LỜI NÓI ĐẦU

Dựa vào kinh nghiệm thực tế trong việc viết các chương trình điều khiển cũng như lập trình cơ sở dữ liệu bằng nhiều ngôn ngữ lập trình khác nhau, tôi đã đúc kết ra các phương pháp học lập trình làm sao để các bạn có thể tiếp cận ngôn ngữ Visual Basic một cách nhanh chóng và có khả năng lập trình trên nó dễ dàng nhất và hiệu quả nhất.

Một số đặc điểm quan trọng nhất, có thể xem nó như một bước tiến mang tính cách mạng của các ngôn ngữ lập trình Visual trên Windows so với các ngôn ngữ lập trình For Dos trước đây. Đó là lập trình hướng đối tượng và giao diện đồ hoạ. Với các ưu điểm tuyệt với này việc lập trình trở nên vô cùng đơn giản cho người lập trình và phía người sử dụng thì chương trình sẽ trở nên rất dể dàng và rất tiện ích trong việc khai thác vận hành.

Tài liệu này là lần biên soạn đầu tiên nên chắn chắn sẽ không tránh khỏi các thiếu sót, trong quá trình giảng dạy tôi sẽ ghi nhận các thiếu sót và hiệu chỉnh vào lần biên tập sau.

TPHCM, ngày 30/08/2004

Tác giả

Bùi Thư Cao

MỤC LỤC

1 .GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH VISUAL BASIC

1.1 XUẤT SỨ

Chúng ta đã từng quen biết với ngôn ngữ lập trình trong môi trường hệ điều hành **MSDOS, PC-DOS, Microsoft** ra đời đã cho ra hàng loạt ứng dụng và **Microsoft Visual Basic** là cách nhanh nhất và tốt nhất để tạo các trình ứng dụng dành cho **Microsoft Windows** . Vậy **Visual Basic** là gì ?.

Phần "**Visual**" đề đến phương pháp được sử dụng để tạo giao diện đồ hoạ người sử dụng Cách viết dòng số mô tả ,việc xuất hiện vào vị trí của các yếu tố giao diện một cách đơn giản là bạn thêm bổ xung các đối tượng được tạo trước đó vào vị trí trên màn hình

Nếu bạn đã từng sử dụng chương trình vẽ như Pain , bạn đã có sẵn các kỹ năng cần thiết để tạo một hiệu ứng giao diện nguòi dùng

Phần "**Basic" (Beginnes – All – Purpose Symbolic Instruction Code**) . Đề cập đến ngôn ngữ **Basic** một ngôn ngữ được các nhà lập trình sử dụng nhiều hơn bất kì ngôn ngữ khác trong lịch sử máy tính

Visual Basic được giới thiệu lần đầu tiên vào năm 1991, từ ngôn ngữ lập trình **Basic** gốc trên hệ điều hành **DOS** . Mãi đến cuối năm 1992 , khi phiên bản 3.0 ra đời với rất nhiều cải tiến so với phiên bản trước đó, **Visual Basic** mới thực sự trở thành một trong những công cụ chính để phát triển các ứng dụng trên Windows . Những người mới bắt đầu có thể viết chương trình bằng cách chỉ học một vài **Commands** ,**Functions** và **Keywords** ..Khả năng của ngôn ngữ cho phép những người chyên nghiệp hoàn thành bất kì điều gì nhờ sử dụng ngôn ngữ lập trình **MSWindows** nào khác

Tóm lại ngôn ngữ lập trình **Visual Basic** là ngôn ngữ lập trình hướng đối tượng nó rất mạnh về lập trình cơ sở dữ liệu

Visual Basic còn có hai dạng khác : ***Visua Basic For Application (VBA)*** và ***VBScript*** .VBA là ngôn ngữ nằm phía sau các chương trình **Word**, **Excel, MSAccess, Msproject** còn gọi là **Macros** . Dùng **VBA** trong **MSOffice** ta có thể làm tăng chức năng bằng cách tự động hoá các chương trình

VBScript được dùng cho **Internet** và chính **Operating System**

Dù cho mục đích của bạn là tạo một tiện ích nhỏ cho riêng bạn , trong một nhóm làm việc của bạn , trong một công ty lớn, hay hay cần phân bố chương trình ứng dụng rộng rãi trên toàn thế giới qua **Internet, VB6** cũng sẽ có các công cụ lập trình mà bạn cần thiết.

1.2. CÁC ẤN BẢN CỦA VISUA BASIC 6.0

Có ba ấn bản VB6: Learning, Professional và Enterprise. Chúng ta hãy gạt qua ấn bản **Learning**. Bạn có thể dùng ấn bản **Learning, Enterprisse** và **Professional**. Ấn bản **Professional** cung cấp đầy đủ những gì bạn cần để học và triển khai một chương trình **VB 6**, nhất là các **Control ActiveX**, những bộ phận lập trình tiền đề về rất hữu dụng cho các chương trình ứng dụng (**Application Programs**) của bạn trong tương lai. Ngoài đĩa

Compact chính cho VB6, tài liệu đính kèm gồm có sách **Visual Studio Professional Features** và hai đĩa **CD Microsoft Developer Network (MSDN).**

Ấn bản **Enterprise** là ấn bản **Professional** cộng thêm các công cụ **Back Office** chẳng hạn như **SQL Server, Microsoft Transaction Server, Internet Information Server**.

Đặc Điểm của Visual Basic:

Khác với môi trường lập trình thủ tục trước đây trong hệ điều hành **Dos** như trong **Pascal** chạy **Foxpro**, **Visual Basic** là môi trình hướng biến cố trên hệ điều hành W**indows**.

Sự khác nhau giữa hướng thủ tục hướng biến cố.

Trong các môi trường lập trình hướng thủ tục, người lập trình phải xác định trước tuần tự thực hiện tuần tự của từng lệnh và từng thủ tục có trong chương trình. Có nghĩa sau lệnh này họ phải thực hiện lệnh tiếp theo, Với môi trường lập trình hướng biến cố như **Visual Basic** thì người lập trình chỉ việc định nghĩa những lệnh nào cần thực hiện khi có một biến cố do người dùng tác động lên chương trình mà không quan tâm đến tuần tự cách xử lý nhập liệu`.

Ví dụ: Một chương trình đơn giản như hình dưới đây có mục tiêu là nhập vào hai số cộng và đơn giá công việc, tính và in ra tiền công phải trả. Với hướng thủ tục, người lập trình phải viết ra các lệnh theo tuần tự được xác định như sau:

- **Chờ người dùng nhập vào giá trị cộng**
- **Chờ người dùng nhập vào giá trị đơn giá công việc**
- **Tính tiền công = Số công * Đơn giá công việc**
- **In giá tiền**

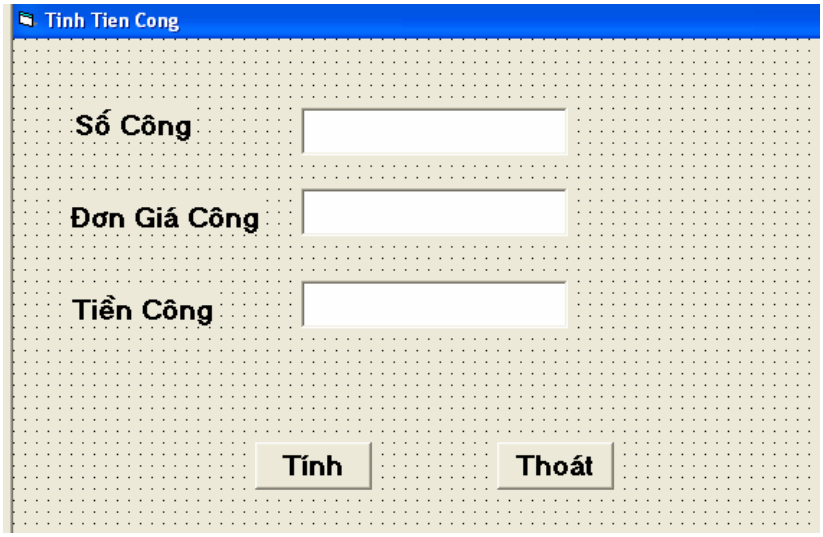
Trong khi đó, với môi trường lập trình hướng biến cố người lập trình sẽ không quan tâm tuần tự thực hiện của các lệnh nhập mà chỉ định nghĩa các lệnh xử lý tương ứng với các lệnh xảy ra

- **Biến cố khi người dùng nhấn chuột tại nút tính**

- Tính tiền công = Số công * Đơn giá
- Gán giá trị tiền công vào ô tiền công

- **Biến cố khi người dùng nhấn chuột tại nút thoát**

- Thoát khỏi ứng dụng



Hình 1.1 Ví dụ cơ bản

1.3 HƯỚNG DẪN CÀI ĐẶT VISUAL BASIC:

Khi thực hành chương trình **Setup** , một chương trình được tạo ra cho **Visual Basic** sau đó ta có thể chọn ra các thành phần của **Visual Basic** mà ta muốn cài đặt

Trừ một số ngoại lệ của tập tin hệ điều hành trong thư mục , các tập tin trên đĩa CD không bị nén, vì vậy chúng có thể dùng trực tiếp trên đĩa

Ví dụ : Thư mục **Tools** chứa vô số các công cụ và các thành phần có thể được thi hành hoặc cài đặt trực tiếp từ **CD_ ROM** .

1.3.1 CÀI ĐẶT TỪ CD:

Đặt đĩa CD vào ổ CD_ROM

Dùng lệnh thích hợp với môi trường hệ điều hành để thi hành chương trình Setup chứa trong thư mục gốc trong ổ đĩa số 1. Nếu hệ thống hỗ trợ Autoplay, Chương trình Setup sẽ tự động nạp khi ta cho đĩa vào ổ đĩa.

Chọn **Install Visual Basic 6.0**

Lần lượt làm theo các hướng dẫn trên màn hình.

1.3.2 THÊM HOẶC XOÁ CÁC THÀNH PHẦN TRÊN VISUAL BASIC:

Ta có thể thi hành lệnh **Setup** trong nhiều lần. Ví dụ: Ta có thể chương trình Setup để cài lại chương trình **Visual Basic** trong thư mục khác hoặc là để cài đặt các phần khác của **Visual Basic**.

- Đặt Đĩa CD vào ổ **CD_ROM,** Dùng lệnh thích hợp với môi trường hệ điều hành để thi hành chương trình **Setup** trong thư mục gốc có trên đĩa CD. Nếu hệ thống hỗ trợ **AutoPlay**, chương trình **Setup** sẽ tự động nạp khi ta đặt đĩa **CD** vào ổ đĩa.
- Chọn nút **Custom** trong hộp thoại **Microsoft Visual Basic 6.0 Setup**.
- Chọn các thành phần cài đặt (hoặc bỏ chọn các thành phần bị xoá) trong hộp danh sách **Options** của hộp thoại **Custom**.
- Lần lượt làm theo các chỉ dẫn trên màn hình.

1.3.3 Khởi động Visual Basic.

Sau khi đã hoàn tất các thủ tục cài đặt, ta có thể khởi động **Visual Basic** bằng cách sử dụng nút **Start** trên thanh công cụ của **Windows** . Nếu hệ thông hỗ trợ **AutoPlay**, ta có thể khởi động **Visual Basic** bằng cách đặt đĩa **Visual Basic** vào ổ đĩa.

Ghi chú :

Trong lúc cài VB6, nhớ chọn **Graphics** nếu không bạn sẽ thiếu một số hình ảnh như **Icons**, **Bitmaps** v.v... Đáng lẽ **Microsoft** chọn tự động cài đặt **Graphics**, tức là **Default** (không có nói gì) thì cài đặt **Graphics**.

2. GIAO DIỆN WINDOWS CHÍNH CỦA VISUAL BASIC

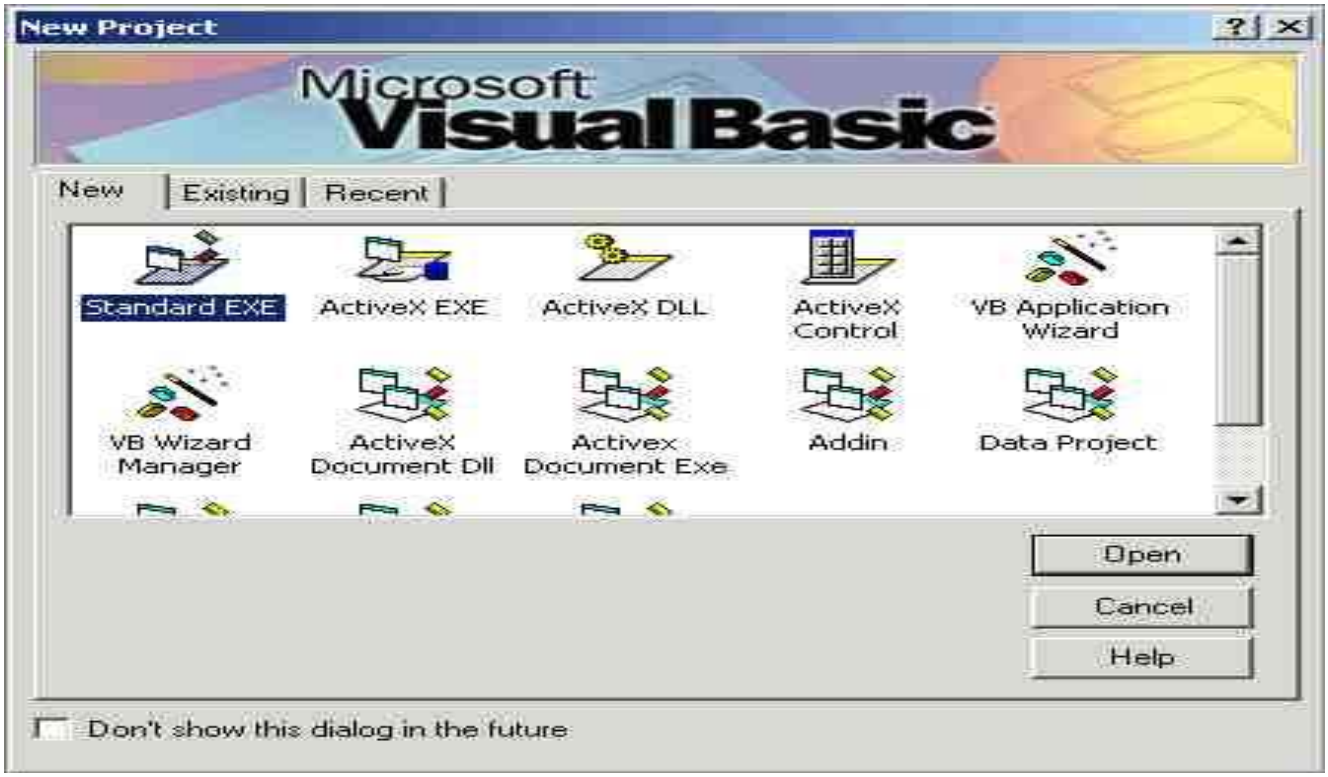
2.1 GIAO DIỆN WINDOWS CHÍNH CỦA VISUA BASIC

2.1.1 KHỞI ĐỘNG VISUA BASIC

Bạn có thể khởi động Visua Basic bằng nhiều cách như :

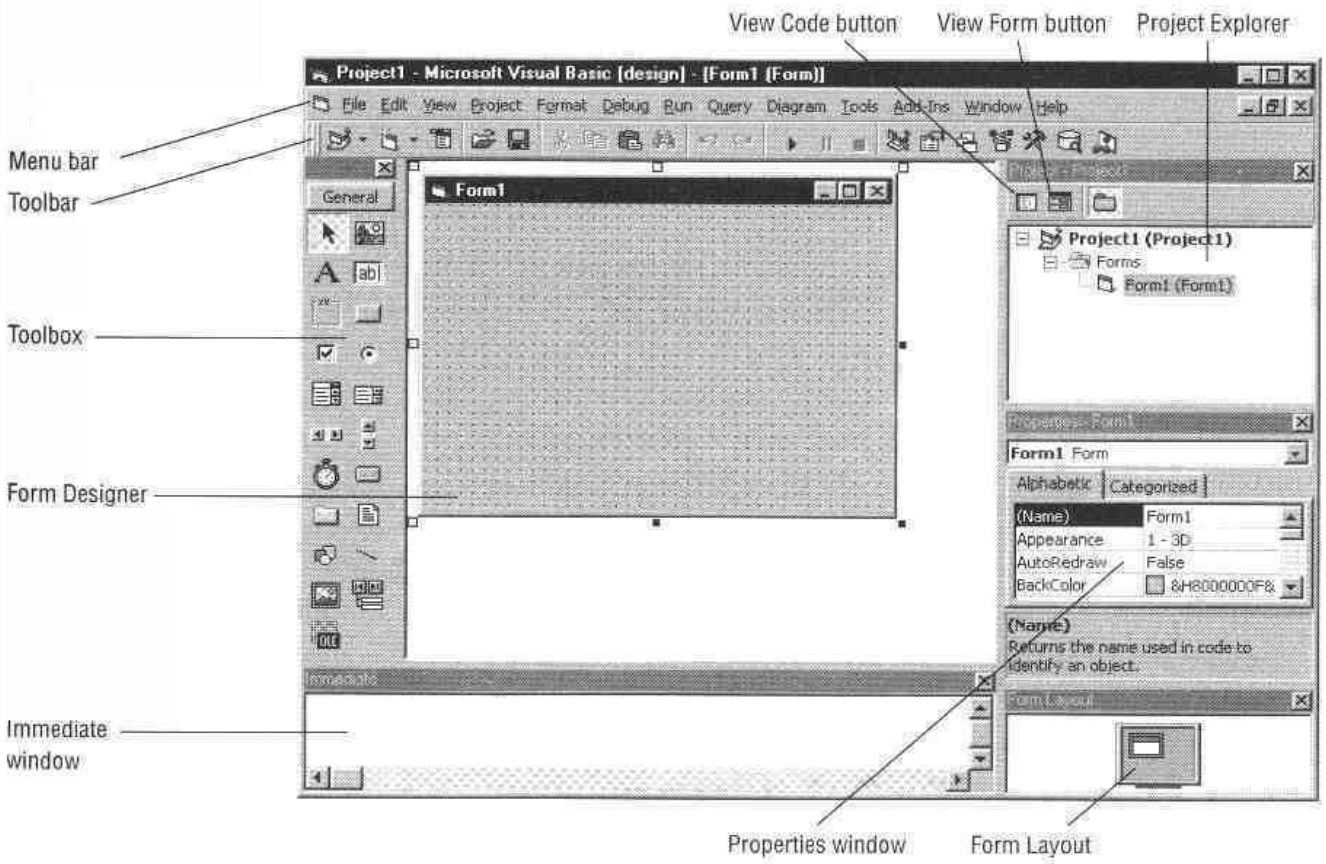
- Khởi động Visua Basic từ Start menu bằng cách chọn Program chọn Microsoft Visual Basic 6.0 chọn Microsoft Visua Basic 6. 0
- Double click vào biểu tượng VB trên Desktop
- Vào Statr chọn Run và nhập dường dẫn đến VB

Màn hình giao diện sẽ hiện lên như sau: Khi khởi động VB6 bạn sẽ thấy mở ra nhiều cửa sổ (**Windows**), **Scrollbars**, v.v.. và nằm chồng lên là **New Project Dialog**. Ở đây **VB6** cho bạn chọn một trong nhiều loại công trình.



Hình 2.1 Giao diện WinDows

Chọn Standard EXE. Một lát sau trên màn ảnh sẽ hiện ra giao diện của môi trường phát triển tích hợp (**Integrated Development Environment - IDE**) giống như dưới đây:



Hình 2.2 Môi trường **Standa EXE**

Newproject: Cho phép người lập trình mở ra **Project** cần tạo ra. Nếu mặc định chọn **Standar.EXE** mở **Button Open** hoặc **Ente**r. Nếu không tạo **Project** mới thì **Click** vào **Button Cancel**

Nếu người lập trình chọn **Newproject(Open)** thì một cái **Windows** con mang tên **Project** mặc định được tạo thành .Trên **Windows** sẽ có một **Form** mặc định là **Form** 1

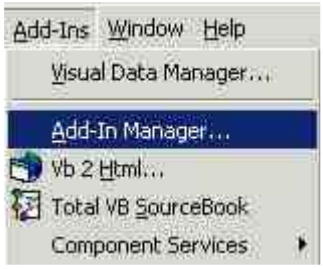
2.2 CÁC MENU CHÍNH CỦA VISUAL BASIC

- **File :** Có chức năng quản lý **Project ,Form ,Print**
- **Edit:** Cho phép các chức năng soạn thảo, lập trình (người lập trình có thể sửa chữađược)
- **View:**Cho phép người lập trình xem các mã nguồn đối tượng và tham khảo toàn bộ cấu trúc của **Project** và các thuộc tính của đối tượng (lệnh này cho phép người lập trình xem nhưng không sửa được)
- **Project:** Là **Menu** có chức năng quản lý các thao tác trong **Project** bao gồm tạo **Module** cũng như xoá bỏ nó . ngoài ra còn có chức năng tham khảo các thành phần trong Project

- **Format :** Là Menu cho phép người lập trình định dạng các đối tượng **Font** chữ , các hình thức kiểm tra lỗi
- **Debug :** Có chức năng kiểm tra lỗi hay cài đặt các hình thức kiểm tra lỗi .bạn có thể thi hành từng câu lệnh trong chương trình VB, xem giá trị dữ liệu và dừng chương trình ở bất cứ nơi đâu
- **Run:** Có chức năng cho phép người lập trình chạy chương trình biên dịch ,kết thúc
- **Windows:** Có chức năng cho phép người lập trình hiển thị các **Windows** con theo các chế độ khác nhau cung như xắp xếp các **Icon** trong một thư mục được mở hiện hành
- **Help:** Có chức năng cung cấp cho người lập trình hiển thị các thông tin về lệnh hướng dẫn lập trình tìm biến và thông tin về Version phần mềm hiện hành (phiên bản)

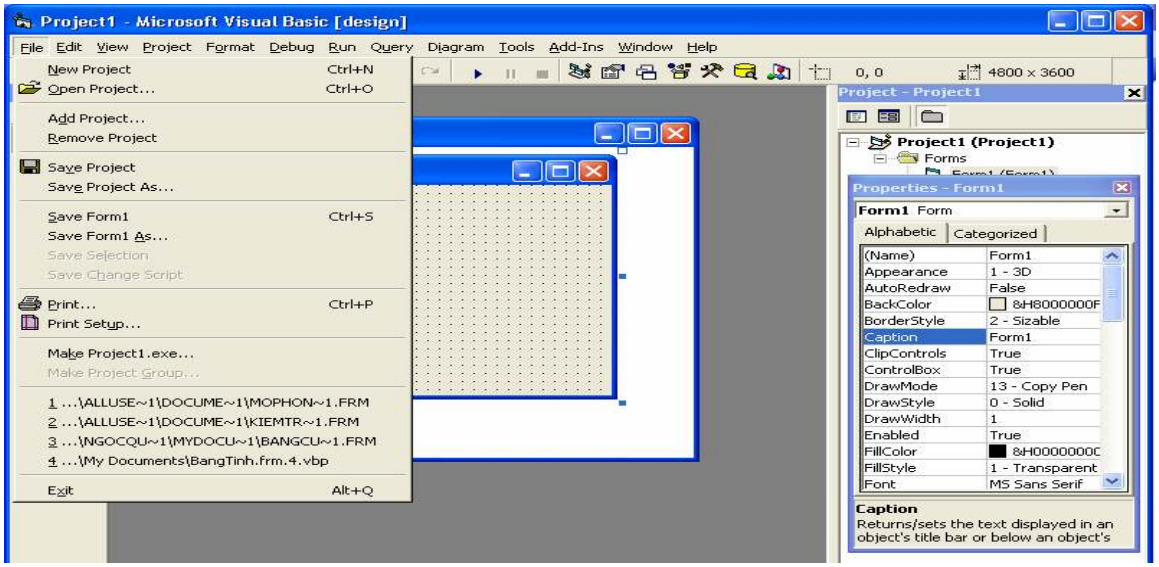
2.2.1 KHẢO SÁT MENU BAR

Chứa đầy đủ các commands mà bạn sử dụng để làm việc với VB6, kể cả các menu để truy cập các chức năng đặc biệt dành cho việc lập trình chẳng hạn như **Project, Format**, hoặc **Debug**. Trong **Menu Add-Ins có Add-Ins Manager** cho phép bạn gắn thêm những menu con giúp cho việc chạy các chương trình có ích cho việc lập trình.



Hình 2.3 **Menu Bar**

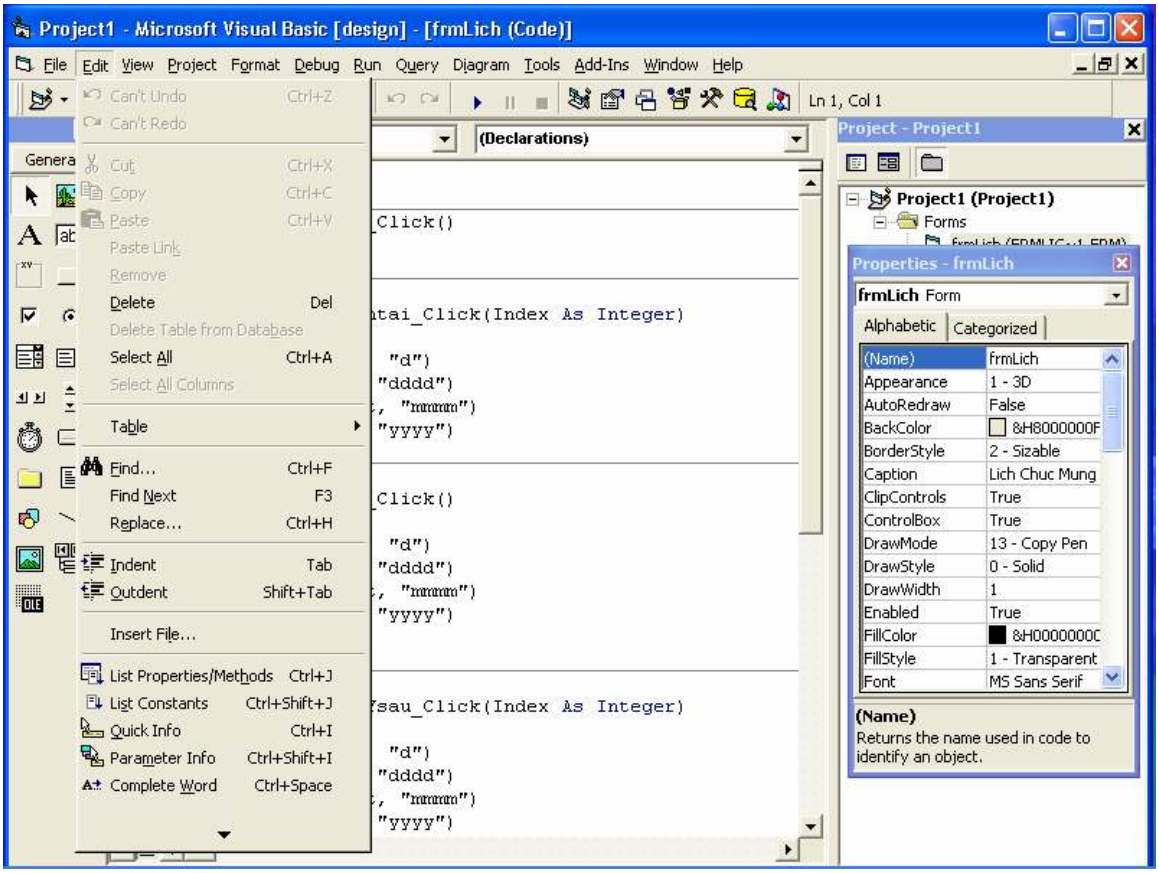
2.2.2 KHẢO SÁT MENU FILE



Hình 2.4 **Menu File**

- **New project (Ctrl +N): T**ạo ra **Project** mới .Khi tạo ra người lập trình phải xem nó chính như thư mục do mình tạo ra, để mở, xem, tạo thư mục và ghi thư mục **(Save) Open Project** mở một **Project** đã được lưu trong bộ nhớ máy tính .(Đường dẫn tên đường dẫn **Enter**)
- **ADD Project :** Cho phép người lập trình thêm một **Project** đã được tạo ra vào **Project**
- **Remove :** Cho phép người lập trình loại bỏ một **Project** khỏi chương trình **Visua Basic** hoặc xoá bỏ **Project** khỏi bộ nhớ máy tính
- **Save Project :** Cho phép người lập trình lưu lại vào bộ nhớ máy tính
- **Save Project/As:** Cho phép người lập trình ghi lại chương trình hiện hành vào một vùng nhớ của máy tính với tên mặc đinh hay tên do người lập trình đặt ra
- **Make EXE File :** Cho phép dịch một chương trình ra tập tin thi hành
- **Print :** Cho phép người lập trình in đối tượng hiện hành đang khảo sát
- **Print Setup:** Cho phép người lập trình định dang kích cỡ trang **Insurance**
- **Exit :** Cho phép người lập trình thoát khỏi chương trình **Visual Basic**

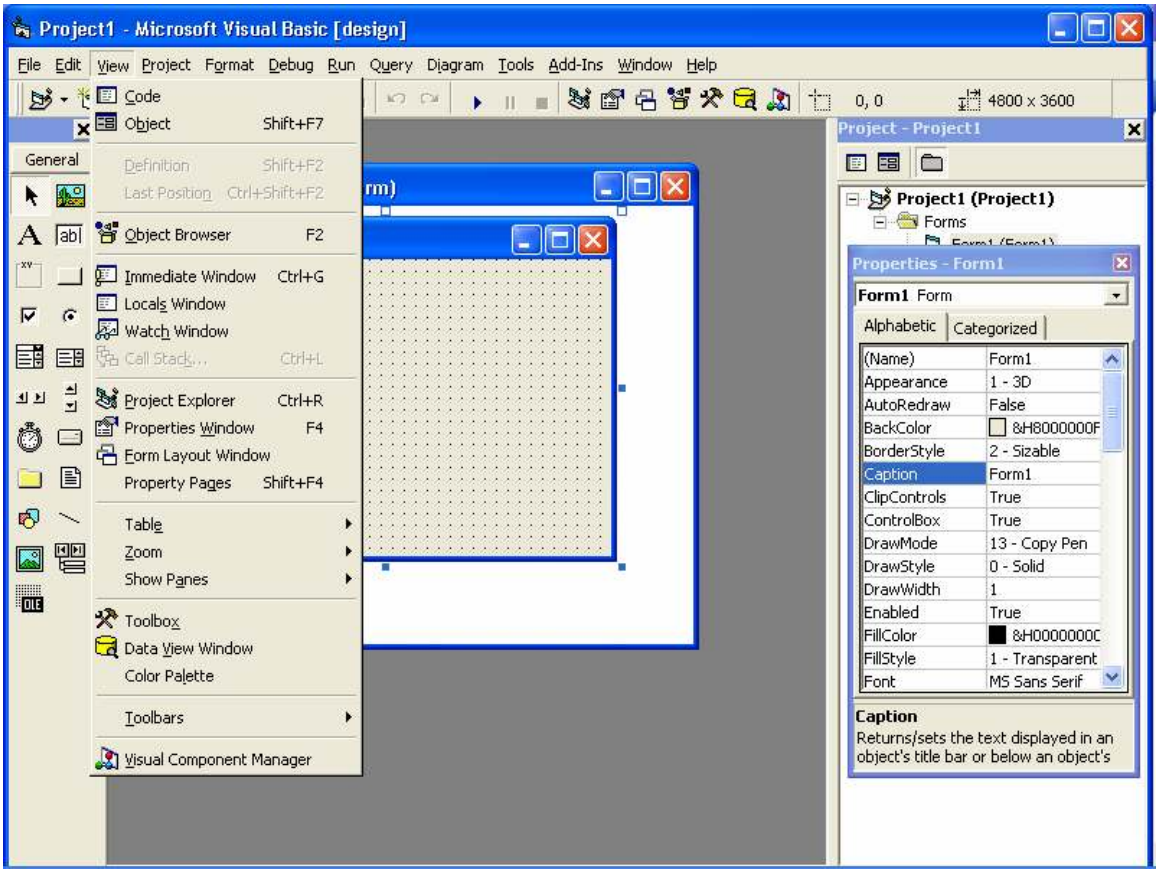
2.2.3 : KHẢO SÁT MENU EDIT



Hình 2.5 **Menu Edit**

- **Undo Insert Text :** Cho phép trở về trạng thái ban đầu trước khi **Insert** đoạn văn
- **Cut:** Cho phép người lập trình đánh dấu và cắt đoạn văn **Clip Board**
- **Copy:** Chức năng giống như **Cut** nhưng chỉ là sao chép
- **Past : Insert** toàn bộ **Clip Board** vào đối tượng hiện hành vào vị trí con trỏ nhập
- **Delete :** Loại bỏ đối tượng khỏi **Project**
- **Select All :** Lựa chọn toàn bộ văn bản hay các **Icon** trong thư mục được hiển thị
- **Table :** Cho phép người lập trình tạo ra bảng thống kê dữ liệu
- **Fild:** Cho phép người lập trình tìm kiếm một chuỗi ký tự trong hai mã nguồn của đối tượng hay toàn bộ **Project**
- **Fild Next :**Tiếp tục tìm kiếm chuỗi ký tự trong đối tượng hiện hành
- **Replace :** Thay thế chuỗi ký tự nhập vào chuỗi ký tự khác trong đối tượng hiện hành
- **Insert File:** Cho phép người lập trình chin toàn bộ nội dung một **File** được chi định vào đối tượng mã nguồn hiện hành

