hiện biểu tượng chữ i (information)
4096	Sử dụng hệ thống MODAL

Ví dụ có câu lệnh sau đây:

MsgBox "May in chua san sang!", 32, "Canh bao ve may in!"

Câu lệnh này làm xuất hiện trên màn hình hộp thông báo về máy in như trong hình dưới đây (Hình 12.7.). Ta thấy rằng trong hộp thông báo có nút lệnh OK. Như đã nói ở trên, khi người sử dụng nhấp chuột vào nút lệnh OK này, hộp thông báo sẽ biến khỏi màn hình.



Hình 12.7. Hộp thông báo về máy in

Chú ý: kiểu và tiêu đề là tùy chọn. Nếu kiểu và tiêu đề không được chỉ ra thì chúng sẽ không xuất hiện trong thông báo.

12.4. HÀM MSGBOX()

Một tình huống tương tự là trong thời gian thực hiện chương trình thường xuất hiện những tình trạng khó xử. Ví dụ người sử dụng muốn lưu dữ liệu vào một tệp mới mà tệp này đã tồn tại trên đĩa từ. Vậy có thứ tự công việc không? Trong trường hợp này cần gửi cho người sử dụng một câu hỏi và nhận câu trả lời từ người sử dụng. Ta hoàn toàn có thể làm công việc này bằng cách viết các câu lệnh thích hợp. Tuy nhiên, cách tốt nhất để thực hiện việc này là dùng hàm MsgBox().

Hàm MsgBox() này tạo ra trên màn hình một hộp thông báo chứa một câu hỏi và một số nút lệnh để người sử dụng trả lời câu hỏi này. Khi người sử dụng đã chọn xong câu trả lời (bằng cách nháy chuột vào một trong các nút lệnh có trong hộp thông báo), hộp thông báo này sẽ biến mất.

Hàm MsgBox() luôn luôn trả lại một giá trị nguyên. Giá trị này sẽ được gán cho một biến nguyên. Biến này được sử dụng để điều khiển chương trình thực hiện công việc phù hợp với câu trả lời của

người sử dụng. Hàm này có qui cách:

MsgBox(*Câu hỏi*, [*Kiểu*], [*Tiêu đề*])

Ở đây:

Câu hỏi là một biểu thức kí tự tạo nên câu hỏi được gửi cho người sử dụng;

Tiêu đề là một biểu thức kí tự tạo nên tiêu đề của hộp thông báo;

Kiểu là một giá trị xác định các nút lệnh sẽ xuất hiện trong hộp hội thoại do hàm MsgBox() tạo ra. Các giá trị của tham số *kiểu* được liệt kê trong bảng sau đây:

Bảng 12.7. Các giá trị của tham số *Kiểu* của hàm MsgBox()

Giá trị số	Giá trị chữ	Giải thích
0	vbOKOnly	Hiện OK
1	vbOKCancel	Hiện OK, Cancel
2	vbAbortRetryIgnore	Hiện Abort, Retry, Ignore
3	vbYesNoCancel	Hiện Yes, No, Cancel
4	vbYesNo	Hiện Yes, No
5	vbRetryCancel	Hiện Retry, Cancel

Như đã nói trên đây, hàm MsgBox() luôn luôn trả lại một giá trị là một số nguyên. Các giá trị do hàm này trả lại được liệt kê trong bảng dưới đây:

Bảng 12. 8. Các giá trị do hàm MsgBox() trả lại:

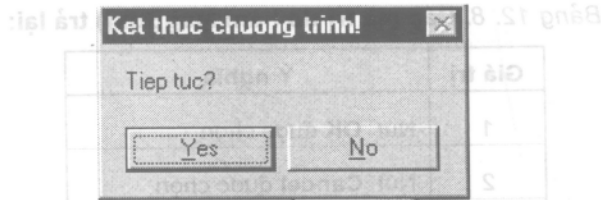
Giá trị	Ý nghĩa
1	Nút OK được chọn
2	Nút Cancel được chọn
3	Nút Abort được chọn
4	Nút Retry được chọn
5	Nút Ignore được chọn
6	Nút Yes được chọn
7	Nút No được chọn

Sau đây là một ví dụ về sử dụng hàm MsgBox() và lệnh MsgBox.

Giả sử có thủ tục - biến cố cmdchayct_Click() như sau:

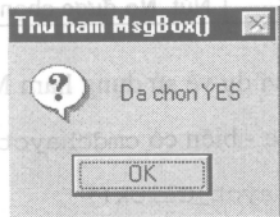
```
Private Sub cmdchayct_Click()
    Dim Traloi%
    Traloi = MsgBox("Tiep tục?", vbYesNo, "Ket thuc
chuong trinh!")
    If Traloi = 6 Then
        MsgBox "Da chon YES", 32, "Thu ham MsgBox()"
    Else
        MsgBox "Da chon NO ", 32, "Thu ham MsgBox()"
    End If
End Sub
```

Trên màn hình xuất hiện hộp thông báo kết thúc chương trình có dạng như trong hình sau đây (Hình 12.8)



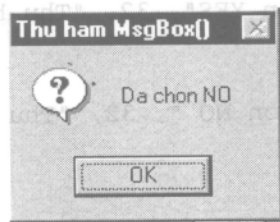
Hình 12.8. Hộp thông báo về kết thúc chương trình

Trong hộp thông báo này có 2 nút lệnh là “Yes” và “No”. Nếu người sử dụng chọn nút lệnh “Yes” thì trên màn hình xuất hiện hộp thông báo Thử hàm MsgBox() có dạng như trong hình sau đây (Hình 12.9.):



Hình 12.9. Hộp thông báo thử hàm MsgBox() khi chọn YES

Còn nếu người sử dụng chọn nút lệnh “No” thì trên màn hình xuất hiện hộp thông báo Thử hàm MsgBox() có dạng như trong hình sau đây (Hình 12.10.):



Hình 12.10. Hộp thông báo thử hàm MsgBox() khi chọn NO

12.5. HỘP DANH SÁCH

Trong mục 6.4.3 của chương 6 ta đã nghiên cứu chương trình **DoiTien**. Như độc giả đã thấy, trong thời gian thực hiện chương trình người sử dụng có thể chọn một trong 4 loại ngoại tệ để đổi là Bảng Anh, Đô la Mỹ, Mác Đức và Rup Nga thông qua việc chọn một trong các nút tùy chọn. Có 3 trường hợp được đặt ra cho chương trình này là:

- Nếu số ngoại tệ tăng lên đến hàng chục, thậm chí hàng trăm thì làm thế nào?
- Nếu số ngoại tệ không biết trước thì làm thế nào?
- Nếu số ngoại tệ thay đổi, lúc nhiều lúc ít thì làm thế nào?

Đối với trường hợp thứ nhất thì không đủ chỗ trên màn hình cho các nút tùy chọn. Đối với 2 trường hợp sau thì cách lập trình này không giải quyết được.

Visual Basic có một đối tượng gọi là hộp danh sách. Sử dụng hộp danh sách để lập chương trình **DoiTien** có thể giải quyết được cả ba trường hợp nêu trên.

12.5.1. Hộp danh sách là gì?

Hộp danh sách (ListBox) là một đối tượng chứa một danh sách. Mỗi phần tử của danh sách này được gọi là một mục (dữ liệu). Trong thời gian thực hiện chương trình người sử dụng có thể chọn một hoặc một số mục của hộp danh sách để giải quyết bài toán của mình. Theo ngầm định các mục dữ liệu của hộp danh sách là các chuỗi kí tự.

Các mục dữ liệu có trong hộp danh sách do người sử dụng xác định trong thời gian thiết kế chương trình hoặc trong thời gian thực hiện chương trình. Các mục dữ liệu được xác định trong thời gian thiết kế chương trình sẽ xuất hiện trong hộp danh sách trên màn hình của biểu mẫu. Nếu trong thời gian thiết kế chương trình lập trình viên

không xác định một mục dữ liệu nào cả thì hộp danh sách có dạng một hình chữ nhật để trống, giống như một hộp văn bản.

12.5.2. Các thuộc tính chính của hộp danh sách

Hộp danh sách có nhiều thuộc tính. Nhiều thuộc tính của hộp danh sách chúng ta đã biết. Sau đây là các thuộc tính chính của hộp danh sách:

Bảng 12.9. Các thuộc tính chính của hộp danh sách

Thuộc tính	Ý nghĩa
Columns	Xác định số cột của các mục dữ liệu xuất hiện trong hộp danh sách. Thuộc tính này có các giá trị sau đây: 0 (ngầm định): Hộp danh sách có 1 cột dữ liệu và có thanh cuộn dọc ở phía phải để cuộn dữ liệu lên xuống theo chiều dọc. 1,2,3 ... : Hộp danh sách có 1,2,3... cột dữ liệu và có thanh cuộn ngang để cuộn dữ liệu theo chiều ngang.
List	Trong thời gian thiết kế chương trình thuộc tính này cho phép người sử dụng xác định các mục dữ liệu của hộp danh sách. Các mục dữ liệu được xác định bằng thuộc tính này sẽ xuất hiện trong hộp danh sách trong thời gian thiết kế chương trình. Trong thời gian thực hiện chương trình thuộc tính này cho giá trị một mục dữ liệu ở vị trí xác định trong hộp danh sách. Ví dụ IstH1.List(2) cho giá trị của mục dữ liệu ở vị trí thứ 3 trong hộp danh sách có tên là IstH1 (các mục dữ liệu được đánh số từ 0).
ListIndex (*)	Cho chỉ số (tức là số thứ tự) của mục dữ liệu đã được chọn. Nếu người sử dụng đã chọn nhiều mục dữ liệu, thì thuộc tính này chứa chỉ số của mục dữ liệu được chọn sau cùng.
MultiSelect	Thuộc tính này quy định cách chọn mục dữ liệu trong hộp danh sách. Thuộc tính này có các giá trị sau đây:

Bảng 12.9 (tiếp)

Thuộc tính	Ý nghĩa
	0 (ngầm định): Cho phép người sử dụng chọn 1 mục dữ liệu bằng cách nháy chuột vào mục dữ liệu đó. 1: Cho phép đa chọn đơn giản. Người sử dụng nháy chuột vào mục dữ liệu nào thì mục dữ liệu đó sẽ được chọn. 2: Cho phép đa chọn mở rộng. Người sử dụng chọn một mục dữ liệu, ấn phím Shift và chọn mục dữ liệu thứ hai. Tất cả các mục dữ liệu nằm giữa hai mục dữ liệu này sẽ được chọn.
Selected (*)	Đây là thuộc tính kiểu logic. Nếu một mục dữ liệu đã được chọn thì thuộc tính này của mục dữ liệu đó sẽ có giá trị True. Ví dụ: Nếu mục dữ liệu 5 đã được chọn thì IstH1.Selected(5) sẽ có giá trị True. Thuộc tính này được sử dụng để nhận biết những mục dữ liệu được chọn trong thời gian thực hiện chương trình.
Sorted	Đây là thuộc tính kiểu logic. Giá trị True của thuộc tính này làm cho các mục dữ liệu trong hộp danh sách được sắp xếp theo thứ tự ABC.
Style	Đây là thuộc tính kiểu logic. Giá trị True của thuộc tính này làm cho các hộp kiểm xuất hiện bên trái các mục dữ liệu của hộp danh sách. Khi một mục dữ liệu được chọn thì hộp kiểm bên trái nó đã đánh dấu chọn (✓).
Text (*)	Thuộc tính này cho giá trị của mục dữ liệu đã được chọn. Giá trị này là một chuỗi kí tự.

Chú ý: Các thuộc tính có dấu (*) là những thuộc tính chỉ có thể được sử dụng trong thời gian thực hiện chương trình.

Toàn bộ các mục dữ liệu của hộp danh sách tạo thành một mảng một chiều. Như ta đã biết, muốn xác định một phần tử trong mảng ta phải dùng chỉ số. Tương tự như vậy, để xác định một mục dữ liệu trong

hộp danh sách ta phải sử dụng cách viết sau đây:

`Tên_hộp_danh_sách.List(chỉ_số)`

Chỉ số là một biến nguyên, có giá trị từ 0 đến $n-1$ (n là số mục dữ liệu của hộp danh sách). Số mục dữ liệu của hộp danh sách đã xác định bằng phương thức `ListCount` (Xem mục tiếp theo).

