HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN I

BÀI GIẢNG

TIN HỌC QUẢN LÝ

NGƯỜI VIẾT: Ths. TRỊNH THỊ VÂN ANH

HÀ NỘI - 2013

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN VỀ BÀI TOÁN QUẢN LÝ VÀ XÂY DỰNG CSDL CHO BÀI TOÁN QUẢN LÝ

1.1. Tổng quan về bài toán quản lý

Công nghệ thông tin có thể hỗ trợ doanh nghiệp cải thiện hiệu quả và hiệu suất của các qui trình nghiệp vụ kinh doanh, quản trị ra quyết định, cộng tác nhóm làm việc, qua đó tăng cường vị thế cạnh tranh của doanh nghiệp trong một môi trường thay đổi nhanh. Tin học hóa công tác quản lý của các đơn vị kinh tế, hành chính...(tin học quản lý) đang là lĩnh vực quan trọng nhất của ứng dụng tin học. Xây dựng và phát triển hệ thống thông tin kinh tế và quản lý hiện đại là nội dung chủ yếu của ứng dụng tin học trong việc tự động hóa từng phần hoặc toàn bộ các quy trình nghiệp vụ, quản lý trong các tổ chức kinh tế.

1.1.1. Tập các bài toán thực tế

Hệ thống là một tập hợp gồm nhiều phần tử tương tác, có các mối quan hệ ràng buộc lẫn nhau và cùng hoạt động hướng tới một mục tiêu chung thông qua chấp thuận các đầu vào, biến đổi có tổ chức để tạo kết quả đầu ra.

Ví dụ:

- Hệ thống điều khiển giao thông
- Hệ thống mạng máy tính
- ...

Các thành phần của hệ thống:

- Nhập vào (Input): Nắm bắt và tập hợp các yếu tố để đưa vào hệthống để xử lý
- Xử lý (Processing): Bước biến đổi nhằm chuyển các yếu tố đưa vào sang các dạng cần thiết
- Kết xuất (Output): Chuyển các yếu tố được tạo ra từ quá trình xử lý thành các kết quả cuối cùng

Hệ thống thông tin: là một tập hợp các phần cứng, phần mềm, hệ mạng truyền thông được xây dựng và sử dụng để thu thập, tạo, tái tạo, phân phối và chia sẻ dữ liệu, thông tin và tri thức nhằm phục vụ các mục tiêu của tổ chức.

Hệ thống thông tin quản lý: Một hệ thống tích hợp "Người - Máy" tạo ra các thông tin giúp con người trong sản xuất, quản lý và ra quyết định là hệ thông tin quản lý. Hệ thông tin quản lý sử dụng các thiết bị tin học, các phần mềm, CSDL, các thủ tục thủ công, các mô hình để phân tích, lập kế hoạch quản lý và ra quyết định.

Những tác động của CNTT tới một trong các ngành sau:

+ Dịch vụ tài chính
+ Chăm sóc sức khỏe
+ Sản xuất
+ Dịch vụ giải trí nghe nhìn
+ Giáo dục
+ Bán lẻ
+ Du lịch và khách sạn

Một số công ty ứng dụng CNTT thành công trên thế giới:

+ Boeing Airplane Company

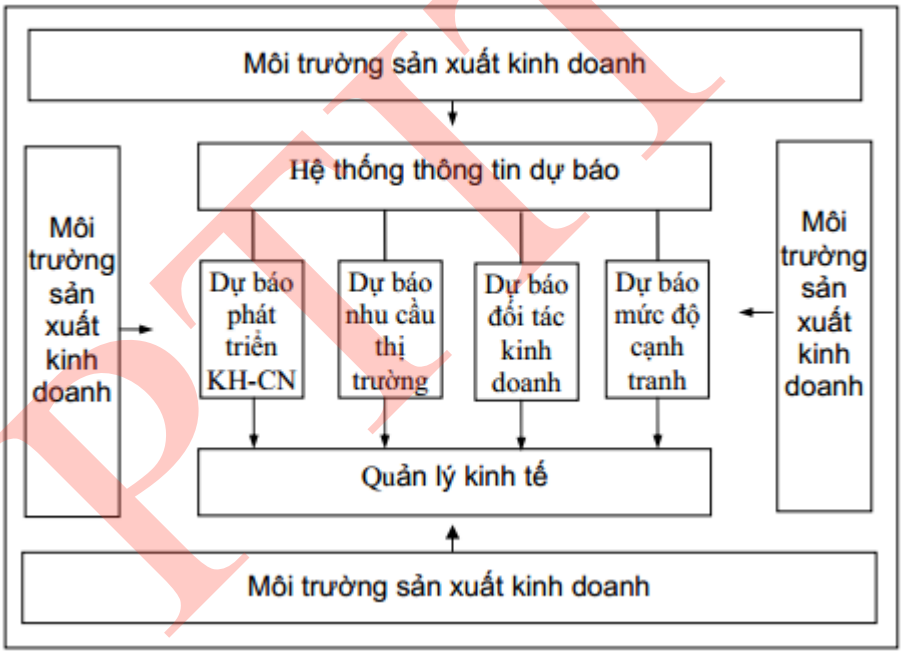
+ Wal-Mart Stores
+ Bissett Nursery Corp.
+ Federal Express
+ Charles Schwab
+ USAA
+ L.L. Bean
+ Progressive Corp.