2.2.4 KHẢO SÁT MENU VIEW



Hình 2.6 :**Menu View**

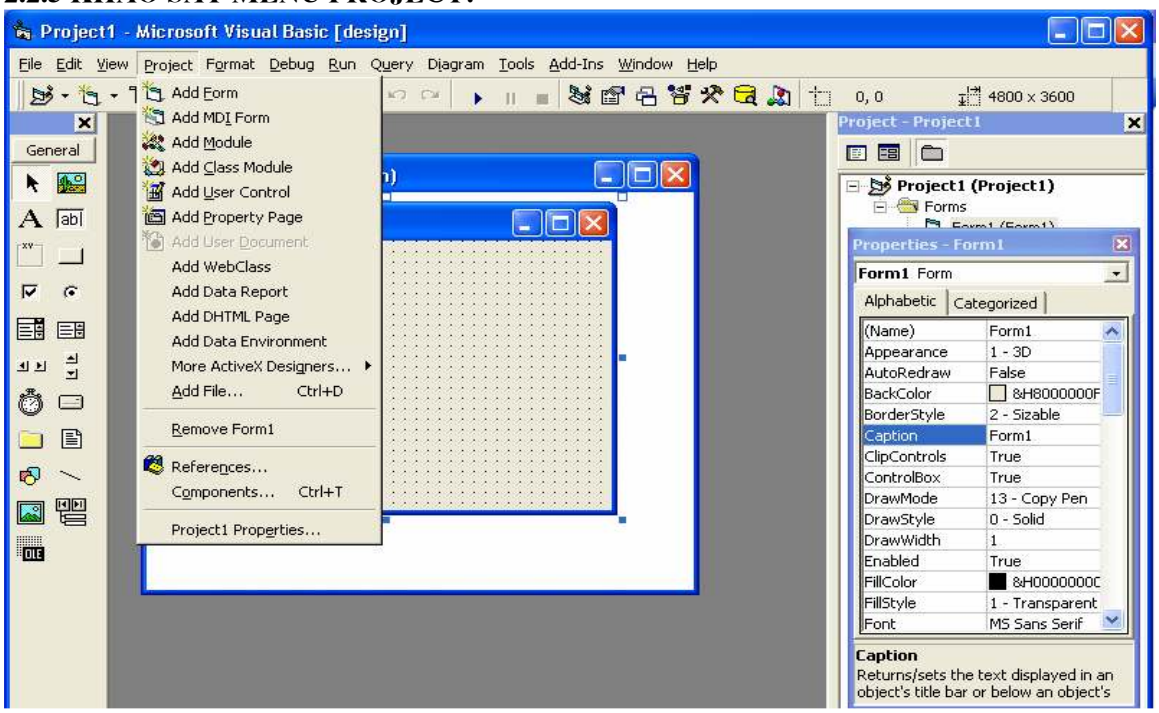
- **Code** cho phép liệt kê các **File** mã nguồn , cho phép liệt kê theo dạng cây thư mục các đối tượng trong **Project**.
- **Oject:** Là cửa sổ hiển thị **Form** theo dạng **Layout**
- **Oject Browser:** Cho phép tìm kiếm các đối tượng, danh sách các hằng có sẵn trong văn bản
- **Immedieate Window:** Cho phép người lập trình liệt kê trực tiếp trên nền **Windows**
- **Local Win dow:** Cho phép người lập trình liệt kê một cách cục bộ trên nền **Windows**
- **Watch Window:** Màn hình **Windows**
- **Project Explorer:**Cho phép lấy lại cửa sổ **Project** lên trên màn hình
- **Property Pages:** Cho phép trọn lựa trang thuộc tính **Property**
- **Table:** Liệt kê các thư mục như **ColumnProperties, ColumnNames, Keys, Name Only, Custom.**
- **Zoom:** Cho phép phóng to hay thu nhỏ đối tượng nên trên màn hình
- **Show Panes:** Liệt kê **Diagram,Grid, SQL, Results**.

- **Toolbox:**Cho phép hiển thị hay không hiển thị thanh công cụ **ToolBox** lên trên màn hình
- **Toolbars:**Cho phép liệt kê tất cả các công cụ **Tool** cho phép việc lập trình. Ở mỗi công cụ sẽ hiển thị trạng thái xem ta có sử dụng nó hay không.

Nếu ta chọn Tool nào thì đánh dấu vsào Tool đó. Nếu không sử dụng Tool nào thì xoá dấu lựa chọn ở Tool đó**.**

- **Colour Palette:** Cho phép người lập trình chọn màu cho đối tượng
- **DataView Window:** Cho phép người lập trình hiển thị cửa sổ thuộc tính Data Window

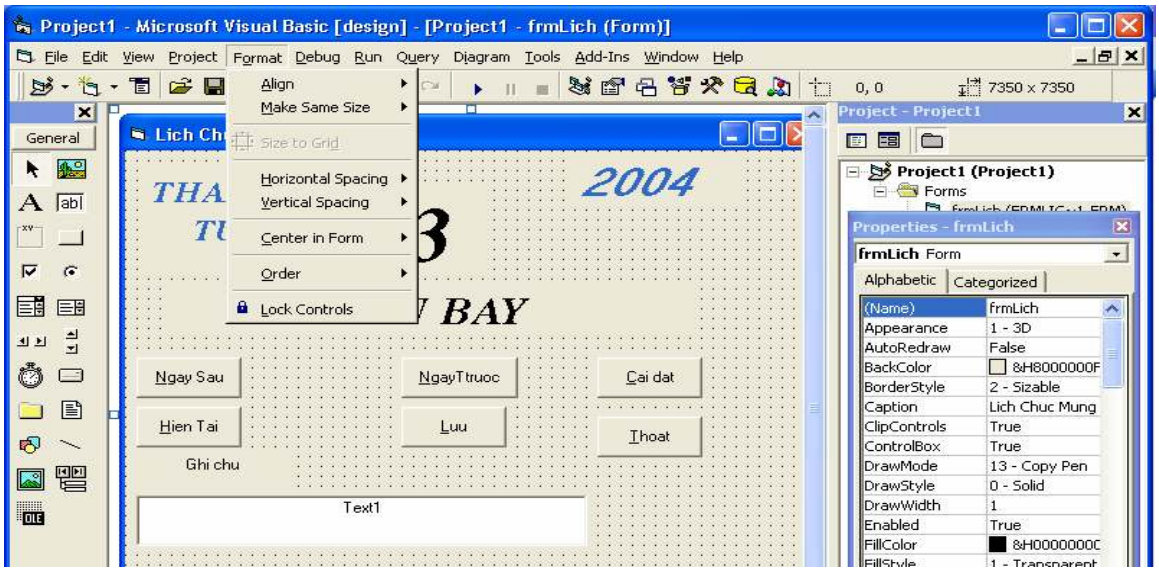
2.2.5 KHẢO SÁT MENU PROJECT:



Hình 2.8: Menu Project

- **Add Form:** Cho phép tạo một Form vào Project
- **Add MDI Form:** Cho phép người lập trình tạo MDI Form vào Project
- **Add Module:** Cho phép thêm một chương trình (Program) vào Program
- **Add Class Module:** Cho phép tạo một Class Module
- **Add Property Page:** Cho phép tạo một trangProperty
- **Add File:**Thêm File vào Project , File đó có thể là một Form của chương trình.
- **Remove Form:** Loại bỏ Form chỉ định khỏi chương trình Visual Basic hay Project.
- **Components:** Liệt kê các thành phần đang sử dụng trên phím
- **Project Properties: L**iệt kê các tính chất

2.2.6 KHẢO SÁT MENU FORMAT:

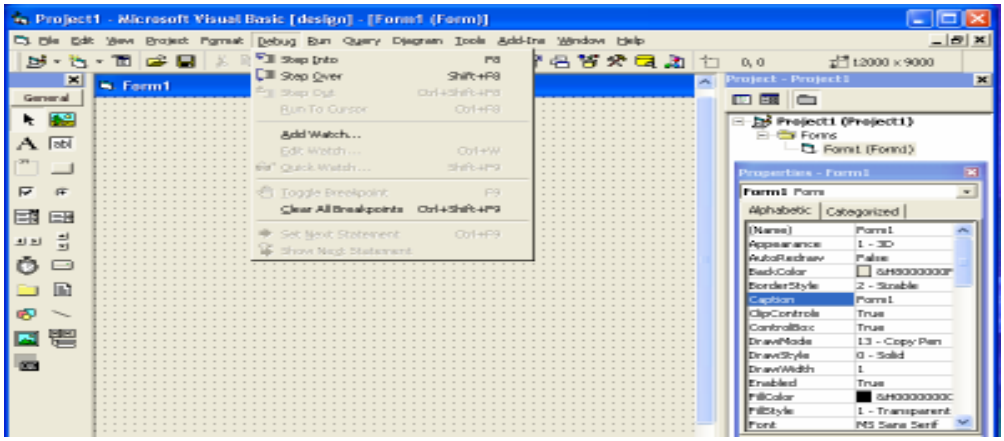


Hình 2.9 Menu Format

- **Align:** Cho phép sắp xếp các đối tượng theo hàng cột.
- **Make Same Size:** Cho phép tạo các đối tượng cùng kích cỡ.
- **Horizontal Spacing:** Cho phép thay đổi thanh cuốn ngang.
- **Vertical Spacing:** Cho phép thay đổi thanh cuốn dọc.
- **Center In Form:** Cho phép chọn đối tượng nằm ở vị trí giữa của Form theo chiều ngang dọc
- **Order:** Cho phép chọn tích cực cho đối tượng tạo ra trước sau
- **Clock Control:** Vô hiệu hoá các điều khiển trên Menu Format

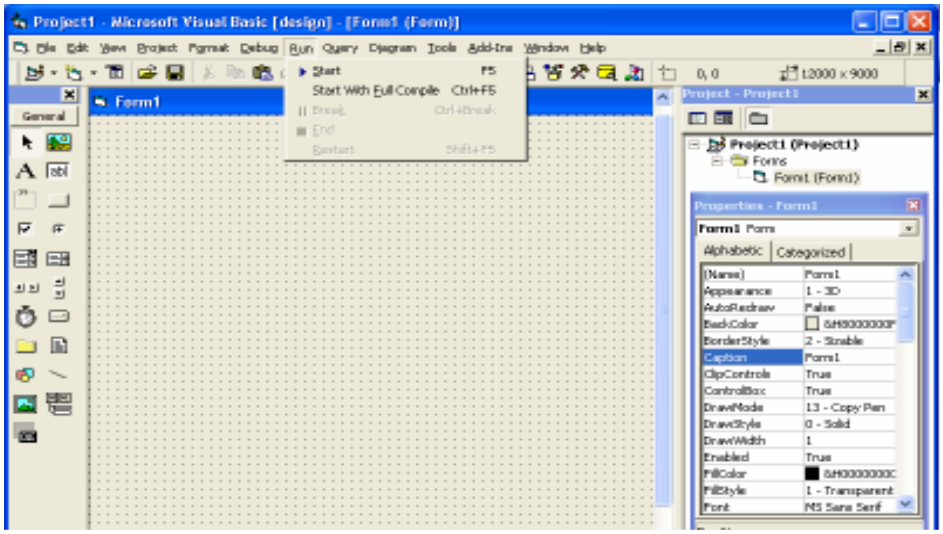
2.2.6: KHẢO SÁT MENU DEBUG:

Step into: (F8)
Step Over: (F8)
Run To Curser: (Ctrl + F8)
Add Watch..
Quick Watch..(Shift + F9)
Toggle Breakpoint: (F9)
Clear All Breakpoint: (Ctrl + Shifl + F9)



Hình 2.10 Menu Debug

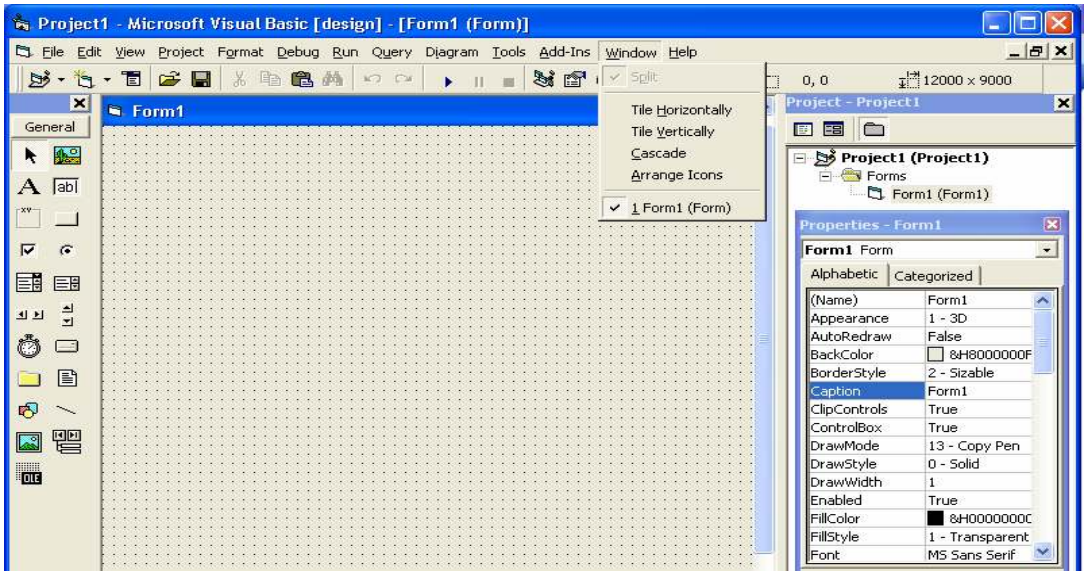
2.2.7: KHẢO SAT MENU RUN:



Hình 2.11: Menu Run

- **Start (F5):**
- **Start With Full Compile:** Chạy với sự phiên dịch toàn bộ những gì liên quan đến Project.
- **Break:** Dừng chương trình.
- **End:** Ngừng chương trình đang khởi động.
- **Restart:** Thoát khỏi tất cả các chương trình đang khảo sát trong Visual Basic

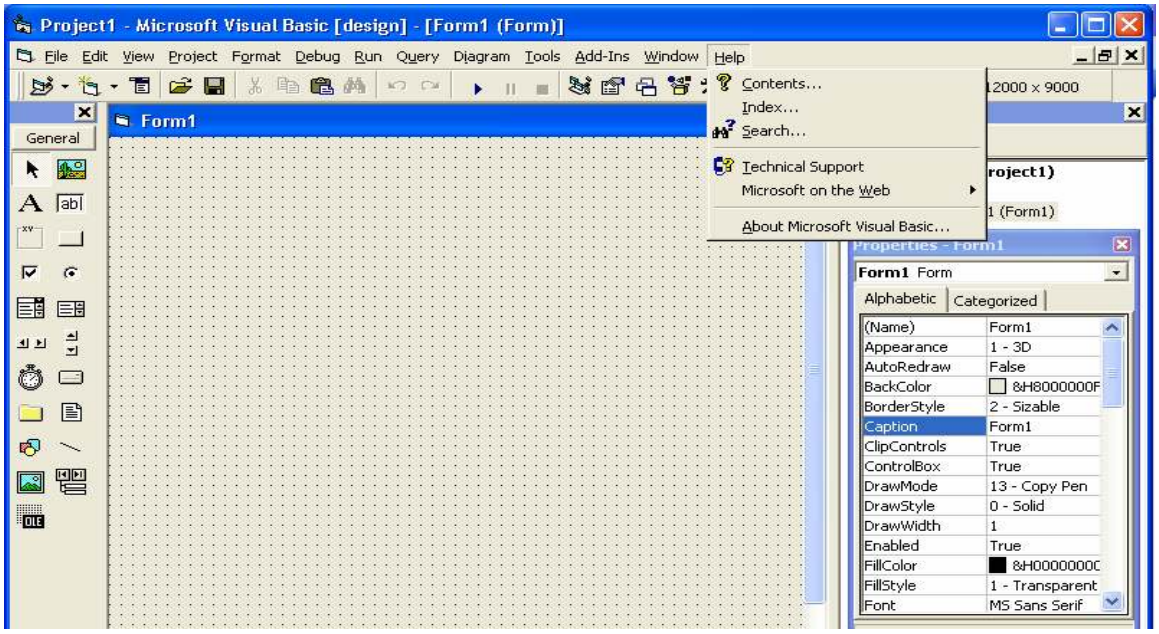
2.2.8: KHẢO SÁT MENU WINDOW:



Hình 2.12: **Menu Windows**

Có chức năng cho phép hiển thị các **Window** con theo các chế độ khác nhau cũng như sắp xếp các Icon trong một thư mục được mở hiện hành.

2.2.9: KHẢO SÁT MENU HELP:



Hình 2.13 **Menu Help**

- **Menu Help:** Có chức năng cho phép hiển thị, cung cấp thông tin về lệnh hướng dẫn lập trình, tìm kiếm và thông tin về
- **Content :**Cho phép người lập trình liệt kê các nội dung VB hỗ trợ cho người lập trình
- **Index:** Cho phép người lập trình gõ vào lệnh hay tên đối tượng, thuộc tính tên đối tượng hoặc tra thông tin về nó
- **Search:** Trong trường hợp thông tin người sử dụng tra cứu Index không được tìm thấy thì cho phép truy suất mở rộng thông tin tìm kiếm
- **Technical Support:** Có chức năng cung cấp thông tin hỗ trợ của VB cho người lập trình
- **About Microsoft Visual Basic:** Có chức năng cung cấp thông tin VB

2.2.10: THANH CÔNG CỤ TOOLBAR:

Thanh công cụ là tập hợp các nút bấm mang biểu tượng chứa trong một thanh thường đặt dưới thanh Menu. Các nút này đảm nhận các chức năng thông dụng trong cấu trúc Menu của Visual Basic. Thanh công cụ rất hữu ích, thay vì phải lần lượt chọn qua Menu và Menu còn ta nhấp một nút bấm nào đó trong thanh công cụ để gọi một chức năng tương tự trên Menu

Chức năng thanh này chứa các Icon nhỏ giúp người lập trình thực hiện nhanh thay thì phải vì phải vào các mục của menu. Người lập trình chỉ phải di chuyển vị trí của con trỏ đến vị trí của Icon và nó sẽ hiện ra

Hình 2.14: Thanh công cụ **Toollbal**

 **Addstandard.EXE:** Tạo **Project** mới, nhập mũi tên xuống bạn có thể chọn các công cụ khác

 **Add Form:** Thêm một **Form** cho **Project**, nhập mũi tên bạn có thể chọn các công cụ khác

 **Menu Editor :**Dùng để thiết kế **Menu** cho chương trình của biểu mẫu hiện hành

 **Open Project :** Dùng để mở một **Project** có sẵn

 **Save Project :**Dùng để lưu một **Project**

 **Cut :** Dùng để cắt bỏ các câu lệnh hoặc các đối tượng đã chọn

 **Copy :** Dùng để sao chép một đối tượng hoặc các câu lệnh đã chọn

 **Paste :** Dùng để dán các câu lệnh hoặc các đối tượng đã chọn

 **Undo :** Dùng để lấy lại hành động trước đó nếu có thể

 **Redo:** Dùng để lấy lại hành động sau đó nếu có thể

 **Start :** Dùng để chạy chương trình sau khi bạn đã thiết kế hoặc chạy thử

 **Break :** Làm ngưng chương trình đang chạy

 **End :** Cho phép chấm dứt chương trình đang hiện hành

 **Project Explorer :** Dùng để xem các Project ,các Form bạn có thể tuỳ chọn

 **Properties Windows :**Đưa ra cửa xổ để bạn xác lập các thuộc tính cho các đối tượng trong hộp **Toolbox** và **Form**

 **Form Layout Windows :** Dùng để điều khiển vị trí xuất hiện của biểu mẫu khi bắt đầu chạy chương trình

 **Objject Browser :** Dùng để mở hộp thoại **Objject Browser**

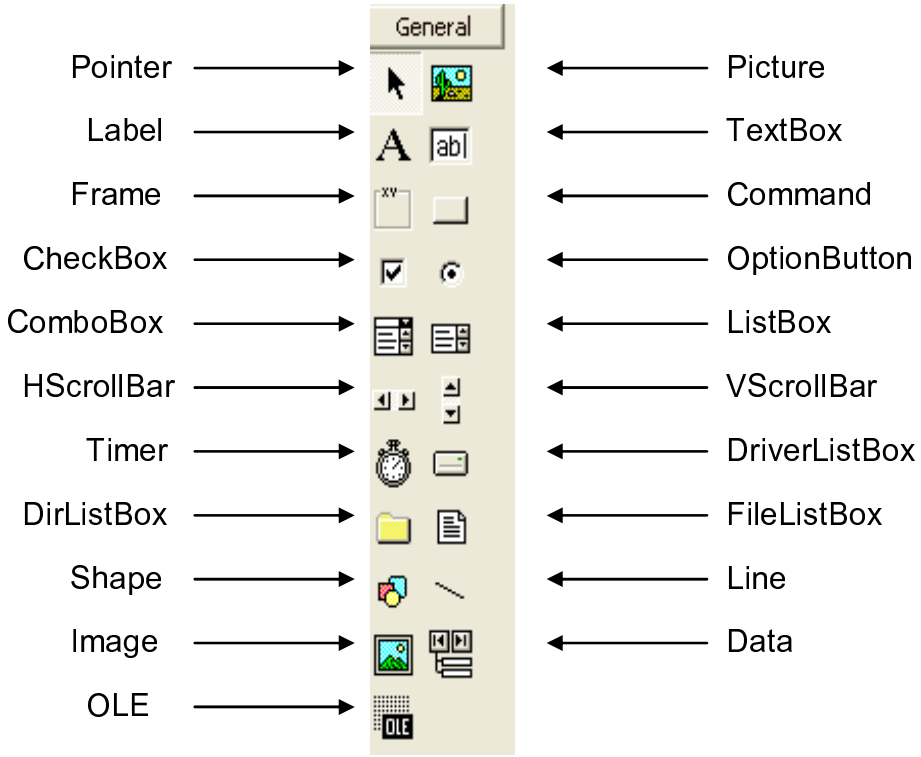
 **Toolbox :**Xuất hiện hộp công cụ phía bên trái màn hình

 **Data View Windows :** Xuất hiện cửa sổ **Data View** để quản lý cơ sở dữ liệu

 **Visual Component Manager:** Xuất hiện **VB Component Manager** để quản lý các đoạn mã của ứng dụng phức tạp

2.2.11: HỘP CÔNG CỤ TOOLBOX:

Hộp công cụ là bảng chứa các điều khiển mà ta thiết kế giao diện người sử dụng bằng cách chọn các điều khiển từ hộp công cụ và đưa chúng vào biểu mẫu



- Muốn hiển thị thanh công cụ, từ menu View, chọn Toolbox, hoặc ấn chuột lên Icon.
- Khi hộp công cụ hiển thị ta có thể di chuyển hộp công cụ xung quanh màn hình bằng cách nhấn trên thanh tiêu đề của nó (Tilebar) giữ chuột và kéo đến nơi ta muốn và thả chuột.
- Muốn đóng hộp công cụ, nhấn chuột nên nút đóng (nằm trên góc phải thanh tiêu đề).

2.2.12 MỘT SỐ KHÁI NIỆM CƠ BẢN

Đối tượng (Object): Là một thực thể được đặc trưng bởi một tập các tính chất(Properties và Method hay Behavier) và hành vi của nó.

Thuộc tính (Property): Là bộ các thông số mà ta có thể gán cho điều khiển. Ví dụ như tên, chiều rộng, chiều cao. Ta có thể xem toàn bộ thuộc tính của một điều khiển bằng cấch chọn vào nó và ấn F4 để mở cửa sổ thuộc tính.

Phương thức(Method): Là những phản ứng của đối tượng lên môi trường ngoài khi có sự tác động từ môi trường ngoài lên nó.

Sự kiện (Event): Là những quá trình xuất hiện trong khi khả thi chương trình, mà quá trình này đã được xác định trước và được cài đặt cho nó các hành vi tương ứng.

Thế mạnh của **VisualBasic** là khi sử dụng điều khiển và tận dụng khả năng lập trình của chúng. Một điều khiển thực chất là một cửa sổ được lập trình sẵn bên trong. Không có gì khác giữa một ứng dụng và một điều khiển. Để thi hành một ứng dụng, ta mở một cửa sổ. Ứng dụng sẽ chiếm điều khiển trên cửa sổ đó và hoạt động thông qua giao diện cũng như những chức năng của nó. Một điều khiển cũng thực hiện tương tự như thế.
Một điều khiển chứa đựng một chương trình được lập trình sẵn và chương trình này có thể tích hợp một cách dễ dàng vào ứng dụng có sử dụng điều khiển. Trước đây lập trình viên thường phảI tự xây dựng toàn bộ Module cần thiết cho chương trình. Điều này có nghĩa là các lập trình viên khác cũng phảI lập lại chương trình đó. Trong khi đó PC, máy tính cá nhân được cấu tạo từ vô số thành phần được cung cấp bởi nhiều nhà sản xuất khác nhau, mỗi thành phần có một công cụ đặc biệt. Khái niệm điều khiển của **VisualBasic** mang ý tưởng như thế. Từng điều khiển có thể được hiệu chỉnh và được tích hợp lại với nhau tạo thành một ứng dụng.
So với các điều khiển có sẵn trong hộp công cụ, mỗi điều khiển hiệu chỉnh (custom control), hay một điều khiển ActiveX là một thành phần có khả năng phát huy cao hơn và sâu hơn các tính năng hiện tại của môi trường. Bằng cách thêm một điều khiển ActiveX vào hệ thống ta đã mở rộng năng lực và tiện ích của môi trường **VisualBasic** duy nhất một lập trình viên có quyền thêm những điều khiển mà họ thích vào công cụ.
Vì là những điều khiển ActiveX nên chúng có thể dùng lại một cách dễ dàng bởi các ứng dụng ActiveX như là bộ **Office**, trình diện **Web Internet Explorer**,v.v… Các điều khiển này được cung cấp bởi phần mềm. Chúng có thể là một sản phẩm thương mại, hoặc được tải xuống miễn phí từ **Internet.**

- **Xét các thuộc tính mà ta thường dùng của các đối tượng trong Toolbox**

 **Poiter:** Dùng để điều khiển các đối tượng sau khi người lập trình đã tạo ra chúng như nhấp con trỏ chuột khi bạn muố chọn, định lại kích cỡ, di chuyển

 **Command Button:** Tạo một nút lệnh cho phép xử lý một số thao tác nào đó trên Form khi ta Click tại nút. Để chọn một nút lệnh ta có thể click tại nút hoặc nhấn Enter tại nút hay có thể nhấn Alt+ ký tự nóng.

- Phương thức click
- Sự kiện: MouseDown, KeyDown
- **Các thuộc tính**
  - Height, Font, Backcolor, Caption
  - Cation: Chứa một nội dung bất kỳ, nội dung này sẽ hiện trên nút nếu muốn sử dụng phím nóng ta đặt dấu & trước ký tự nóng(Hot Key)

 **Image:** Tạo một Farme chứa ảnh, dùng để hiển thị hình ảnh, người sử dụng có thể sử dụng các tập tin hình ảnh (Bmp , Dib), ở trongVB hoặc trong các tập tin bạn tự tạo để hiện thị ở trong đối tượng này.

- **Các thuộc tính:**
  - Picture: Cho phép chon tên một tập tin ảnh (Bmp, Dib),
  - Image: sẽ chứa một nội dung ảnh
  - Strectch: (True/ False) Cho phép hoặc không cho phép kéo giãn khung ảnh Image

 **Texxtbox: Dùng để tạo một ô bên trong để người dùng nhấp vào một nội dung nào đó hoặc xuất hiện thông tin khi hiện chương trình**

- **Các thuộc tính:**
  - Text: Chứa một nội dung để hiện trong Textbox.
  - MultiLine: (True/ False) Cho phép hay không cho phép Textbox chứa nhiều dòng(xuống dòng).
  - ReadOnly: (True/ False) Cho phép hay không cho phép nội dung bên trong Texxtbox là "chỉ đọc" hoặc có thể thay đổi đọc ghi được
  - ScrollBars: Quy định các thanh cuốn trong Texxtbox(chỉ có tác dụng khi Mutilline = True).
    0- None: Không có thanh cuốn nào
    1- Horizontal: Có thanh cuốn ngang
    2- Vertical: Có thanh cuốn dọc
    3- Both: Có cả hai thanh cuốn
  - TabIndex: Chứa một số nguyên (0 trở đi) quy định thứ tự nhận con trỏ.
  - TabStop: (True/False) cho phép hay không cho phếp con trỏ dừng trong
  - Textbox: Tuy nhiên ta có thể cho con trỏ vào bằng cách click chuột trong Textbox
  - MaxxLenght: Chứa một số nguyên quy định số thứ tự tối đa có thể nhận vào Textbox(Nếu = 0 thì không giới hạn).

 **Lable:** Dùng để ghi chú cho một đối tượng nào đó, hoặc thể hiện một dòng chữ khi chạy chương trình

- **Các thuộc tính:**
  - Caption: Chứa một nội dung bất kỳ, nội dung này sẽ hiện trên Lable
  - BackStyle: 0- Transparent: màu nền ẩn trên Control
  - Opaque: Màu nền hiện lại trên Control
  - FontItalic: (True/ False) cho phép hiện hoặc ẩn Font chữ nghiêng.
  - FontBold: (True/ False) cho phép hiện hoặc ẩn Font chữ đậm.
  - FontUnderline: (True/ False) cho phép hiện hoặc ẩn Font chữ nghiêng
  - FrontStrikethru: (True/False) cho phép hoặc không cho phép hiện Font gạch chân
  - Fontside: Chứa một số nguyên quy định cỡ chữ đến tối đa 2160 points.
  - WwordWrap: (Tue/ False) cho phép hay không cho phép Lable mở rộng kích thước theo chiều dọc để mở rộng nội dung.
  - FontName: Quy định tên một Font áp dụng cho Lable

CheckBox: Cho phép tạo một một hộp kiểm tra, cho phép chọn dấu X trong hộp nếu chọn đánh dấu thì trả ra giá trị 1 còn không trả ra giá trị 0

- **Các thuộc tính:**
  - Capton: Chứa một nội dung hiện trên Checkbox
  - Value: Dùng quy định dấu trên Checkbox hoặc nhận giá trị trả về từ Checkbox(1 hoặc 0)
    0- Uncheck: Hộp CheckBox không chọn(không đánh dấu).
    1- Checkded: Hộp CheckBox được chon có đánh dấu.
    2- Grayed: Hộp CheckBox có màu xám.

Chú ý : *Để chọn ta click chuột hoặc nhấn phím **SpaceBar, Checkbox** không có tính loại trừ nhau nghĩa là ta có thể chọn trên nhiều **Checkbox***

Optionbutton: Cũng giống như Checkbox ở trên nhưng khác ở chỗ là chỉ chon được một trong những Option(mang tính loại trừ).

 **PictureBoox:** Dùng để toạ một hộp chứa ảnh và đặt hình ảnh lên Form, cũng giống như đối tượng Image ở trên

- **Các thuộc tính:**
  - Picture: Cho phép tạo tên một tập tin ảnh (.Bmp, Dib..), Picturebox sẽ chứa nội dung của ảnh.
  - CurrentX: Chứa một số (single) là tọa đọ góc trái của Picture.
  - CurrrentY: Chứa một (single) là toạ độ góc trên của PictureBox

 **ComboBox:** Là một đối tượng kết hợp giữa Textbox Và Listbox. Trong đó người lập trình có thể chọn một mục nào đó trong danh sách có sẵn của nó hay có thể nhập một nội dung bất kỳ trong Textbox trên.