Ví dụ:

`LstH1.List(2)` : Mục dữ liệu thứ 3 của hộp danh sách `LstH1`.

`LstH1.List(i)` : Mục dữ liệu thứ i của hộp danh sách `LstH1`.

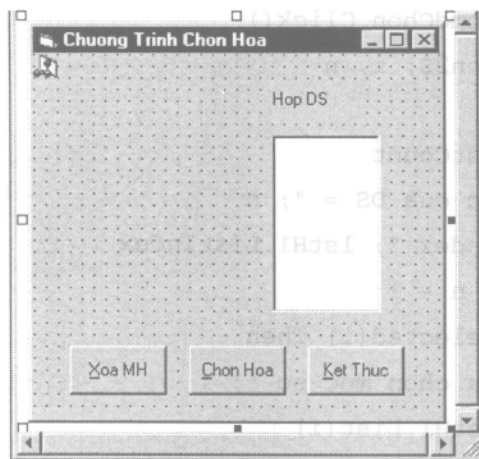
12.5.3. Các phương thức của hộp danh sách

Hộp danh sách có các phương thức sau đây:

Bảng 12.10. Các phương thức của hộp danh sách

Phương thức	Ý nghĩa
AddItem	Thêm một mục dữ liệu vào cuối hộp danh sách. Mục dữ liệu vừa được thêm này luôn luôn là mục dữ liệu cuối cùng của hộp danh sách.
Clear	Xoá tất cả các mục dữ liệu của hộp danh sách.
ListCount	Đếm số mục dữ liệu có trong hộp danh sách.
RemoveItem	Xoá một mục dữ liệu xác định của hộp danh sách. Ví dụ câu lệnh <code>LstH1.RemoveItem(2)</code> sẽ xoá mục dữ liệu thứ 3 của hộp danh sách có tên là <code>LstH1</code> .

Sau đây là một ví dụ về sử dụng hộp danh sách. Hộp danh sách trong ví dụ này chứa danh sách tên các loại hoa. Chương trình này cho phép người sử dụng chọn một hoặc một số loại hoa trong danh sách đó. Khi đã chọn xong thì in các tên hoa đã được chọn trên màn hình. Vì vậy chương trình trong ví dụ này có tên là **ChonHoa** (Chọn Hoa). Màn hình giao diện của chương trình này có dạng sau đây:



Hình 12.11. Màn hình giao diện của chương trình ChonHoa

Ta thấy rằng màn hình giao diện của chương trình này có một biểu mẫu, 3 nút lệnh lần lượt có tiêu đề là “Xoa MH”, “Chon Hoa”, “Ket Thuc” và một hộp danh sách (LstH1). Hộp danh sách này có các thuộc tính sau đây:

Bảng 12.11. Các thuộc tính của hộp danh sách
LstH1 của CT ChonHoa

Thuộc tính	Giá trị
Name	LstH1
Columns	0
List	(List)
MultiSelect	1 - Simple
Sorted	True
Style	False

Các thủ tục - biến cố của chương trình này như sau:

```

Private Sub cmdChon_Click()
Dim ten1$, ten2$, i, n
Print: Print
n = lstH1.ListCount
Print "So muc cua DS = "; n
Print "ListIndex "; lstH1.ListIndex
For i = 0 To n - 1
    If lstH1.Selected(i) Then
        Print "Da chon muc so "; i
        ten1 = lstH1.List(i)
        Print "Noi dung = "; ten1
        Print "Muc "; lstH1.List(i); " is Selected"
        Print
    End If
Next
End Sub

Private Sub cmdKetThuc_Click()
End
End Sub

Private Sub cmdXoa_Click()
Cls
End Sub

Private Sub Form_Load()
lstH1.AddItem "Lan"
lstH1.AddItem "Hue"
lstH1.AddItem "Que"

```

```

lstH1.AddItem "Sen"
lstH1.AddItem "Hong"
lstH1.AddItem "Phuong"
lstH1.AddItem "Dao"
lstH1.AddItem "Cuc"
lstH1.AddItem "Mai"
lstH1.AddItem "Ly"
lstH1.AddItem "Mo"
lstH1.AddItem "Dai"
lstH1.AddItem "Le"
End Sub

```

Giải thích chương trình

Trong chương trình này có thủ tục - biến cố `Form_Load()`. Thủ tục này sẽ được thực hiện trong thời gian tải biểu mẫu vào bộ nhớ, trước khi thực hiện chương trình. Trong thủ tục - biến cố này có nhiều câu lệnh gán. Mỗi câu lệnh gán thêm vào hộp danh sách một mục dữ liệu, là tên một loại hoa. Như vậy chức năng của thủ tục - biến cố này là tạo ra danh sách tên các loại hoa trong hộp danh sách `lstH1`. Trong thời gian thực hiện chương trình ta luôn luôn thấy tên các loại hoa trong hộp danh sách này.

Trong thời gian thực hiện chương trình, người sử dụng có thể chọn một hoặc một số mục dữ liệu trong hộp danh sách. Các mục dữ liệu được chọn sẽ được bôi đậm. Lúc này, nếu sử dụng nút lệnh "Chọn Hoa" thì thủ tục - biến cố `cmdChon_Click()` sẽ được thực hiện và trên màn hình xuất hiện kết quả như trong hình dưới đây (Hình 12.12)



Hình 12.12. Màn hình của chương trình ChonHoa khi chạy chương trình

Trong thủ tục - biến cố `cmdChon_Click()` có câu lệnh `ten1 = lstH1.List(i)`. Câu lệnh này được thực hiện khi mục dữ liệu thứ `i` được chọn. Câu lệnh này gán cho biến `ten1` giá trị của mục dữ liệu đó (mục dữ liệu được chọn). Giá trị của biến `ten1` được in trên màn hình.

Ta cũng có thể gán giá trị của mục dữ liệu được chọn cho biến `ten1` bằng các câu lệnh gán sau đây:

```
Ten1 = lstH1.Text
```

```
Ten1 = lstH1.List(lstH1.ListIndex)
```

Nếu người sử dụng đã chọn nhiều mục dữ liệu, thì hai câu lệnh này gán cho `ten1` giá trị của mục dữ liệu được chọn cuối cùng.

Visual Basic lấy thông tin từ DOS để đưa vào hộp danh sách ổ đĩa. Người sử dụng không cần (và không thể) nhập các thông tin vào hộp danh sách này.

12.6. HỘP DANH SÁCH Ổ ĐĨA

Hộp danh sách ổ đĩa là một hộp danh sách kiểu đồ xuống, ngầm định là tên ổ đĩa chủ xuất hiện ở đầu danh sách này. Trong thời gian thực hiện chương trình người sử dụng có thể dễ dàng chọn một ổ đĩa trong hộp danh sách ổ đĩa. Khi người sử dụng nháy chuột vào mũi tên xuống bên phải hộp danh sách ổ đĩa thì toàn bộ các ổ đĩa của máy tính sẽ xuất hiện trong hộp này. Nếu người sử dụng chọn một ổ đĩa thì tên ổ đĩa này sẽ xuất hiện trên đầu hộp danh sách ổ đĩa.

Có thể dùng câu lệnh sau đây để xác định ổ đĩa được chọn.

```
DrvH.Drive = "c:\"
```

Trong câu lệnh này **DrvH** là tên hộp danh sách ổ đĩa. Câu lệnh này gán cho thuộc tính Drive của hộp danh sách ổ đĩa drvH tên ổ đĩa "c:\". Câu lệnh này biến ổ đĩa c: thành ổ đĩa chủ.

Việc người sử dụng chọn một ổ đĩa trong danh sách ổ đĩa không làm thay đổi ổ đĩa chủ. Để thay đổi ổ đĩa chủ cần phải thực hiện câu lệnh sau đây:

```
ChDrive DrvH.Drive
```

Câu lệnh này biến ổ đĩa xác định bởi thuộc tính Drive của hộp danh sách ổ đĩa DrvH thành ổ đĩa chủ.

12.7. HỘP DANH SÁCH THƯ MỤC

Hộp danh sách thư mục là một hộp danh sách chứa cấu trúc thư mục của ổ đĩa chủ, bắt đầu từ thư mục gốc. Cấu trúc của thư mục được thể hiện dưới dạng hình cây, phân thành các nhánh. Thư mục hiện thời được tô đậm. Thư mục cha của thư mục hiện thời hơi thụt về phía trái và cứ thế các thư mục ở các cấp cao hơn càng thụt về phía trái nhiều hơn cho đến tận thư mục gốc.

Mỗi thư mục trong hộp danh sách thư mục có một số hiệu riêng, gọi là chỉ số (ListIndex). Các hộp hội thoại chung không có khả năng này. Thư mục xác định bởi thuộc tính Path của hộp danh sách thư mục luôn luôn có chỉ số là -1, thư mục ngay trên nó có chỉ số là -2, sau đó là -3 ...cho đến thư mục gốc. Thư mục con đầu tiên của thư mục được xác định bởi thuộc tính Path của hộp danh sách thư mục có chỉ số là 0. Nếu có nhiều thư mục con ở mức này thì thư mục tiếp theo có chỉ số là 1...

Dùng thuộc tính Path của hộp danh sách thư mục để xem hoặc xác định thư mục hiện thời. Như đã nói ở trên, thư mục này có chỉ số là -1. Ví dụ có câu lệnh:

```
DirH.Path= "c:\Baitapvb"
```

Thì thư mục "Baitapvb" trở thành thư mục hiện thời. Trong câu lệnh này **DirH** là tên hộp danh sách thư mục.

Tương tự, ta có thể gán cho thuộc tính Path của hộp danh sách thư mục giá trị thuộc tính Drive của hộp danh sách ổ đĩa bằng câu lệnh sau đây:

```
DirH.Path = DrvH.Drive
```

Khi đó trong hộp danh sách thư mục sẽ xuất hiện tất cả các thư mục có trong ổ đĩa. Theo ngầm định, trong hộp danh sách thư mục có tất cả các thư mục của ổ đĩa chủ. Thư mục hiện thời được chọn (được chiếu sáng hoặc bôi đậm) và có chỉ số là -1.

Để biến thư mục xác định bởi thuộc tính Path của hộp danh sách thư mục thành thư mục hiện thời cần phải dùng câu lệnh:

```
chDir DirH.Path
```

Chú ý: Chỉ có thể sử dụng thuộc tính Path của hộp danh sách thư mục trong thời gian thực hiện chương trình. Trong thời gian thiết kế chương trình không sử dụng được thuộc tính này.

Khi người sử dụng nháy chuột vào một thư mục trong hộp danh sách thư mục thì thư mục này được chọn. Khi người sử dụng nháy đúp chuột vào một thư mục trong hộp danh sách thư mục thì thư mục này được gán cho thuộc tính Path của hộp danh sách thư mục, chỉ số của thư mục này được đổi thành -1. Hộp danh sách thư mục sẽ được vẽ lại để có thể hiện ra các thư mục con của thư mục hiện thời.

Cũng như đối với hộp danh sách ổ đĩa, Visual Basic lấy thông tin từ DOS để đưa vào hộp danh sách thư mục. Người sử dụng không cần (và không thể) nhập các thông tin vào hộp danh sách này.

12.8. HỘP DANH SÁCH TẬP

Trong phần 12.2 ta đã nghiên cứu hộp hội thoại mở tệp. Trong hộp hội thoại đó ta đã thấy danh sách tất cả các tệp có trong thư mục hiện thời. Người sử dụng có thể dễ dàng chọn một trong các tên tệp đó để mở tệp.

Visual Basic còn có một đối tượng khác cũng cho phép ta xem danh sách các tệp có trong một thư mục xác định. Đó là hộp danh sách tệp.