1.1.2. Sơ đồ khối của bài toán cụ thể

Để định hướng ra các hệ thống thông tin quản lý người ta dựa vào định hướng hoạt động của hệ thống thông tin và tổng thể các bài toán quản lý mà hệ thống giải quyết. Theo cách này thì có thể chia các hệthống thông tin thành một số dạng sau:

- Hệ thống thông tin dự báo
- Hệ thống thông tin khoa học
- Hệ thống thông tin kế hoạch
- Hệ thống thông tin thực hiện

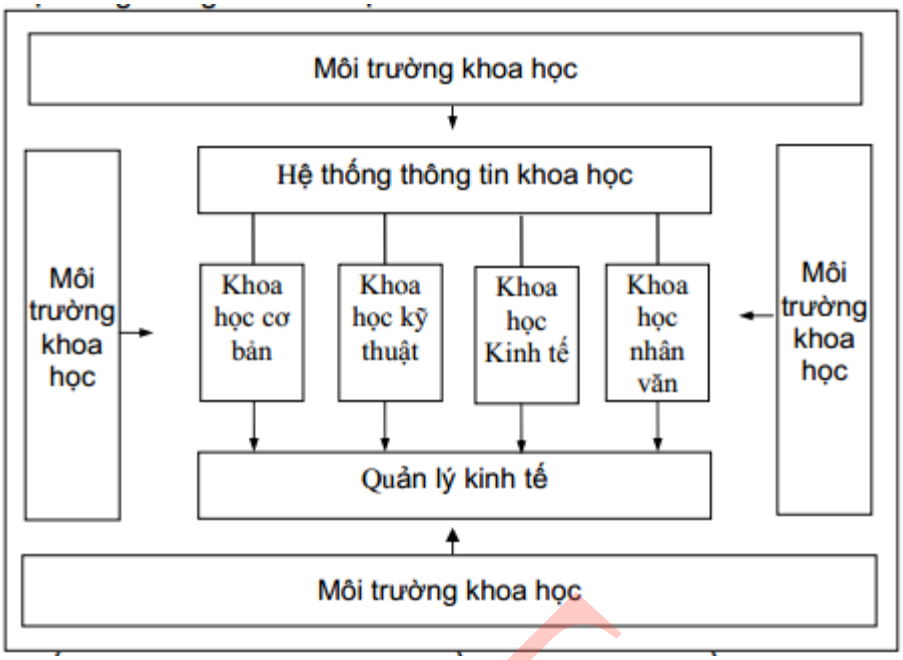
Hệ thống thông tin dự báo:



Hình 1.1: Mô hình hệ thống thông tin dự báo

Hệ thống thông tin dự báo bao gồm DB dài hạn, DB trung hạn và DB ngắn hạn về các vấn đề liên quan đến doanh nghiệp, như DB các tiến độ KH-CN, dự báo qui mô sản xuất, dự báo về nhu cầu của thị trường, về mức độ cạnh tranh trên thị trường,… Hệ thống thông tin dự báo càng quan trọng trong hoạt động kinh tế thị trường.