 **Timer:** Tạo bộ đếm thời gian có thể xử lý thời gian trôi qua trong lúc lập trình chạy. Timer thể hiện ở chế độ "tàng hình".

- **Các thuộc tính:**
  - Enable: (True/ False) có hoặc không có hiệu lực
  - Interval: Chứa một giá trị số là khoảng thời gian đếm tính bằng MiliSecond (1/1000 giây) cũng là thời gian lập lại xử lý

***Chú ý**: Ta có thể tạo nhiều **Timer** trên Form, các **Timer** xử lý trên các khoảng thời gian khác nhau.*
Viết một chương trình xử lý cho Timer bằng cách Double Click tại Timer.

 **Fame:** Nó chỉ trình bày một khung chữ nhật ở trên Form, thường dùng để chứa các Checkbox hay OptionButton như một nhóm giá trị lựa chọn. Các Checkbox hay Otionbutton như một nhóm giá trị lựa chọn. Các Checkbox hay Optionbutton trong một Frame hoàn toàn độc lập với các Check Box, Option khác

- **Các thuộc tính:**
  - **Caption:** Chứa một nội dung bất kỳ, nội dung này sẽ hiện trên Frame
    Sử dụng: Click tại công cụ Frame đặt vào trong Form, thực hiện nhiều lần click chọn công cụ CheckBox hoặc OptionButton thuộc về Frame

 **Line:** Chỉ trình bày một đường thẳng ở trên From, dùng nó để trang trí cho From thay vì phảI gọi phương thức Line**.**

- **Các thuộc tính:**
  - BorderStyle: Cho phép chọn một dạng đường (chọn 1 trong 7 dạng).
  - BorderWidth: Chứa một số nguyên từ(1 đến 32767 tính bằng pixel) quy định độ rộng của đường.
  - BorderColor: cho phép chọn một màu cho đường thẳng.
  - X1: Chứa toạ độ ngang của góc trái
  - X2: Chứa toạ độ ngang của góc phải
  - Y1: Chứa toạ độ dọc của góc trên
  - Y2: Chứa toạ độ dọc của góc dưới

 **Shape:** Giống như đối tượng Line, đối tượng Shape cũng được dùng để trình bày From, nó thể hiện hình chữ nhật, hình vuông, hình Ellipse, hình tròn tuỳ theo chúng ta chọn.

- **Các thuộc tính:**
  - Shape: Cho phép chọn một hình vẽ:
    1- Square: Vẽ hình vuông
    2- Oval: Hình Ellipse
    3- Cricle: Vẽ hình tròn
    4- Rectangle: Vẽ hình chữ nhật
    5- Rounder Rectangle: Vẽ hình chữ nhật tròn góc
  - FillColor: Cho phép một màu tô nền
  - FillStyle: Cho phép chọn một dạng tô nền
    1- Solid: Mẫu thuần theo Fillcolor
    2- TransParent: Màu theo nền của From
    3- Horizontal Line: Tô sọc ngang
    4- Upward Diagonal:Tô sọc chéo từ dưới và từ trái qua phải (\\\)
    5- Downward Diagonal: Tô sọc tréo từ dưới nên và từ trái qua phải (///).
    6- Cross: Tô đường ca rô
    7- Diagonal Cross: Tô đường ca rô chéo

 **DriveListBox :** Là một CombooBox trong đó liệt kê tất cả các tên có trong hệ thống, nó dùng để chọn ổ đĩa dễ dàng**.**

- **Các thuộc tính:**
  - Name: Chứa tên đặt cho DriveListbox.
  - Drive: Chứa một chuỗi là ổ đĩa (Volume) đã chọn hặc cài đặt mặc định ổ đĩa làm việc**.**

 **DirlistBox:** Là một Listbox trình bày cấu trúc trong ổ đĩa đang chọn hiện thời, dùng để chọn thư mục.

- **Các thuộc tính:**
  - Name: Chứa tên đặt cho DirListBox
  - Path: Chứa các thông tin về thư mục

 **Fist ListBox:** Là một Listbox trình bày các File trong thư mục nào đó.

- **Các thuộc tính:**
  - Name: Chứa tên đặt cho FileListbox.
  - FileName: Chứa tên đã chọn.
  - Path: Chứa các thông tin về thư mục hiện hành

- Archive: True/ False cho/ không cho hiện các tập tin có thuộc tính Archive (thuộc tính thường "Normal" và thuộc tính chỉ đọc"Read only")
- Hidden:True/ False cho\ không cho hiện các tập tin có thuộc tính Hidden (thuộc tính ẩn Hidden) và thuộc tính hệ thống System.
- Normal: True/ False: Cho/ không cho hiện các tập tin có thuộc tính thường.
- System: True/ False: Cho/ không cho hiện các tập tin có tính hệ thống.
- Pattern: Chứa một dạng mẫu quy định các tập tin hiện trong FileListBox phải tuân theo dạng này (Ví dụ: *.*; exe; *.* .txt).
- Multiselect: Quy định cách chọn nhiều tập tin trong danh sách:
  1. None: Không cho phép chọn nhiều tập tin
  2. Single: Cho phép chọn nhiều tập tin trong danh sách và huỷ chọn lần lượt từng giá trị.
  3. Exxtended: Cho phép chọn nhiều tập tin trong danh sách và huỷ trọn một lần các tập tin dã chọn.

Chú ý:

- Chọn các tập tin liên tục nhau : Đè Shift chon tập tin đầu và tập tin cuối.
- Chọn xen kẽ : Đè Ctrl, chọn các tập tin tiếp theo
- Khi Multiselect =1 – Single hoặc 2- Extended thì thuộc tính Filename chỉ chứa tập tin huỷ chọn hoặc chọn gần nhất.
- Khi FileListbox có kích thước nhỏ không chứa hết các giá trị ta sẽ thấy một thanh cuộn dọc hiện trên FileListBox

 **HscrollBox :**Thanh cuộn ngang cho ta chọn một số nguyên khi ta di chuyển con chạy (từ giá trị Min đến giá trị Max)

 **Vscrollbox:**Thanh cuộn dọc cho ta chọn một số nguyên khi ta di chuyển con chạy (từ giá trị Min đến giá trị Max)

 **DaTa:**Dùng để kết hợp với các cơ sở dữ liệu khác như FoxPro,Access, để lấy thông tin OTL Container:cho phép thêm chức năng lập trình của một điều khiển vào ứng dụng

 **LitsBox:**Thường dùng để liệt kê một danh sách gồm nhiều thư mục và cho phép chọn lựa từ các Table

Các thuộc tính:

- ✓ List:Chứa một hay nhiều dòng giá trị chọn
- ✓ Text:Chứa giá trị trả về khi chọn từ danh sách(chỉ một giá trị mà thôi) người ta dùng thuộc tính này để nhận giá trị trả về
- ✓ Sorrted:True/False = Có /Không sắp sếp thứ tự (tăng dần) cho các giá trị trong danh sách
- ✓ Mutilselect: Quy định cách chọn nhiều giá trị trong danh sách
  0-None: Không cho phép chọn nhiều giá trị

1-Single: Cho phép chọn nhiều giá trị trong danh sách và huỷ chọn lần lượt từng giá trị

2-Extended: Cho phép chọn nhiều giá trị trong danh sách và huỷ bỏ một lần các giá trị đã chọn

- ✓ LangeChange: Chứa giá trị (số nguyên) có thể thay đổi lớn nhất
- ✓ SmallChange: Chứa giá trị (số nguyên)có thể thay đổi nhỏ nhất

 **OLE Container:** Cho phép chèn vào From một đối tượng (có thể là hình ảnh**,** âm thanh, đoạn phim, tài liệu, đồ thị được nhúng Embed hoặc liên kết Link).

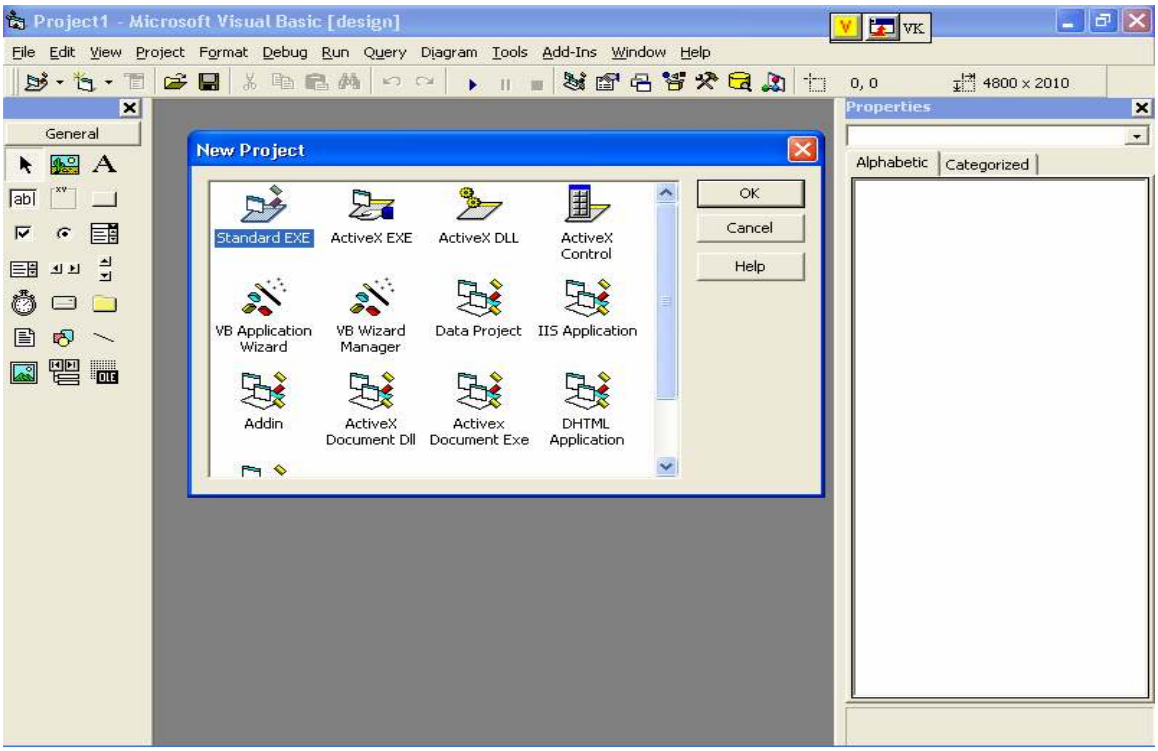
- **Các thuộc tính:**
  - ✓ OleTypeAllowed: Quy định kiểu của đối tượng Ole:
    1. Linked: Đối tượng là liên kết
    2. Embeded: Đối tượng là nhúng
    3. Either: Đối tượng là cả hai (Nhúng hoặc liên kết).
  - ✓ AutoActive: Định chế độ khởi động đối tượng:
    1. Manual: Ole sẽ không tự khởi động được.
    2. GetFocus: Ole ssex khởi động khi Control nhận con trỏ.
    3. DoubleClick: Ole sẽ khởi động khi Double Click trong Control.
    4. Automatic: Ole sẽ tự khởi động khi chạy chương trình
  - ✓ Display Type: Quy định chế độ hiển thị của Ole
    1. Content: Ole sẽ hiện dung của nó
    2. Icon: Ole sẽ hiện biểu tượng không có nội dung.
  - ✓ Size Mode: Quy định kích cỡ đối tượng:
    1. Clip: Ole sẽ bị cắt theo khung (hiện một phần theo kích thước khung, phần còn lại bị khuất).
    2. Stretch: Ole sẽ kéo giãn cho vừa khung (to nhỏ tuỳ kích thước khung).
    3. Auto Size: Ole tự động thay đổi kích thước để chứa đủ đối tượng
    4. Zoom: Ole tự động thay đổi kích thước chứa đối tượng và cân đối tượng về dạng nguyên thuỷ để xem.
  - ✓ OLEDropAllowed:True / False cho phép hay không cho phép thao tác Drag và Drop (rê và thả) một Ole khác trên Ole này.

3. GIAO DIỆN CỦA FORM

3.1: GIỚI THIỆU VỀ FROM:

Trong VB, một biểu mẫu From là một cửa sổ mà bạn chuyên biệt hoá để tạo ra một giao diện người dùng của chương trình. Một biểu mẫu có thể chứa các trình đơn, nút lệnh, các lệnh liệt kê (List Box) thanh trượt (Scrollbar) và những đề mục khác mà ta có thể thấy trong một chương trình chuẩn trên nền Windows

Khi khởi động VB sẽ xuất hiện một cửa sổ cho ta chọn kiểu để thiết kế chương trình, giá trị mặc nhiên là kiểu **Standard.EXE**. ta **Click Open**.



Hình 3.1.1 Màn hình chính khi khởi động **Windows**

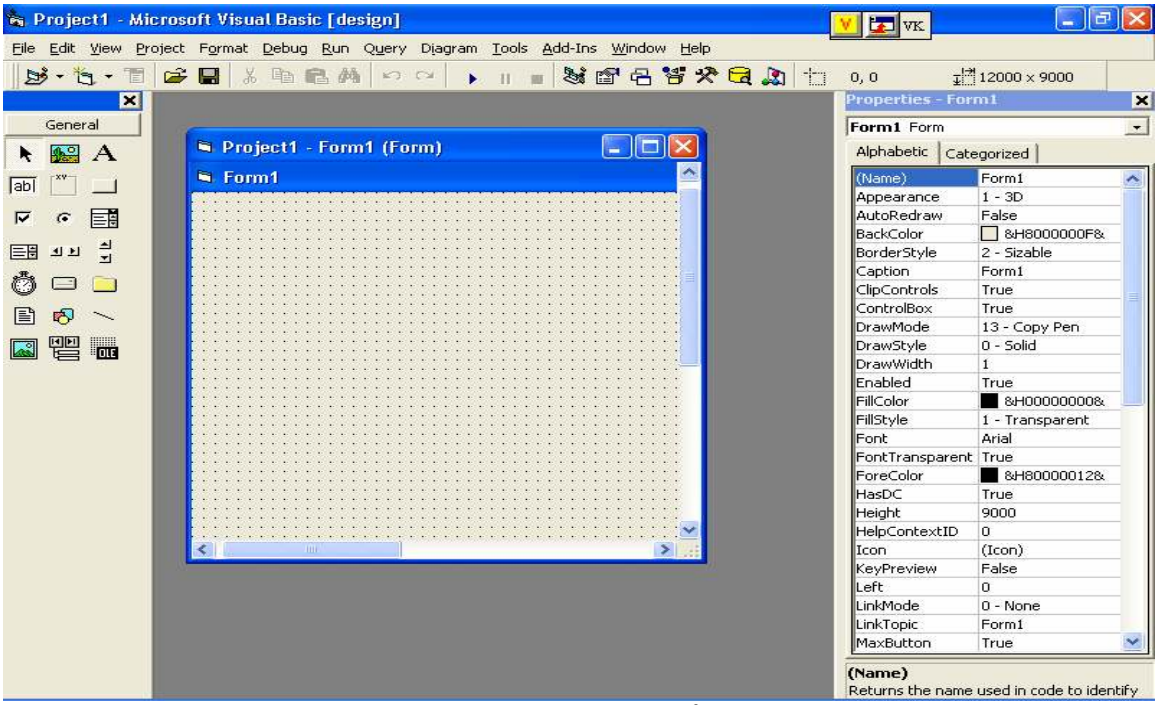
Cửa sổ khi ta khởi động VB:

Một cửa sổ From mới hiện ra trên màn hình với tên mặc định là From1. Người lập trình sẽ thiết kế trên Form1 này bằng cách đưa vào các điều kiểm lấy từ hộp công cụ ToolBox nằm ở bên trái màn hình, cửa sổ sẽ có hình dạng như sau:

Cửa sổ khi chọn kiểu thiết kế Standard.EXE

Trong trường hợp này ta sẽ khởi tạo VB rồi, sau khi đã thực hiện song một chương trình nào đó. Bây giờ ta muốn thiết kế một chương trình khác, ta vào **Menu File** chọn **New Project**. Thì quá trình cũng thực hiện hai cửa sổ như trên.

Trong trường hợp ta đã hoàn tất chương trình ở phần nào đó, nếu ta muốn vào để thay đổi hoặc thiết kế thêm ta khởi động VB, vào **Menu File** chọn **Project**, chọn chương trình mà ta cần mở. Khi **Click** chọn màn hình không hiện ra biểu mẫu. (Ta cũng có thể vào Menu File và nhìn ở phía cuối nếu có tên của chương trình cần mở thì ta Click vào chương trình gần nhất mà ta muốn thực hiện). Sau khi đã thực hiện như trên, để xuất hiện cửa sổ thiết kế ta có thể thực hiện theo hai cách sau:
Một là ta nhấn F5 hoặc là dấu mũi tên trên thanh công cụ để chạy chương trình, khi biểu mẫu chương trình đang thực hiện ta đóng chương trình lại . Sau đó vào **View** chọn **Object**.
Hai là ta mở cửa sổ **Project Eplorer, Click** chọn biểu mẫu (From) cần mở trong cửa sổ này, sau đó Click chọn View Object.



Hình 3.1.2 Cửa sổ **Form**

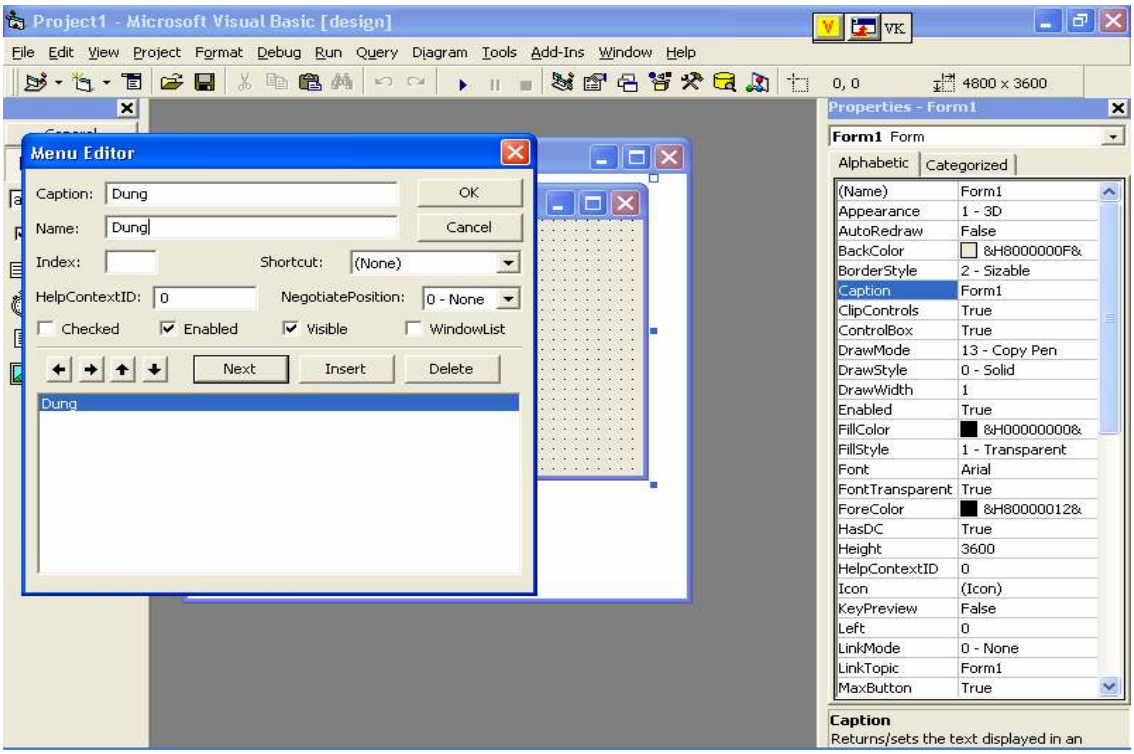
Khi cửa sổ thiết kế đã xuất hiện. Hộp công cụ ToolBox chứa các điều kiểm ở bên trong ta sẽ đưa các điều kiểm vào **Form** bằng bằng hai cách

<u>Cách 1:</u>
Ta **Double Click** vào điều kiểm tronh hộp công cụ mà ta muốn đưa vào Form dấu điều kiểm sẽ xuất hiện ở giữa Form
Ví dụ: Giả sử ta chọn điều kiểm **Command**, ta **Double Click** vào nó. Khi đó trên màn hình như sau sẽ xuất hiện Một **Command** 1
Nếu muốn xoá điều kiểm, ta Click chọn nó sau đó **Click** nút phải chuột, sau đó ta chọn Delete.

<u>Cách2:</u>

Ta Click chuột trái chọn điều kiểm trong Toolbox. Đưa con trỏ chuột vào trong Form lúc này con trỏ có dấu +.
Ta dùng chuột di chuyển dấu + đó đến vị trí nào đó Và Drag kéo đến khi có kích thước như ta muốn và thả nút trái chuột.
Sau đó ta có thể kéo đến vị trí nào đó. Hoặc có thể thưy đổi kích thước, cũng như xoá giống như các bước trên.

Tạo menu trên Form: Trên Form ta di chuyển con trỏ đến các Icon thì lập tức một Windows có tên Editor. Ta click chuột trái sẽ hiện ra



Hình 3.1.3: WinDows Editor

Caption: Chính là tên gọi được hiển thị bởi các Menu đó
Nane: Là tên chính của VB

<u>Chú ý</u>: Thông thường với các tên ngắn của Caption đều được đặt chung cho Name

- ✓ Index: Số thứ tự của Menu đó
- ✓ Short Cut: Là menu cho phép lựa chọn chế độ truy suất cho bàn phím tắt (hot key).
- ✓ Enabled: Cho phép hay không cho phép menu tích cực
- ✓ Menu: Được gọi là trạng tháI tích cực (Active) khi double Click vào Menu đó thì sẽ khả thi được khả năng tương ứng.

<u>Chú ý:</u>Trong VB những Menu nào không tích cực (Active) thì mở Visible: là tích cực cho phép Menu này có được hiển thị hay không hiển thị.

3.2: CÁC TÍNH CHẤT VỀ FORM:

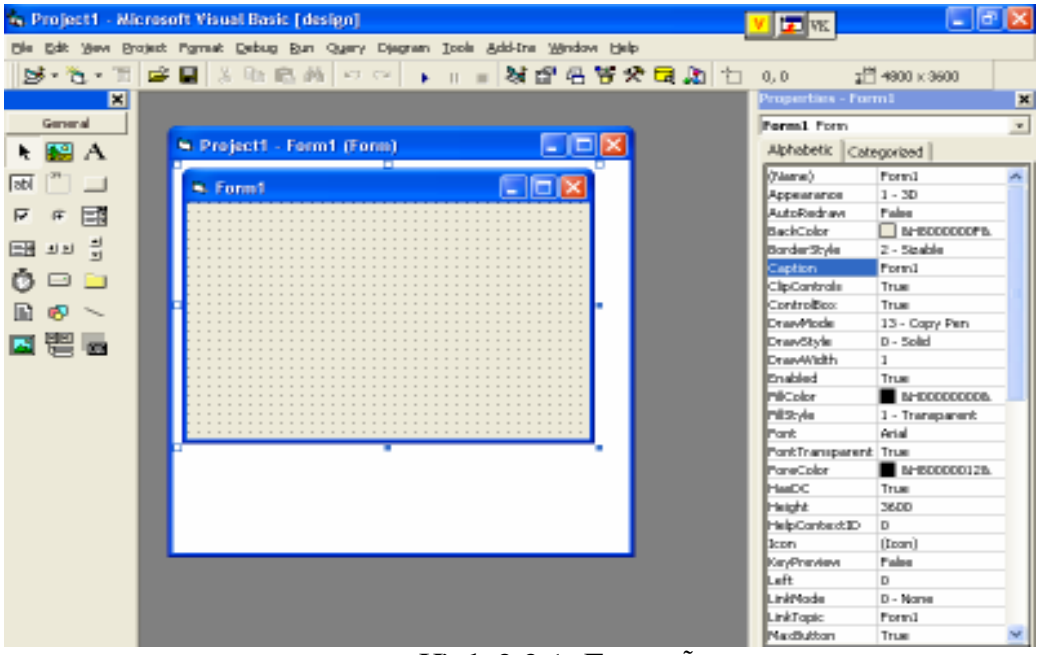
- ✓ **Name:** Cho phép người lập trình đặt tên cho Form
- ✓ **Appearance (xuất hiện):** Chế độ hiển thị gồm có chế độ hai chiều, ba chiều.
- ✓ **Backcolor (màu nền):** Hiển thị trạng tháI màu nền, giá trị của nó gồm có Red, Blue, Green và các màu odo chúng ta tự chọn trên Winzard.
- ✓ **BordeStyle:** (Khung Form) Quy định loại đường viền của Form gồm các loại
- ✓ **Sizable:** Cho phép thay đổi khung Form trong lúc chạy chương trình
- ✓ **None:** Không cho phép thay đổi kích thước trong lúc chạy chương trình
- ✓ **Caption:** Cho phép người lập trình đặt tên cho Form
- ✓ **ControlBox (hộp điều khiển):** cho phép hay không cho phép hiển thị hộp điều khiển của Form.
- ✓ **Enable:** Cho phép hay không cho phép Form này được hiển thị (Active).
- ✓ **FillColor:** Cho phép người lập trình chọn màu của chức năng View.
- ✓ **Filltyle:** Cho phép người lập trình chọn loại chức năng Fill.
- ✓ **Font:** Cho phép người lập trình chọn Font của Form.
- ✓ **FontTransporent:**
- ✓ **ForeColor:** Cho phép người lập trình chọn màu chữ trên Form
- ✓ **Height:** Cho phép ta chọn chiều cao của Form
- ✓ **Width:** Cho phép ta chọn chiều rộng của Form
- ✓ **Left:** Cho phép ta đặt lề trái của Form.
- ✓ **Top:** cho phép ta đặt đỉnh của Form
- ✓ **Icon:** Cho phép ta tạo biểu cho Form

3.3: GIỚI THIỆU VỀ MDI FORM:

3.3.1: HIỂN THỊ IDE

IDE là viết tắt của môi trường phát triển tích hợp (Intergrated Dovelopment Environment). IDE là nơi tạo những chương trình VisualBasic

Hiển thị IDE thể hiện một cửa sổ rỗng chứa các điểm và thanh tiêu đề với từ Form1



Hình 3.3.1: Form rỗng

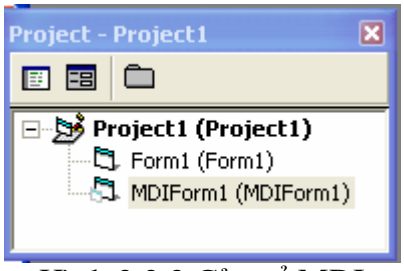
IDE của VissualBasic là nơi tập trung các Menu, thanh công cụ và cửa sổ để tạo ra chương trình. Mỗi phần của IDE có các chức năng ảnh hưởng đến hoạt động lập trình khác nhau. Thanh Menu cho bạn tác động cũng như quản lý trực tiếp trên toàn bộ ứng dụng. Thanh công cụ cho phép truy cập các chức năng của thanh Menu qua các nút của thanh công cụ. Các biểu mẫu Form khối xây dựng chính của các chương trình VisualBasic xuất hiện trong cửa sổ Form. Hộp công cụ để thêm các điều khiển vào các biểu mẫu của đề án . Project Explorer hiển thị các đề án mà bạn đang làm cũng như các phần khác của đề án. Cuối cùng bạn bố chí xem xét một hoặc nhiều biểu mẫu trên màn hình thông qua cửa sổ Form Layout.

Ta có thể xem IDE của VisualBasic bằng hai cách chính MDI hoặc SDI.

Kiểu MDI là giao diện đa tài liệu (Multiple Document Inter Face) có nhiều cửa sổ lớn chứa nhiều cửa sổ bên trong

Hiển thị kiểu MDI cho phép trình bày tất cả các cửa sổ thành phần trong IDE. Các ứng dụng như Microsoft Excel, Microsoft Word là những MDI cho phép mở nhiều tài liệu cùng một lúc, mỗi tài liệu chứa trong cửa sổ riêng.

Nếu MDI hiển thị ở chế MDI bạn có thể quy định vị trí lúc khởi động. Kiểu SDI là giao diện đơn tài liệu (Single document interface) chương trình chỉ có duy nhất một cửa sổ.



Hình 3.3.2 Cửa sổ MDI

3.3.2: TÌM HIỂU VỀ ỨNG DỤNG MDI:

- ✓ Khi thiết kế , đổi thuộc tính MDIchild thành True biểu mẫu thường trở thành cửa sổ con trong ứng dụng MDI
- ✓ Trong cửa sổ Project Exxplorer, các biểu mẫu MDI dược hiển thị với một biểu tượng đặc biệt.
- ✓ Khi thi hành:
- ✓ Các biểu mẫu con hiển thị bên trong vùng hoạt động của ứng dụng, người sử dụng có thể di chuyển, co dãn các cửa sổ con, nhưng những thay đổi này không thể vượt khỏi cửa sổ chính.
- ✓ Khi cửa sổ con bị thu nhỏ thành biểu tượng Icon , biểu tượng sẽ xuất hiện bên trong ứng dụng thay vì trên thanh tác vụ (Taskbar) . Khi cửa sổ chính của ứng dụng MDI bị thu nhỏ, toàn bộ cửa sổ con cũng bị thu nhỏ và hiển thị thành một biểu tượng duy nhất trên Takbar.
- ✓ Khi cửa sổ chính được phục hồi, cửa sổ chính và toàn bộ cửa sổ con được trả về trạng thái trước khi chúng bị thu nhỏ.
- ✓ Khi phóng to cửa sổ con, tiêu đề của nó được kết hợp với tiêu đề của cửa sổ chính và hiển thị trên thanh tiêu đề của ứng dụng.
- ✓ Đổi thuộc tính AutoshowChildrent thành true các cửa sổ con sẽ được hiển thị tự động khi cửa sổ chính nạp. Trái lại toàn bộ cửa sổ con bị giấu nếu thuộc tính này đổi thành False
- ✓ Menu của cửa sổ con đang làm việc sẽ hiển thị trên thanh Menu của cửa sổ chính ,không phải trên cửa sổ con.

3.3.3 ỨNG DỤNG MDI:

Ứng dụng MDI có nhiều cửa sổ con nên việc xác định cửa sổ này đang hoạt động rất cần thiết. Ví dụ: Muốn coppy một chuỗi ký tự trên cửa sổ con vào Clipboard, nghĩa là thủ tục xử lý xự kiện Click trên mục coppy của Menu Edit phảI gọi hàm Editcoppy Proc. Hàm này cần biết tên cửa sổ đang chứa chuỗi ký tự. Thuộc tính ActiveForm của cửa sổ chính (MDIForm) trả về tên cửa sổ còn đang focus. Lưu ý là cửa sổ chính phảI được nạp và hiển thị khi ta truy cấp đến thuộc tính này . Nừu không VisualBasic sẽ báo lỗi . trong một cửa sổ có nhiều điều khiển , thuộc tính Activecontrol của biểu mẫu xác định điều khiển đang có focus

Trong ứng dụng MDI , nhiều Insstance của biểu mẫu chia sẻ cùng đoạn chương trình. Vì vậy, khi truy cập đến một điều khiển trong một cửa sổ, ta không dùng tên biểu mẫu để tham chiếu đến điều khiển. Ví dụ: Để lấy chiều cao của hộp văn bản , ta dùng Text1.Height thay vì Form1. Text1.Height. Như vậy đoạn chương trình sẽ lấy cửa sổ hiện hành, một cách khác để lấy cửa sổ hiện hành là dùng từ khoá Me.

Muốn tạo nhiều Instance của biểu mẫu dùng từ khoá New với khai báo Dim

Quản lý các thông tin trạng thái của cửa sổ con.

Trước khi thoát khỏi chương trình, người sử dụng được yêu cầu lưu lại công việc. Như vậy ứng dụng phải luôn theo dõi các thay đổi của dữ liệu trên các cửa sổ con kể cả lần gần nhất. Ta khai báo một biến toàn cục (Public) cho mỗi cửa sổ con

```
Public booolDirty As Booolean
```

Mỗi khi người sử dụng gõ ký tự vào hộp văn bản Text1, sự kiện change của hộp văn bản được kích hoạt và thủ tục xử lý sự kiện đổi cờ BooolDirty thành true:

```
Private Sub Text1_change
BoolDirty=True
End Sub
```

Khi người sử dụng lưu bằng cách nhấn Menu File Save, cờ này đổi thành False.

```
Sub menuFileSave_Click()
    FileSave
    Set the state variable
    BooolDirty=False
End sub
```

3.3.4NẠP THOÁT BIỂU MẪU:

Khi nạp một cửa sổ con, cửa sổ chính được nạp tự động. Trong trường hợp ngược lại, cửa sổ con không nạp tự động.