Hộp danh sách tệp (List file Box) là một loại hộp danh sách trong đó có danh sách tất cả các tệp có trong một thư mục được xác định bởi thuộc tính path của hộp này. Ta cũng có thể làm xuất hiện tên các tệp có trong thư mục hiện thời của ổ đĩa chủ bằng câu lệnh:

```
FileH.Path = DirH.Path
```

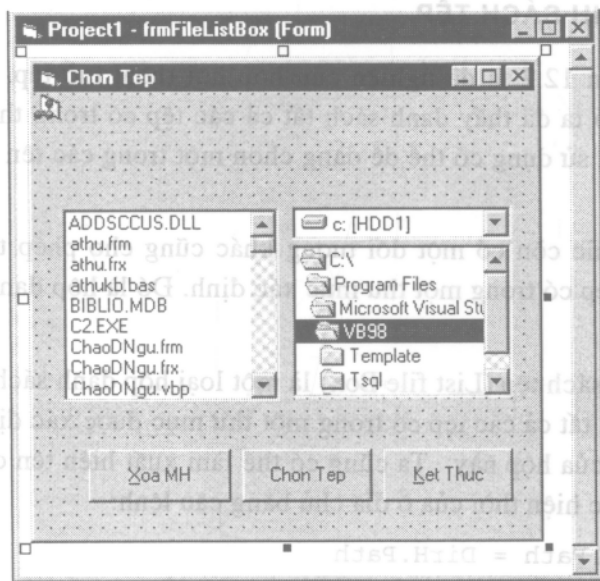
Trong câu lệnh này thì FileH là tên của hộp danh sách tệp. DirH là tên của hộp danh sách thư mục.

Ta có thể xem một loại tệp hoặc một số loại tệp nhờ sử dụng thuộc tính Pattern (mẫu) như sau:

```
FilH.Pattern = "*.frm; *.bas"
```

Cũng như đối với hộp danh sách ổ đĩa và hộp danh sách thư mục, Visual Basic lấy thông tin từ DOS để đưa vào hộp danh sách tên tệp. Người sử dụng không cần (và không thể) nhập các thông tin vào hộp danh sách này.

Sau đây là một ví dụ về sử dụng kết hợp ba loại hộp danh sách vừa trình bày trên đây. Chương trình trong ví dụ này có tên là ChonTep (Chọn Tệp). Chương trình này sử dụng một hộp danh sách ổ đĩa, một hộp danh sách thư mục và một hộp danh sách tệp để người sử dụng có thể chọn một tệp bất kỳ trong hệ thống máy tính của mình. Màn hình giao diện của chương trình này có dạng sau đây (Hình 12.13):



Hình 12.13. Màn hình giao diện của chương trình ChonTep

Ta thấy rằng màn hình giao diện của chương trình này có một biểu mẫu, 3 nút lệnh lần lượt có tiêu đề là “Xoa MH”, “Chon Tep”, “Ket Thuc” và một hộp danh sách ổ đĩa (DrvH), một hộp danh sách

thư mục (DirH), một hộp danh sách tệp (fileH). Các thủ tục - biến cố của chương trình này như sau:

```
Private Sub cmdChon_Click()  
Print  
Print "O dia chu = "; DrvH.Drive  
Print "Duong dan = "; DirH.Path  
TenFile = FileH.List(ListIndex) ' Chay qua tot  
TenFile = FileH.FileName 'Chay qua tot  
Print "Ten tep = "; TenFile  
Print "Tep da chon la: "; FileH.Path; "\";  
FileH.FileName  
End Sub  
  
Private Sub cmdKetThuc_Click()  
End  
End Sub  
  
Private Sub cmdXoa_Click()  
Cls  
End Sub  
  
Private Sub DirH_Change()  
TenPath = DirH.Path  
FileH.Path = DirH.Path  
Print FileH.Path  
End Sub  
  
Private Sub DrvH_Change()  
DirH.Path = DrvH.Drive
```

```

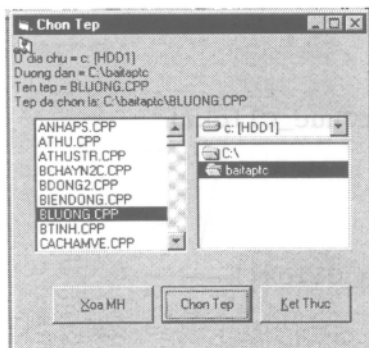
End Sub

Private Sub FileH_Click()
FileH.Pattern = "*.cpp ;*.vbp; *.bas "
End Sub

```

Giải thích chương trình

Trong thời gian thực hiện chương trình nội dung của 3 hộp danh sách sẽ xuất hiện trên màn hình. Người sử dụng có thể dùng chuột để chọn một ổ đĩa, một thư mục, một tên tệp bất kỳ. Sau khi làm như vậy, nếu nháy chuột vào nút lệnh “Chon Tep” thì máy in trên màn hình kết quả có dạng sau đây (Hình 12.14):



Hình 12.14. Màn hình của chương trình ChonTep khi chạy chương trình

Ta thấy rằng dòng đầu tiên là tên ổ đĩa đã được chọn. Dòng thứ hai là tên thư mục đã được chọn. Dòng thứ ba là tên tệp đã được chọn. Dòng cuối cùng là dạng đầy đủ của tên tệp đã được chọn, gồm tên ổ đĩa, tên đường dẫn và tên tệp đã chọn.

Nếu sử dụng nút lệnh “Xoa MH” thì máy xóa hết các thông tin có trên màn hình biểu mẫu. Nếu sử dụng nút lệnh “Ket Thuc” thì máy sẽ kết thúc chương trình.

12.9. ĐỐI TƯỢNG TIMER

Đối tượng Timer là đối tượng tạo ra biến cố có tên là Timer sau một khoảng thời gian đều đặn. Đối tượng này độc lập với người sử dụng. Trong thời gian thực hiện chương trình người sử dụng không thể thấy đối tượng thời gian trên màn hình. Có thể dùng đối tượng Timer để thực hiện một công việc đều đặn theo những khoảng thời gian bằng nhau. Kiểu ứng dụng điển hình của đối tượng Timer là kiểm tra xem đã đến lúc thực hiện một công việc nào đó hay chưa. Đối tượng Timer có các thuộc tính sau đây:

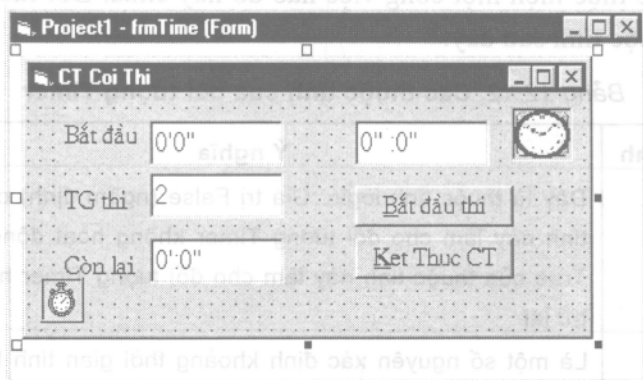
Bảng 12.12. Các thuộc tính của đối tượng Timer

Thuộc tính	Ý nghĩa
Enabled	Đây là thuộc tính logic. Giá trị False (ngầm định) của thuộc tính này làm cho đối tượng Timer không hoạt động. Giá trị True của thuộc tính này làm cho đối tượng Timer hoạt động trở lại.
Interval	Là một số nguyên xác định khoảng thời gian tính bằng mili giây để đối tượng Timer tạo ra một biến cố Timer. Giá trị của thuộc tính này nằm trong khoảng từ 1 đến 64767.
Left	Xác định khoảng cách trái của đối tượng Timer.
Name	Xác định tên của đối tượng timer. Tiền tố của tên đối tượng này là tmr.
Top	Xác định khoảng cách trên của đối tượng Timer.

Visual Basic có một thủ tục - biến cố đặc biệt, có tên là Timer. Thủ tục - biến cố này sẽ được thực hiện mỗi khi xảy ra thủ tục - biến cố Timer do đối tượng Timer tạo ra. Như vậy, cứ sau một khoảng thời gian được xác định bởi thuộc tính Interval, thủ tục - biến cố Timer được thực hiện 1 lần. Ta sẽ sử dụng thủ tục - biến cố này để viết chương trình giải bài toán của mình.

Xin đọc giả lưu ý rằng từ tiếng Anh Timer (tạm dịch là bộ định thời gian) được dùng làm tên cho đối tượng Timer, cho biến cố Timer và cho thủ tục - biến cố Timer. Vì gọi Timer là bộ định thời gian thì không chính xác và hơi dài, nên tác giả dùng luôn từ tiếng Anh Timer.

Sau đây là một ví dụ về sử dụng đối tượng Timer. Chương trình trong ví dụ này có tên là CoiThi. Chương trình này sử dụng một đối tượng Timer để xác định thời điểm hết giờ thi. Màn hình giao diện của chương trình này có dạng sau đây (Hình 12.15):



Hình 12.15. Màn hình giao diện của chương trình CoiThi

Ta thấy rằng màn hình giao diện này gồm một biểu mẫu, 3 nhãn, 4 hộp văn bản, một đối tượng Timer và một hộp ảnh.

Ba nhãn dùng để giải thích nội dung của hộp văn bản đứng phía phải. Ba hộp văn bản đầu tiên lần lượt chứa thời điểm bắt đầu thi, thời gian thi và thời gian còn lại. Hộp văn bản thứ tư (bên cạnh ảnh đồng hồ) chỉ thời gian theo đồng hồ.

Đối tượng Timer có ở góc trái - dưới của màn hình giao diện. Đối tượng Timer nằm ở đâu trên màn hình giao diện không quan trọng. Vì trong thời gian thực hiện chương trình người sử dụng không nhìn thấy

đối tượng này. Đối tượng Timer trong chương trình này có các thuộc tính như sau:

Bảng 12.13. Các thuộc tính của đối tượng Timer của chương trình CõiThi

Thuộc tính	Giá trị
Enabled	False
Interval	1000
Name	TmrTimer

Chương trình này có 2 nút lệnh. Nút lệnh “Bắt đầu thi” khởi động quá trình thi. Quá trình sẽ kết thúc khi hết thời gian thi. Nút lệnh “Kết Thúc CT” sẽ kết thúc chương trình CõiThi.

Các thủ tục - biến cố của chương trình này như sau:

```
Option Explicit
Dim  tgian%
Private Sub CmdKThuc_Click()
End
End Sub
Private Sub cmdBDau_Click()
txtTGThi.Text = InputBox("TGian thi (phut) = ", "Xac
Dinh TG thi ", "1")
tgian = Val(txtTGThi.Text) * 60
txtTGThi.Text = txtTGThi.Text + " ' : 0""
TmrTimer.Enabled = True
txtBDau.Text = Time()
End Sub
```

```

Private Sub TmrTimer_Timer()
Dim Phut%, Giay%
TxtDongHo.Text = Time()
tgian = tgian - 1
Phut% = tgian \ 60
Giay% = tgian - 60 * Phut%
TxtCLai.Text = Str(Phut%) & " ' : " & Str(Giay%) & "'' "
If Phut = 0 And Giay = 0 Then
    MsgBox "Het gio thi!", 32, "Thong bao het gio"
    TmrTimer.Enabled = False
End If
End Sub

```

Giải thích chương trình

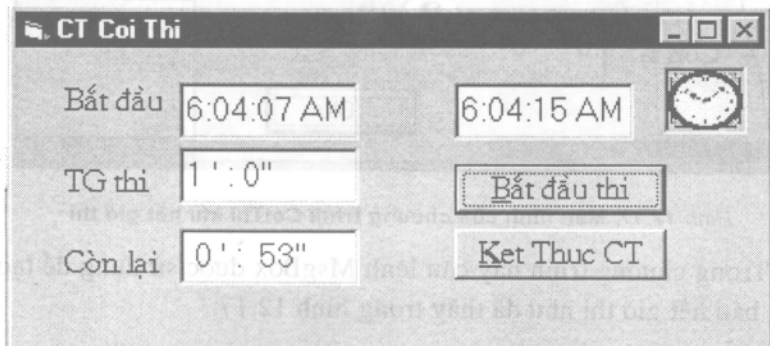
Khi bắt đầu thực hiện chương trình, trên màn hình sẽ xuất hiện màn hình giao diện của chương trình CoiThi. Vì thuộc tính Enabled của đối tượng Timer có giá trị False, nên lúc này đối tượng Timer không hoạt động.