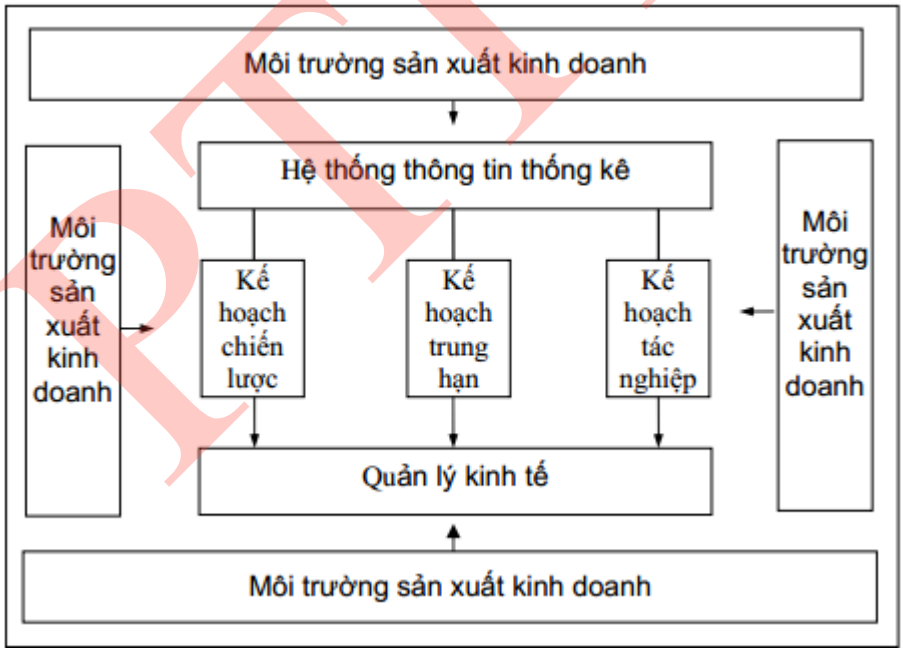
Hệ thống thông tin khoa học:



Hình 1.2: Mô hình hệ thống thông tin khoa học

Hệ thống thông tin khoa học bao gồm các thông tin về KHCB, KHCN, KHKT và KHTN. Từ môi trường KH rộng lớn hệ thống thông tin khoa học thu thập các thông tin liên quan đến sản xuất - kinh doanh để đáp ứng yêu cầu quản lý của doanh nghiệp.

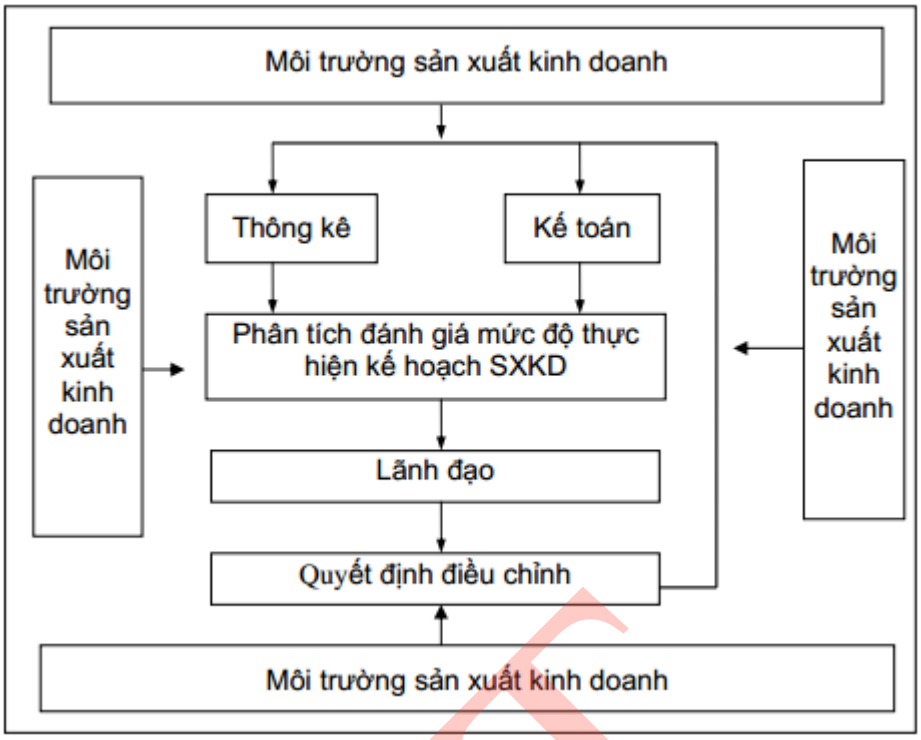
Hệ thống thông tin kế hoạch:



Hình 1.3: Mô hình hệ thống thông tin kế hoạch

Hệ thống thông tin kế hoạch bao gồm toàn bộ các thông tin về công tác kế hoạch hoá của doanh nghiệp. Các kế hoạch được đề cập đến ở 3 mức độ: kế hoạch hóa chiến lược, kế hoạch hóa trung hạn, kế hoạch hóa cơ động. HTTTKH bao quát tất cả các lĩnh vực hoạt động của doanh nghiệp gồm cả trong lĩnh vực sản xuất và quản lý.