Khi ứng dụng thoát, sự kiện **QueryUnload** của cửa sổ chính được gọi trước và đến lần lượt các cửa sổ con. Nếu ta không xử lý nó sẽ gọi sự kiện **Unload**, từng cửa sổ con lần lượt thoát và cuối cùng là cửa sổ chính . Bởi vì sự kiện **QueryUnload** được gọi trước khi cửa sổ thoát, ta có thể đề nghị người sử dụng lưu trước khi thoát.

```
Private sub mnFExit_Click()
  ' When the user chopses File Exit in an MDI
  ' application, unload the MDI form, invoke
  ' the QueryUnload event for each open child.
    Unload frmMDI
     End
    End sub
  Private sub Form_QueryUnload(Cancel As
Integer_UnloadMode As Integer )
   If booolDirty Then
    ' Call routine to query the user and save
    ' file if necessary.
     FileSave
  End if
End Sub
```

Tuỳ theo mục đích ứng dụng mà ta có thể chộn giao diện. Để chọn giao diện ta có thể dùng trình tạo ứng dụng tự động của **VisualBasic**.

3.3.5 CÁC TÍNH CHẤT CỦA FORM

- ✓ **Active Control** (quyền điều khiển): Tính chất này được lấy về quyền điều khiển hay hội tụ đối tượng – **Focus**. Đặt quyền điều khiển cho đối tượng chỉ định
  Ví dụ: **Lable1.Caption= Screen.ActiveControl.Text**
  (gán **Lable1** bằng text của **Screen**)
- ✓ **Appearance**: Tính chất này được sử dụng chung cho tất cả có nội dung tính chất giống nhau. Tính chất này có hai giá trị
  - Giá trị: 1 cho phép hiển thị đối tượng ở dạng 3D
  - Giá trị: 0 cho phép hiển thị đối tượng 2D
- ✓ **Backcolor**: Tính chất này áp dụng chung cho các đối tượng , nó cho phép chọn màu nền của đối tượng. Để chọn màu cho nó ta sang giá trị **Few** của nó sau đó double click để chọn
- ✓ **ForeColor**: Tính chất này áp dụng chung cho các đối tượng, nếu đối tượng là đối tượng đồ hoạ như Button , Lable thì màu này là màu chữ của **Caption** của đối tượng. Nếu đối tượng này là đối tượng hiển thị như **Text, Textbox, Listbox, Combobox** thì màu này là màu chữ của **Text** đó.
- ✓ **BorderStyle**(Kiểu đường biên) tính chất này cho phép chọn kiểu đường biên chung cho các đối tượng
  - Giá trị 0 : Không có đường biên
  - Giá trị 1 : Cho phép tạo ra một đường biên với kích thước cố định
  - Giá trị 2 : Cho phép tạo ra đường biên với kích thước có thể thay đổi được
  - Giá trị 3: Cho phép tạo ra kích thước cố định nhưng không lồng vào các nút Maxmze và Minimize
- ✓ **Fillcolor**: Cho phép trở về mã màu hoặc đặt mã màu được dùng cho hành vi tô màu (Fill)
- ✓ **Controlbox** : Cho phép tạo ra nút Control hay không
  - Giá trị là True : Cho phép tạo
  - Giá trị là False : Không cho phép tạo
- ✓ **Drawmode** : cho phép lựa chọn **Mode** vẽ (Mode : cơ chế vẽ)
- ✓ **Drawstyle** : cho phép chọn kiểu vẽ, chế độ vẽ trong chế độ đồ hoạ
  - Giá trị là 0 : cho phép vẽ nét liền (Solid)
  - Giá trị là 1 : cho phép vẽ nét gạch (Dash)
  - Giá trị là 2 : cho phép vẽ net chấm (Dot)
  - Giá trị là 3 : cho phép vẽ nét chám gạch (Dash – Dot)
  - Giá trị là 4 : cho phép vẽ nét gạch – chấm – chấm (Dash – Dot –Dot)
  - Giá trị là 5 : cho phép vẽ với cơ chế trong suốt so với màu nền
- ✓ **Drawwith**:Cho phép ta chọn kích thước, chiều rộng của bút vẽ đơn vị tính bằng **Pixel** (**Pixel:** điểm ảnh)
- ✓ **Mause Icon**:Cho phép tạo các **Icon** tuỳ thích cho con trỏ để đặt giá trị màu Icon ta chuyển con trỏ đến Fill giá trị
- ✓ **Mause Pointer:** Chức năng cũng giống như **Mause** Icon nhưng chỉ khác Mause Pinter cho phép đặt các giá trị danh sách liệt kê của nó còn Mause Icon đặt các hình ảnh tuỳ thích do người sử dụng tạo ra. Để lựa chọn giá trị Mause Icon ta Click vào Fill giá trị rồi chọn giá trị tương ứng

- ✓ **Negotiate Menus:** Cho phép hay không cho phép lồng Menu vào Form
- ✓ **Showintaskbar:** Cho phép trở về hay đặt giá trị thể hiện trạng tháI của Windows
  - Giá trị 0: hiển thị bình thường
  - Giá trị 1: Form đó được hiển thị nhỏ nhất
- ✓ **Start Up Position:** Cho phép trở về hoặc đặt giá trị thể hiện các vị trí bắt đầu của đối tượng khi nó được khả thi
  - Giá trị =0 (Manual) Cho phép ấn định kích thước trong lúc người sử dụng lập trình nó
  - Giá trị =1 Cho phép hiển thị ở góc phải màn hình
  - Giá trị =2 Cho phép hiển thị ở trung tâm màn hình
  - Giá trị =3 cho phép hiển thị ở góc trái màn hình

4. LẬP TRÌNH MÃ NGUỒN TRÊN VISUAL BASIC

4.1. KHAI BÁO BIẾN:

4.1.1: BIẾN

Biến dùng để chứa dữ liệu tạm thời cho tính toán so sánh và các hoạt động khác ta dùng toán tử (=) để tính toán và chứa giá trị vào biến.

Ví dụ:

```
Applesold = 10
Applesold = Applesold + 1
```

Dấu bằng là toán tử gán, không phải toán tử so sánh (=), giá trị 10 được gán cho biến Applesold.

4.1.2: KHAI BÁO BIẾN:

Tuỳ theo phạm vi biến cần sử dụng chúng ta có thể dùng các cấu trúc lệnh khác nhau để khai báo biến.

Dùng lệnh Dim:

```
Dim ( tên biến) As ( kiểu dữ liệu ).
Dim ( tên biến ) ( kiểu gán chuỗi ).
```

Ví dụ:

```
Dim a,b As integer
Dim a%, b%
```

Biến khai báo trong thủ tục chỉ tồn tại khi thủ tục thi hành. Nó sẽ biến mất khi thủ tục chấm dứt. Giá trị biến trong thủ tục là cục bộ đối với thủ tục đó. Nghĩa là không thể truy cập biến từ bên ngoài thủ tục. Nhờ đó ta có thể dùng tên biến cục bộ trong thủ tục khác nhau.

Kiểu dữ liệu khai báo trong Dim có thể là kiểu cơ bản như Integer, String hoặc Currentcy.Ta cũng có thể dùng đối tượng của VB (Như Object, Form, TextBox) hoặc của các ứng dụng khác.

Khai báo phần Declaration của một modul nghĩa là biến đó tồn tại và có tầm hoạt động trong Modul đó.

Khai báo biến với từ kháo Public nghĩa là biến đó tồn tại và có tầm hoạt động trong toàn bộ ứng dụng.
Khai báo biến cục bộ với từ khoá Static, nghĩa là mặc dù biến đó biến mất khi thủ tục chấm dứt, nhưng giá trị của nó vẫn được giữ lại để tiếp tục hoạt động khi thủ tục được gọi trong lần sau.

4.1.3: KHAI BÁO NGẦM:

Nghĩa là ta không cần khai báo tường minh, trước khi sử dụng biến
Ví dụ:

```
Funtion Safe Sqr (num)
Temp Val = Abs (mim)
Safe Sqr  = Sqr (Temval)
End Funtion
```

4.1.4: KHAI BÁO TƯỜNG MINH:

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

Cách1:

Chọn Tools chọn Option, chọn Tab Edtor và đánh dấu vào tuỳ chọn Require Variable Declaration.Từ thời điểm này các màn hình lớp hay thư viện được tạo ra sẽ mặc nhiên có sẵn dòng lệnh Option Explicit trong phần Decrlarations với các màn hình giao tiếp (Form), lớp class hay thư viện (Module) đã được tạo trước đó, chúng ta sẽ tự thêm vào dòng lệnh này như cách 2.

Cách 2:

Trong cửa sổ đặt dòng lệnh sau đây Option Explicit ở đầu phần declarations của màn hình giao tiếp (Form), lớp (Class) hay thư viện Module.
Tầm hoạt động của biến:

	Private	**Public**
Thủ tục	**Biến chỉ tồn tại và hoạt động trong thủ tục**	**Không có**
Modul	**Biến chỉ tồn tại và hoạt động trong Modul**	**Biến tồn tại và hoạt động trên Modul**

4.1.5: KHAI BIẾN STATIC:

Kiểu dữ liệu	Kích thước		
Integer	2 byte	-32,768 -> 32,767	
Long (Long integer)	4 byte	-2,147,483,648 -> 2,147,483,647	&
Single (Single – precision, floating - point)	4 byte	-3.402823E38 -> -1.401298E –45 Với giá trị âm; 1.401298E – 45 ->	
Double (double-precision, floating-point)	8 byte	- 1.79769313486232 E308->
- 4.94065645841247E – 324 Với các giá trị âm: 4.94065645841247E – 324 1.79769313486232 E308 Với các giá trị dương	#
Currency	8 byte	- 922,337,203,685,477.5808 ->
922,337,203,685,477.5808.	#
String	1 byte mỗi character	0 ->65,500 byte	
Varriant		+ Dùng để biểu diễn các giá trị số (với tầm vực double) hay các ký tự	

- ✓ Để khai báo tất cả các biến cục bộ trong một thủ tục Static, ta đặt từ khoá Static vào thủ tục.
- ✓ Static Function Runningtotal (mim) VB sẽ hiểu là tất cả biến khai báo trong thủ tục này đều là Static, là cho chúng được khai báo là Private, là Dim hoặc thậm chí là khai báo ngầm.
- ✓ Từ kháo Static có thể đặt ở đầu thủ tục Sub hoặc Funtion , kể cả thủ tục xử lý sự kiện hoặc các hàm Private.

4.1.6 CÁC TOÁN TỬ CỞ SỞ:

4.1.6.1 TOÁN TỬ GÁN:

Đây là toán tử cở sở của hầu hết các ngôn ngữ lập trình.

Toán tử dùng để gán giá trị cho các biến có kiểu dữ liệu cơ sở trong VisualBasic là dấu (=), cú pháp chung lệnh gán có dạng sau:

<Tên biến> = <Biểu thức>

Biểu thức ở bên phải của cú pháp trên có thể là một giá trị hằng, một biến hay một biểu thức tính toán. Khi đó VisualBasic sẽ thực hiện việc tính giá trị của biểu thức trước rồi sau đó mới gán giá trị có được cho biến.

Ví dụ: Dòng lệnh gán sau đây sẽ tăng giá trị biến k lên thêm một (k=k+1).
Thông thường, giá trị của biểu thức và biến trong cú pháp lệnh gán phải cùng kiểu dữ liệu, tuy nhiên chúng ta vẫn có thể gán biểu thức số vào một biến kiểu chuỗi. Trong trường hợp này, VisualBasic sẽ tự động đổi giá trị biểu thức thánh chuỗi sau đó mới gán vào biến

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

Set <Tên biến>= <Biểu thức>

4.1.6.2. TOÁN TỬ TÍNH TOÁN:

Những toán tử trình bày trong phần này là những toán tử cơ sở liên quan đến các giá trị số, bao gồm số nguyên và số thực

❁ **Các toán tử số nguyên :**

+ : Cộng (cộng hai toán hạng)

- : Trừ (Trừ hai toán hạng)

*****: Nhân (Dùng để nhân hai toán hạng)

\ : Chia (chia nguyên bỏ phần thập phân)

Ví dụ 1: X=5\2 thì X sẽ nhận giá trị là 2
/ : Chia làm tròn

Ví dụ 2: Dim x as Double
X=3/2

^ : Luỹ thừa (Dùng để tính luỹ thừa)

MOD: Chia lấy phần dư

❁ Các toán tử số thực

+ : Cộng (cộng hai toán hạng)

- : Trừ (Trừ hai toán hạng)

*****: Nhân (Dùng để nhân hai toán hạng)

/ : Chia (chia hai toán hạng)

^ : Luỹ thừa (Dùng để tính luỹ thừa)

4.1.6.3.TOÁN TỬ NỐI CHUỖI & :

Đây là toán tử cơ sở dùng để nối các dữ liệu lại với nhau

Ví dụ :Trong dòng lệnh dưới đây:

S = ”Viasua” & ”Basic”

Biến chuỗi S sẽ có giá trị là “VisuaBasic”

Tương tự như lệnh gán chuỗi với các biểu thức số, Visuabasic sẽ tự động thực hiện việc chuyển dữ liệu chuỗi thành số trước rồi sau đó mới nối với dòng tiếp theo

S=S & 1 =>Giá trị của biến S sau lệnh gán sẽ là “VisualBasic 1”

4.1.6.4 TOÁN TỬ SO SÁNH

Như mọi ngôn ngữ lập trình khác ,VisualBasic cũng sử dụng các toán tử so sánh như :

Kí hiệu	Ý Nghĩa	Ví dụ
=	So sánh bằng	A=B
<	So sánh nhỏ hơn	A<B
<=	So sánh nhỏ hơn hay bằng	A<=B
<>	So sánh khác	A<>B
>	So sánh lớn hơn	A>B
>=	So sánh lớn hơn hay bằng	A=B

Giá trị kết quả của một biểu thức so sánh sẽ thuộc kiểu luận ý (True hay False). Khi cần kết nối các biểu thức so sánh lại với nhau chúng ta có thể dùng các toán tử luận lý And, Or.

4.2 TẬP LỆNH CƠ BẢN CỦA VISUALBASIC:

4.2.1 KIỂU DỮ LIỆU:

Kiểm soát nội dung của dữ liệu, VisualBasic dùng kiểu Variant như là kiểu mặc định. Ngoài ra, một số kiểu dữ liệu khác cho phép tối ưu hoá về tốc độ và kích cỡ chương trình, khi dùng Variant ta không phảI chuyển đổi giữa các kiểu dữ liệu,VB tự làm việc đó.

- **Kiểu số:**

Integer, Long, Double và currrencey kiểu số chứa ít vùng nhớ hơn kiểu Variant. Tất cả biến kiểu số có thể được gán cho nhau và biến Variant VisualBasic làm tròn thay vì chặt bỏ phần thập phần trước khi gán nó cho số Integer

Kiểu Integer tốn ít nhất vùng dữ liệu, nó thường dùng làm biến đếm trong vòng lặp ForNext.

Kiểu Singe, double, Currrency cho các số thập phân. Currency hỗ trợ bốn chữ số thập phân và mười năm chữ số phần nguyên.

- **Kiểu String** Mặc định biến hay tham số kiểu chuỗi có chiều dài thay đổi nó có thể tăng giảm tuỳ theo ta gán dữ liệu. Nhưng ta có thể khai báo chuỗi có chiều dài cố định.

- **Kiểu Boolean:** Là kiểu có hai giá trị Yes/ No, True/ False, Yes/ No, On/ Off.

❁ **Kiểu Date:** Khi các kiểu dữ liệu khác được chuyển sang Date, giá trị đứng trước là ngày, giá trị đứng sau dấu chấm là giờ. Nửa đêm là 0, nửa ngày là 0,5. Dờu âm trước thể hiện ngày trước

Ví dụ:

```
Click : cmd Exprole Date
Dim d Date As Date
If Is Date ( text Date text )
Then. Date = CV Date ( Text. date. Text )
Msg Box " Day = " & Day (d Date ) & - "Month= " Month ( d Date)
& - " Year = "Year ( d Date )
Else
   Msg Box " that isn't a valid date. Pleasetry again."
End if.
```

❁ **Kiểu Variant:**

Có thể chứa nhiều loại dữ liệu chuỗi số thậm chí mảng. Ta không cần chuyển đổi dữ liệu, Visual Basic làm việc đó một cách tự động .

❁ **Kiểu mảng Array:**

Mảng là một sâu các biến có cùng tên và cùng kiểu dữ liệu. Dùng Array làm chương trình đơn giản và rút gọn vì ta có thể dùng vòng lặp mảng có biến trên và biến dưới và các thành phần trong mảng là liên tục giữa hai biến

Có 2 loại biến mảng: Mảng có chiều dài cố định và mảng động có chiều dài thay đổi lúc thi hành mảng có chiều dài cố định có thể được khai báo Public trong ứng dụng, Private trong Modul hoặc Private trong thủ tục.

4.2.2 IF….THEN

CP1: Một dòng lệnh

If < Điều kiện > ***Then*** < Dòng lệnh >

CP2: Haidòng lệnh:

If < Điều kiện > ***Then*** < khả thi lệnh 1 >
Else < Khả thi lệnh 2 >
End If

CP3: Nhiều dòng lệnh

If < Điều kiện 1 > Then < Khả thi lệnh 1>
Else If < điều kiện 2> Then < Khả thi lệnh 2 >
Else < khả thi lệnh 3 >
End if

Điều kiện là một so sanh hay một biểu thức mang giá trị số. VisualBasic thông dịch giá trị thành True / False (0 là False, giá trị khác 0 là True). Nếu điều kiện là True, Visual Basic thi hành tất cả dòng lệnh sau từ khoá Then.

Ví dụ :

If any Date < Now Then any Date = Now
If any Date < Now Then
Any Date = Now
End If

Note : Trường hợp một dòng lệnh không có End If

4.2.3: SELECT CASE:

Giải quyết trường hợp có quá nhiều Else If được dùng để chương trình dể hiểu, dễ đọc. Biểu thức so sánh được tính toán một lần vào đầu cấu trúc. Sau đó, VisualBasic so sánh kết quả biểu thức với từng Case. Nếu bằng, nó thi hành khối lệnh trong Case đó.

Select Case < Tên biến >
Case < giá trị 1 hay phạm vi giá trị 1 >
< phát biểu 1 >
Case Else
< phát biểu n >
End Else

Mỗi danh sách biểu thức chứa một hoặc nhiều giá trị. Các giá trị cách nhau bằng dấu phẩy. Mỗi khối lệnh có thể chứa từ 0 đến nhiều dòng lệnh. Nếu có hơn một Case thoả điều kiện thì khối lệnh của Case đầu tiên được thi hành. Case Else không nhất thiết phải có, dùng trong trường hợp còn lại của các Case trước.

4.2.4 FOR….NEXT (vòng lặp):

Biết trước số lần lặp. Ta dùng biến đếm tăng dần hoặc giảm dần trong vòng lặp.

For < tên biến > = < giá trị đầu > To < giá trị cuối >
(Các lệnh khả thi)

4.2.5: DO … WHILE(vòng lặp):

Thi hành một khối lệnh với số lần không định trước, trong đó một biểu thức điều kiện dùng so sánh để quyết định vòng lặp có tiếp tục hay không . Điều kiện phảI quy về False (0) hoặc true (khác 0).

Kiểu1: Lặp trong khi điều kiện True

Do While < điều kiện >
(các phát biểu khả thi) hay (khối lệnh)

Kiểu 2: Lặp trong khi điều kiện là False

Do until < điều kiện >
Các phát biểu khả thi
Loop

4.2.6: PHÁT BIỂU GÁN:

< Tên biến > = Giá trị
< Tên đối tượng >.<Tính chất giá trị> = giá trị
<Tên đối tượng >.<Tính chất giá trị >= <Tên đối tượng 2>
< Tính chất giá trị>

4.3 : CÁC HÀM CƠ BẢN CỦA VISUALBASIC:

- ✓ **Thủ tục:**

  Khi lập trình ta thường gặp những đoạn chương trình lặp đi lặp lại nhiều lần ở những chỗ khác nhau. Để chương trình đỡ phức tạp, các đoạn này được thay thế bằng những chương trình con tương ứng, khi cần ta chỉ cần gọi nó ra mà không cần viết lại cả đoạn .

  Mặc khác, đối với những chương trình lớn và phức tạp. Việc xem xét tổng quan cũng như việc gỡ rối, hiệu chỉnh sẽ rất khó khăn.Việc lập các chương trình con sẽ chia chương trình ra từng khối , từng Modul. Điều này giúp cho chúng ta kiểm tra, gỡ rối và điều khiển rễ ràng.

  Cấu trúc của một thủ tục có dạng như sau:

  Sub < tên thủ tục >
  ' Các lệnh .. '
  End sub

  Nếu bạn muốn dùng thủ tục này trong toàn bộ chương trình thì bạn dùng thêm, từ khoá Public trước từ khoá Sub, còn nếu ta chỉ muốn dùng trong Form chứa nó thì bạn thêm từ khoá Private trước từ khoá Sub.

- ✓ **Hàm:**

  Hàm cũng tương tự như thủ tục, chỉ khác là ta dùng hàm khi muốn nhận lại một kết quả trả lại của hàm, cấu trúc hàm như sau:

  Funtion < tên hàm > (tham số As kiểu) **As** < Kiểu trả về >
  ' các lệnh .. '
  Tên hàm = giá trị trả về
  End Function

4.3.1 HÀM ABS (Number)

Hàm này trả về một số giá trị tuyệt đối của Number

4.3.2 HÀM SIN (Number as Double)

Tính sin của một góc

4.3.3 HÀM COS (Number As Double)

Tính Cos của một góc

4.3.4 HÀM TAN (Number As Double)

Tính hàm tang của một góc.

4. 3.5 HÀM ATN (Number As Double)

Tính hàm Arctang của một góc

4. 3.6 HÀM INT (Number)

Trả về phần nguyên của số nếu là số dương. Còn nếu là số âm thì trả về phần nguyên có giá trị nhỏ hơn phần nguyên một đơn vị.

4. 3.7 HÀM FIX (Number)

Hàm trả về phần nguyên của số

4.3.8 HÀM SQN (Number)

Hàm trả về một số nguyên
Nếu số > 0 sẽ trả về giá trị là 1
Nếu số < 0 sẽ trả về giá trị là - 1
Nếu số = 0 sẽ trả về giá trị là 0

4.3.9 HÀM SQR (Number)

Tính căn bậc hai của một số

4.3.10 HÀM EXP (Number)

Tính e mũ của số

4.3.11 HÀM RND (Number)

Hàm này trả lại một số thực ngẫu nhiên

4.3.14 HÀM DAY (Biến)

Trả về số ghi ngày của biến bạn nhập vào

4.3.15 HÀM MONTH (Biến)

Trả về số ghi tháng bạn nhập vào

4.3.16 HÀM YEAR (Biến)

Trả về số ghi năm của biến bạn cần nhập vào

4.3.17 Hàm Now

Trả về ngày tháng năm và thời gian hiện tại

4.3.18 HÀM WEEKDAY (biến)

Cho biết thứ mấy trong tuần

4.3.19 HÀM HOUR (thời gian)
Cho biết giờ ứng với biến

4.3.20 HÀM MINUTE (thời gian)
Cho biết phút ứng với biến

4.3.21 HÀM SECOND (thời gian)
Cho biết giây ứng với biến

4.3.22 HÀM REPLACE(chuỗi, chuỗi cần tìm, chuỗi thay thế, vị trí thay thế, số lần thay thế)
Thay thế chuỗi này bằng một chuỗi khác.

4.3.23 HÀM VAL (STRING)
Trả về một số thực với chuỗi String. String phảI là một chuỗi goomfg các ký tự hợp lệ.
Giá trị của hàm là 0 nếu chuỗi có ký tự đầu là ký tự .
Giá trị hàm là 1 nếu chuỗi đó toàn là các ký tự số
Nếu các ký tự số viết cách nhau thì hàm này sẽ cắt bỏ các khoảng trắng và cho trả về giá trị bằng với giá trị này.

4.3.24 HÀM STR (NUMBER)
Hàm này trả về một chuỗi ký tự.
Chuỗi này luôn có một kí tự đầu ghi dấu trong trường hợp số âm hoặc một khoảng trống trong trương hợp số dương.

4.3.25 HÀM QBCOLOR (COLOR)
Hàm cho màu của một đối tượng nào đó

4.3.26 HÀM RGB (RED , GREEN, BLUE)
Chọn màu bất kỳ qua tỷ lệ ba màu chuẩn Red, Green, Blue

4.3.27 HÀM ASC (STRING)
Trả về mã Ascii của ký tự String

4.3.28 HÀM CHR (CHARCODE)
Hàm trả về một ký tự tương ứng với một mã Ascii nào đó

4.3.29 HÀM LEN (EXPRESSION)
Cho biết chiều dài của chuỗi

4.3.30 HÀM LTRIM (STRING)
Hàm này trả về một chuỗi sau khi cắt bỏ các khoảng trống bên trái của chuỗi

4.3.31 HÀM RTRIM (STRING)

Hàm này trả về một chuỗi sau khi cắt bỏ các khoảng trống bên phải của chuỗi

4.3.32 HÀM TRIM (STRING)

Hàm này trả về một chuỗi sau khi cắt bỏ các khoảng trống bên phải và bên trái của chuỗi

4.3.33 HÀM LEFT (STRING,N)

Trả về một chuỗi ký tự được cắt từ n ký tự (kể cả ký tự trắng) bên trái của chuỗi

4.3.34 HÀM RIGHT (STRING, N)

Trả về một chuỗi ký tự được cắt từ n ký tự (kể cả ký tự trắng) bên phải của chuỗi

4.3.35 HÀM MID (STRING, N,[LEGTH])

Hàm này trả về một chuỗi ,chuỗi này được lấy từ chuỗi String bắt đầu từ vị trí n với chiều dài length

4.3.36 HÀM SPACE (N)

Hàm trả về một chuỗi gồm n khoảng trắng

4.3.37 HÀM STRING(N, KÝ TỰ)

Trả về một chuỗi gồm ký tự n giống nhau

4.3.38 HÀM INSTR (START, S1, S2, COMPARE)

Dùng để tìm một chuỗi con nằm trong chuỗi mẹ hay không, nếu có thì nằm ở vị trí mấy
Start :Vị trí bắt đầu tìm
S1 và S2 là chuỗi mẹ và chuỗi con
Compare có giá trị :
0: So sánh chính sác từng ký tự
1: So sánh không phân biệt chữ hoa hay chữ thường
2: Chỉ dùng trong khi lập trình cho Microsift Acsess

4.3.39 HÀM UCASE

Đổi thành chữ hoa

4.3.40 HÀM LCASE

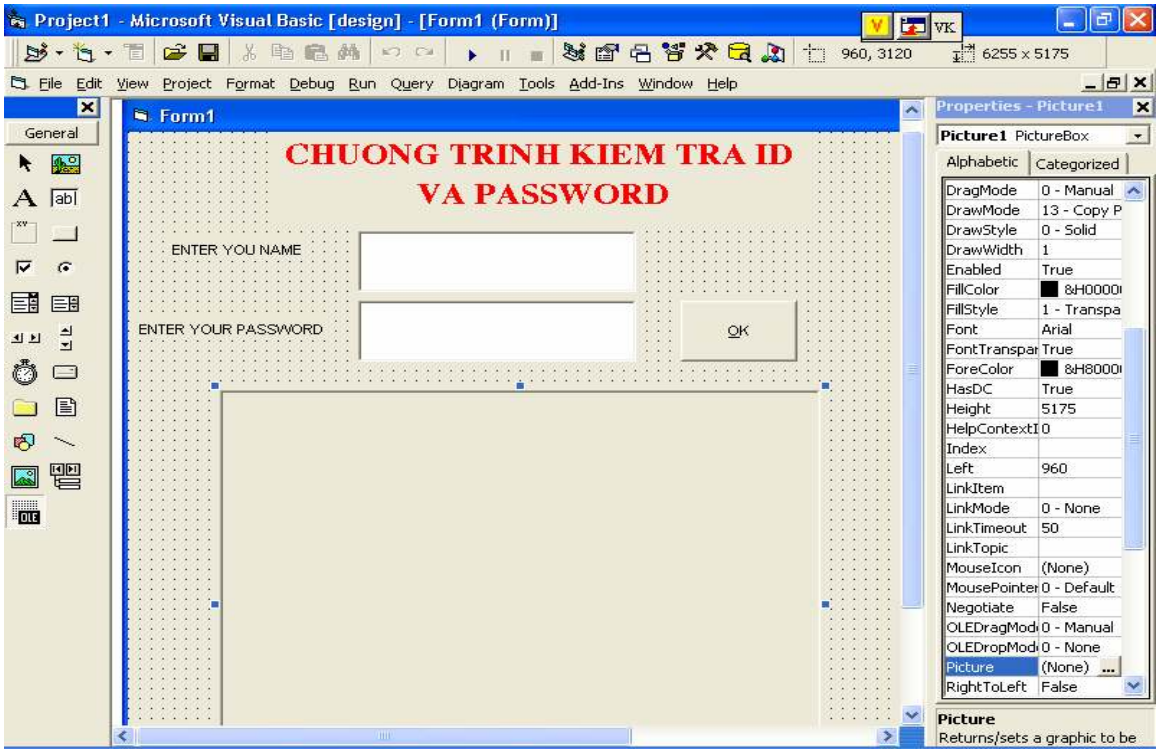
Đổi thành chữ thường

4.3.41 HÀM FORMAT (VALUE, FORMAT)

Hàm này dùng để định dạng

4.4 MỘT SỐ VÍ DỤ MINH HOẠ CHO CÁC HÀM

4.4.1 THIẾT LẬP MỘT FORM KHI CHẠY CHƯƠNG TRÌNH SẼ XUẤT HIỆN YÊU CẦU NHẬP PASSWORD NHU SAU:



Hình 4.4.1

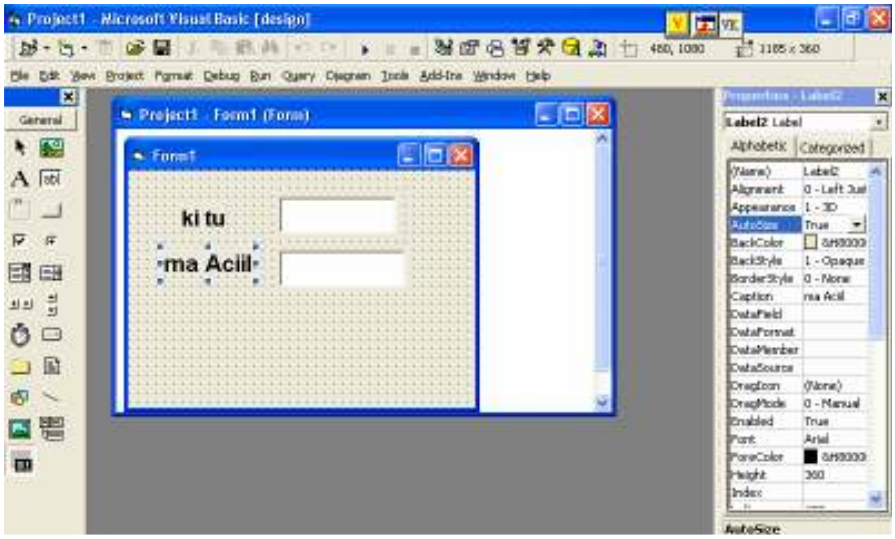
Để thực hiện chương trình khi viết Code cho chương trình sẽ áp dung hàm Len
Chương trình sẽ được viết mã nguồn như sau

```
Private Sub Command1_Click()
If ((Text1.Text = "NGUYEN NGOC QUANG") And (Text2.Text = 0904307746) And
Len(Text1.Text) = 17) Then
Picture1.Visible = True
MsgBox "YOU ARE VIP MEMBER !"
Else
MsgBox "I DONT KNOW YOU "
End If
End Sub

Private Sub Form_Load()
Picture1.Visible = False
End Sub
```

4.4.2 VIẾT MỘT CHƯƠNG TRÌNH NHẬP MỘT KÝ TỰ VÀ IN RA MÃ ASCII CỦA KÝ TỰ ĐÓ

Trong bài nay ta áp dụng hàm Str(Asc)



Hình 4.4.2

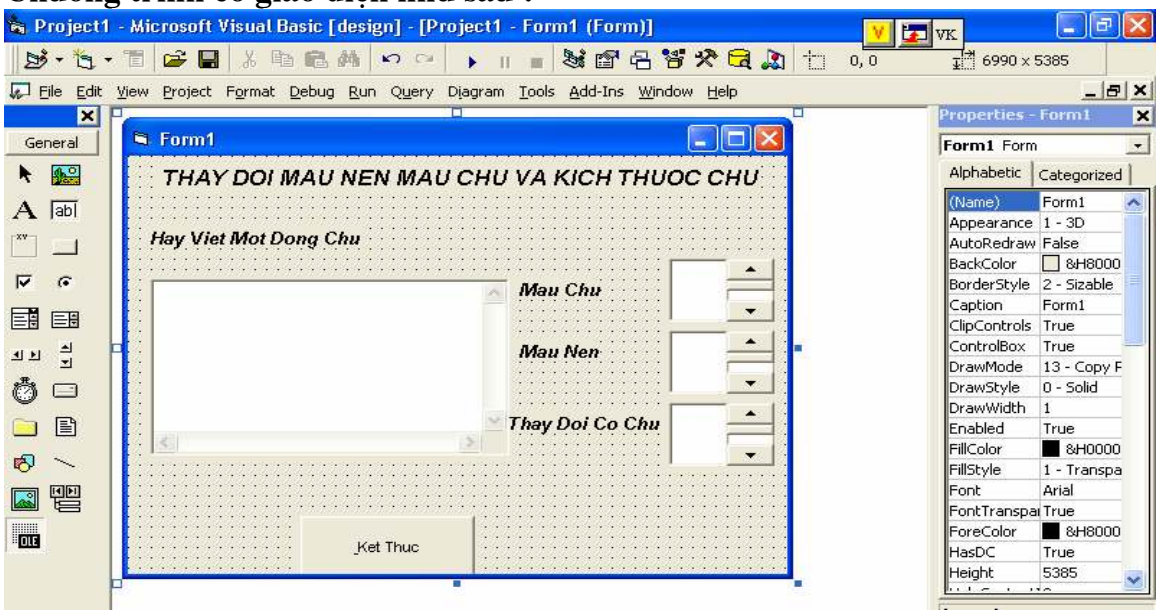
Ta viết Code cho chương trình như sau :

```
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
Text2.Text = Str(Asc(Text1.Text))
Text1.Text = ""
End Sub
```

4.4.3 VIẾT MỘT CHƯƠNG TRÌNH THAY ĐỔI CỠ CHỮ ,MÀU NỀN , MÀU CHỮ

Trong bài nay ta sử dung hàm Qbcolor và các thuộc tính Max Min cho Vscrollbar,Hscrollbar

Chương trình có giao diện như sau :



Hình 4.4.3

Ta viết Code cho chương trình như sau :

```
Private Sub Cmdketthuc_Click()
        End
End Sub
Private Sub VScroll1_Change()
        Txtfc = VScroll1.Value
        TxtT.ForeColor = QBColor(VScroll1.Value)
End Sub
Private Sub VScroll2_Change()
        Txtbc = VScroll2.Value
        TxtT.BackColor = QBColor(VScroll2.Value)
End Sub
Private Sub VScroll3_Change()
        Txtfs = VScroll3.Value
        TxtT.FontSize = VScroll3.Value
End Sub
Private Sub form_load()
        VScroll1.Min = 15
        VScroll1.Max = 0
        VScroll1.Value = 5
        Txtfc = VScroll1.Value
        VScroll2.Min = 15
        VScroll2.Max = 0
        VScroll2.Value = 6
        Txtbc = VScroll2.Value
        VScroll3.Min = 80
        VScroll3.Max = 0
        VScroll3.Value = 10
        Txtfs = VScroll3.Value
End Sub
```

5. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

5.1. GIỚI THIỆU VỀ ĐỐI TƯỢNG

Từ đầu quyển sách đến giờ chúng ta chỉ sử dụng biến để chứa những dữ liệu tạm thời trong ứng dụng, chẳng hạn như những giá trị do người sử dụng nhập vào qua giao diện. Tuy nhiên đây chỉ là phần nhỏ của VisualBasic. Vb6 thực chất là một công cụ lập trình hướng đối tượng rất mạnh

Bạn có thể cho rằng kỹ thuật này vượt quá khả năng một người mới mới học lập trình VB. Tuy nhiên, không hẳn như vậy. **Lập trình hướng đối tượng** (Object Oriented Programming – OOP) giúp lập trình dễ dàng hơn.