Khi người sử dụng nháy chuột vào nút lệnh “Bắt đầu thi” thì thủ tục - biến cố cmdBDau_Click() được thực hiện. Máy sẽ yêu cầu người sử dụng nhập thời gian thi (tính bằng phút). Sau khi người sử dụng nhập thời gian thi, các công việc sau đây sẽ được thực hiện:

- Đổi thời gian thi sang giây.
- Cập nhật nội dung hộp văn bản chứa thời gian thi.
- Gán giá trị True cho đối tượng Timer TmrTimer.
- Đặt thời gian theo đồng hồ hệ thống vào hộp văn bản chứa thời

điểm bắt đầu thi.

Sau khi thực hiện các công việc đó thủ tục - biến cố này sẽ kết thúc và bắt đầu tính thời gian thi (xem Hình 12.16). Nội dung của hộp văn bản chứa thời gian còn lại và hộp văn bản chứa thời gian theo đồng hồ hệ thống không ngừng thay đổi.



Hình 12.16. Màn hình của chương trình CòiThi khi chạy chương trình

Cứ sau 1 giây (tức là 1000 mili giây) đối tượng Timer lại tạo ra một biến cố Timer. Khi đó thủ tục - biến cố `TmrTimer_Timer()` sẽ được thực hiện. Thủ tục - biến cố này thực hiện các công việc sau đây:

- Đặt thời gian theo đồng hồ hệ thống và hộp văn bản chứa thời gian theo đồng hồ.
- Trừ bớt thời gian còn lại đi 1 (giây).
- Cập nhật nội dung hộp văn bản chứa thời gian còn lại.
- Kiểm tra thời gian còn lại. Nếu thời gian này đã hết (Phut = 0 và Giay = 0) thì in thông báo kết thúc thi và gán giá trị False cho đối tượng `TmrTimer`. Đối tượng Timer sẽ ngừng làm việc. Quá trình thi kết thúc. Nội dung các hộp văn bản không thay đổi nữa (Xem hình 12.17.). Lúc này hộp văn bản chứa thời gian còn lại có dạng (0' : 0''), tức là 0 phút 0 giây.



Hình 12.17. Màn hình của chương trình CòiThi khi hết giờ thi

Trong chương trình này câu lệnh MsgBox được sử dụng để tạo ra thông báo hết giờ thi như đã thấy trong hình 12.17.

CÂU HỎI ÔN TẬP

1. Thực đơn là gì? Sử dụng thực đơn có những lợi ích gì?
2. Hãy trình bày cách thêm thực đơn vào màn hình giao diện.
3. Hộp hội thoại chung là gì? Hãy trình bày các thuộc tính và các phương thức của hộp hội thoại chung.
4. Hộp danh sách là gì? Hộp danh sách được sử dụng để làm gì? Hãy cho một ví dụ có sử dụng hộp danh sách.
5. Hãy nêu tác dụng của các hộp danh sách ổ đĩa, hộp danh sách thư mục, hộp danh sách tệp. Hãy cho một ví dụ về cách sử dụng các đối tượng này.
6. Hãy viết chương trình sử dụng đối tượng Timer để tạo ra một đồng hồ báo thức.

ĐỒ HOẠ TRONG VISUAL BASIC

Các hình ảnh, các hình vẽ có trong màn hình giao diện luôn luôn làm cho màn hình giao diện sinh động, đa dạng và gây ấn tượng hơn rất nhiều. Lập trình trong môi trường Visual Studio có nhiều điều kiện thuận lợi để đưa các hình ảnh và hình vẽ vào các giao diện của chương trình ứng dụng. Chương này nghiên cứu các đối tượng của Visual Basic dùng để đưa các hình ảnh và hình vẽ vào màn hình giao diện của chương trình. Các đối tượng của Visual Basic được nghiên cứu trong chương này là đối tượng ảnh (Image), đối tượng hộp hình (PictureBox), đối tượng đường thẳng (Line) và đối tượng hình vẽ (Shape). Các đối tượng này có thể được dùng để đưa hình ảnh vào biểu mẫu trong thời gian thiết kế chương trình cũng như trong thời gian thực hiện chương trình.

Trong hình 12.0. có các biểu tượng của các công cụ điều khiển đồ hoạ của Visual Basic.

13.1. ĐỐI TƯỢNG ẢNH

Đối tượng ảnh là một trong hai đối tượng cho phép đưa hình ảnh vào màn hình giao diện. Các hình ảnh xuất hiện trong đối tượng ảnh được lưu trữ trong các tệp. Thuộc tính Picture của đối tượng ảnh xác

định tên tệp chứa hình ảnh sẽ xuất hiện trong đối tượng ảnh. Khi đối tượng ảnh vừa mới được thêm vào biểu mẫu, nó rất mờ nhạt vì lúc này thuộc tính Picture chưa được xác định, nên chưa có ảnh trong đối tượng này. Hình ảnh chỉ xuất hiện trong đối tượng ảnh sau khi ta xác định tên tệp cho thuộc tính Picture.

13.1.1. Các loại tệp được sử dụng cho đối tượng ảnh

Các loại tệp đồ họa sau đây có thể được sử dụng cho đối tượng ảnh:

Bảng 13.1. Các loại tệp được sử dụng cho đối tượng ảnh

Phần mở rộng	Mô tả tệp
.bmp	Tệp hình ảnh ánh xạ bit của Windows
.cur	Tệp chứa con trỏ sinh động
.dib	Tệp hình ảnh ánh xạ bit cũ
.emf	Các siêu tệp của Windows nâng cấp
.gif	Các tệp có qui cách Graphic Interchange Format
.ico	Tệp chứa các biểu tượng
.jpg	Tệp chứa các hình ảnh nén
.wmf	Siêu tệp của Windows

13.1.2. Các thuộc tính chính của đối tượng ảnh

Đối tượng ảnh có gần 30 thuộc tính. Nhiều thuộc tính của đối tượng ảnh đã quen thuộc đối với chúng ta. Sau đây là các thuộc tính chính của đối tượng này.

Bảng 13.2. Các thuộc tính chính của đối tượng ảnh

Thuộc tính	Ý nghĩa
Name	Xác định tên đối tượng ảnh. Tiền tố của tên đối tượng ảnh là img
BorderStyle	Xác định kiểu đường viền cho đối tượng ảnh.
Enabled	Cho phép đối tượng đáp ứng hoặc không đáp ứng những thay đổi xảy ra trong thời gian thực hiện chương trình.
Picture	Xác định tên tệp chứa hình ảnh sẽ xuất hiện trong đối tượng ảnh.
Stretch	Đây là thuộc tính logic, xác định khả năng tự điều chỉnh kích thước của đối tượng ảnh. Stretch = False (ngầm định): đối tượng ảnh tự điều chỉnh kích thước của mình cho phù hợp với kích thước của hình ảnh được đặt vào nó. Stretch = True: đối tượng ảnh không tự điều chỉnh kích thước.
Visible	Cho phép đối tượng xuất hiện hoặc không xuất hiện trên màn hình trong thời gian thực hiện chương trình.

13.1.3. Xác định kích thước cho đối tượng ảnh

Khi ta thêm đối tượng ảnh vào biểu mẫu ta không biết xác định kích thước của đối tượng này như thế nào cho phù hợp với hình ảnh sẽ được chứa trong đối tượng này. Vì vậy các thuộc tính Height và Width không có ý nghĩa lắm trong thời gian thiết kế chương trình khi hình ảnh chưa được tải vào đối tượng ảnh.

Để xác định kích thước đối tượng ảnh, đầu tiên cần cần tải hình ảnh vào đối tượng ảnh (bằng cách xác định tên tệp chứa hình ảnh cho thuộc tính Picture của đối tượng ảnh), rồi điều chỉnh kích thước của đối tượng ảnh cho phù hợp.

Đối tượng ảnh có khả năng tự điều chỉnh kích thước của mình

cho phù hợp với hình ảnh được đặt vào nó. Muốn vậy thuộc tính Stretch của đối tượng ảnh này phải có giá trị False.

13.1.4. Tải hình ảnh vào đối tượng ảnh trong thời gian thực hiện chương trình

Trên đây ta đã biết cách tải (tức là chuyển hoặc đặt) hình ảnh vào đối tượng ảnh trong thời gian thiết kế chương trình. Trong thời gian thực hiện chương trình, để tải hình ảnh vào đối tượng ảnh cần thực hiện câu lệnh sau đây:

Tên_đối_tượng_ảnh.Picture = LoadPicture("Đặc tả tệp")

Ví dụ có câu lệnh:

```
ImgAnh1.Picture = LoadPicture("c:\clipart\car.wmf")
```

Câu lệnh này tải vào đối tượng ảnh có tên là ImgAnh1 hình ảnh một chiếc ô tô được xác định bởi tệp car.wmf nằm trong thư mục \clipart\ của ổ đĩa C.

Chú ý: để xoá hình ảnh trong đối tượng ảnh ta cũng dùng chính câu lệnh gán này với đặc tả tệp là rỗng. Ví dụ có câu lệnh:

```
ImgAnh1.Picture = LoadPicture()
```

Câu lệnh này xoá hình ảnh trong đối tượng ảnh có tên là ImgAnh1.

13.2. ĐỐI TƯỢNG HỘP HÌNH

Đối tượng hộp hình (Picture Box) cũng cho phép đưa hình ảnh vào màn hình giao diện. Có thể sử dụng đối tượng hộp hình tương tự như đối tượng ảnh. Đối tượng hộp hình hoạt động hầu như giống hệt đối tượng ảnh, ngoại trừ các điểm sau đây:

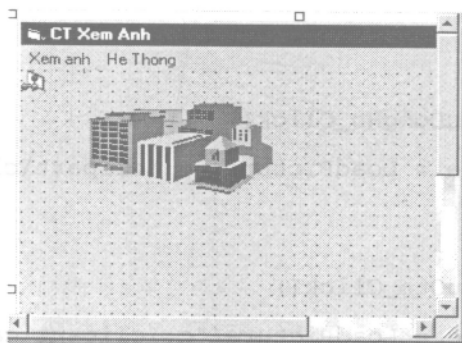
- Đối tượng hộp hình có nhiều thuộc tính, nhiều phương thức và đáp ứng được nhiều biến cố hơn đối tượng ảnh.

- Đối tượng hộp hình sử dụng nhiều tài nguyên hơn đối tượng ảnh, nên sử dụng nó không hiệu quả lắm.

Đối tượng hộp hình có thuộc tính `AutoSize`. Đây là thuộc tính logic. Khi `AutoSize = False` (ngầm định) thì kích thước của đối tượng hộp hình không được điều chỉnh. Các hình ảnh được điều chỉnh kích thước cho vừa với kích thước của đối tượng hộp hình. Khi `AutoSize = True` thì kích thước của đối tượng hộp hình được điều chỉnh cho phù hợp với hình ảnh được vào đối tượng hộp hình này.

Chú ý: đối tượng hộp hình cho phép vẽ vào nó các đối tượng khác, giống như đối tượng `Frame`. Khi đối tượng hộp hình xuất hiện thì những đối tượng đó cũng xuất hiện. Khi đối tượng hộp hình ẩn (tức là biến mất) thì các đối tượng đó cũng biến mất.

Sau đây là một ví dụ về chương trình có sử dụng đối tượng ảnh. Chương trình trong ví dụ này có tên là **XemAnh** (Xem ảnh). Màn hình giao diện của chương trình này có dạng sau đây (Hình 13.1.):



Hình 13.1. Màn hình giao diện của chương trình **XemAnh**.