Hệ thống thông tin thực hiện:



Hình 1.4: Mô hình hệ thống thông tin thực hiện

Hệ thống thông tin thực hiện sử dụng các công cụ thống kê và kế toán để kiểm tra, đánh giá, phân tích các quá trình thực hiện kế hoạch theo thời gian. Trên cơ sở các số liệu của HTTT thực hiện, mà cán bộ lãnh đạo có thể ra các quyết định điều chỉnh.

Như vậy, phương pháp phân loại thông tin theo nội dung giúp chúng ta định hướng rõ mục đích của các dòng thông tin trong hệ thống quản lý. Trên cơ sở đó tiến hành phát triển hoàn thiện một nội dung nào đó trong hệ thống quản lý.

1.1.3. Phân tích và thiết kế bài toán cụ thể

Giả sử với bài toán xây dựng hệ thống ứng dụng cẩm nang hỗ trợ người dùng trong ẩm thực và nấu ăn.

Đặc tả chức năng:

Chức năng tìm kiếm món ăn: Cho phép tìm kiếm món ăn theo ba miền bắc, trung, nam. Mỗi miền gồm mười hai loại món ăn khác nhau. Với mỗi loại món ăn cho phép người dùng xem một danh sách món ăn đặc sắc thuộc loại đó. Cho phép người dùng xem chi tiết về món ăn: ảnh món ăn, các loại gia vị cùng số lượng (nếu có), tác dụng gia vị, cách chế biến món ăn. Cho phép người dùng đánh dấu món ăn vào danh sách món ăn yêu thích.

Chức năng xem video nấu ăn: Cho phép người dùng xem danh sách các video nấu ăn của các đầu bếp nổi tiếng. Người dùng chọn một video về món ăn của đầu bếp mà người dùng thích để xem hướng dẫn chế biến.

Chức năng tim kiếm nhà hàng: Cho phép người dùng tim kiếm nhà hàng theo từng khu vực Huế, Hà Nội, Sài Gòn. Với mỗi khu vực hiển thị danh sách các nhà hàng thuộc khu vực đó. Cho phép người dùng xem thông tin chi tiết về nhà hàng: ảnh nhà hàng, địa chỉ, điện thoại liên lạc, giờ mở cửa đóng cửa, website.

Chức năng xem quảng cáo nhà hàng: Cho phép người dùng xem thông tin về các chương trình khuyến mại của các nhà hàng, người dùng liên hệ với nhà hàng thông qua số điện thoại để biết thêm thông tin.

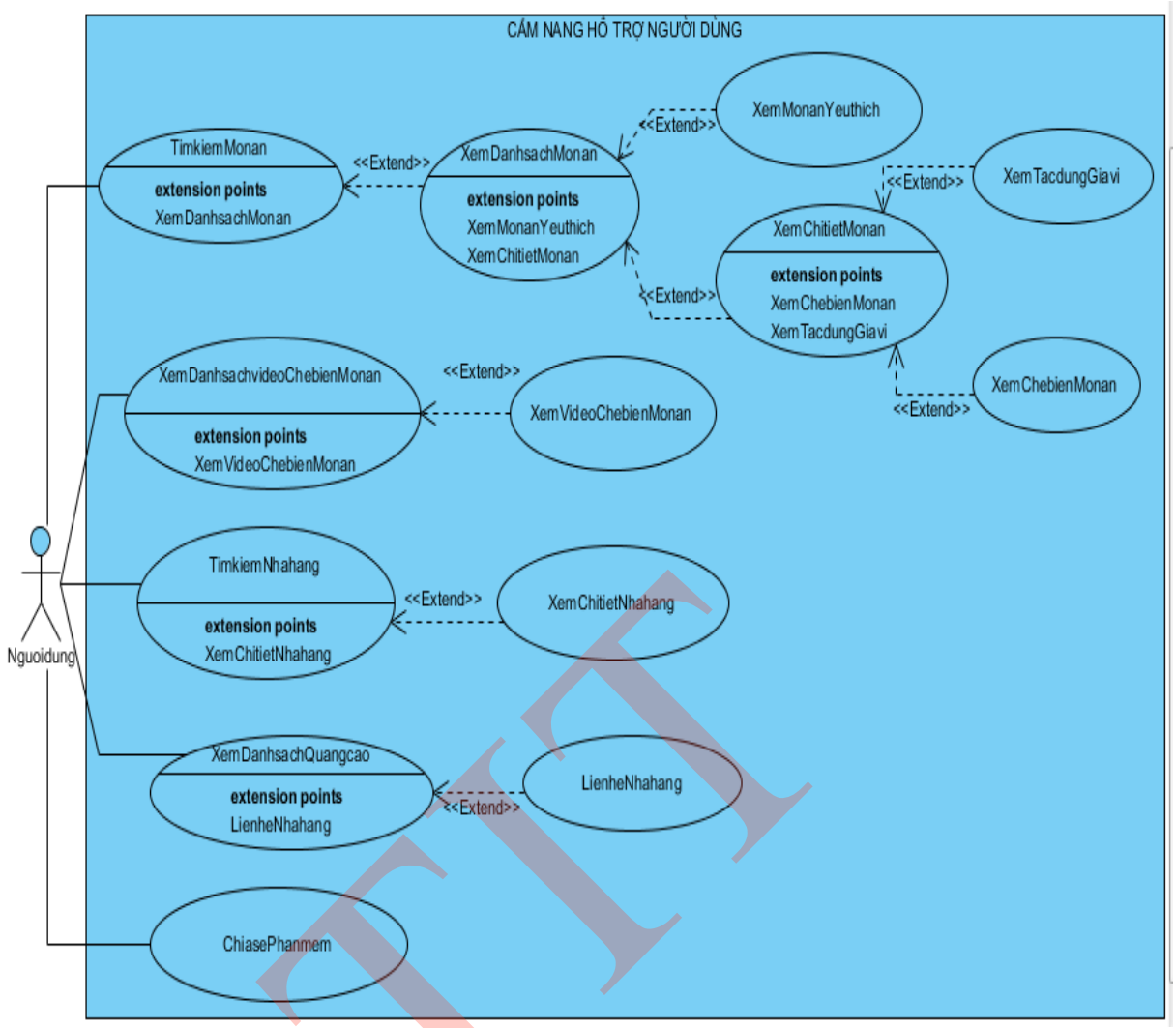
Chức năng chia sẻ phần mềm: Cho phép người dùng chia sẻ thông tin về phần mềm qua sms, facebook, gmail bao gồm: tên phần mềm, mô tả phần mềm, đường dẫn.