Các ví dụ ta dùng trước đây được lập trình theo kiểu lập trình cổ điển. Điều này không có gì sai bởi vì đây là những chương trình nhỏ và việc sử dụng OOP cho chúng cũng không phù hợp. Với kiểu lập trình cổ điển, còn gọi là **phát triển phần mềm theo cấu trúc** (Structured Software Development), ta phải xác định phần mềm cũng như cách thức để xử lý dữ liệu trong ứng dụng. Một giao diện người sử dụng được cung cấp để hiển thị và nhập liệu từ người sử dụng, sau đó các hàm và thủ tục còn được xây dựng để xử lý dữ liệu. Điều này có vẻ đơn giản. Để giải quyết một vấn đề lớn, ứng dụng chia thành nhiều vấn đề lớn, ứng dụng chia thành nhiều vấn đề nhỏ để giải quyết trong các hàm, thủ tục.
OOP hơi khác một chút. Với lập trình có cấu trúc, cách thức xây dựng ứng dụng, cách chúng kết hợp ở mức chương trình rất thực tế với cuộc sống. Lấy một ứng dung tính lương làm ví dụ. Khi nhân viên được nhận vào để làm việc, các thông tin chi tiết của nhân viên đó sẽ được nhập vào hệ thống tính lương. Sử dụng kỹ thuật lập trình có cấu trúc, ta sẽ dùng một biểu mẫu để chứa những thông tin nhân viên và viết chương trình copy tất cả những thông tin đã nhập vào biểu mẫu đó vào cơ sở dữ liệu chứa ở đâu đó trên mạng công ty. Để tạo ra phiếu trả lương ta cần phảI có một biểu mẫu in trả lương cho phép người sử dụng chương trình chọn một nhân viên trả lương, rồi viết chương trình để thu nhập tất cả thông tin từ cơ sở dữ liệu và định dạng nó và đưa ra máy in.
Ta có thể thấy rằng, giải pháp này nặng về kỹ thuật và nghiêng về xử lý máy hơn là cách thực hiện trong thực tế cuộc sống. Lập trình hướng đối tượng sẽ làm cho mọi việc trở nên đơn giản hơn nhiều.
Với OOP, ta viết một chương trình dựa trên các đối tượng của thực tế cuộc sống. Ví dụ nếu ta đang viết một chương trình ứng dụng tính lương đối tương mà ta cần làm việc sẽ là **phòng ban** và **nhân việc.** Mỗi đối tượng có cac thuộc tính :ví dụ một nhân viên có tên và số;một phòng ban có vị trí và trưởng phòng.Thêm vào đó,có một số phương thức để phòng phát lương ứng dụng cho cac đối tượng trên – mỗi tháng một lần,nó quyết định phương thức phát lương cho các đối tượng nhân viên .lập trình OOP cũng tương tự như thế ta quyết định đối tượng naò là cần thiết, đối tượng có những thuộc tính nào , và ta sẽ áp dụng những phương thức cho đối tượng đó .

Ta có thể thấy rằng; đây là một phương thức hết sức gần gũi với những vấn đề của thực tế cuộc sống mà ta thương xuyên gặp phải . nhân viên được xem là đối tượng trong một ứng dụng ,và phòng ban là đối tượng có liên quan với nhân viên .
Với lập trình có cấu trúc, ta có xu hướng xem dữ liệu và cach thức sử lý dữ liệu là hai phần tách biệt nhau ,hoàn toàn khác với đối tượng và cách xử lý trong thực tế cuộc sống mà ta vẫn thường làm .Vói OOP ,ta đóng dữ liệu và các chức năng xử lý dữ liệu trong một đối tượng (Object) giống hệt với đối tượng trong thực tế cuộc sống
Vói lập trình có cấu trúc có thể xem dữ liệu và cách thức sử lý dữ liệu là hai phần tách biệt nhau hoàn toàn khác với các đối tợng và cách sử lý trong thực tế cuộc sống mà ta vẫn thường làm. Với OPP ta đóng gói dữ liệu và các chức năng sử lý dữ liệu trong một đối tợng (Object) . Bằng cách chia ứng dụng thành nhiều đối tượng. Nó giúp tạo ra những chương trình để đọc chương trình và dễ bảo trì.

5.1.1 ĐỐI TƯỢNG TRONG VISUAL BASIC

Khi ta quyết định đặt một hộp văn bản vào biểu mẫu ta phải gọi một thủ tục con để tạo hộp văn bản, một thủ tục con khác để đặt văn bản vào vị trí rồi gọi một thủ tục con khác nữa để định giá trị khởi động

Những gì ta cần làm là kéo và thả một đối tượng (hay một điều khiển) chẳng hạn như thả hộ văn bản vào biểu mẫu rồi dùng các thuộc tính để sửa đổi cách thể hiện của chúng. Khi người sử dụng nhập dữ liệu vào hộp văn bản, hộp văn bản sẽ thông báo cho ta biết các sự kiện Change, và sự kiện Keypress

Ngoài các đối tượng hay điều khiển được cung cấp sẵn, Visual Basic còn cho phép lập trình viên tạo ra các đối tượng thông qua cơ chế Modun lớp (Class Module). Trong Mô-dun ta định nghĩa các thuộc tính và phương thức của một đối tượng. Sau khi hoàn tất, để sử dụng đối tượng trước hết, ta tạo ra đối tượng và gọi các hàm / thủ tục trong mô-đun lớp

Các đối tượng này có một số đặc tính chung :

Từng đối tượng phải có chức năng tổng quát, được định nghĩa vừa đủ để hiểu nhưng khá mềm dẻo để có thể sử dụng được : nhưng cho phép phát triển thêm tuỳ theo yêu cầu .

Ví Dụ : một nút lệnh phải có chức năng chung là nhấn vào để thi hành một công việc gì đó. Tuy nhiên cách thể hiện và hoạt động của nó trong từng trường hợp có thể thay đổi chút ít tuỳ theo cách ta cài thuộc tính và viết code choi phương thức để phản với sự kiện

- Đối tượng giao tiếp với bên ngoài thông qua thuộc tính phương thức và sự kiện được định nghĩa trớc cho nó. Tổ hợp của 3 khái niệm này gọi là giao diện (interface). Đó là những yếu tố cần biết về một đối tượng để sử dụng chúng
- Có thể sử dụng nhiều đối tợng trong một đề án, ta cũng có nhiều thể hiện khác nhau của một kiểu đối tượng
- Người sử dụng đối tượng không quan tâm đến cách lập trình bên trong đối tượng. Bởi vì người sử dụng chỉ thấy đối tượng điều khiển, ta có thể thay đổi hoạt động bên trong của đối tợng sao cho những thay đổi này không ảnh hưởng đến ứng dụng đang dùng đối tượng

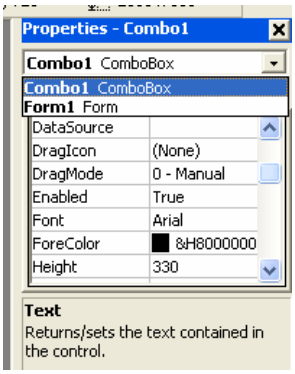
5.1.1.2 MỘT SỐ ĐỐI TƯỢNG CƠ BẢN

- ✓ **Keydow:** Đối tượng này xuất hiện và thi hành khi có một phím bất kỳ nhấn xuống
- ✓ **KeyUp:** Đối tượng này xuất hiện và thi hành khi ta nhả phim nhấn của lần thi hành chương trình trước
- ✓ **GotFocus :** Đối tượng này xuất hiện và thi hành khi có sự hội tụ của các điều kiện được nhập từ bàn phím
- ✓ **MouseDown:** Đối tượng này xuất hiện và thi hành khi ta nhấn chuột xuóng các đối tượng
- ✓ **MouseUp:** Đối tượng này xuất hiện và thi hành khi ta nhả chuột ra sau khi đã thi hành chương trình trước đó

5.1.2 MÔ-ĐUN LỚP

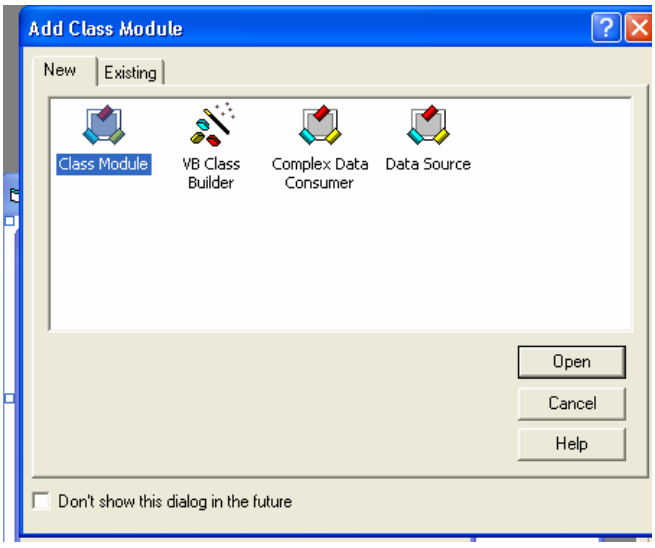
Khuôn mẫu để tạo đối tượng là Mo-đun lớp. Sau này mô- dun lớp còn được dùng để tạo điều khiển ActiveX, một kỹ thuật cao hơn của lập trình hướng đối tượng

Trong bước lập trình căn bản với VB ta dùng mô - đun để chứa các hàm hay thủ tục. Tuỳ theo tầm hoạt động của hàm / thủ tục này ta có thể gọi chúng trực tiếp từ mô- đun
Nhưng mô- đun lớp thì không bao giờ đợc gọi trực tiếp. Để sử dụng một lớp, ta phải tạo đối tượng từ lớp thông qua lệnh New
Ở đây đối tượng được tạo từ lớp MyClass, còn biến đối tợng MyObject cung cấp một tham chiếu đến đối tượng



Dim MyObject as new MyClass
Dòng lệnh trên tạo một đối tượng gọi là MyObject theo mô tả của lớp MyClass. Hành động này gọi là tạo một instance từ lớp
Trong cửa sổ Properties ta có thể phân biệt tên lớp và tên đối tượng :
Combo1 là tên đối tượng trong khi combobox là tên lớp
Ta có thể tạo ra vô số instance từ một lớp .Mỗi instace có thể khác nhau một chút tuỳ theo cách ta quy định thuộc tính và sử dụng phương thức

5.1.2.1 THUỘC TÍNH VÀ PHƯƠNG THỨC CỦA LỚP



Bên trong một lớp ta có thủ tục phơng thức và thủ tục thuộc tính . Quy định một thuộc tính nghĩa là ta đang gọi hàm xử lý sự kiện Property Let
Trong Visual Basic 6phiên bản Professional và Enterprice hỗ trợ trình xây dựng lớp (Class Builder) giúp lập trình dễ dàng với lớp .Nó cung cấp một loạt các hộp thoại hướng dẫn ta từng bước để tạo lớp

Ví dụ mẫu – thiết kế lớp có chức năng di chuyển hộp trên màn hình

1. Tạo đề án mới , Kiểu Standard EXE

2. Từ Menu Project chọn Add Class Module . Một hộp thoại xuất hiện

3. Chọn Class Module và nhấn Open. Cửa sổ Code sẽ hiển thị. Nếu nhìn vào cửa sổ Project Expoler, ta sẽ thấy một lớp mới xuất hiện

4. Vì ta muốn tạo một lớp Box, nên ta đổi tên Class1 sao cho gợi nhớ ClsBox. Cụm từ " cls" thể hiện đây là lớp nhờ đó , chương trình trở nên dễ đọc hơn. Để thực hiện điều này , tìm lớp class1 trong cửa sổ Properties, đổi thuộc tính Name của nó thành clsBox.

5.1.2.1.1. THUỘC TÍNH CỦA LỚP – PUBLIC VÀ PRIVATE

Lớp Box có 4 thuộc tính là toạ độ góc trái trên (X, Y) va chiều cao (Height) và chiều rộng (Width). Bây giờ ta cần khai báo thuộc tính trên public và Private.
Khi một thuộc tính được khai báo là Public trong một lớp thì nó sẽ không được truy cập bởi bất kỳ đoạn chương trình nào khác
So sánh khai báo của một thuộc tính Public. Biến Public trong một lớp giống như một biến Public bất kỳ nào khác, chỉ có điều là khi ta xử lý nó trong chương trình thì giống như ta đang xử lý với một thuộc tính.
Ví dụ Nếu ta khai báo thuộc tính X là một biến Public, sau đó khai báo một đối tượng gọi là Mybox dựa trên lớp này ta có dòng lệnh sau MyBox.x =1000
Tham chiếu đến X tương tự như ta xử lý với một thuộc tính thông thường trên các đối tượng hoặc điều khiển bất kỳ khác. Nhưng những gì chúng ta làm là cho phép người sử dụng đối tượng chúng ta đổi X thành giá trị mà họ mong muốn.
Bây giơ ta sẽ khai báo X là thuộc tính Public nó cũng tương tự. Nhưng nó cũng không giống hẳn. Đối với thuộc tính Public mỗi khi đổi giá trị một đoạn chương trình bên trong sẽ thi hành. Trong đoạn chương trình này ta có thể quyết định một đoạn chương trình nào mà người sử dụng chỉ . Thuộc tính luôn có một ddoanju chương trình chạy bên trong mỗi khi nó được truy cập.
Dùng thuộc tính thay cho biến cũng hạn chế khả năng sai sót vì giá trị truyền vào lớp được kiểm nghiệm nhờ đoạn chương trình kiểm tra bên trong lớp.
Trong thực tế thuộc tính hữu dụng hơn biến vì đôI khi ta cần một xử lý hơn là chỉ gán giá trị

Ví dụ

1. Ta khai báo biến để chứa giá trị thuộc tính

```
Option Explicit
```

Private mvarX as Integer

Biến này có tầm hoạt động bên trong mô - đun lớp.

2. Thêm chương trình vào thuộc tính X

```
Public Property Let X ( ByVal vData As Integer)
    mvarX = v Data
End Property
Pub lic property Get X () As Integer
   X = mvarX
End Property
```

Đoạn chương trình này không thi hành trực tiếp trừ khi nó được gọi thông qua thuộc tính đối tượng

```
Dim Mybox As New clsBox
My Box.X = 100
```

Khi ta gán giá trị 100 cho thuộc tính X, thực chất ta đang gọi thủ tục ***Property Let X***

```
New _ Position = Mybox.X
```

Nghĩa là thủ tục ***Property Get X thi hành***

```
Public Property Get X () As Integer
    X=mvarX
End Property
```

Thủ tục ***Property Let*** được gọi khi đổi giá trị thuộc tính. Giá trị đổi sẽ được chứa vào một biến cục bên trong lớp .

Thủ tục ***Property Get*** được gọi khi cần đọc giá trị thuộc tính. Giá trị được chứa trong biến cục bộ được trả về ***Property Get.***

Tuy hai thủ tục thuộc tính này chỉ làm việc với các dữ liệu cơ bản như **Variant,** ***String, Integer,*** vv… Đối với thuộc tính chứa đối tượng, thay vì dùng Property Let, ta dùng ***Property Set***

Ví dụ

```
Public Property Set Font ( ByVal New_Font As StdFont)
    Set mvarFont = New_Font
New Property
```

Để thuộc tính Font của đối tượng MyOject từ ứng dụng ta cho nó đối tượng Font MyFont. Tuy nhiên để đảm bảo VisualBasic dùng thủ tục Property Set trước thuộc tính :

```
Dim myFont As New StdFont ' StdFont is the VB class name ' for a Font
object
    myFont .Name = "Courier"
   myFont .Bold = True
  Set myobject.Font = myFont    ' class  the
```

```
                                                    ' Property  Set  procedure
```

Tương tự ta hoàn tất các thuộc tính còn lại của lớp ClsBox

3. Trong phần General Declarations, thêm, các biến cục bộ

```
Option  Explicit
Private mvarY  As  Integer
Private  mvarWidth AS  Integer
Private  mvarHeight AS  Integer
```

4. Thêm các thủ tục tiếp theo

```
Public Propety Let Y (ByVal  vData As Integer )
      mvarY  =  vData
End Property

Public Property Get Y As Integer
     Y  = mvarY
End Property

Public Propety Let Y (ByVal  vData As Integer )
      MvarWidth  =  vData
End Property

Public Property Get Width As Integer
     Width  = mvar Width
End Property

Public Propety Let Height (ByVal  vData As Integer )
      Mvar Hieght =  vData
End Property

Public Property Get Height As Integer
     Width  = mvar Height
End Property
```

5. Lưu mô - đun thành tập tin clsBox.cls
6. Đến đây, ta cần hai phương thức nữa là vẽ hộp (***DrawBox***) và xoá hộp (***ClearBox***). Cả hai phương thức có một phương thức truyền là đối tượng để vẽ hộp lên. Nó có thể là biểu mẫu, hộp hình,vv…

5.1.2.1.2 PHƯƠNG THỨC CỦA LỚP

Ví dụ

1. Thêm đoạn chương trình sau vào mô - đun lớp

Public Sub DrawBox (Canvas As Object)
Canvas.Line (mvarX, mvarY) - (mvarX + mvar Width, _ mvarY + mvarHeight) , , B
End sub
Đoạn chương trình này sử dụng phương thức Line của đối tượng Canvar

2. Kế đến ta thêm ClearBox vào Lớp
Public Sub ClearBox (Canvas As Object)
Canvas.Line (Canvas.Line (mvarX, mvarY) - (mvarX + mvar Width, _ mvarY + mvarHeight) , Canvas.backcolor , B

3. Lưu mô - đun với tên clsBox.cls
Vì hai thủ tục này sẽ được dùng làm phương thức của đối tượng nên chúng được khai báo Public, nghĩa là chúng có thể được gọi từ bên ngoài mô - đun .

5.1.2.2 TẠO INSTANCE CHO LỚP
Ví dụ mẫu – Tạo hoạt động cho đối tượng hộp
1. Nếu bạn đang mở đề án trong ví dụ trước, chỉ cần nhấn đúp chuột lên biểu mẫu để mở cửa sổ Code .Nếu không tạo đề án mới kiểu stand EXE .Từ menu Project chọn Add Class M odule; sau đó chọn **Tab Existing** trong hộp thoại và chọn clsBox.cls.

2. Tìm sự kiện Click trong danh sách và đưa đoạn chương trình sau vào:
Dim A_ Box As New clsBox
Biến đối tượng A_Box sẽ giữ một Instance của lớp .Từ khoá New rất quan trọng ,nếu thiếu nó ,VB sẽ cho rằng ta muốn tạo một bản sao của đối tượng clsBox hiện hành .Khi tham chiếu đến nó, ta sẽ gặp lỗi .

3. Đưa vào đoạn chương trình sử dụng đối tượng :
```
Private Sub Form_Click()
    Dim A_Box As New clsBox
    Dim nindex As integer
    Width A_Box
      .x=0
      .y=0
      .width=1000
      .Height=1000
     For nIndex
     .DrawBox  Me
Next
End Width
End sub
```

4. Thi hành chương trình nhấn chuột vào biểu mẫu ta sẽ thấy hộp trược dọc theo biểu mẫu.

5.1.2.2.1. KIỂM TRA GIÁ TRỊ THUỘC TÍNH

Trong thủ tục thuộc tính ***Property Let X***, nếu ta chuyền vào chuỗi ký tự ***" Hello Word "***, trình biên dịch sẽ báo lỗi

```
Public Property Let X( ByVal   vData  As  Integer )
   mvarX =   vData
End property
```

Tuy nhiên nếu ta chuyền số -7983, hộp chắc chắn sẽ không hiển thị trên biểu mẫu. Ta có thể cấm điều này bằng cách

```
Public Property Let X( ByVal   vData  As  Integer )
   IF vData >0 then mvarX = vData
End property
```

Đối tượng sẽ bỏ qua giá trị âm truyền vào.

5.1.2.1.3 THUỘC TÍNH CHỈ ĐƯỢC ĐỌC (READ – ONLY)

Đối với thuộc tính chỉ được đọc, ta không thể thay đổi giá trị thuộc tính, muốn vậy ta chỉ cần loại bỏ giá trị thủ tục ***Property Let*** trong mô - đun lớp

5.1.3 THAM SỐ TUỲ CHỌN

Ta có thể sử dụng tham số tuỳ chọn trong các phương thức sau

Ví dụ

1. Dừng chương trình, trong cửa sổ *Project Exxplorer*, nhấn đúp chuột nên clsBox để mở cửa sổ Code
2. Tìm phương thức ***ClearBox***, đánh dấu khối thủ tục rồi ấn Delete xoá nó đi
3. Sửa phương thức ***DrawBox*** để thêm vào tham số tuỳ chọn màu

```
Public Sub DrawBox ( Canvas  As  Object ,_Optional  1 color As
Variant)
    If Is  MissLng( 1color ) Then
      Canvas.Line   (mvarX, mvarY ) – ( mvarX  + mvar Width, _ mvarY
+ mvarHeight) ,       ,   B
    Else
      Canvas.Line   (mvarX, mvarY ) – ( mvarX  + mvar Width, _ mvarY
+ mvarHeight) ,   1color      ,   B
End if
End sub
```

4. Đến đây chương trình chưa thể biên dịch, vì vẫn còn dòng lệnh tham chiếu đến phương thức *ClearBox*

ClearBox Me

Xoá dòng này và thay thế dòng lệnh

DrawBox Me, Me.Backcolor

5. Thi hành dòng lệnh. Không có thay đổi trong kết quả

Từ khoá Optional cho biết tham số phía sau nó không nhất thiết là phải truyền khi gọi phương thức. Để biết được khi nào có tham số được truyền, ta dùng hàm IsMissing. Hàm này trả về giá trị True/ False. Khi có hàm truyền ta gọi hàm Line có chỉ định màu.

Thận trọng

Mặc dù tham số chỉ định có tính linh hoạt , giúp ta giảm số dòng chương trình nhưng nó cũng cho ta rắc rối kèm theo

Một trong những lý do để truyền từ lập trình theo cấu trúc sang lập trình hướng đối tượng ta làm cho chương trình dễ đọc, dễ hiểu

Tham số tuỳ chọn cung cấp tính năng táI sử dụng chương trình.

5.1.4 SỰ KIỆN CỦA LỚP

Định nghĩa

Sự kiện cho lớp đã có trong VB5. Nó vẫn hữu dụng với VB6. Chẳng hạn ta muốn mỗi lần hộp được vẽ trên màn hình, sự kiện Draw gây ra hoạt động cập nhật trên chương trình

Ví dụ

1. Định nghĩa sự kiện ***Draw*** . Một trong những thông tin cập nhật nhất là toạ độ (x,y) của

hộp. Mở cửa sổ ***Code*** và thêm dòng lệnh sau ***General Declarations***

Public Event Draw (X as Integer, Y as Integer)

Tuy nhiên dòng lệnh này chưa thể hiện lúc nào thì sự kiện được kích hoạt.

2. Ta muốn sự kiện ***Draw*** được sinh ra mỗi khi hộp được vẽ trên biểu mẫu. Ta tìm phương thức ***DrawBox*** và thêm dòng lệnh in đậm

```
Public Sub DrawBox ( Canvas As Object ,_Optional 1 color As Variant)
    If Is MissLng( 1color ) Then
      Canvas.Line (mvarX, mvarY ) – ( mvarX + mvar Width, _ mvarY + mvarHeight) ,  ,  B
    Else
      Canvas.Line (mvarX, mvarY ) – ( mvarX + mvar Width, _ mvarY + mvarHeight) ,  1color  ,  B

    End if
    RaiseEvent Draw( mvarX, mvarY)
    End sub
```

3. Tìm thủ tục sử lý sự kiện Click của biểu mẫu. Tìm và xoá dòng lệnh tạo đối tượng A_Box và thêm một dòng vào phần GeneralDeclarations

```
Private withevents A_Box As clsBox
Private sub Form _click()
    ‘ remove the line here
```

```
Dim nIndex As Integer
With A_Box
……
```

4. Thêm một dòng vào sự kiện Form_load

```
Private sub Form _click()
  Set A_Box = New clsBox
End sub
```

5. Chọn A_Box từ danh sách trong cửa sổ Code. Chọn Draw trong danh sách các sự kiện
6. Trong sự kiện này ta dùng Print để in ra toạ độ của hộp

```
Public Sub A_Boxx_Draw ( XAs Integer, Y As Integer )
   Debug. Print " The box just got drawn at " _
  & / & ", " & Y
End sub
```

7. Thi hành chương trình, nhấn chuột trên biểu mẫu ta thấy hộp trượt trên màn hình, và cửa sổ Immediate và các dòng văn bản.

Ta dùng phương thức RaisEvent để yêu cầu VB phát ra sự kiện Draw

Để có thể sử lý sự kiện của một đối tượng tạo ra, ta cần khai báo đối tượng hơi khác một chút.

5. 2.BIẾN ĐỐI TƯỢNG

Với biến đối tượng, ta có thể

- Tạo điều khiển mới trong lúc hiện hành
- Coppy điều khiển để sinh ra một instance mới của điều khiển hiện hành
- Tạo bản sao biểu mẫu với cùng tên cùng điều khiển và chương trình
- Biến đối tượng cung cấp các khả năng xây dựng các thủ tục tổng quát để xử lý với những điều khiển nhất định

5.2.1 TẠO ĐIỀU KHIỂN LÚC HIỆN HÀNH

Cách đơn giản nhất là tạo một mảng điều khiển vào lúc thiết kế. Nếu ta định thuộc tính Index của điều khiển đầu tiên là 0 lúc thiết kế ta có thêm điều khiển.

Tương tự biến mảng ta có thể mở rộng và rút gọn mảng điều khiển.

Ví dụ

Thử tạo một dãy nút lệnh trên biểu mẫu.

1. Tạo đề ấn mới và vẽ một nút lệnh lên biểu mẫu
2. Đổi thuộc tính ***Index*** thành 0 khi ấy một mảng điều khiển có một phần tử được tạo ra
3. Đưa đoạn chương trình sau vào thủ tục ***Click*** của nút lệnh

```
Private sub command1_Click(Index As Integer )
  Static sNextOperation As String
  Dim nIndex As Integer
  For nIndex =1 to 5
```

```
    If sNextOperation  = "Unload"  then
     Unload Command1(nIndex)
   .Top = Command1 ( n Index - 1).Top – Command1 ( nIndex - 1).Height
   .Caption = nIndex
Visible=True
End width
End if
Next
If sNextOperation = "Unload" Then
  SnextOperation = "LOAD"
Else
NextOperation = "Unlooad"
End if
End sub
```

4. Thi hành đoạn chương trình và nhấn nút lệnh vài lần Mỗi lần nhấn 5 nút lệnh được tạo hoặc xoá

5. Lưu vào đĩa với tên NewCtrl.vbp

Vòng lặp For… Next tạo hoặc xoá nút lệnh tuỳ theo nội dung biến sNext Operation. . Trước hết nội dung của SnextOperation được kiểm tra để xem cần Looad hay UnLoad các phần tử . ần đầu sNext Operation chưa được gán nó rơi vào phần Else nghĩa là Load Bởi mặc định điều khiển mới tạo lúc thi hành xuất hiện tại cùng vị trí với điều khiển gốc và không hiển thị. Do đó ta có thể đổi vị trí và điều chỉnh kích cỡ mà không để người sử dụng thấy .Ta chỉ cho chúng hiển thị sau khi đã có vị trí mới

5.2.2 SỰ KIỆN CỦA MẢNG ĐIỀU KHIỂN

Mặc dù hiển thị khác nhau 6 phần tử gồm một bản gốc và một bản coppy vẫn chia sẻ một thủ tục xử lý xự kiện, vì nhấn vào một nút bất kỳ chúng đều đáp ứng như nhau.