Màn hình giao diện của chương trình này có một biểu mẫu, một đối tượng ảnh và thực đơn. Trong đối tượng ảnh có hình ảnh một số toà nhà cao tầng. Đối tượng ảnh của chương trình này có các thuộc tính sau đây:

**Bảng 13.3. Các thuộc tính chính của đối tượng ảnh
của chương trình XemAnh**

Thuộc tính	Giá trị
Name	ImgAnh1
BorderStyle	0
Enabled	True
Picture	C:\clipart\Building
Stretch	False
Visible	True

Các thủ tục - biến cố của chương trình này như sau:

```
Private Sub mnuKetThuc_Click()
    Beep
    End
End Sub

Private Sub mnuDongHo_Click()
    imgAnh1.Picture = LoadPicture("c:\clipart\clock.wmf")
End Sub

Private Sub mnuOto_Click()
    imgAnh1.Picture = LoadPicture("c:\clipart\car.wmf")
End Sub

Private Sub mnuPTo_Click()
    imgAnh1.Height = 2 * imgAnh1.Height
    imgAnh1.Width = 2 * imgAnh1.Width
```

```

End Sub

Private Sub mnuTienXu_Click()
imgAnh1.Picture = LoadPicture("c:\clipart\coins.wmf")
End Sub

Private Sub mnuTNho_Click()
imgAnh1.Height = 0.5 * imgAnh1.Height
imgAnh1.Width = 0.5 * imgAnh1.Width
End Sub

Private Sub mnuXoa_Click()
imgAnh1.Picture = LoadPicture()
End Sub

```

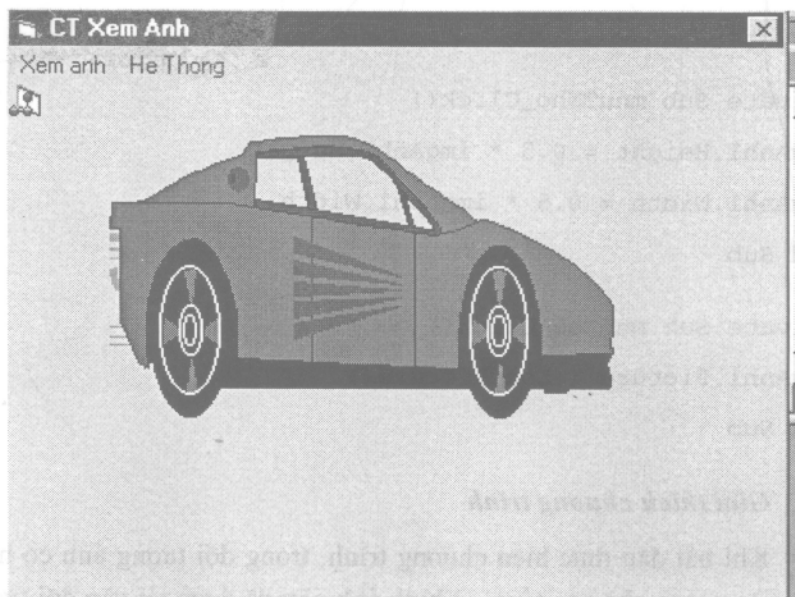
Giải thích chương trình

Khi bắt đầu thực hiện chương trình, trong đối tượng ảnh có hình ảnh một số toà nhà cao tầng, vì hình ảnh này đã được tải vào đối tượng ảnh trong thời gian thiết kế chương trình. Nếu người sử dụng chọn chức năng “Xem Oto” thì hình ảnh xe ô tô xuất hiện trong đối tượng ảnh. Tương tự như vậy, nếu người sử dụng chọn chức năng “Xem Dong Ho” (hoặc “Xem Tien xu”) thì hình ảnh cái đồng hồ (hoặc hình ảnh các chồng tiền xu) sẽ xuất hiện trong đối tượng ảnh.

Nếu người sử dụng chọn chức năng “Phong To” thì hình ảnh trong đối tượng ảnh được phóng to gấp đôi. Nếu người sử dụng chọn chức năng “Thu Nho” thì hình ảnh trong đối tượng ảnh được thu nhỏ chỉ còn một nửa. Nếu người sử dụng chọn chức năng “Xoa Anh” thì hình ảnh trong đối tượng ảnh bị xoá. Còn nếu người sử dụng chọn chức

năng “Ket Thuc” thì chương trình XemAnh sẽ kết thúc.

Khi người sử dụng chọn liên tiếp hai chức năng “**Xem Oto**” và “**Phong To**”, màn hình kết quả có dạng như hình sau đây (Hình 13.2):



Hình 13.2. Màn hình của chương trình XemAnh

khi chọn liên tiếp hai chức năng “Xem Oto” và “Phong To”

Trong chương trình này đã sử dụng đối tượng ảnh. Độc giả có thể sử dụng đối tượng hộp hình để viết một chương trình tương tự.

13.3. ĐỐI TƯỢNG ĐƯỜNG THẲNG

Đối tượng đường thẳng (**Line**) được sử dụng để vẽ một đường thẳng trong biểu mẫu. Đối tượng này có những thuộc tính chính sau đây:

Bảng 13.4. Các thuộc tính chính của đối tượng đường thẳng

Thuộc tính	Ý nghĩa
Name	Xác định tên đối tượng đường thẳng. Tiền tố của tên đối tượng này là Lin.
BoderColor	Xác định màu của đường thẳng.
BorderStyle	Xác định kiểu đường thẳng. Thuộc tính này có 7 giá trị. Các giá trị của thuộc tính này được giải thích trong bảng tiếp theo.
BoderWidth	Xác định độ dày của đường thẳng tính bằng twip.
DrawMode	Xác định kiểu vẽ đường thẳng. Giá trị ngầm định là 13 - Copy Pen hoạt động tốt đối với các ứng dụng của Visual Basic.
Visible	Cho phép đường thẳng xuất hiện hoặc không xuất hiện trên màn hình trong thời gian thực hiện chương trình.
X1	Toạ độ X của điểm đầu của đối tượng đường thẳng.
X2	Toạ độ X của điểm cuối của đối tượng đường thẳng.
Y1	Toạ độ Y của điểm đầu của đối tượng đường thẳng.
Y2	Toạ độ Y của điểm cuối của đối tượng đường thẳng.

Bảng 13.5. Các giá trị của thuộc tính *BorderStyle* của đối tượng đường thẳng

Giá trị	Tác dụng
0 - Transparet	Đường thẳng trong suốt, không nhìn thấy được.
1 - Solid	Đường thẳng là một đường liền (ngám định).
2 - Dash	Đường thẳng gồm những gạch ngang các nhau.
3 - Dot	Đường thẳng gồm các dấu chấm.
4 - Dash - Dot	Đường thẳng gồm các dấu gạch chấm liên tiếp nhau.
5 - Dash-Dot-Dot	Đường thẳng gồm hai dấu chấm, một dấu gạch liên tiếp nhau.
6 - Inside Solid	Đường thẳng là một đường liền, tương tự 1.

Để vẽ một đường thẳng trong biểu mẫu ta làm như sau:

- Nháy đúp vào biểu tượng của công cụ điều khiển ***Line*** trong hộp công cụ. Một đường thẳng sẽ xuất hiện trên biểu mẫu.
- Xác định thuộc tính ***x1, x2, y1, y2*** để điều chỉnh độ nghiêng và độ dài của đường thẳng.
- Xác định giá trị thích hợp các thuộc tính khác (***BorderStyle*** và ***BorderWidth***).

13.4. ĐỐI TƯỢNG HÌNH VẼ

Đối tượng hình vẽ (***Shape***) dùng để vẽ 6 loại hình vẽ khác nhau, như hình vuông, hình chữ nhật, hình tròn, hình elip, hình vuông có góc tròn và hình chữ nhật có góc tròn trong biểu mẫu. Đối tượng này có những thuộc tính chính sau đây:

Bảng 13.6. Các thuộc tính chính của đối tượng hình vẽ

Thuộc tính	Ý nghĩa
Name	Xác định tên đối tượng đường thẳng. Tiền tố của tên đối tượng này là ba kí tự Shp.
BackColor	Xác định màu nền của đối tượng hình vẽ.
BoderColor	Xác định màu của đường dùng để vẽ hình.
BorderStyle	Xác định kiểu đường vẽ của hình. Thuộc tính này có 7 giá trị, ngầm định là solid.
BoderWidth	Xác định độ dày của đường bao quanh hình vẽ tính bằng twip.
DrawMode	Xác định kiểu vẽ hình. Giá trị ngầm định là 13 - Copy Pen hoạt động tốt đối với các ứng dụng của Visual Basic.
FillColor	Xác định màu của đường dùng để tô màu hình vẽ.
FillStyle	Xác định kiểu tô màu hình vẽ. Thuộc tính này có đến 8 giá trị (Xem bảng 13.7).
Shape	Xác định loại hình vẽ do đối tượng hình vẽ tạo ra. Thuộc tính này có 6 giá trị (Xem bảng 13.8).
Visible	Cho phép đường đối tượng hình vẽ xuất hiện hoặc không xuất hiện trên màn hình trong thời gian thực hiện chương trình.

Bảng 13.7. Các giá trị của thuộc tính FillStyle của đối tượng Shape

Giá trị	Tác dụng
0 - Solid	Tô màu đặc (ngầm định).
1 - Transparet	Không tô màu. Hình vẽ chỉ có đường bao quanh.
2 - Horizontal Line	Tô màu bằng những gạch ngang các nhau.
3 - Vertical Line	Tô màu bằng những vạch đứng.
4 - Upward Diagonal	Tô màu bằng những đường thẳng chéo đi lên

Giá trị	Tác dụng
5 - Downward Diagonal	Tô mẫu bằng những đường thẳng chéo đi xuống
6 - Cross	Tô mẫu bằng những ô vuông
7 - Diagonal Cross	Tô mẫu bằng những ô chéo

Bảng 13.8. Các giá trị của thuộc tính Shape

Giá trị	Tác dụng
0 - Rectangle	Hình chữ nhật
1 - Square	Hình vuông
2 - Oval	Hình elip
3 - Circle	Hình tròn
4 - Rounded Rectangle	Hình chữ nhật có góc tròn
5 - Rounded Square	Hình vuông có góc tròn

Để thêm đối tượng hình vẽ vào biểu mẫu ta làm như sau:

- Nháy đúp vào **Shape**
- Xác định thuộc tính **Shape** thành **Oval**, **Rectangle**...theo yêu cầu sử dụng.
- Xác định các giá trị thích hợp cho các thuộc tính khác của đối tượng hình vẽ.

Một phương thức hay được sử dụng của đối tượng hình vẽ là phương thức Move. Phương thức này có dạng sau đây:

Tên_đối_tượng_hình_vẽ. Move x, y

Phương thức này sẽ đưa hình vẽ có tên đã chỉ ra đến vị trí được xác định bởi cặp tọa độ (x, y).

13.5. XÁC ĐỊNH MẪU KHI VẼ HÌNH

Trong thời gian thiết kế chương trình ta đã sử dụng các cửa sổ chọn mẫu của Visual Basic để chọn mẫu cho các đối tượng khác nhau. Ta cũng có thể viết câu lệnh xác định mẫu cho một đối tượng xác định. Muốn vậy ta phải dùng hàm RGB(). Đây là một hàm chuẩn của Visual Basic. Hàm này có dạng sau đây:

RGB(*r, g, b*)

Hàm này xác định một mẫu cho phép trong Visual Basic. Hàm này có 3 tham số là ***r***, ***g*** và ***b***. Kiểu của các tham số này là kiểu nguyên. Tác dụng của các tham số này được giải thích trong bảng sau đây:

Bảng 13.9. Các tham số r, g, b của hàm RGB

Tham số	Tác dụng
<i>r</i>	Là một số nguyên nằm trong khoảng 0 - 255, xác định thành phần mẫu đỏ (Red) trong mẫu được sử dụng.
<i>g</i>	Là một số nguyên nằm trong khoảng 0 - 255, xác định thành phần mẫu xanh lá cây (Green) trong mẫu được sử dụng.
<i>b</i>	Là một số nguyên nằm trong khoảng 0 - 255, xác định thành phần mẫu xanh thẫm (Blue) trong mẫu được sử dụng.

Bảng sau đây liệt kê một số mẫu chuẩn và các thành phần *r*, *g*, và *b* của mỗi mẫu đó.