Ví dụ

1. Mở đề án NewCtrl.vbp và chọn sự kiện Click trên nút lệnh.
2. Thêm đoạn chương trình sau vào

```
Private sub command1_click( Index As Integer)
    Dim nIndex As Integer

     Select case Index
      Case=0
        For nIndex =1 To 5
          If  sNextOperation  = " Unload"  Then
          Unload Command1 "nIndex "
          Else
            Load Command1 ( nIndex)
            With  Command1 ( nIndex)
             .Top  =  Command1  (  n  Index  -  1).Top  +Command1  (  nIndex  -
1).Height
```

```
    .Caption = nIndex
  Visible= true
  End width
  End if
  Next
  If sNextOperation = "Unload" Then
     SnextOperation = "LOAD"
  Else
  sNextOperation = "Unlooad"
  End if
  Case 1, 2, 3, 4, 5
    Msg Box " You presses Button " & nIndex
  End Select
  End sub
```

3.Thi hành chương trình

Nhấn chuột trên từng nút lệnh, một thông điệp xuất hiện cho biết thứ tự nút nhấn

5.2.3 QUẢN LÝ ĐIỀU KHIỂN NHƯ BIẾN ĐỐI TƯỢNG

Không chỉ dùng biến đối tượng như mảng điều khiển ta còn có thể truyền biến đối tượng và mảng đối tượng vào thủ tục hay hàm

5.2.3.1 HÀM KIỂM TRA HỘP VĂN BẢN

Hàm này sẽ tự động biến kiểu dữ liệu mà mỗi hộp văn bản cần cũng như chiều dài tối đa của dữ liệu

Ví dụ

1. Tạo một đề án mới và vẽ một biểu mẫu gồm các điều khiển như sau
2. Thiết lập thuộc tính cho hộp văn bản như sau

Mô tả	Thuộc tính	Giá Trị
Hộp " alphabetic only" (Chỉ nhận chữ)	Name Index Tag	Txt Validate 0 N12
Hộp " Number Only " (Chỉ nhận số)	Name Index Tag	Txt Validate 1 N5

Mô tả	Thuộc tính	Giá Trị
Hộp " Anything "	Name	Txt Validate

(Nhận mọi thứ)	Index Tag	2 *4

3. Viết chương trình

```
Option  Explicit
 Private sub ValidateKeyPress ( TxtControl As TextBox, _ nKeyAsci As Integer)
     Dim sMaxLength As string
 Dim sKey As string * 1
  If  nKeyAscii <32 or nKey Ascii >126 then Eexxit  Sub
      sMax.Hieght = Rights (txtControl.Tag,_
          Len(txt Control.Tag) -1)
  If  Len ( TxtControl.Text) >= Val ( sMaxLenght ) Then
        Beep
        nKey.Ascii =0
        Exit Sub
  End if
 sKey = UCase (Chr$ (nkey Ascii))
   Select case Let$( TxtControl.Tag,   1)
  Case "A"
    If Asc (sKey) < 65 or Asc (sKey) >90 Then
          Beep
        nKey.Ascii =0
        Exit Sub
  End if
        Case "N"
    If Asc (sKey) < 48 or Asc (sKey) > 57 Then
          Beep
        nKey.Ascii =0
        Exit Sub
    End if
  End select
End sub

          Private sub TxtValidate_KeyPress ( Index As Integer,_ KeyAscii As Integer)
          ValidatedatekeyPress TxtValidate ( Index ) KeyAscii
   End sub
```

4. Thi hành chương trình

ValidateKeyPress là thủ tục ở mức biểu mẫu và được khai báo trong phần General. nKeyascii là mã phím nhấn. Vì từ khoá ByVal không được nêu ra, nên tham số được truyền bằng tham chiếu. Đổi KeyAscii về 0 trong sự kiện CasePress là tắt phím nhấn. Thuộc tính Tab được dùng như một nhãn riêng đa năng cho các điều khiển. Nó cho biết kiểy dữ liệu được cho phép trong mỗi hộp vawn bản
Ta có thể đổi kiểu dữ liệu của một hộp văn bản bằng cách đổi thuộc tính Tab trong cửa sổ Properties.
Giai đoạn đầu tiên, chương trình kiểm tra nKeyascii cho các ký tự đặc biệt(như Bksp hay các phím bấm định hướng). Nếu chúng được nhấn phím đó sẽ bỏ qua

```
If nKeyascii <32 or nKeyascii >126  then Eixt Sub
```

Dòng lệnh kế lấy giá trị từ thuộc tính Tab đưa vào biến sMaxLenght
Sau đó kiểm tra chiều dài tối đa

```
If  Len ( TxtControl.Text) >= Val ( sMaxLenght ) Then
    Beep
    nKey.Ascii =0
    Exit Sub
End if
```

Select Case so sánh phím ký tự với kiểu dữ liệu trong thuộc tính Tab
Hàm Chr$ chuyển mã ký tự nKeyascii thành chuỗi ký tự
Hàm Asc làm ngược lại trả về mã ký tự ASCII.

5.2.4 KHAI BÁO BIẾN ĐỐI TƯỢNG

Ta có thể khai báo một biến đối tượng một cách tường minh bằng cách cung cấp một kiểu dữ liệu mà VB nhận ra

```
Dim TxtControl As TextBox
```

Chương trình sẽ hiệu quả dễ gỡ rối và sẽ chạy nhanh hơn

```
Dim ct1Control As Control
```

Tham số hàm và thủ tục có thể khai báo theo kiểu này nó chó phép ta truyền một điều khiển bất kỳ. Sau đó dùng dòng lệnh TypeOf để kiểm tra kiểu điều khiển liên quan đến một đối tượng
Nếu khai báo tường minh VB kiểm tra thuộc tính của đối tượng ngay lúc biên dịch. Nếu khai báo ẩn (không tường minh) VB chỉ kiểm tra thuộc tính lúc thi hành

5.2.4.1 KIỂU BIẾN ĐỐI TƯỢNG

Danh sách các kiểu đối tượng mà VB có thể nhận ra

CheckBox	ComboBox	CommandButton	MDIForm
Data	DirrListBox	DriveListBox	FileListBox
Grid	Frame	HscrollBar	Image
Lable	Line	ListBox	Menu
OptionButton	OLE	PictureBox	Shape
TextBox	Timer	VscrollBar	Form

Đây là đối tượng chuẩn chúng là tên lớp và được đặt kế bên tên điều khiêntrong cửa sổ Properties

5.3 TẬP HỢP (COLLECTION)

Một biểu mẫu trong ứng dụng có một tập hợp các điều khiển nội tại. Trong lúc thi hành, ta có thể dùng tập hợp để truy cập đến điều khiển trên biểu mẫu. Khi thêm hoặc xoá một điều khiển trên biểu mẫu, VB tự động quản lý việc thêm hay xoá điều khiển trên tập hợp
Truy cập phần tử trong tập hợp tương tự như truy cập phần tử trong mảng thông thường.
Tập hợp rất tiện lợi cho các biểu maauxnhaapj liệu.
Khác với mảng điều khiển tập hợp không hỡ trợ các điều khiển. Tuy nhiên từng điều khiển trong tập hợp đều có những thuộc tính, phương thức và sự kiện

5.3.1 THUỘC TÍNH CONTROLS

Thuộc tính Control của biểu mẫu chỉ được truy cập thông tin qua chương trình, nó thực chất là một mảng các biến đối tượng. Trong đó mỗi phần tử của mảng là một điều khiển đơn: phần tử số 0 là điều khiển đầu tiên trên biểu mẫu, phần tử số 1 là điều khiển thứ hai….
Số thứ tự trong mảng được gán tự độn. Nếu có 2 hộp văn bản trên biểu mẫu, ta có thể đổi thuộc tính Text của từng điều khiển sau.

```
Form1.Controls(0).Text = " Control 0 "
Form1.Controls(1).Text = " Control 1 "
```

Cú pháp

```
<Tên biểu mẫu>.Controls(< Số thứ tự >).< Thuộc tính> = < Giá trị>
```

5.3.2 XÁC ĐỊNH ĐIỀU KHIỂN TRÊN BIỂU MẪU

Mảng Control có một thuộc tính gọi là Count, xác định số điều khiển trên biểu mẫu. Lưu ý rắng mảng được đánh số thứ tự từ 0. Nếu Count là 3 nghĩa là các phần tử sẽ được đánh là 0, 1, 2
Ta có thể dùng TypeOf để xử lý với nhóm điều khiển, mặc dù không chỉ ra chính xác một phần tử nhưng ta có thể xác định các điều khiển cùng kiểu.

```
For nControlNo = 0 To Form1.Controls.Count – 1
If Type Of Form1.Controls(nControlNo) If TextBox then
        .---lenh---
End If
 Next
```

Ta duyệt qua từng phần tử trên Form1 từ 0 đến phần tử cuối cùng, Count – 1. Với từng điều khiển ta dùng TypeOf để kiểm tra xem nó có phảI là hộp văn bản hay không.
Tuy nhiên ta có thể làm theo cách khác

```
For Each  objControl Is Form1.Controls
 If Type ObjControls Is Text1Box Then
 :
End If
 Next
```

5.4 BIỂU MẪU MDI

Biểu mẫu MDI cho phép nhóm các biểu mẫu và chức năng trong một cửa sổ lớn. Tuy nhiên biểu mẫu MDI có một số nhược điểm : chỉ có vài điều khiển được vẽ trên biểu mẫu MDI. Đó là điều khiển định giờ (timer) và hộp hình. Trong phiên bản Professional và Enterpirse ta có thể vẽ thêm thanh trạng tháI và thanh công cụ.

5.4.1 BIỂU MỘU CON (CHILD FORM)

Thuộc tính MDIChild của một biểu mẫu là mộ giá trị True/ False cho biết biểu mẫu có phảI là biểu mẫu con trong một biểu mẫu MDI hay không. Bởi vì VB chỉ cho phép tòn tại một biểu mẫu MDI trong ứng dụng, biểu mẫu con tự động nhận định cửa sổ cha(Paren) và khi thi hành nó chỉ được hoạt động trong cửa sổ cha

Vào lúc thiết kế không thể nhận biết cửa sổ độc lập với cửa sổ con và chỉ khác nhau ở chỗ thuộc tính MĐIChild thuộc tính này không được gán vào lúc thi hành nếu không sẽ báo lỗi khi thi hành chương trình

5.4.2 TẠO INTANCE CỦA BIỂU MẪU

Sử dụng biến đói tượng để tạo ra những bản sao của một biểu mãu. Từng bản sao có các điều khiển và Menu như nhau nhưng có những dữ liệu khác nhau . Mặc dù chương trình cũng như tên biến và tên điều khiển như nhau nhưng dữ liệu được chứa ở những nơi khác nhau trong bộ nhớ

5.4.3 XÁC ĐỊNH BIỂU MẪU

Vì ta có thể tạo ra 10 biểu mẫu đồng nhất có cùng tên nên việc xác định cửa sổ là điều cần thiết. Từ khoá Me cho phép ta tham chiếu đến cửa sổ hiện hành là cửa sổ đang có Focus hay nói cách khác là cửa sổ nhận được mọi phím nhấn hay click chuột bất kỳ

Ta có thể dùng :

Activeform.txtEmployee.text = “peter”

Nhưng Me là cách thông dụng nhất

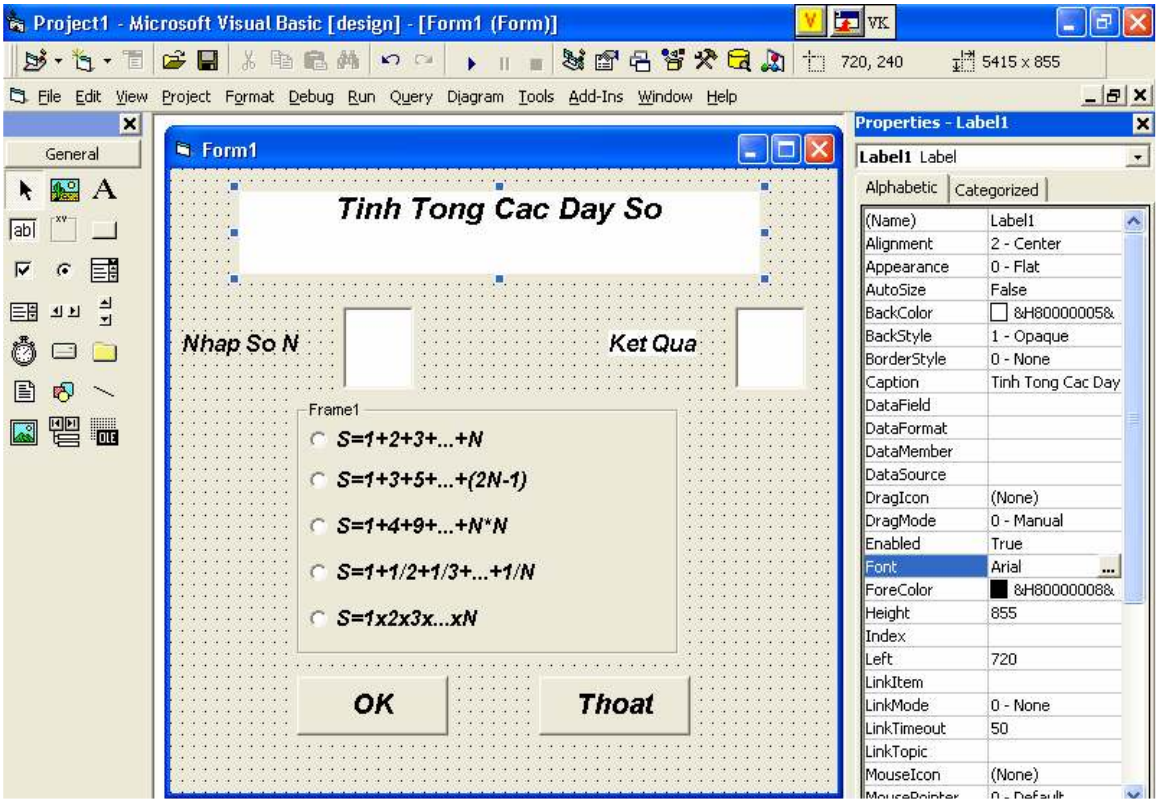
6. CÁC BÀI ỨNG DỤNG CỦA FORM

6.1 TÍNH TỔNG CỦA DÃY SỐ

Với yêu cầu của đề bài như sau:

1. **S=1+2+3+..+N**
2. **S=1+3+5+..+(2N-1)**
3. **S=1+4+9+..+N^2**
4. **S=1+1/2+1/3+..+1/N**
5. **N!=1x2x3x..xN**

Khi chạy chương trình xuất hiện một biểu mẫucó tên là : FRMTongCuaCacDaySo như trong hình dưới :



Hình 6.1

Hướng Dẫn :

Để thiết kế chương trình này bạn lần lượt làm theo các bước sau:

1. Mở một dự án mới
2. Lưu tập tin Dự án với tên TongCacDaySo.VBP và tập tin biểu mẫu với tên TongCâcDaySo.Frm
3. Thiết kế Biểu mẫu theo như bảng mô tả dưới đây

Đối Tượng	Thuộc Tính	Giá Trị
Form	Name Caption	FRMTongcacdayso Tính Tổng Các Dãy Số
Label	Name Caption	lblNhan Tinh Tổng Các Dãy Số
Label	Name Caption	
Label	Name Caption	
Text Box	Name Tetx	TXTSoN (Để Trống)
Text Box	Name Tetx	TXTKQ (Để Trống)

CommandButton	**Name** **Caption**	**CmdOK** **&OK**
CommandButton	**Name** **Caption**	**Cmdthoat** **&Thoat**
Option Button	**Name** **Caption**	**Opt1** **S=1+2+3+…+N**
Option Button	**Name** **Caption**	**Opt2** **S=1+3+5+…+(2N-1)**
Option Button	**Name** **Caption**	**Opt3** **S=1+4+9+…+N*N**
Option Button	**Name** **Caption**	**Opt4** **S=1+1/2+1/3+…+1/N**
Option Button	**Name** **Caption**	**Opt5** **S=1x2x3x…xN**

4.Lưu Dự án .Sau đó ta viết Code cho các nút lệnh

- **Viết Lệnh cho nút OK**

```
Private Sub CmdOK_Click()
N = Val(txtsoN.Text)
TxtKQ.Text = " "

If Opt1.Value = True Then
S = 0
For i = 1 To N
S = S + i
Next i
TxtKQ.Text = S

End If
If Opt2.Value = True Then
S = 0
For i = 1 To N
S = S + (2 * i - 1)
Next i
TxtKQ.Text = S

End If
If Opt3.Value = True Then
S = 0
For i = 1 To n
S = S + i * i
Next i
TxtKQ.Text = S
```

```
End If
If Opt4.Value = True Then
S = 0
For i = 1 To n
Next i
S = S + 1 / i
TxtKQ.Text = s

End If
If Opt5.Value = True Then
GT = 1
For i = 1 To n
GT = GT * i
Next i
TxtKQ.Text = gt
End If
End Sub
```

- **Viết lệnh cho nút thoát**

```
Private Sub CmdThoat_Click()
End
End Sub
```

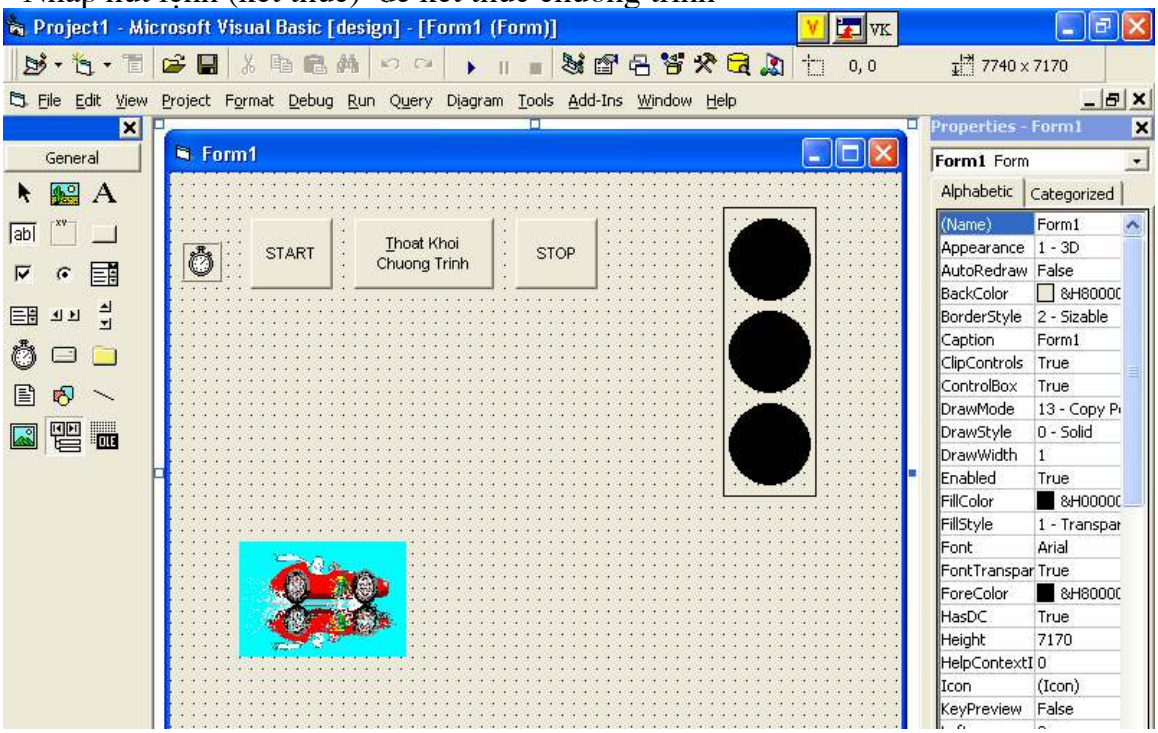
- **Viết các lệnh cho thủ tục tính tổng**

```
Option Explicit
Dim n, i As Integer
Dim s, gt As Integer
```

6.2 MÔ TẢ HOẠT ĐỘNG CỦA ĐÈN GIAO THÔNG

Trước hết ta tạo một Form theo Yêu cầu :

- Khi chạy chương trình thì xe chuyển động theo điều khiển của đèn giao thông
- Nhấp nút lệnh (kết thúc) để kết thúc chương trình



Hình 6.2

Ta thiết kế Biểu Mẫu theo như bảng mô tả dưới đây

Đối Tượng	Thuộc Tính	Giá Trị
Form	Name Caption	FrmGiaoThong Điều Khiển Giao Thông
Command Button	Name Caption FontName FontSize	CmdStart &Start VNI- Helve 10
Command Button	Name Caption FontName FontSize	CmdStop &Stop VNI- Helve 10
Command Button	Name Caption FontName FontSize	CmdKetThuc &Kết Thúc VNI- Helve 10
Image	Name Picture	ImgXe Car.WMF

	Stretch	**True**
Shape	**Name** **Shape** **FillStyle**	**ShpVang** **3-Circle** **0- Solid**
Shape	**Name** **Shape** **FillStyle**	**ShpDo** **3-Circle** **0- Solid**
Shape	**Name** **Shape** **FillStyle**	**ShpXanh** **3-Circle** **0- Solid**
Shape	**Name** **Shape** **FillStyle**	**ShpChuNhat** **3-Circle** **0- Solid**
Label	**Name** **Text** **Alignment**	**lblNhan** **Điều Khiển Giao Thông** **2-Center**
Timer	**Name** **Interval**	**TmrXe** **200**

Viết Code cho Chương trình

- **Viết lệnh cho Nút Timer**

```
Private Sub Tmrxe_Timer()
Static a As Integer
Select Case a
Case 0
Shpdo.FillColor = vbRed
Shpvang.FillColor = vbBlack
Shpxanh.FillColor = vbBlack
a = 1
If Imgxe.Left > 8000 Then
Imgxe.Left = 100
End If
Imgxe.Move Imgxe.Left + 300
Case 1
Shpdo.FillColor = vbBlack
Shpvang.FillColor = vbYellow
Shpxanh.FillColor = vbbalck
a = 2
Case 2
Shpdo.FillColor = vbbbalck
Shpvang.FillColor = vbbalck
```

```
Shpxanh.FillColor = vbGreen
a = 0
End Select
End Sub
```

- **Viết Lệnh cho nút Start và Stop**

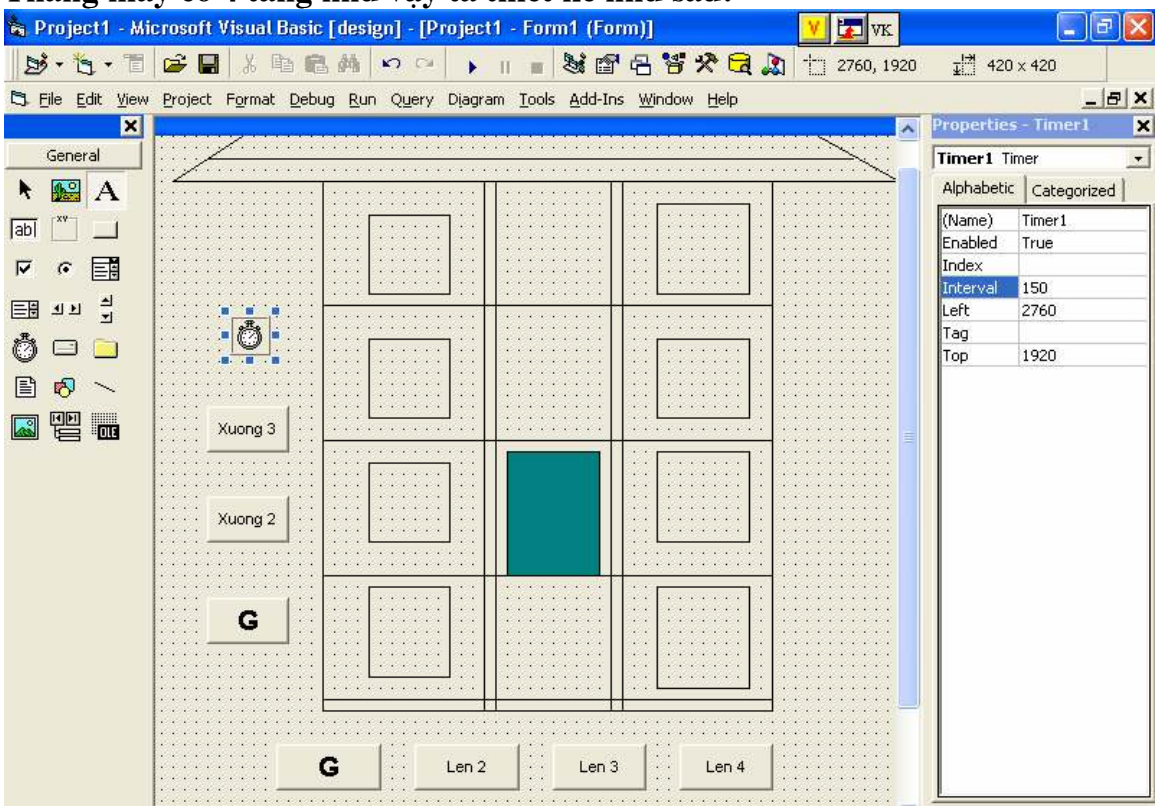
```
Private Sub CmdStart_Click()
Tmrxe.Enabled = True
End Sub
Private Sub CmdStop_Click()
Tmrxe.Enabled = False
End Sub
```

- **Viết lệnh cho nút Kết Thúc**

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

6.3 MÔ PHỎNG THANG MÁY

Để thực hiện bài này cần chú ý một số tính chất của đối tượng Shape
Thang máy có 4 tầng như vậy ta thiết kế như sau:



Hình 6.3

Bản mô tả các thuộc tính của Biểu Mẫu

Đối Tượng	Thuộc Tính	Giá Trị
Form	**Name** **Caption**	**FrmThangMay** **Mô Phỏng Thang Máy**
Command Button1	**Name** **Caption** **Font**	**CmdG** **G** **VNI-Center**
Command Button2	**Name** **Caption** **Font**	**CmdLen2** **Len 2**
Command Button3	**Name** **Caption** **Font**	**CmdLen3** **Len 3**
Command Button4	**Name** **Caption** **Font**	**CmdLen4** **Len 4**
Command Button5	**Name** **Caption** **Font**	**CmdXuong3** **Xuong 3**
Command Button6	**Name** **Caption** **Font**	**CmdXuong2** **Xuong 2**
Shape	**Name** **Left** **Top** **Width**	**ShpThang** **5640** **3360** **975**

Viết code cho từng nút lệnh

-Viết Code cho Nút G

```
Private Sub Command1_Click()
        Timer1.Enabled = True
        TANG = 1
End Sub
```

-Viết Code cho nút Lên2

```
Private Sub CmdLen2_Click()
        Timer1.Enabled = True
        TANG = 2
End Sub
```

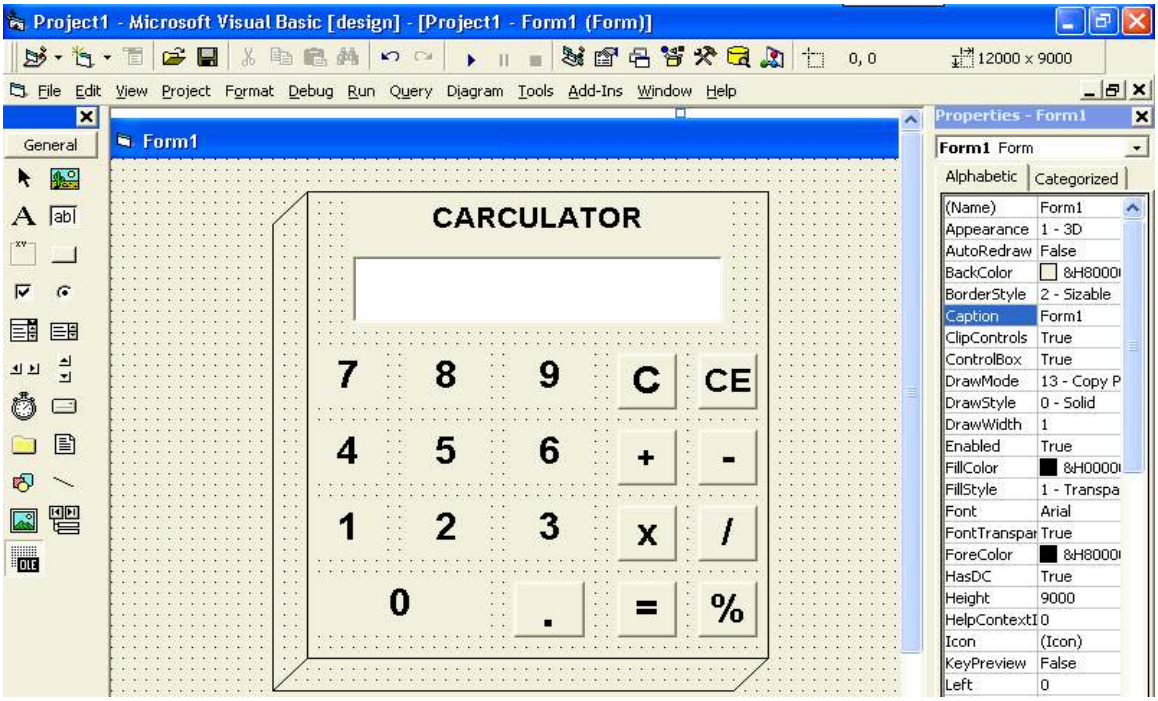
-Viết Code cho nút Len3

```
Private Sub CmdLen3_Click()
        Timer1.Enabled = True
        TANG = 3
```

```
End Sub
```
-Viết Code cho nút Lên4
```
Private Sub Cmdlen4_Click()
        Timer1.Enabled = True
        TANG = 4
End Sub
```
-Viết Code cho nút Xuống3
```
Private Sub CmdXuong3_Click()
        Timer1.Enabled = True
        TANG = 5
End Sub
```
-Viết Code cho nút Xuống2
```
Private Sub CmdXuong2_Click()
        Timer1.Enabled = True
        TANG = 6
End Sub
```
Nhấp đúp vào cửa sổ Form suất hiện cửa sổ CođeWindows ta vieet lệnh sau:
```
Private Sub Form_Load()
        TANG = 0
        Timer1.Enabled = False
End Sub
Private Sub Timer1_Timer()
        Shape1.BackColor = &HFF&
        Select Case TANG
        Case 1
                Shape1.Top = Shape1.Top + 90
                If (Shape1.Top > 4680) Then
                        Timer1.Enabled = False
                        Shape1.BackColor = &HFF00&
                End If
        Case 2
                Shape1.Top = Shape1.Top - 90
                If (Shape1.Top < 3240) Then
                        Timer1.Enabled = False
                        Shape1.BackColor = &HFF00&
                End If
        Case 3
                Shape1.Top = Shape1.Top - 90
                If (Shape1.Top < 2000) Then
                        Timer1.Enabled = False
                        Shape1.BackColor = &HFF00&
                End If
```

```
        Case 4
                Shape1.Top = Shape1.Top - 90
                If (Shape1.Top < 500) Then
                        Timer1.Enabled = False
                        Shape1.BackColor = &HFF00&
                End If
        Case 5
                Shape1.Top = Shape1.Top + 90
                If (Shape1.Top > 2000) Then
                        Timer1.Enabled = False
                        Shape1.BackColor = &HFF00&
                End If
        Case 6
                Shape1.Top = Shape1.Top + 90
                If (Shape1.Top > 3240) Then
                        Timer1.Enabled = False
                        Shape1.BackColor = &HFF00&
                End If
        End Select
End Sub
```

6.4 MÔ TẢ HOẠT ĐỘNG CỦA MÁY TÍNH BỎ TÚI



Hình 6.4

Bảng mô tả thuộc tính của biểu mẫu

Đối Tượng	Thuộc Tính	Giá Trị
Form	Name Caption	Frmmaytinh Máy Tính Bỏ Túi
Command Button	Name Caption	Cmdcong +
Command Button	Name Caption	Cmdtru -
Command Button	Name Caption	Cmdnhan x
Command Button	Name Caption	CmdChia /
Command Button	Name Caption	CmdCancel C
Command Button	Name Caption	CmdCancelEntry CE
Command Button	Name Caption	CmdDecimal .
Command Button	Name Caption	CmdTinh =
Label	Name Caption Alignment	Lbl0 0 2-Center

Label	**Name** **Caption** **Alignment**	**Lbl1** **1** **2-Center**
Label	**Name** **Caption** **Alignment**	**Lbl2** **2** **2-Center**
Label	**Name** **Caption** **Alignment**	**Lbl3** **3** **2-Center**
Label	**Name** **Caption** **Alignment**	**Lbl4** **4** **2-Center**
Label	**Name** **Caption** **Alignment**	**Lbl5** **5** **2-Center**
Label	**Name** **Caption** **Alignment**	**Lbl6** **6** **2-Center**
Label	**Name** **Caption** **Alignment**	**Lbl7** **7** **2-Center**
Label	**Name** **Caption** **Alignment**	**Lbl8** **8** **2-Center**
Label	**Name** **Caption** **Alignment**	**Lbl9** **9** **2-Center**
Label	**Name** **Caption** **Alignment**	**CmdMaytinhbotui** **Máy Tính Bỏ Túi** **2-Center**