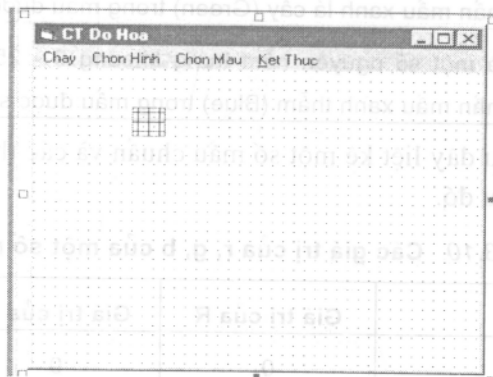
Bảng 13.10. Các giá trị của r, g, b của một số mẫu chuẩn

Mẫu	Giá trị của R	Giá trị của G	Giá trị của B
Đen (Black)	0	0	0
Xanh thẫm (Blue)	0	0	255
Xanh lá cây (Green)	0	255	0

Mẫu	Giá trị của R	Giá trị của G	Giá trị của B
Xanh cẩm thạch (Cyan)	0	255	255
Đỏ (Red)	255	0	0
Tía (Magenta)	255	0	255
Vàng (Yellow)	255	255	0
Trắng (White)	255	255	255

Trong thủ tục - biến cố `mnuChayNN_Click()` của chương trình **HoatHinh** dưới đây có sử dụng hàm `RGB()` để xác định màu cho hình vẽ.

Sau đây là một ví dụ về chương trình có sử dụng đối tượng hình vẽ. Chương trình trong ví dụ này có tên là **HoatHinh** (Hoạt Hình). Chương trình này sử dụng một đối tượng hình vẽ để tạo ra các hình vẽ khác nhau, với màu sắc khác nhau và chạy (chuyển động) theo các hướng khác nhau. Màn hình giao diện của chương trình này có dạng sau đây (Hình 13.3):



Hình 13.3. Màn hình giao diện của chương trình HoatHinh

Màn hình giao diện của chương trình này có một biểu mẫu, một đối tượng hình vẽ và thực đơn. Trong đối tượng hình vẽ có một hình vuông. Đối tượng ảnh của chương trình này có các thuộc tính sau đây:

Bảng 13.11. Các thuộc tính của đối tượng hình vẽ của chương trình HoatHinh

Thuộc tính	ý nghĩa
Name	HVuong
BackColor	0 - Transparent
BorderStyle	Solid
BoderWidth	1
DrawMode	13 - Copy Pen
FillColor	&H00FF0000&
FillStyle	6 - Cross
Shape	1 - Square
Visible	True

Thực đơn của chương trình này gồm có các chức năng sau đây:

Tiêu đề của chức năng	Tên thủ tục - biến cố của chức năng
Chạy	
----Chạy Dọc	mnuChayDoc_Click()
----Chạy Ngang	mnuChayNgang_Click()
----Chạy Xìen	mnuChayXien_Click()
----Chạy Ngẫu nhiên	mnuChayNN_Click()
Hình	
----Hình Vuông	mnuVuong_Click()
----Hình Tròn	mnuTron_Click()
Màu	
----Màu Đỏ	mnuDo_Click()
----Màu Xanh	mnuXanh_Click()
----Màu Tím	mnuTim_Click()
Kết Thúc	mnuKThuc_Click()

Các thủ tục - biến cố của chương trình này như sau:

```
Option Explicit
Dim i%, j%, k%, x%, y%, dx%, dy%, tt%, ketThuc%

Private Sub mnuChayDoc_Click()
    ' Chay doc o giua man hinh
    x = Int(FrmDoHoa.Width / 2) - HVuong.Width / 2
    y = 0
    dy = 200
    For k = 1 To 50
        HVuong.Move x, y
        ' Tao thoi gian cho doi
        For i = 1 To 5000
            For j = 1 To 1000
                Next j
            Next i
            y = y + dy
            If y > FrmDoHoa.Height - 1500 Then
                y = FrmDoHoa.Height - 1500
                dy = -100
            Else
                If y < 0 Then
                    y = 0
                    dy = 100
                End If
            End If
        End If
    End If
```



```

Next
MsgBox "Ket thuc ve hinh !", 16
End Sub

Private Sub mnuChayNgang_Click()
' Chay ngang o giua man hinh
y = Int(FrmDoHoa.Height / 2) - (HVuong.Height)
x = 0
dx = 200
For k = 1 To 50
    HVuong.Move x, y
    ' Tao thoi gian cho doi
    For i = 1 To 5000
        For j = 1 To 1000
            Next j
        Next i
        x = x + dx
        If x > FrmDoHoa.Width - 1500 Then
            x = FrmDoHoa.Width - 1500
            dx = -100
        Else
            If x < 0 Then
                x = 0
                dx = 100
            End If
        End If
    Next
MsgBox "Ket thuc ve hinh !", 16

```

```

End Sub

Private Sub mnuChayXien_Click()
    ' Chay xien theo Duong Cheo Chinh
    Dim tyle!
    tyle = FrmDoHoa.Height / FrmDoHoa.Width
    y = 0
    x = 0
    dx = 200
    dy = tyle * dx
    For k = 1 To 40
        HVuong.Move x, y
        ' Tao thoi gian cho doi
        For i = 1 To 5000
            For j = 1 To 1000
                Next j
            Next i
            x = x + dx
            'y = y + dy
            y = x * tyle
            If x > FrmDoHoa.Width - HVuong.Width Then
                x = FrmDoHoa.Width - HVuong.Width
                y = FrmDoHoa.Height - HVuong.Height
                dx = -200
                dy = tyle * dx
            Else
                If (x < 0) Or (y < 0) Then
                    x = 0

```

```

        y = 0
        dx = 200
        dy = tyle * dx
    End If
End If

Next
MsgBox "Ket thuc ve hinh !", 16
End Sub

Private Sub mnuTim_Click()
HVuong.FillColor = RGB(255, 0, 255)
End Sub

Private Sub mnuDo_Click()
HVuong.FillColor = RGB(255, 0, 0)
End Sub

Private Sub mnuChayNN_Click()
' Hinh chay ngau nhien
Dim g, r, b, hinh%
    For k = 1 To 10
        g = Rnd() * 255
        r = Rnd() * 255
        b = Rnd() * 255
        x = Rnd() * ScaleWidth
        y = Rnd() * ScaleWidth
        hinh = Rnd() * 5
        HVuong.Move x, y
        HVuong.BackColor = RGB(r, g, b)
    
```

```

        HVuong.BorderColor = r
        HVuong.FillColor = g
        HVuong.Shape = hình
        For i = 1 To 5000
            For j = 1 To 3000
                Next j
            Next i
        Next
    MsgBox "Ket thuc ve hình !", 16
End Sub

Private Sub mnukThuc_Click()
    End
End Sub

Private Sub mnutron_Click()
    HVuong.Shape = 3
End Sub

Private Sub mnuvuong_Click()
    HVuong.Shape = 1
End Sub

Private Sub mnuxanh_Click()
    HVuong.FillColor = RGB(0, 255, 0)
End Sub

```

Giải thích chương trình

Khi vừa mới bắt đầu thực hiện chương trình, đối tượng hình vẽ chứa một hình vuông. Nếu người sử dụng chọn chức năng Hình Tron,

thì thủ tục - biến cố `mnuTron_Click()` sẽ được thực hiện, làm cho hình vẽ sẽ trở thành hình tròn. Bây giờ, nếu người sử dụng chọn chức năng Hình Vuông thì, thủ tục - biến cố `mnuVuong_Click()` sẽ được thực hiện, làm cho hình vẽ sẽ trở thành hình vuông.

Khi vừa mới bắt đầu thực hiện chương trình, hình vuông trong đối tượng hình vẽ có màu xanh thẫm. Nếu người sử dụng chọn chức năng Mau Do, thì thủ tục - biến cố `mnuDo_Click()` sẽ được thực hiện, làm cho màu của hình vẽ sẽ trở thành màu đỏ. Nếu người sử dụng chọn chức năng Mau Tim, thì thủ tục - biến cố `mnuTim_Click()` sẽ được thực hiện, làm cho màu của hình vẽ sẽ trở thành màu tím.

Chỉ có các chức năng của nhóm chức năng đầu tiên mới vẽ các hình chuyển động. Khi người sử dụng chọn chức năng Chạy Dọc, thì thủ tục - biến cố `mnuChayDoc_Click()` sẽ được thực hiện. Máy sẽ vẽ hình vuông (hoặc tròn) đang có trong đối tượng hình vẽ chạy đi chạy lại theo đường thẳng đứng ở giữa biểu mẫu.

Khi người sử dụng chọn chức năng Chạy Ngang, thì thủ tục - biến cố `mnuChayNgang_Click()` sẽ được thực hiện. Máy sẽ vẽ hình vuông (hoặc tròn) đang có trong đối tượng hình vẽ chạy đi chạy lại theo đường thẳng nằm ngang ở giữa biểu mẫu.

Nếu người sử dụng chọn chức năng Chạy Xìen, thì thủ tục - biến cố `mnuChayXien_Click()` sẽ được thực hiện. Máy sẽ vẽ hình vuông (hoặc tròn) đang có trong đối tượng hình vẽ chạy đi chạy lại theo đường chéo chính của biểu mẫu.

Khi người sử dụng chọn chức năng Chạy Ngẫu Nhiên, thì thủ tục - biến cố `mnuChayNN_Click()` sẽ được thực hiện. Thủ tục - biến cố này sẽ vẽ một hình vẽ ngẫu nhiên (trong 6 hình vẽ cho phép) bằng màu ngẫu nhiên (trong các màu cho phép dùng trong Windows) ở vị trí ngẫu nhiên trên màn hình của biểu mẫu.

Trong thủ tục - biến cố `mnuChayNN_Click()` của chương trình này ta đã sử dụng hàm `RGB()` để xác định màu dùng để vẽ hình.

CÂU HỎI ÔN TẬP

1. Đối tượng ảnh dùng để làm gì?
2. Hãy trình bày các thuộc tính chính của đối tượng ảnh.
3. Đối tượng hộp hình có những điểm gì giống với đối tượng ảnh?
4. Đối tượng hộp hình có những điểm gì khác với đối tượng ảnh?
5. Đối tượng đường thẳng đầy đủ là gì? Nó có những thuộc tính nào?
6. Đối tượng hình vẽ dùng để là gì? Đối tượng này có những thuộc tính nào?

DỰ ÁN ĐA BIỂU MẪU

Các bài toán trong thực tế thường phức tạp, có quy mô lớn, thực hiện nhiều công việc khác nhau và đòi hỏi nhiều giao diện khác nhau. Trong trường hợp này chương trình giải bài toán cần phải sử dụng nhiều biểu mẫu.

Chương này trình bày kỹ thuật sử dụng nhiều biểu mẫu trong một dự án của Visual Basic và trình bày một ví dụ về dự án có nhiều biểu mẫu.

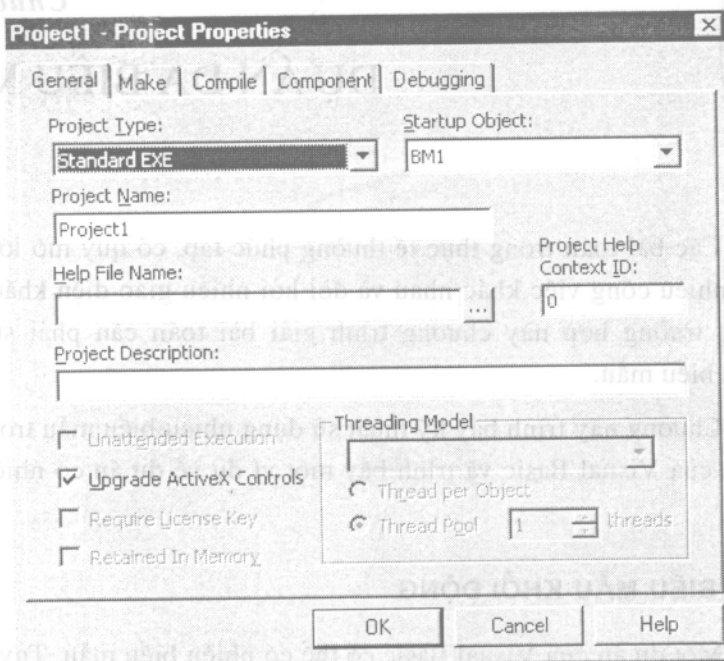
14.1. BIỂU MẪU KHỞI ĐỘNG

Một dự án của Visual Basic có thể có nhiều biểu mẫu. Tuy nhiên, quá trình thực hiện chương trình luôn luôn bắt đầu từ một biểu mẫu xác định. Biểu mẫu này được gọi là biểu mẫu khởi động (*Startup*). Khi trong dự án có nhiều biểu mẫu, lập trình viên cần phải chỉ ra cho Visual Basic biết biểu mẫu nào là biểu mẫu khởi động.