❁ **Viết Code cho từng nút lệnh**

Viết lệnh khai báo cho biểu mẫu

Option Explicit
Dim op1, op2
Dim decimalflag As Integer
Dim numops As Integer
Dim Lastlnput
Dim OpFlag
Dim TempReadOut

Viết lệnh cho phím C

```
Private Sub cmdcancel_Click()
    readout = Format(0, "0.")
    op1 = 0
    op2 = 0
    Form_Load
End Sub
```

Viết lệnh cho phím CE

```
Private Sub cmdcancelEntry_Click()
    readout = Format(0, "0.")
    decimalflag = False
    LastInput = "CE"
    Form_Load
End Sub
```

Viết lệnh cho dấu chấm thập phân

```
Private Sub Cmddecimal_Click()
    If LastInput = "neg" Then
            readout = Format(0, "-0.")
        ElseIf LastInput <> "NumS" Then
        readout = Format(0, "0.")
    End If
    decimalflag = True
    LastInput = "NumS"
End Sub
```

Viết lệnh cho thủ tục FormLoad

```
Private Sub Form_Load()
    decimalflag = False
    numops = 0
    LastInput = "None"
    OpFlag = ""
    readout = Format(0, "0.")
    decima.Caption = Format(0, "")
End Sub
```

Viết lệnh cho các phím từ 0 đến 9

```
Private Sub number_click(index As Integer)
    If LastInput <> "NUMS" Then
```

```
        readout = Format(0, "")
        decimalflag = False
    End If
    If decimalflag Then
        readout = readout + Number(index).Caption
    Else
        readout = Left(readout, lnStr(readout, Format(0, "")) - 1) +
        Number(index).Caption + Format(0, "")
    End If
    If Lastlnput = "NEG" Then readout = "-" & Readoutlastlnput = "Num"
End Sub

Private Sub operator_click(index As Integer)
    If TempReadOut = readout then
        numops = numops + 1
    End If
    Select Case numops
    Case 0
        If operator(index).Caption = "-" And Lastlnput <> "neg" Then
            readout = "-" & readout
            Lastlnput = "neg"
        End If
    Case 1
        If operator(index).Caption = "-" And Lastlnput <> "nums" Then
            readout = "-"
            Lastlnput = "neg"
        End If
    Case 2
        op2 = TempReadOut
        Select Case OpFlag
        Case "+"
            op1 = CDbl(op1) + CDbl(op2)
        Case "-"
            op1 = CDbl(op1) - CDbl(op2)
        Case "*"
            op1 = CDbl(op1) * CDbl(op2)
        Case "/"
            If op2 = 0 Then
                magBox "Can'", 48, "Carculator"
            Else
                op1 = CDbl(op1) / CDbl(op2)
            End If
```

```
            Case "="
                op1 = CDbl(op2)
            case "%"
                op1 = CDbl(op1) * CDbl(op2)
            End Select
            readout = op1
            numops = 1
        End Select
        If Lastlnput <> "NEG" Then
            Lastlnput = "OPS"
            OpFlag = operator(index).Caption
        End If
    End Sub

    Private Sub percent_click()
        readout = readout / 100
        Lastlnput = "ops"
        OpFlag = "%"
        numops = numops + 1
        decimalflag = True
    End Sub
```

7. NHỮNG KHÁI NIỆM CƠ BẢN VỀ CƠ SỞ DỮ LIỆU

7.1 CƠ SỞ DỮ LIỆU LÀ GÌ

Cơ sở dữ liệu là cốt lõi của nhiều ứng dụng phần mềm kinh doanh. Cơ sở dữ liệu rất phổ biến trong thế giới lập trình cao cấp vì chúng cho phép truy cập đến các thông tin theo một cách nhất quán, hiệu quả và tương đối dễ dàng cho việc thiết lập bảo trì. Vậy cơ sở dữ liệu là gì :
Cơ sở dữ liệu là một kho chứa thông tin. Trong tập đồ án này chỉ quan tâm đến cơ sở dữ liệu quan hệ, là kiểu cơ sở dữ liệu phổ biến nhất hiện nay
Một cơ sở dữ liệu quan hệ gồm có :

- Chứa dữ liệu trong các bảng, được cấu tạo bởi các dòng còn gọi là các mẩu tin , và cột còn gọi là trường
- Cho phép lấy về (hay truy cập) các tập hợp dữ liệu con từ các bảng
- Cho phép nhối các bảng với nhau cho mục đích truy cập các mẩu tin liên quan với nhau chứa trong các bảng khác nhau

7.1.1 BỘ MÁY (ENGINE) CƠ SỞ DỮ LIỆU LÀ GÌ ?

Chức năng cơ bản của một bộ máy cơ sở dữ liệu được cung cấp bởi một bộ máy cơ sở dữ liệu là hệ thống chương trình quản lý cách thức chứa và trả về dữ liệu
Bộ máy cơ sở dữ liệu được trình bày ở đây là Microsoft jet. Jet ở đây là một hệ thống con được nhiều ứng dụng của Micrôst sử dụng
Lưu ý: Ngoài jet còn có nhiều bộ máy cơ sở dữ liệu khác nhưng vì Visual Basic hỗ trợ Jet một cách nội tại, hơn nữa Jet có thể hỗ trợ các bộ máy cơ sở dữ liệu khác

7.1.2 BẢNG VÀ TRƯỜNG

Các cơ sở dữ liệu được cấu tạo từ các bảng dùng để thể hiện các phần nhóm dữ liệu
Ví dụ : Nếu ta tạo một cơ sở dữ liệu để quản lý các tài nguyên trong công việc kinh doanh ta phảI tạo một bảng cho nhân viên. Bảng có cấu trúc định nghĩa sẵn và chứa dữ liệu phù hợp với cấu trúc sau ;

- Bảng : chứa các mẫu tin là các mẫu dữ liệu riêng rẽ bên trong phân nhóm dữ liệu
- Mẫu tin: Chứa các trường, mỗi trường thể hiện một bộ phận dữ liệu trong một mẫu tin

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

7.1.2.1 THIẾT KẾ CƠ SỞ DỮ LIỆU

Để tạo một cơ sở dữ liệu trước hết ta phải xác định thông tin gì cần theo dõi. Sau đó ta thiết kế cơ sở dữ liệu, tạo bảng chứa các trường định nghĩa kiểu dữ liệu sẽ có. Sau khi tạo ra cấu trúc cơ sở dữ liệu có thể chứa dữ liệu dưới dạng mẩu tin. Ta không thể đưa dữ liệu vào mà không có bảng hay định nghĩa trường vì dữ liệu sẽ không có chỗ để chứa. Do đó thiết kế cơ sở dữ liệu cực kỳ quan trọng nhất là rất khó thay đổi thiết kế một khi đã tạo ra nó

Ví dụ : Ta tạo ra bảng với nội dung sau

Bảng Khách Hàng	**Bảng** TblRegion	**Bảng Hoá Đơn**
TblCustomer	TblRegion	ID
ID	State	CustomerID
FirstName	RegionName	OrderDate
Lastname		Amount
Company		
Address		
City		
State		
Zip		
Pone		
Fax		
Email		

Có những mối quan hệ giữa hai bảng thông qua trường State. Đây là mối quan hệ Một chiều – Nhiều chiều.
Đối với mật mẩu tin trong LblRegion có thể không có, hoặc có một hoặc nhiều mẩu tin tương tự trong bảng LblCustomer
Cụm từ " Lbl " thể hiện tên bảng tên trường hiển thị đầy đủ không chứa khoảng trắng hay những ký tự đặc biệt khác như dấu gách dưới

7.1.3 RECORDEST LÀ GÌ ?

Thao tác trên bảng liên quan đến việc nhập và lấy về dữ liệu từ các bảng khác cũng như việc kiểm tra và sửa đổi cấu trúc bảng. Để thao tác với cấu trúc bảng ta dùng các câu lệnh

định nghĩa dữ liệu " Truy vấn " hoặc một đối tượng TableDef " các đối tượng truy cập dữ liệu ". Để thao tác dữ liệu trong một bảng ta dùng Recordset
Một Recordset là một cấu trúc dữ liệu thể hiện một tập hợp con các mẩu tin lấy về từ cơ sở dữ liệu
Các Recordset được thể hiện như là các đối tượng, về khái niệm tương tự như là các đối tượng giao diện người sử dụng. Cũng như các kiểu đối tượng khác trong VB các đối tượng Recordset có các thuộc tính và phương thức riêng

7.1.4 CÁC KIỂU DỮ LIỆU

Cơ sở dữ liệu nội tại của VB Jet cung cấp 21 kiểu dữ liệu khác nhau

Kiểu Dữ Liệu	**Ý Nghĩa**
Binary	Dùng để chứa các khối dữ liệu như là đồ hoạ và các tập tin âm thanh số hoá
Boolean	Giá trị đúng hoặc sai
Byte	Giá trị số nguyên một Byte từ 0 đến 255
Curency	Trường số có thuộc tính đặc biệt để chứa các giá trị tiền tệ chính xác
Date/Time	Giá trị 8 byte thể hiện ngày hoặc giờ từ ngày 1 tháng 1 năm 100 đến ngày 31 thang 12 năm 9999
Double	Kiểu dữ liệu 8 byte, dấu phẩy động
Guid	Là một số 128 byte cũng được gọi là định danh toàn thể duy nhất. Ta dùng số này để định dạng duy nhất các mẩu tin
Integer	Số 2 Byte đầy đủ từ -32,768 đền 32,767
Long	Số 4 Byte đầy đủ từ -2.147.483.648 đến 2.147.483.687 .Ta có thể quy định trường này là trường tăng tự động
Long Binary	Trường giá trị lớn có thể chứa các cấu trúc dữ liệu nhị phân như là hình ảnh hay tập tin
(OLE Object)	Kiểu Ole Object trong cơ sở dữ liệu có thể lên đến 1 GigaByte
Memo	Trong giá trị lớn có thể chứa đến 65,335 ký tự. Ta không cần thiết phảI khai báo trước chiều dài của trường này
Single Type	Dữ liệu 4 Byte, dấu phẩy đơn
Text	Kiểu dữ liệu có chiều dài cố định, đò hỏi ta phải khai báo chiều dài của trường khi ta khai báo kiểu dữ liệu.Trường văn bvar có thể dài từ 1 đến 255 ký tự
VarBinary	Mẫu dữ liệu nhị phân biến đổi

Không thể có tương đương một – một giữa kiểu dữ liệu VB và kiểu dữ liệu trong trường cơ sở dữ liệu
Ví dụ : Ta không thể quy định một trường cơ sở dữ liệu là kiểu định nghĩa bởi người sử dụng hay biến Object của Visual Basic

7.1.5:TẠO LƯỢC ĐỒ CƠ SỞ DỮ LIỆU

Mặc dù việc tạo danh sách các bảng và trường là cách tốt nhất để xác định cấu trúc cơ sở dữ liệu, ta còn có một cách khác để xem các bảng và trường dưới dạng đồ hoạ. Không chỉ xem được các bảng và trường mà ta còn thấy được mối quan hệ giữa chúng. Để làm được điều này ta tạo lược đồ

Lược đồ là bản đồ các con đường trong cơ sở dữ liệu. Lược đồ thể hện các bảng, trờng và mối quan hệ trong cơ sở dữ liệu. Có lược đồ cơ sở dữ liệu là phần quan trọng trong quá trình thiết kế phần mềm bởi vì nó cho ta một cách nhìn quan trọng về những gì trong cơ sở dữ liệu

Các lược đồ vẫn có vị trí quan trọng lâu dài sau khi quá trình thiết kế cơ sở dữ liệu hoàn tất. Ta xẽ cần đến lược đồ để thi hành các câu lệnh truy vấn trên nhiều bảng

Ta dùng công cụ Visio để thực hiện tạo lược đồ. Công cụ Visio tích hợp với các ứng dụng Windows khác nhất là Microsoft Office

7.1.5.1 DÙNG VISIO ĐỂ TẠO LƯỢC ĐỒ

7.1.5.1.1 VẼ BẢNG THỨ NHẤT

1. Khởi động Visio hộp thoại New xuất hiện
2. Chọn Basic Template
3. Trong khuông mẫu vẽ chọn hình dạng Rectangle và đặt nó vào vùng vẽ. Một hình chữ nhật xuất hiện.
4. Điều chỉnh hình chữ nhật sao cho diện rộng là 1,5 inch và chiều cao là 0.25 inch
5. Gõ tên bảng, TblCustomer vào hình chữ nhật
6. Từ khuôn mẫu vẽ một hình chữ nhật khác vào vùng vẽ
7. gõ tên trường cho bảng này vào hình chữ nhật. Bởi vì ta đang chọn công cụ Text, ta có thể gõ ngay lập tức

8, khi đã hoàn tất gõ tên các trường dùng con trỏ để điều chỉnh hình chữ nhật sao cho nó đủ rộng để hiển thị mọi trường rõ ràng

7.1.5.1.2 VẼ BẢNG THỨ HAI

1. Từ Menu Visio chọn Edit , Select All Rồi chọn Edit,Duplicate để copy hình đồ hoạ mà ta đã có
2. Bản sao Tbl Customer xuất hiện. Dùng chuột chọn và kéo nó ra khỏi bản gốc để không bị chùng hình
3. Chọn vào hình chữ nhật bản sao. Dùng công cụ Text, sửa tên nó thành TblOrder
4. Chọn vào hiùnh chữ nhật TblOrder. Dùng công cụ Text sửa danh sách các trường của nó đúng với thiết kế ban đầu
5. Chọn và rê cạnh dưới của hình chữ nhật để co nó lại ngắn hơn cho hợp với số trường ít hơn

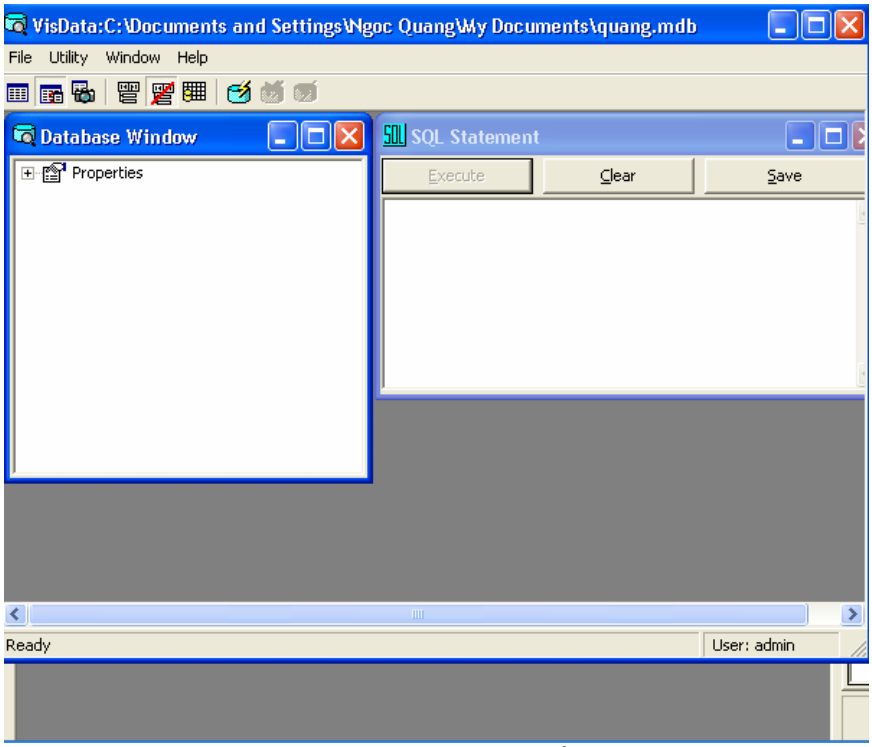
7.1.5.1.3 VẼ MỐI QUAN HỆ GIỮA HAI BẢNG

1. Trong thanh công cụ của Visio chọn công cụ Line
2. Chọn và kéo trường ID trong Tblcustomer và kéo đến trường CstomerID của Tbl Order. Nếu ta nhấn và kéo nhiều lần ta có thể tạo một đường gấp khúc

7.1.6 DÙNG MỘT VISUA BASIC ĐỂ TẠO MỘT CƠ SỞ DỮ LIỆU

Sau khi tạo song một lược đồ ta bắt đầu tạo cơ sở dữ liệu thực sự. Để tạo cơ sở dữ liệu Jet dùng VB ta có thể dùng tiện ích gọi là Visual Data Manage
Để có được Vissual Data Manager ta làm theo các bước sau

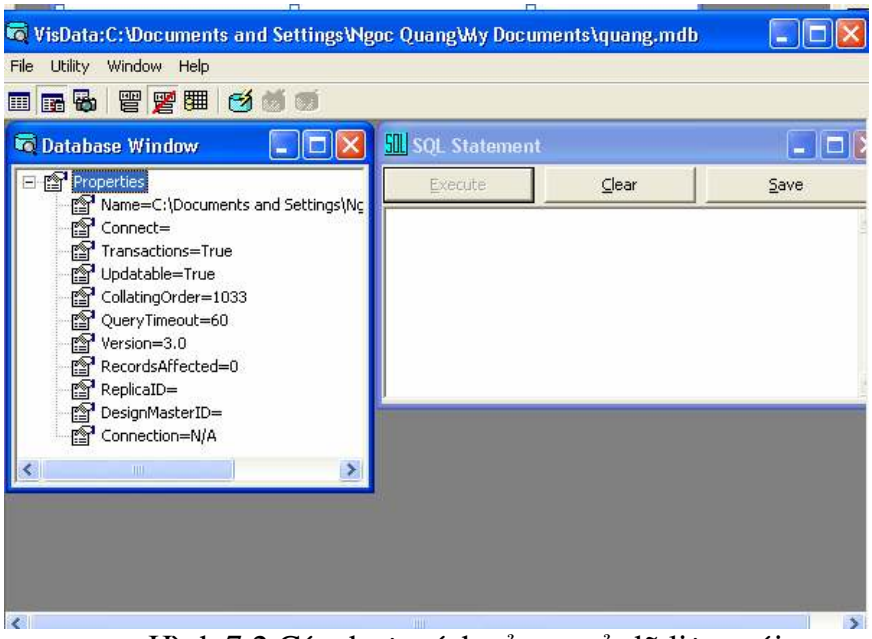
5. Từ thanh Menu của VB ta chọn mục Add-In, Visial Data Manager, cửa sổ Visual Data manager sẽ xuất hiện
6. Từ Menu của Visual Data Manager chọn File, New. Từ Menu con chọn Micrrosoft Access, Version 7.0.MDB. Một hộp thoại tin xuất hiện
7. Chọn thư mục ta muốn lưu cơ cở dữ liệu mới rồi gõ tên (Vì mục đích minh hoạ cho cuôn sách này, bạn có thể chọn cơ sở dữ liệu là novelty.mdb)
8. Nhấn nút Save. Cơ sở dữ liệu mới được tạo ra và Visual Data Manager sẽ hiển thị một vài cửa sổ cho phép ta làm việc với cơ sở dữ liệu được hiển thị như hình dưới đây



Hình 7. Cửa sổ Visual Data Manager

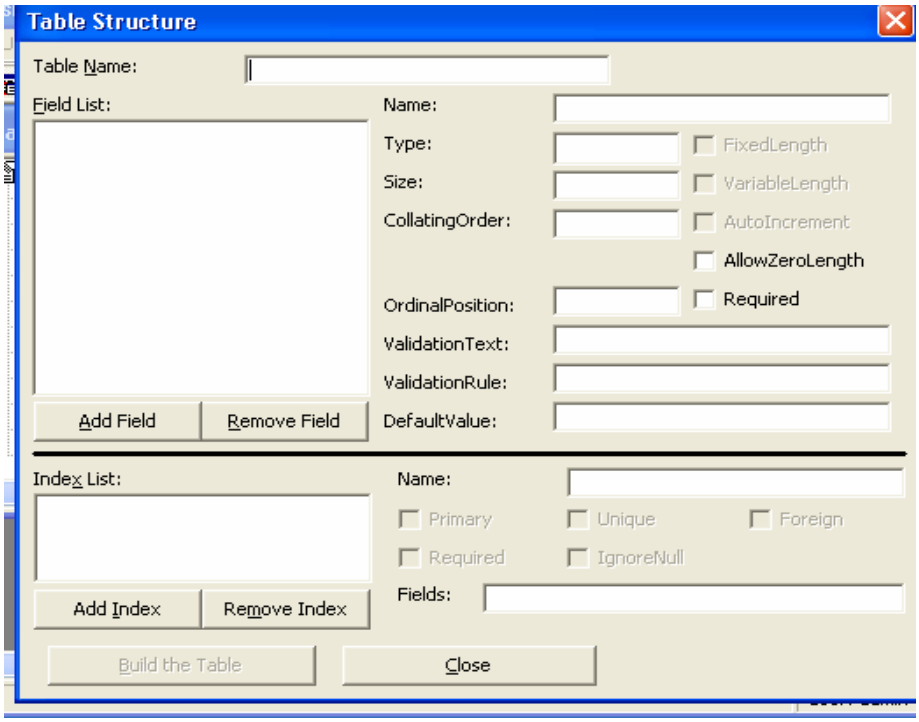
7.1.6.1 : SỬ DỤNG CỬA SỔ CƠ SỞ DỮ LIỆU

Cửa sổ Database của Visual Data Manager chứa tất cả các thành phần của cơ sở dữ liệu. Treong cửa sổ này ta có thể xem các thuộc tính kiểm tra các bảng và các phần tử khác và thêm các thành phần mới vào cơ sở dữ liệu
Để xem các thuộc tính của cơ sở dữ liệu ta vừa tạo nhấn chuột vào dấu cộng bên trái của mục Properties. Mục này sẽ mở ra như hình dưới đây



Hình 7.2 Các thuộc tính của cơ sở dữ liệu mới

7.1.6.2 TẠO BẢNG

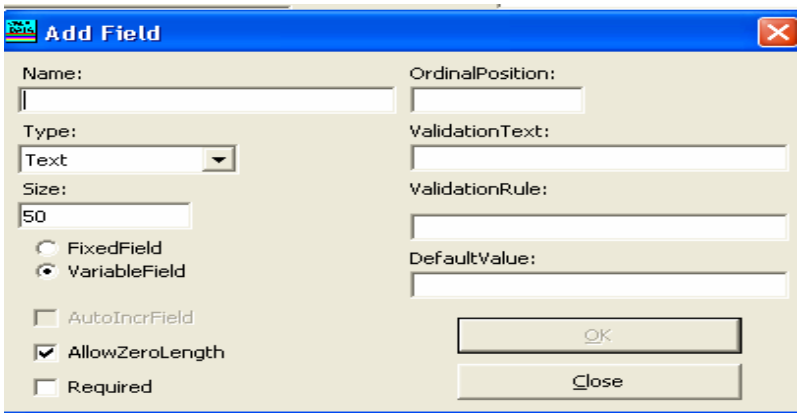


Một đặc tính của Visual Data Manager là nó không thể cho ta cách rõ ràng để tạo ra bảng mới trong cơ sở dữ liệu mà ta vừa tạo
Để tạo một bảng mới ta làm theo các bước sau

1. Trong cửa sổ Data Base của Visual Data Manager nhấn chuột phải vào Properties. Menu ngữ cảnh của cửa sổ sẽ xuất hiện
2. Chọn New Table. Hộp thoại Table Structure sẽ xuất hiện như hình dưới đây

Trong hộp thoại Table Structure, ta có thể tạo cấu trúc bảng chỉ định các trường, kiểu dữ liệu và chỉ mục

Trong hộp thoại Table ta nhấn chuột vào Add Field. Hộp thoại Add Field xuất hiện như hình sau

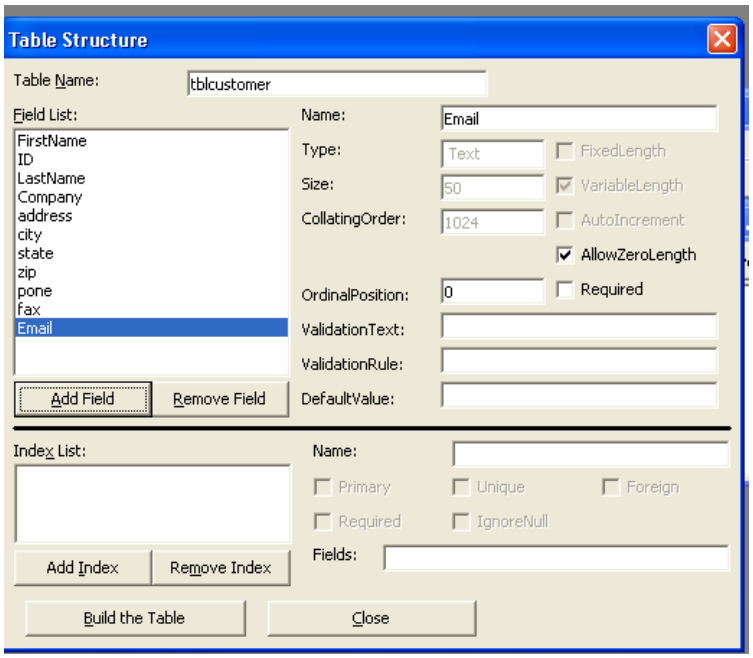


Hộp thoại Add Field cho phép ta thêm một trường vào một bảng tạo bởi hộp thoại Table Structure của Visual Data Manager .Tại hộp thoại có các ô chứa các lựa chon do người sử dụng nhập vào như sau :

1. Trong ô Name ta gõ FirstName. Đây tên trờng mà ta tạo trong bảng
2. trong ô SiZe ta gõ một số bất kỳ.Ví dụ: ta gõ 25 điều này chỉ ra răng ten có thể len đến 25 ký tự
3. Trong hai OptionButton. Nếu ta chọn FixedField sẽ có nghĩa rằng đây không phải là trờng có chiều dài biến đổi
4. Ta có thể thêm các trường khác nhau vào cấu trúc bảng. Sử dụng hộp thoại AddField thêm các trường như sau:

Name	DaTa Type	SiZe	Fixed
First Name	Text	25	Yes
ID	Long.AutoIncrField =True	N/A	N/A
LastName	Text	45	Yes
Company	Text	100	Yes
Address	Text	100	Yes
City	Text	100	Yes
State	Text	2	Yes
Zip	Text	9	Yes
Phone	Text	25	Yes
Fax	Text	25	Yes
Email	Text	255	Yes

5. Cần kiểm tra lại hộp AutolncrField khi tạo trường ID để bảo đảm mọi khách hàng ta tạo ra đều có số hiệu duy nhất
6. Khi hoàn tất việc nhập trường ta nhấn Close .Lúc đó sẽ xuất hiện hộp thoại Table Structure :



Hình 7.1.7.

7.1.6.3 THAY ĐỔI THUỘC TÍNH CỦA CÁC TRƯỜNG SẴN CÓ

Để thay đổi thuộc tính của trường trong Visual Data Manager ta phải thay đổi bằng cách xoá trường để làm lại

7.1.6.3.1: SỬA ĐỔI CHIỀU DÀI CỦA TRƯỜNG LASTNAME

Để thay đổi chiều dài của trường LastName ta phải làm các bước sau:
1. Trong cửa sổ DataBase của Visual Data Manager ta nhấn chuột phải lên lblCustomer
2. Từ Menu ngữ cảnh , chọn Desgin. Hộp thoại table Struncture xuất hiện

7.1.6.3.2: XOÁ CHỈ MỤC

Để xoá được LastName trước hết ta phải xoá chỉ mục của nó. Ta làm các bước sau
1. Chọn LastNameIndex trong danh sách các chỉ mục
2. Nhấn nút Remove index
3. khi xuất hiện thông báo hỏi ta có xoá thư mục này hay không ta nhấn Yes thì chỉ mục được xoá

7.1.6.3.3: XOÁ TRƯỜNG LASTNAME

1. Chọn trường LastName trong danh sách các trường

2. Nhấn Remove Field xuất hiện thông báo hỏi ta có muốn xoá hay không, ta nhấn Yes => giờ thì ta có thể sửa đổi trường bằng cách thêm nó trở lại bảng, lần này với chiều dài là 50

7.1.6.4: DÙNG VISUAL DATA MANAGER ĐỂ TẠO GIAO DIỆN

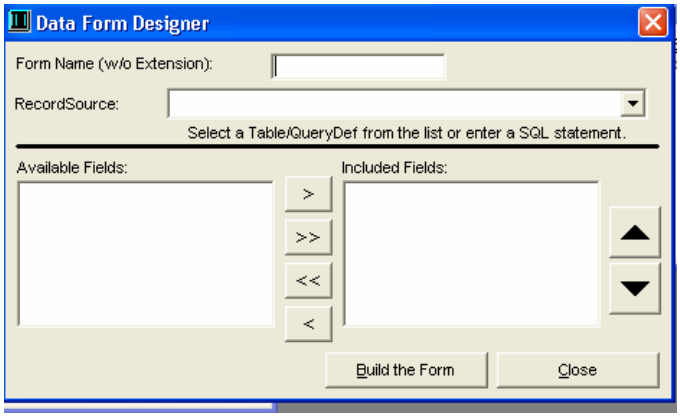
Một ưu điểm của Visual Data Manager là khả năng tạo biểu mẫu VB dựa trên cấu trúc dữ liệu được tạo

Nếu ta đã hoàn tất khâu thiết kế TblCustomer và ta muốn thêm một biểu mẫu Vb vào đề án dựa trên thiết kế .Ta làm theo các thiết kế sau

1. Từ Menu của Visual Data Manager, chon Utility, Data Form Design Hộp thoại Data Form Design xuất hiện

2. Trong hộp văn bản Form Name, gõ Cusstomer

3. Trong hộp kết hợp RecordSource, chọn TblCustomer ,Data Form Design điền danh sách các trường tìm thấy trong TblCustomer vào Available Fields



4. Nhấn nút mũi tên phải cho tất cả các trường hiện có trừ ID để thêm chúng vào biểu mẫu (không sửa được trường ID)

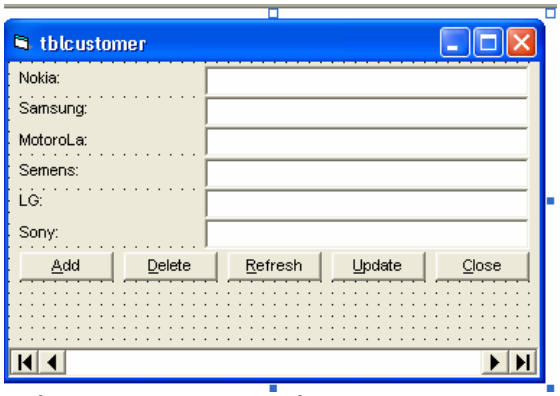
5. Chọn trường và sắp sếp chúng lại

Tiếp theo khi ta thoát ra khỏi Visual Data Manager để xem biểu mẫu của ta như thế nào .

Nhưng ta muốn quay lại để thêm các phần tử mới hoặc muốn sửa đổi ta làm như sau :

1. Chọn Utility, Preferences. Từ Menu con chọn Open Last Dataabase từ Startup

2. Để thoát khỏi Visual Data Manager ta chọn File, Exit. Khi trở về Visual basic ta sẽ thấy giao diên như sau với biểu mẫu có tên gọi Tblcustomer



Để làm việc với biểu mẫu mới chọn Menu Project, từ Menu này chọn Project1 Properties .Xuất hiện hộp thoại

1.Trong hộp kết hợp StartUp Object chọn FrmCustomer và nhấn nút OK

2.Từ Menu Run của Visual Basic chọn Start. ứng dụng thi hành nó sẽ hiển thị giao diện nhập liệu trong FrmCustomer

Bây giờ ta có thể nhập dữ liệu vào giao diện mà Visual Basic cung cấp cho ta. Để làm được điều này ta thực hiện các bước sau :

1. Nhấn nút Add. Ta sẽ thấy ứng dụng không có phản hồi một cách trực quan để cho thấy rằng có một điều gì đó thay đổi. Tuy nhiên cần phải chắc chắn ta đang thay đổi một dữ liệu mới
2. Nhập dữ liệu vào mỗi hộp văn bản trong biểu mẫu
3. Khi thực hiện xong nhấn nút Update. Dữ liệu sẽ được lưu trrữ. chỉ còn một thông tin phản hồi ta thấy là điều khiển dữ liệu hiển thị " Record 1"

Giao diện nhập liệu cơ bản được tạo bởi Data Form Designer cho ta ý nghĩa chương trình mà ta phải viết để có một ứng dụng mạnh mẽ bằng cách sử dụng điều khiển Data. Điều khiển Data đợc xem như là một giải pháp " không cần lập trình "

7.1.7 CHUẨN HOÁ

Chuẩn hoá làmột khái niệm liên quan đến mối quan hệ. Về cơ bản nguyên tắc của chuẩn hoá là các bảng cơ sở dữ liệu sẽ loại trừ tính không nhất quán và giảm thiểu sự kém hiệu quả

Các cơ sở dữ liệu được mô tả không nhất quán khi dữ liệu trong một bảng không tương ứng với các dữ liệu nhập vào trong bảng khác

Một cơ sở dữ liệu kém hiệu quả không cho phép ta chích ra các dữ liệu chính xác.Mặt khác cơ sở dữ liệu được chuẩn hoá đầy đủ chứa chứa từng mẩu thông tin của cơ sở dữ liệu tropng bảng riêng và xa hơn

Các cơ sở dữ liệu chuẩn hoá cho phép ta tham chiếu đến một mẩu thông tin trong một bảng bất kỳ chỉ bằng khoá chính của thông tin đó

Ta quyết định cách thức để chuẩn hoá một cơ sở dữ liệu khi ta thiết kế và khởi tạo một cơ sở dữ liệu. Thông thường mọi thứ về ứng dụng cơ sở dữ liệu- từ thiết kế bảng đến thiết kế

truy vấn, từ giao diện người sử dụng đến các hoạt động của báo cáo - đều xuất phát từ cách chuẩn hoá dữ liệu

7.1.7.1 QUAN HỆ MỘT - MỘT

Là một quan hệ dễ hiểu và dễ thực hiện bởi vì trong những mối quan hệ như vậy một bảng sẽ lấy vị trí của một trường trong một bảng khác. Tuy nhiên quan hệ Một – Một không phải là một mối quan hệ thông dụng trong ứng dụng của cơ sở dữ liệu. Vì 2 lý do:

- Hầu như ta không cần biểu diễn mối quan hệ Một – Một với 2 bảng. Ta có thể dùng nó để cải tiến khả năng hoạt động ,dù vaayj ta mất tính linh động khi chứa dữ liệu liên hệ trong một bảng tách biệt
- Thể hiện quan hệ Một – Nhiều thì linh hoạt hơn quan hệ Một – Một

7.1.7.2 QUAN HỆ MỘT – NHIỀU

Phổ biến hơn quan hệ Một – Một trong đó mỗi mẩu tin trong một bảng này không có hoặc có một hoặc nhiều mẩu tin trong bảng liên hệ

7.1.7.3 QUAN HỆ NHIỀU – NHIỀU

Quan hệ Nhiều – Nhiều là bước phát triển của quan hệ Một – Nhiều
Để thiết lập mối quan hệ Nhiều – Nhiều ta có thể sửa lại cho cơ sở dữ liệu có thể chứa nhiều dữ liệu

7.2 SỬ DỤNG CỬA SỔ XEM DỮ LIỆU

Điểm mới trong Visual basic là cửa sổ Data View cho phép ta làm việc với một cơ sở dữ liệu không cần phải sử dụng công cụ bên ngoài hay công cụ bổ xung Add-In

Để dùng cửa sổ Data View ta thực hiện các bước sau :

1. Từ Menu của VB chọn Data View
2. Cửa sổ Data View xuất hiện cho ta hai thư mục, Data Links và Data Environment Conections

Liên kết dữ liệu (Data links) là một cách kết nối môi trường phát triển của Visual basic với một cơ sở dữ liệu nào đó. Một kết nối môi trường dữ liệu (Data Environment Conections) là một cách sử dụng cơ sở dữ liệu cấy trong một đề án của VB. Điểm khác biệt của hai thành phần này là khi ta tạo một liên kết dữ liệu nó xuất hiện trong cửa sổ Data View mỗi khi cửa sổ hiển thị trong VB, ngay cả khi ta đóng đề án hiện hành và mở một đề án mới. Trái lại trình thiết kế môi trường dữ liệu gắn liền với đề án ta đang làm việc

Muốn dùng dùng một liên kết dữ liệu để duyệt dữ liệu ta thực hiện như sau;

1. Trong cửa sổ Data View nhấn chuột phảI lên thư mục Data Links. Từ Menu mới ta chọn Add a Data Links
2. cửa sổ data links properties xuất hiện
3. Chọn trình cung cấp Microsoft Jet, rồi nhấn Next.
4. Tab Connection xuất hiện. Nhập đường dẫn và tên tập tin cơ sở dữ liệu ta muốn dùng.

5. Nhập nút Test Connection tại nphaanf dưới của cửa sổ. Ta sẽ có một thông điệp thông báo kết nối đến cơ sở dữ liệu thành công. kiểm tra kết nối không có gì ghê gớm với cơ sở dữ liệu Access. Nhưng với các cơ sở dữ liệu Client / Server, nó có thể olaf một giảI pháp cấp cứu.
6. Nhấn ok liên kết ữ liệu được thiết lập, và cửa sổ Data View nhắc ta nhập vào tên của liên kết. Gõ vào Novelty, rồi nhấp Enter.

Liên kết dữ liệu cung cấp một cách nhìn tóm lược về nguồn dữ liệu . Mỗi lần ta tạo một liên kết dữ liệu., ta có thể mduyeetj bằng cách mở rộng phần từ trong danh sách tóm lược.

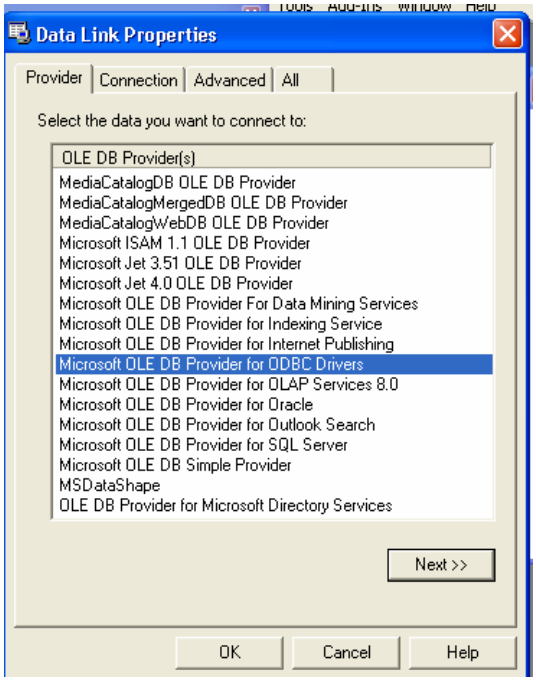
Tuỳ theo công cụ ta dùng để tạo cơ sở dữ liệu, ta có thể thấy thêm một số bảng trong danh sách

Giờ đây ta có thể xem các dữ liệu sống động. Thực hiện điều này bằng cách nhấn chuột lên bảng TblCustomer trong cửa sổ Data View.

Cách thể hiện khởi đầu của dữ liệu thì không đặc sắc lắm, vì ta chưa nhập mẩu tin nào cả.Tuy nhiên ta có thể nhập mẩu mtin bằng cách gõ trên lưới

Đến đây hoặc ta có thể tiếp tục với cơ sở dữ liệu đã tạo từ trước hoặc bắt đầu sử dụng cơ sở dữ liệu có sẵn :

1. Chọn trình cung cấp MicroSoft Jet rồi nhấn Next
2. Tab Connection xuất hiện. Nhập đường dãn và tên tập tin cơ sở dữ liệu ta muốn dùng
3. Nhấn nút Test Connection tại phần dưới của cửa sổ. Ta sẽ có một thông điệp khác thông báo kết nối thành công
4. Nhấn OK. Liên kết dữ liệu đợc thiết lập và cửa sổ Data View nhắc ta nhập vào tên của liên kết . Gõ tên vào Novelty, rồi nhấn Enter
5. Liên kết dữ liệu cung cấp một cách tóm lược về nguồn dữ liệu. Mỗi lần ta tạo một liên kết dữ liệu ta có thể duyệt bằng cách mở rộng phần tử trong danh sách tóm lược



Liên kết dữ liệu cung cấp một cách nhìn tóm lược về nguồn dữ liệu. Mỗi lần ta toạ một liên kết dữ liệu , ta có thể duyệt bằng cách mở rộng phần tử trong danh sách tóm lược bằng cách nhán chuột trái vào mỗi phần tử

Tuỳ theo công cụ ta dùng để tạo cơ sở dữ liệu, ta có thể thấy thêm một số bảng trong danh sách

Giơ đây ta có thể xem các dữ liệu sống động. Thực hiện điều này bằng cách nhấn chuột lên bảng TblCustomer

Cách thể hiện khởi đầu của cơ sở dữ liệu thì không đặc sắc lắm, vì ta chưa nhập mẩu tin nào vào cả. Tuy nhiên ta có thể nhập mẩu tin bằng cách gõ vào trên lưới.

Đến đây hoặc ta có thể tiếp tục với cơ sở dữ liệu đã tạo từ trước hoặc bắt đầu sử dụng cơ sở dữ liệu có sẵn. Ngoài ra cơ sở dữ liệu còn được sử dụng trong tất cả các ví dụ tiếp theo.

7.3 TẠO TRÌNH THIẾT KẾ MÔI TRƯỜNG DỮ LIỆU

Ta có thể tạo một thiết kế DataEnterVironment để quản lý một cách trực quan kết nối với cơ sở dữ liệu. Khi ta có một thiết kế DataEnterVironment trong ứng dụng ta có thể quản lý tất cả thông tin gắn liền với kết nối opwr một nơi cơ sở dữ liệu trong chương trình.

Chương này trình bày cách dùng thiết kế DataEnvironment. để tạo một giao diện người ta sử dụng được điều khiển với cơ sở dữ liệu tuy nhiên có nhiều cách để thực hiện. để có thêm thông tin về cách dùng thiết kế xem chương 27

Để thêm một thiết kế DataEnvironment vào ứng dụng cửa sổ DataView, theo các bước.

1. Trong cửa sổ Data View, nhấp nút Add DataEnvironment
2. Thiết kế DataaEnvironment mới sẽ xuất hiện trong đề án. một kết nối mặc định , gọi là connection1, xuất hiện trong thiết kế.

Có thể điều chỉnh một cách thủ công kết nối mặc định trong một DataEnvironment để nó trỏ đến cơ sở dữ liệu. Nhưng nếu cơ sở dữ liệu đã có sẵn trong cửa sổ trong Dataview ta chỉ cần kéo và thả bảng thiết kế.

1. Khởi động cửa sổ Dataview , chọn một bảng trong thư mục tables.
2. Kéo bảng lên trên thiết kế DataEnvironment .
3. Một thiết kế mới gọi là Connection2 xuất hiện trên bảng thiết kế, với bảng xuất hiện bên dưới.

Đến đây ta có thể kéo các bảng khác vào thiết kế nếu thích . Khi hoàn tất ta có.

7.3.1:TẠO MỘT GIAO DIỆN NGƯỜI SỬ DỤNG VỚI THIẾT KẾ DATAENVIRONMENT

Ta có thể tạo một giao diện người sử dụng nhanh chóng bằng cách dùng thiết kế Dataenvironment . Thiết kế kết hợp với cơ chế biểu maayux của VB cho phép ta dùng kỹ thuật kéo để thả và kéo để tạo giao diện người sử dụng. Để thực hioện điều này ta làm theo các bước sau.

1. Mở biểu mẫu ta muốn dùng làm giao diện người sử dụng .
2. Chọn bảng thiết kế DataEnvironment .(không phảI trong cửa sổ DataView).

3. Thả bảng vào biểu mẫu.

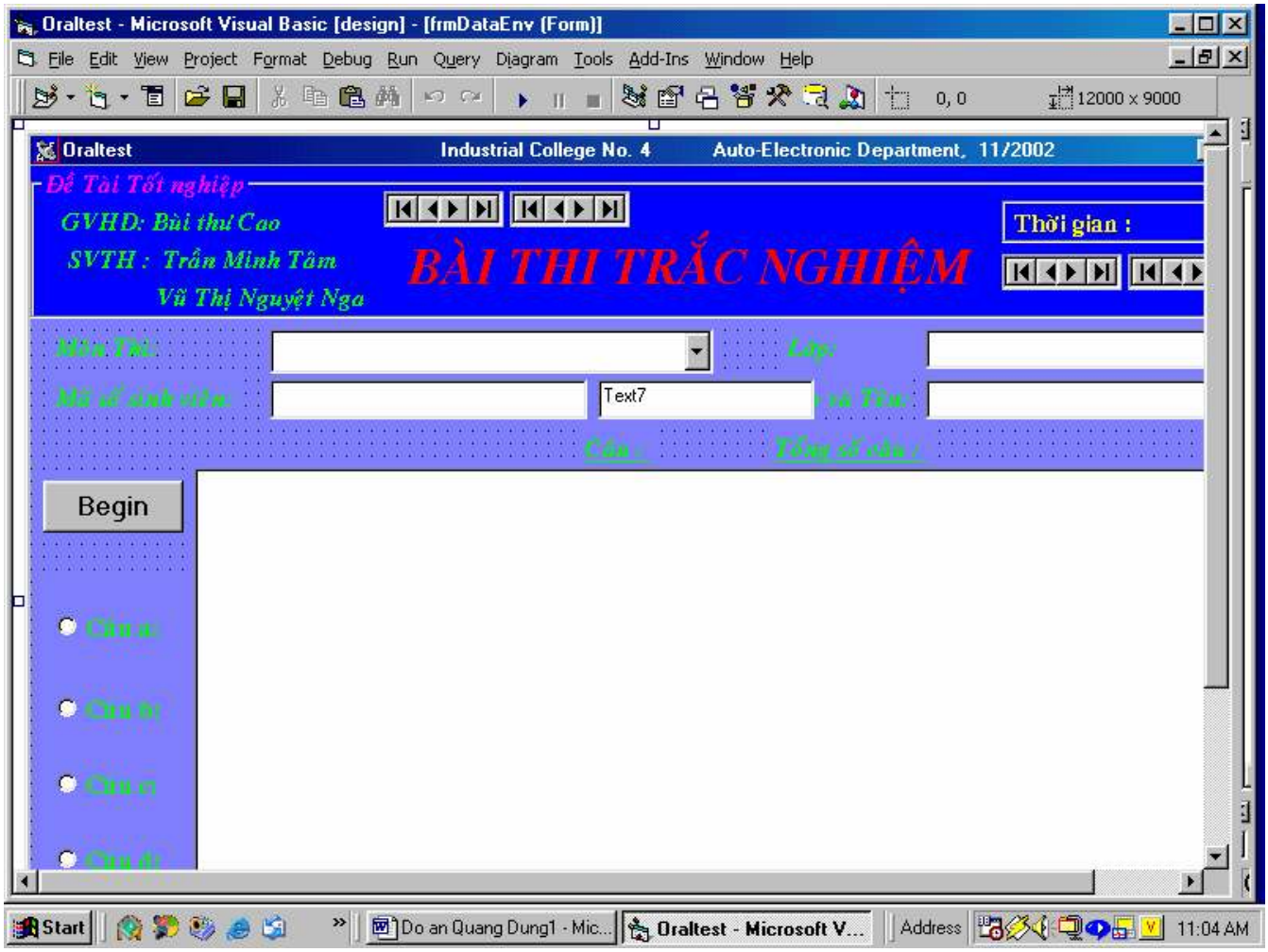
Một giao diện người sử dụng ràng buộc dữ liệu sẽ được tạo trên biểu mẫu.
Thi hành ứng dụng để xem mẫu tin thứ nhất trong cơ sở dữ liệu tuy nhiên, không có chức năng duyệt từ mẩu tin này sang mẩu tin khác để thực hiện điều đó ta phải lập trình hoặc dùng một điều khiển dữ liệu, mổ ta trong phần sau.

7.4 SỬ DỤNG ĐIỀU KHIỂN DỮ LIỆU ĐỂ TẠO GIAO DIỆN NGƯỜI SỬ DỤNG

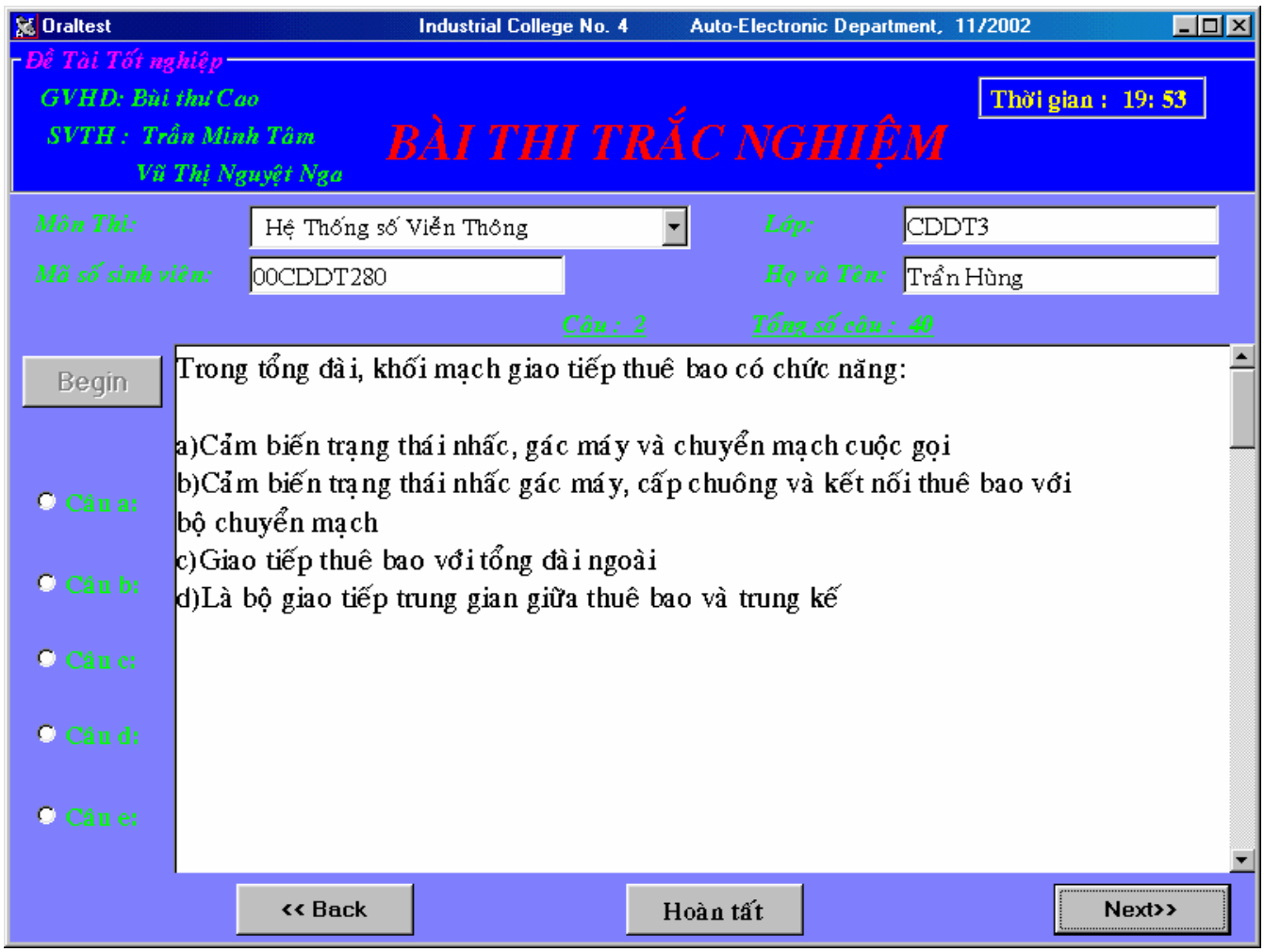
Ta có thể dùng một điều khiển dữ liệu để quản lý kết nối giữa biểu mẫu VisualBasic và một cơ sở dữ liệu. điều khiển dữ liệu còn cung cấp chức năng duyệt dữ liệu đơn giản, cho phép dữ liệu duyệt qua một recordset, thêm và cập nhật mẩu tin. Phiên bản trước của VB cung cấp 2 loại điều khiển dữ liệu, DAO Data, thường được dùng để mkeets nối để nối với cơ sở dữ liệu trên máy tính cá nhân như Microsoft Access, và điều khiển Remote Data (RDC), dùng cho dữ liệu Client/ server. VB6 thêm một Remote Data (RDC) điều khiển dữ liệu mới. ADO Data , cho pháp ta truy cập mọi dữ liệu , bao gồm nguồn dữ liệu trên máy tính cá nhân, trên Client / Server và không được mô hình quan hệ.
Vậy ta nên dùng điều khiển dữ liệu nào? Đối với các ứng dụng cơ sở dữ liệu mới trong VB6 ta có thể dùng ADO Data trình bày . Vì các điều khiển dữ liệu hoạt động tương tự nhau, ta chỉ trình bày dữ liệu ADO
Do hạn chế này của điều khiển Data trrong ấn bản Leaning để sử dụng điều khiển với đầy đủ chức năng ta giả sử rằng điều khiển Data ở đây là bản giành cho Professional và Enterprise

VÍ DỤ MINH HOẠ ỨNG DỤNG LẬP TRÌNH CƠ SỞ DỮ LIỆU

Chương trình thi trắc nghiệm trên máy con (ứng dụng Client). Chương trình này được viết bởi các sinh viên thực hiện đồ án tốt nghiệp



Hình minh hoạ ở chế độ **Program time**



Hình minh họa ở chế độ **Run Time**

Mã nguồn VB:

```
Dim phut, giay, socau As Integer
Dim db As Adodc
Dim datapath As String

Private Sub Combo1_Click()
Dim boctham, i, size As Integer
Dim path, dapan As String
   With Data1.Recordset
      If Data1.Recordset.RecordCount > 0 Then
         .MoveFirst
         Do While .EOF = False
```

```
            .Delete
            .MoveNext
        Loop
    End If
End With
With Data2.Recordset
    .MoveFirst
    .FindFirst "tenmonhoc = '" & Combo1.Text & "'"
    If .NoMatch = False Then
        socau = .Fields(2)
        phut = .Fields(3)
        Text7.Text = .Fields(1)
        dapanpath = .Fields(5) & "\dapan.mdb "
        datapath = .Fields(5)
    Else
        MsgBox ("Error !, Mamonhoc notfound ")
    End If
End With
Data3.DatabaseName = dapanpath
Data3.Refresh
Data3.Recordset.MoveFirst
With Data3.Recordset
    .MoveLast
    size = .RecordCount
End With
giay = 0
For i = 1 To Second(Time)
    boctham = Int(size * Rnd) + 1
Next
i = 1
Do While i <= socau
    boctham = Int(size * Rnd) + 1
    With Data3.Recordset
        .MoveFirst
        .FindFirst "stt =" & boctham
        path = .Fields(2)
        dapan = .Fields(1)
```

```
        End With
        Data1.Refresh
        With Data1.Recordset
           If .RecordCount > 0 Then
              .MoveFirst
              .FindFirst "caugoc =" & boctham
              If .NoMatch = True Then
                 .AddNew
                 .Fields(0) = i
                 .Fields(1) = boctham
                 .Fields(2) = path
                 .Fields(4) = dapan
                 .Update
                 i = i + 1
              End If
           Else
              .AddNew
              .Fields(0) = i
              .Fields(1) = boctham
              .Fields(2) = path
              .Fields(4) = dapan
              .Update
              i = i + 1
           End If
        End With
    Loop
    Label6.Caption = "Tổng số câu : " + Str(socau)
    Label2.Caption = Str(phut)
End Sub

Private Sub Command1_Click()
Dim paths As String
    Data1.Refresh
    'ChDir (datapath)
    paths = datapath & "\" & Data1.Recordset.Fields(2)
    OLE1.CreateLink (paths)
    Command2.Enabled = True
```

```
    Command3.Enabled = True
    Command1.Enabled = False
    Command4.Enabled = True
    OLE1.Enabled = True
    Timer1.Enabled = True
    Label5.Caption = "Câu : " + Str(Data1.Recordset.AbsolutePosition + 1)
End Sub

Private Sub Command2_Click()
Dim paths As String
    Data1.Recordset.Edit
    If Option1.Value = True Then
       Data1.Recordset.Fields(3) = "a"
    End If
    If Option2.Value = True Then
       Data1.Recordset.Fields(3) = "b"
    End If
    If Option3.Value = True Then
       Data1.Recordset.Fields(3) = "c"
    End If
    If Option4.Value = True Then
       Data1.Recordset.Fields(3) = "d"
    End If
    If Option5.Value = True Then
       Data1.Recordset.Fields(3) = "e"
    End If
    Option1.Value = False
    Option2.Value = False
    Option3.Value = False
    Option4.Value = False
    Option5.Value = False
    Data1.Recordset.Update
    Data1.Recordset.MovePrevious
    If Data1.Recordset.BOF Then
       Data1.Recordset.MoveFirst
    Else
       paths = datapath & "\" & Data1.Recordset.Fields(2)
```

```
        OLE1.Delete
        OLE1.CreateLink (paths)
    End If
    Select Case Data1.Recordset.Fields(3)
          Case "a": Option1.Value = True
          Case "b": Option2.Value = True
          Case "c": Option3.Value = True
          Case "d": Option4.Value = True
          Case "e": Option5.Value = True
    End Select
    Label5.Caption = "Câu : " + Str(Data1.Recordset.AbsolutePosition + 1)
End Sub
Private Sub Command3_Click()
Dim paths As String
    Data1.Recordset.Edit
    If Option1.Value = True Then
      Data1.Recordset.Fields(3) = "a"
    End If
    If Option2.Value = True Then
      Data1.Recordset.Fields(3) = "b"
    End If
    If Option3.Value = True Then
      Data1.Recordset.Fields(3) = "c"
    End If
    If Option4.Value = True Then
      Data1.Recordset.Fields(3) = "d"
    End If
    If Option5.Value = True Then
      Data1.Recordset.Fields(3) = "e"
    End If
    Option1.Value = False
    Option2.Value = False
    Option3.Value = False
    Option4.Value = False
    Option5.Value = False
    Data1.Recordset.Update
    Data1.Recordset.MoveNext
```

```
    If Data1.Recordset.EOF Then
       Data1.Recordset.MoveLast
    Else
       paths = datapath & "\" & Data1.Recordset.Fields(2)
       OLE1.CreateLink (paths)
    End If
    Select Case Data1.Recordset.Fields(3)
          Case "a": Option1.Value = True
          Case "b": Option2.Value = True
          Case "c": Option3.Value = True
          Case "d": Option4.Value = True
          Case "e": Option5.Value = True
       End Select
    Label5.Caption = "Câu : " + Str(Data1.Recordset.AbsolutePosition + 1)
End Sub
Private Sub Command4_Click()
Dim cau As Integer
Dim da As String
'------update dapan----
    Data1.Recordset.Edit
    If Option1.Value = True Then
      Data1.Recordset.Fields(3) = "a"
    End If
    If Option2.Value = True Then
      Data1.Recordset.Fields(3) = "b"
    End If
    If Option3.Value = True Then
      Data1.Recordset.Fields(3) = "c"
    End If
    If Option4.Value = True Then
      Data1.Recordset.Fields(3) = "d"
    End If
    If Option5.Value = True Then
      Data1.Recordset.Fields(3) = "e"
    End If
    Option1.Value = False
    Option2.Value = False
```

```
    Option3.Value = False
    Option4.Value = False
    Option5.Value = False
    Data1.Recordset.Update
'----------------------
    Me.Hide
    diem = 0
    With Data1.Recordset
       .MoveFirst
       Do While .EOF = False
          cau = .Fields(1)
          Data3.Recordset.MoveFirst
          Data3.Recordset.FindFirst "stt =" & cau
          If Data3.Recordset.NoMatch = False Then
             da = Data3.Recordset.Fields(1)
             .Edit
             .Fields(4) =  Asc(da)
             .Update
          Else
             MsgBox ("Error 101, Contact to provider ")
          End If
          If .Fields(3) = .Fields(4) Then
             diem = diem + 1
          End If
          .MoveNext
       Loop
    End With
    diem = diem * 10 / socau
    If diem > 1 Then
      MsgBox ("Diem so : " & Format(diem, "#,#"))
    Else
      MsgBox ("Diem so : 0" & Str(diem))
    End If
    Data2.Refresh
    With Data2.Recordset
       .FindFirst "tenmonhoc = '" & Combo1.Text & "'"
       If .NoMatch = False Then
```

```
            mamon = .Fields(1)
        Else
            MsgBox ("MaMonhoc not found !")
        End If
    End With
    With Data5.Recordset
        .AddNew
        .Fields(0) = Text2.Text
        .Fields(1) = mamon
        .Fields(2) = diem
        .Fields(3) = Text3.Text
        .Fields(4) = Text1.Text
        .Update
    End With
    Unload Me
End Sub

Private Sub Command5_Click()

End Sub

Private Sub Form_Activate()
    Option1.Value = False
    Option2.Value = False
    Option3.Value = False
    Option4.Value = False
    Option5.Value = False
    Data2.DatabaseName = Data4.Recordset.Fields(0)
    Data5.DatabaseName = Data4.Recordset.Fields(0)
    Data6.DatabaseName = Data4.Recordset.Fields(0)
    Data5.Refresh
    Data6.Refresh
    Data2.Refresh
    With Data2.Recordset
        .MoveFirst
        Do While .EOF = False
            Me.Combo1.AddItem (.Fields(0))
```

```
            .MoveNext
        Loop
    End With
    Combo1.ListIndex = -1
    VScroll1.SmallChange = 5
    VScroll1.LargeChange = 20
    VScroll1.Max = 100
    VScroll1.Min = 0
End Sub

Private Sub OLE1_DblClick()
    If Frame4.Top = 0 Then
        Frame4.Top = 2880
        Frame4.Height = 5055
        VScroll1.Height = Frame4.Height
        VScroll1.Top = Frame4.Top
    Else
        Frame4.Top = 0
        Frame4.Height = 7935
        VScroll1.Height = Frame4.Height
        VScroll1.Top = Frame4.Top
    End If
End Sub

Private Sub Text2_KeyDown(KeyCode As Integer, Shift As Integer)
If KeyCode = 13 Then
    Text2.Text = UCase(Text2.Text)
    With Data6.Recordset
        .MoveFirst
        .FindFirst "masosv = '" & Text2.Text & "'"
        If .NoMatch = False Then
            Command1.Enabled = True
            Text3.Text = .Fields(0)
            Text1.Text = .Fields(2)
        Else
            MsgBox ("Nhap lai ma so sinh vien ! ")
        End If
```

```
    End With
End If
End Sub
Private Sub Timer1_Timer()
Dim diem, cau As Integer
Dim da As String
   If giay = 0 Then
      giay = 60
      phut = phut - 1
      If phut = 0 Then
         Me.Hide
         diem = 0
         With Data1.Recordset
            .MoveFirst
            Do While .EOF = False
               cau = .Fields(1)
               Data3.Recordset.MoveFirst
               Data3.Recordset.FindFirst "stt =" & cau
               If Data3.Recordset.NoMatch = False Then
                  da = Data3.Recordset.Fields(1)
                  .Edit
                  .Fields(4) = Asc(da)
                  .Update
               Else
                  MsgBox ("Error 101, Contact to provider ")
               End If
               If .Fields(3) = .Fields(4) Then
                  diem = diem + 1
               End If
               .MoveNext
            Loop
         End With
         diem = diem * 10 / socau
         If diem > 1 Then
            MsgBox ("Diem so : " & Format(diem, "#,#"))
         Else
            MsgBox ("Diem so : 0" & Str(diem))
```

```
            End If
            Data2.Refresh
            With Data2.Recordset
               .FindFirst "tenmonhoc = '" & Combo1.Text & "'"
               If .NoMatch = False Then
                  mamon = .Fields(1)
               Else
                  MsgBox ("MaMonhoc not found !")
               End If
            End With
            With Data5.Recordset
               .AddNew
               .Fields(0) = Text2.Text
               .Fields(1) = mamon
               .Fields(2) = diem
               .Fields(3) = Text3.Text
               .Fields(4) = Text1.Text
               .Update
            End With
            Unload Me
         End If
      Else
         giay = giay - 1
      End If
      Label2.Caption = Str(phut) & ":" & Str(giay)
End Sub

Private Sub VScroll1_Change()
  OLE1.Top = (-VScroll1.Value / 100) * (7000 - 4000)
End Sub
```

TÀI LIỆU THAM KHẢO

1. Microsoft Visual Basic 6.0 – Programmer's Guide
   Published by Microsoft' Press
2. Microsoft Visual Basic 6.0 – Lập trình cơ sở dữ liệu
   TG: GSTS Nguyễn Hữu Anh