Để xác định biểu mẫu khởi động cần thực hiện các thao tác sau:

- Chọn Project
- Chọn Project Properties. Hộp hội thoại Project Properties sẽ xuất hiện (Hình 14.1).

- Chọn General
- Xác định biểu mẫu khởi động trong hộp Startup Object:
- Chọn OK



Hình 14.1. Hộp hội thoại Project Properties

Chú ý: nếu lập trình viên không xác định biểu mẫu khởi động, thì Visual Basic luôn luôn coi biểu mẫu được tạo ra đầu tiên là biểu mẫu khởi động.

14.2. CÁC PHƯƠNG THỨC ĐƯỢC SỬ DỤNG ĐỐI VỚI BIỂU MẪU

Để điều khiển các biểu mẫu trong dự án đa biểu mẫu cần phải sử dụng các phương thức sau đây đối với biểu mẫu:

DỰ ÁN ĐA BIỂU MẪU

Các bài toán trong thực tế thường phức tạp, có quy mô lớn, thực hiện nhiều công việc khác nhau và đòi hỏi nhiều giao diện khác nhau. Trong trường hợp này chương trình giải bài toán cần phải sử dụng nhiều biểu mẫu.

Chương này trình bày kỹ thuật sử dụng nhiều biểu mẫu trong một dự án của Visual Basic và trình bày một ví dụ về dự án có nhiều biểu mẫu.

14.1. BIỂU MẪU KHỞI ĐỘNG

Một dự án của Visual Basic có thể có nhiều biểu mẫu. Tuy nhiên, quá trình thực hiện chương trình luôn luôn bắt đầu từ một biểu mẫu xác định. Biểu mẫu này được gọi là biểu mẫu khởi động (*Startup*). Khi trong dự án có nhiều biểu mẫu, lập trình viên cần phải chỉ ra cho Visual Basic biết biểu mẫu nào là biểu mẫu khởi động.

Để xác định biểu mẫu khởi động cần thực hiện các thao tác sau:

- Chọn Project
- Chọn Project Properties. Hộp hội thoại Project Properties sẽ xuất hiện (Hình 14.1).

Bảng 14.1. Các phương thức của biểu mẫu.

Phương thức	Ý nghĩa
FormName.Show	Hiện biểu mẫu trên màn hình. Nếu biểu mẫu chưa được tải vào bộ nhớ thì trước khi hiện biểu mẫu trên màn hình máy sẽ tự động tải biểu mẫu vào bộ nhớ.
Load FormName	Tải biểu mẫu vào bộ nhớ nhưng không hiện nó trên màn hình.
FormName.Hide	Xoá biểu mẫu khỏi màn hình nhưng không xoá biểu mẫu khỏi bộ nhớ.
Unload.FormName	Xoá biểu mẫu khỏi bộ nhớ.

14.3. NGUYÊN TẮC LẬP CHƯƠNG TRÌNH ĐA BIỂU MẪU

Giả sử một chương trình có nhiều biểu mẫu có tên là BM1, MB2,...và giả sử biểu mẫu BM1 là biểu mẫu khởi động. Ta không quan tâm đến nội dung cụ thể của các biểu mẫu này. Tuy nhiên, muốn sử dụng chính xác các biểu mẫu có trong dự án phải tuân theo nguyên tắc sau đây:

- Trong biểu mẫu khởi động (đã giả sử là BM1) phải có một nút lệnh (hoặc một chức năng của bảng thực đơn) dùng để gọi một trong các biểu mẫu còn lại (giả sử là BM2). Trong thủ tục - biến cố liên quan đến nút lệnh này nhất thiết phải có câu lệnh Hide dùng để xoá biểu mẫu hiện thời khỏi màn hình và câu lệnh Show dùng để hiện biểu mẫu được gọi trên màn hình.

- Trong biểu mẫu được gọi (giả sử là BM2) phải có một nút lệnh (hoặc một chức năng của bảng thực đơn) dùng để kết thúc quá trình làm việc với biểu mẫu này. Trong thủ tục - biến cố liên quan đến nút lệnh này cũng nhất thiết phải có câu lệnh Hide dùng để xoá biểu mẫu này khỏi màn hình và câu lệnh Show dùng để hiện biểu mẫu cần dùng tiếp theo. Biểu mẫu tiếp theo này có thể là biểu mẫu đã gọi biểu mẫu

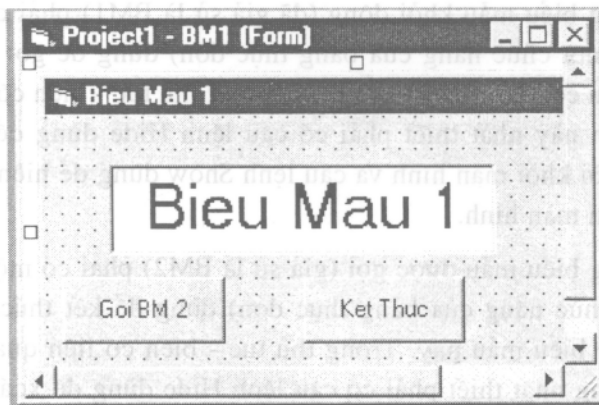
hiện thời (trong trường hợp này là BM1) hoặc một biểu mẫu khác.

Nguyên tắc nêu trên được thể hiện trong chương trình dưới đây. Chương trình này có tên là *DaBieuMau*. Chương trình này có 2 biểu mẫu là BM1 và BM2. BM1 có một hộp văn bản có nội dung là “Biêu Mau 1” và 2 nút lệnh có tên là *Goi BM2* và *Ket Thuc*. BM1 được xác định là biểu mẫu khởi động.

BM2 cũng có một hộp văn bản có nội dung là “Biêu Mau 2” và có một nút lệnh có tên là “Ket Thuc”.

Khi thực hiện chương trình này, đầu tiên BM1 xuất hiện trên màn hình (vì BM1 đã được xác định là biểu mẫu khởi động) như hình 14.2. Khi đó nếu ta sử dụng nút lệnh “Goi BM2” thì BM1 bị xóa khỏi màn hình và BM2 sẽ xuất hiện trên màn hình như hình 14.3.

Lúc này nếu ta sử dụng nút lệnh “Ket Thuc” của BM2 thì biểu mẫu này (BM2) bị xóa khỏi màn hình và BM1 sẽ xuất hiện trở lại trên màn hình. Bây giờ, nếu ta sử dụng nút lệnh “Ket Thuc” của BM1 thì Chương trình *DaBieuMau* sẽ kết thúc.



Hình 14.2. BM1 của chương trình *DaBieuMau*



Hình 14.3. BM2 của chương trình DaBieuMau

Các thủ tục - biến cố của các nút lệnh của biểu mẫu BM1 như sau:

Option Explicit

```
Private Sub cmdGoiBM2_Click()
```

```
    BM1.Hide
```

```
    BM2.Show
```

```
End Sub
```

```
Private Sub CmdKetThuc_Click()
```

```
    Unload BM1
```

```
    Unload BM2
```

```
End
```

```
End Sub
```

Thủ tục - biến cố của nút lệnh “Ket Thuc” của BM2 như sau:

```
Private Sub cmdKetThuc_Click()
```

```

BM2.Hide
BM1.Show
End Sub

```

Các nguyên tắc sử dụng đa biểu mẫu sẽ trở nên rõ ràng hơn sau khi nghiên cứu chương trình trong ví dụ sau đây.

14.4. VÍ DỤ TỔNG HỢP

Trong phần này ta sẽ nghiên cứu một ví dụ tổng hợp, sử dụng nhiều công cụ khác nhau của Visual Basic để viết chương trình giải một bài toán thực tế. Nội dung bài toán như sau:

Cho biết họ và tên, điểm toán, điểm lý, điểm hoá của n sinh viên. Hãy viết chương trình in bảng học bổng có dạng:

Bảng học bổng của sinh viên

Số TT	Tên sinh viên	Điểm TB	Học bổng
	Tổng cộng		

Học bổng được tính theo công thức sau đây: (dtb: điểm trung bình)

$$\text{Học bổng} = \begin{cases} 0 & \text{nếu } dtb < 7 \\ 120 & \text{nếu } 7 \leq dtb < 9 \\ 180 & \text{nếu } dtb \geq 9 \end{cases}$$

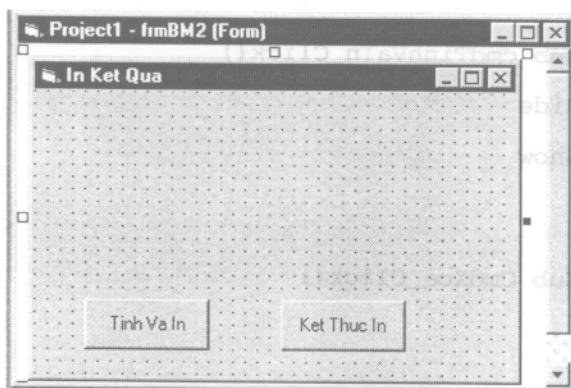
Chương trình này có tên là *InBangHB* (In Bảng Học Bổng). Chương trình này gồm 2 biểu mẫu có tên là *frmBMI* và *frmM2* và một đơn thể chương trình độc lập có tên là *nhapdl*.

Biểu mẫu frmBM1 là biểu mẫu khởi động, chứa màn hình giao diện chính của chương trình. Biểu mẫu này có dạng sau đây:



Hình 14.4. BM1 của chương trình InBangHB

Biểu mẫu frmBM2 dùng để in bảng kết quả. Biểu mẫu này có dạng sau đây:



Hình 14.5. BM2 của chương trình InBangHB

Đơn thể chương trình độc lập *nhapdl* chứa các thủ tục dùng chung cho cả 2 biểu mẫu của chương trình.

Phần mã lệnh của frmBM1

```

Private Sub CmdKetThuc_Click()
End
End Sub

Private Sub cmdNhapDL_Click()
Dim i%
Call nhap
Print " Nhap xong DL"
For i = 1 To n
    Print t(i), l(i), h(i)
Next
End Sub

Private Sub cmdTinhVaIn_Click()
    frmBM1.Hide
    frmBM2.Show
End Sub

Private Sub CmdXoa_Click()
    Cls
End Sub

```

Phần mã lệnh frmBM2

```

Option Explicit

Private Sub CmdKetThuc_Click()
    frmBM2.Hide

```

```

frmBM1.Show
End Sub

Private Sub CmdTVI_Click()
Dim thb%, hb%, stt%, dtb!, i%
Cls
Print
Print "    BANG HOC BONG CUA SINH VIEN"
Print
stt% = 1
For i = 1 To n
    dtb = (t(i) + l(i) + h(i)) / 3
    If dtb < 7 Then
        hb = 0
    Else
        If dtb < 9 Then
            hb = 120
        Else
            hb = 180
        End If
    End If
    Print stt, ten(i), dtb, hb
    stt% = stt% + 1
    thb = thb + hb

```



```

Next
Print
Print " Tong Cong", " ", " ", thb
End Sub

```

Phân mã lệnh của đơn thể chương trình độc lập *nhapdl*

```

Option Explicit
Public ten$(100), t%(100), l%(100), h%(100)
Public n%
Public Sub nhap()
Dim i%
n = 4
ten$(1) = "Le Hong Mai": ten$(2) = "Dao My Linh"
ten$(3) = "Le Hai Ha": ten$(4) = "Vu Van Ba"
t(1) = 8: t(2) = 9: t(3) = 8: t(4) = 6
l(1) = 9: l(2) = 10: l(3) = 6: l(4) = 5
h(1) = 8: h(2) = 9: h(3) = 9: h(4) = 6
End Sub

```

Giải thích chương trình

Trước hết hãy chú ý đến đơn thể chương trình độc lập *Nhapdl*. Trong đơn thể chương trình này từ khoá **Public** được sử dụng để khai báo 4 mảng: *ten\$(100)*, *t%(100)*, *l%(100)*, *h%(100)* và biến *n%*. Vì vậy 3 mảng và biến *n* là các mảng và biến toàn cục, có thể truy nhập trong toàn bộ dự án này. Thủ tục - biến cố **Public Sub nhap()** dùng để gán (chứ không nhập từ bàn phím) cho các biến những giá trị xác định. Các giá trị này được sử dụng để tạo ra các kết quả tính toán của chương trình.

Khi bắt đầu thực hiện chương trình, frmBM1 sẽ xuất hiện trên màn hình vì nó đã được xác định là biểu mẫu khởi động. Lúc này nếu ta sử dụng nút lệnh “Nhập DL” thì thủ tục - biến cố

```
Private Sub cmdNhapDL_Click()
```

của biểu mẫu này sẽ được thực hiện. Trong thủ tục - biến cố này có câu lệnh

```
Call nhập
```

Câu lệnh này làm cho thủ tục nhập() trong đơn thể chương trình độc lập được thực hiện. Sau đó máy in các giá trị các biến vừa được nhập trên màn hình biểu mẫu.

Nếu sử dụng nút lệnh “Tính Va In” thì thủ tục - biến cố cmdTinhVaIn_Click() của biểu mẫu này sẽ được thực hiện. Thủ tục - biến cố này chỉ có 2 câu lệnh sau đây:

```
frmBM1.Hide ` (1)
```

```
frmBM2.Show ` (2)
```

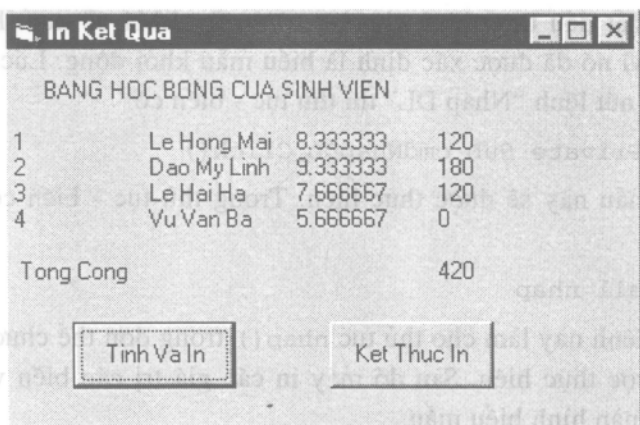
Câu lệnh (1) làm cho frmBM1 bị xóa khỏi màn hình. Câu lệnh (2) làm cho frmBM2 xuất hiện. Lúc này nếu ta sử dụng nút lệnh “Tính Va In” của biểu mẫu này thì in bảng học bổng (xem hình 14.6). Nếu ta sử dụng nút lệnh “Ket Thuc In” thì thủ tục - biến cố CmdKetThuc_Click() của biểu mẫu này được thực hiện. Thủ tục - biến cố này chỉ có 2 câu lệnh sau đây:

```
frmBM2.Hide ` (3)
```

```
frmBM1.Show ` (4)
```

Câu lệnh (3) làm cho frmBM2 sẽ biến mất, và câu lệnh (4) làm cho frmBM1 sẽ xuất hiện trở lại trên màn hình. Bây giờ, nếu ta sử dụng nút lệnh “Ket Thuc” của frmBM1 thì chương trình InBangHB sẽ kết thúc.

Đây là màn hình in kết quả khi thực hiện chương trình.



Hình 14.6. Màn hình in kết quả của chương trình InBangHB

Độc giả có thể cải tiến chương trình này, như thêm phần in dòng tiêu đề, in các vạch ngang và cột dọc, in điểm trung bình có 2 chữ số phần lẻ... để kết quả đẹp hơn.

CÂU HỎI ÔN TẬP VÀ BÀI TẬP

Lý thuyết

1. Tại sao cần dùng nhiều biểu mẫu trong một dự án?
2. Biểu mẫu khởi động là gì? Căn cứ vào đâu mà Visual Basic xác định được biểu mẫu khởi động?
3. Hãy trình bày những phương thức được sử dụng để điều khiển các biểu mẫu trong dự án đa biểu mẫu.
4. Hãy trình bày nguyên tắc sử dụng đa biểu mẫu.

Bài tập

5. Hãy viết chương trình đa biểu mẫu để giải bài toán: Cho một tập phiếu tính lương, mỗi phiếu có các thông tin sau đây:

- Họ và tên công nhân
- Tiền lương
- Tiền thưởng
- Tiền nợ

Chương trình cần thực hiện các công việc là:

- a) Tạo ra một tệp có tên PTL để chứa các phiếu tính lương đã cho.
- b) Nhập các phiếu tính lương vào tệp PTL.
- c) Sử dụng tệp PTL để in bảng lương sau :

Số TT	Họ và tên	Tiền lương	Tiền thưởng	Tiền nợ	Tiền được lĩnh
	Tổng cộng				

MỤC LỤC

Trang

Lời nói đầu	3
-------------------	---

CHƯƠNG 1. GIỚI THIỆU VISUAL BASIC

1.1. Từ Basic đến Visual Basic	5
1.2. Cài đặt Visual Basic	8
1.3. Khởi động Visual Basic	9
1.4. Môi trường phát triển tổng hợp của VB.....	11
1.5. Hai bước lập trình trong VB.....	17
1.6. Các ưu, nhược điểm của lập trình trong Visual Basic.....	18
Câu hỏi ôn tập.....	18

CHƯƠNG 2. CHƯƠNG TRÌNH ĐẦU TIÊN

2.1. Tạo một thư mục chứa chương trình Visual Basic	19
2.2. Các đối tượng được sử dụng trong chương trình	20
2.3. Giới thiệu chương trình "Hello"	20
2.4. Tạo một dự án mới cho chương trình Hello	23
2.5. Lưu dự án mới lên đĩa từ	24
2.6. Thiết kế giao diện cho chương trình.....	25
2.7. Viết các lệnh cho chương trình	33
2.8. Thực hiện chương trình HELLO	39
2.9. Tạo ra chương trình khả thi.....	39
Câu hỏi ôn tập.....	40

CHƯƠNG 3. ĐỐI TƯỢNG VÀ THUỘC TÍNH

3.1. Khái niệm đối tượng	41
3.2. Một số thuộc tính thường dùng của các đối tượng.....	42
3.3. Biểu mẫu	48
3.4. Nhân.....	51
3.5. Hộp văn bản.....	53
3.6. Nút lệnh.....	55
Câu hỏi ôn tập	57

CHƯƠNG 4. NHỮNG KIỂU DỮ LIỆU CƠ SỞ CỦA VISUAL BASIC

4.1. Khái niệm kiểu dữ liệu.....	59
4.2. Các hằng trong VB.....	61
4.3. Biến	64
4.4. Các phép toán	69
4.5. Hàm chuẩn	71
4.6. Biểu thức	72
Câu hỏi ôn tập	77

CHƯƠNG 5. LẬP CHƯƠNG TRÌNH ĐƠN GIẢN

5.1. Quy định về cách viết các dòng lệnh trong chương trình VB.....	77
5.2. Cách viết dòng chú thích trong chương trình	78
5.3. Một số câu lệnh của Visual Basic	78
5.4. Chuyển đổi kiểu dữ liệu trong quá trình tính giá trị biểu thức	80
5.5. Chương trình không có phần nhập dữ liệu.....	82
5.6. Lập trình có phần nhập - xuất dữ liệu	95
Câu hỏi ôn tập và bài tập.....	103

CHƯƠNG 6. CÁC CẤU TRÚC LỰA CHỌN

6.1. Khái niệm chương trình phân nhánh (lựa chọn).....	104
6.2. Câu lệnh If	105
6.3. Câu lệnh SELECT	117
6.4. Đối tượng nút tùy chọn.....	132
6.5. Đối tượng hộp kiểm.....	140
Câu hỏi ôn tập và bài tập	149

CHƯƠNG 7. CÁC CẤU TRÚC LẶP

7.1. Khái niệm chương trình lặp	151
7.2. Câu lệnh chu trình số học (câu lệnh for)	153
7.3. Lệnh chu trình logic Do While ... Loop.....	160
7.4. Lệnh chu trình logic Do Until ... Loop	167
7.5. Lệnh chu trình logic Do ... Loop While	169
7.6. Câu lệnh Exit	178
7.7. Chu trình lồng nhau.....	181
Câu hỏi ôn tập và bài tập	184

CHƯƠNG 8. KHAI BÁO VÀ SỬ DỤNG MẢNG

8.1. Khái niệm mảng.....	187
8.2. Khai báo mảng	189
8.3. Một số ví dụ xử lý mảng.....	195
Câu hỏi ôn tập và bài tập	201

CHƯƠNG 9. HÀM VÀ THỦ TỤC

9.1. Lập trình có cấu trúc	203
9.2. Các khái niệm	204
9.3. Một số hàm chuẩn của VB.....	206

9.4. Hàm và thủ tục tự định nghĩa	209
9.5. Module độc lập	223
9.6. Một số ví dụ về hàm tự định nghĩa	224
Câu hỏi ôn tập và bài tập	228

CHƯƠNG 10. XUẤT DỮ LIỆU TRONG VISUAL BASIC

10.1. In dữ liệu trên biểu mẫu	230
10.2. In dữ liệu trên máy in	242
10.3. In dữ liệu trong hộp ảnh	244
Câu hỏi ôn tập	247

CHƯƠNG 11. TẠO LẬP VÀ XỬ LÝ TỆP

11.1. Khái niệm tệp	248
11.2. Các loại tệp trong Visual Basic	249
11.3. Tệp tuần tự	250
11.4. Tệp ngẫu nhiên	260
Câu hỏi ôn tập	269

CHƯƠNG 12. HOÀN TẤT GIAO DIỆN

12.1. Thực đơn	271
12.2. Hộp hội thoại chung	278
12.3. Lệnh MsgBox	285
12.4. Hàm MsgBox()	287
12.5. Hộp danh sách	291
12.6. Hộp danh sách ổ đĩa	299
12.7. Hộp danh sách thư mục	299
12.8. Hộp danh sách tệp	301
12.9. Đối tượng Timer	305
Câu hỏi ôn tập	310

CHƯƠNG 13. ĐỒ HOẠ TRONG VISUAL BASIC

13.1. Đối tượng ảnh.....	311
13.2. Đối tượng hộp hình.....	314
13.3. Đối tượng đường thẳng.....	318
13.4. Đối tượng hình vẽ.....	320
13.5. Xác định mẫu khi vẽ hình.....	323
Câu hỏi ôn tập.....	332

CHƯƠNG 14. DỰ ÁN ĐA BIỂU MẪU

14.1. Biểu mẫu khởi động.....	333
14.2. Các phương thức được sử dụng đối với biểu mẫu.....	334
14.3. Nguyên tắc lập chương trình đa biểu mẫu.....	335
14.4. Ví dụ tổng hợp.....	338
Câu hỏi ôn tập và bài tập.....	344
Tài liệu tham khảo.....	346

Chịu trách nhiệm xuất bản: **PGS. TS. Tô Đăng Hải**

Biên tập: **Ths. Vũ Thị Minh Luận**

Trình bày bìa: **Hương Lan**

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

70 TRẦN HUNG ĐẠO - HÀ NỘI

In 1000 cuốn, khổ 14,5 x 20,5 cm, tại Xí nghiệp in 19 - 8 số 3
đường Nguyễn Phong Sắc - Nghĩa Tân - Cầu Giấy - Hà Nội.

Giấy phép xuất bản số: 06-293, ngày 5/1/2004.

In xong và nộp lưu chiểu tháng 01 năm 2005.

204236



Giá: 36.000đ