Phân tích hệ thống:

Biểu đồ Usecase tổng quát của hệ thống

Bảng 1.1: Danh sách các usecase

TT	Tên usecase	Ý nghĩa
1	TimkiemMonan (Tìm kiếm món ăn)	Chức năng này cho phép người dùng xem danh sách các loại món ăn ba miền Bắc, Trung, Nam.
2	XemDanhsachMonan (Xem danh sách món ăn)	Chức năng này cho phép người dùng xem danh sách các món ăn của một loại món ăn.
3	XemChitietMonan (Xem món ăn)	Chức năng này cho phép người dùng xem chi tiết món ăn.
4	XemDanhsachMonanYeuthich (Xem danh sách món ăn yêu thích)	Chức năng này cho phép người dùng xem danh sách các món ăn yêu thích được người dùng đánh dấu.
5	XemTacdungGiavi (Xem tác dụng gia vị)	Chức năng này cho phép người dùng xem tác dụng của gia vị trong món ăn đã chọn.
6	XemChebienMonan(Xem chế biến món ăn)	Chức năng này cho phép người dùng xem các bước chế biến món ăn đã chọn.
7	TimkiemNhahang (Tìm kiếm nhà hàng)	Chức năng này cho phép người dùng tìm kiếm các nhà hàng ở Hà Nội, Huế, Sài Gòn.
8	XemChitietNhahang (Xem chi tiết nhà hàng)	Chức năng này cho phép người dùng xem chi tiết các thông tin về nhà hàng (Hình ảnh, địa chỉ, số điện thoại liên hệ, website nhà hàng, giờ mở cửa, giờ đóng cửa, các đánh giá của người dùng đối với nhà hàng…)
9	XemDanhsachQuangcao (Xem danh sách quảng cáo)	Chức năng này cho phép người dùng xem danh sách quảng cáo của các nhà hàng.
10	Lienhe (Liên hệ)	Chức năng này cho phép người dùng liên hệ với nhà hàng.
11	XemDanhsachvideoChebienMonan(Xem danh sách video chế biến món ăn)	Chức năng này cho phép người dùng xem danh sách các video hướng dẫn nấu ăn của các món.
12	XemVideoChebienMonan (Xem video chế biến món ăn)	Chức năng này cho phép người dùng xem video hướng dẫn chế biến món ăn người dùng đã chọn.
13	ChiasePhanmem (Chia sẻ phần mềm)	Chức năng này cho phép người dùng chia sẻ phần mềm bạn bè qua sms, gmail, facebook.



Hình 1.5: Sơ đồ usecase tổng quát của hệ thống cẩm nang hỗ trợ người dùng

Biểu đồ lớp phân tích

Lớp biên:

- GDDanhsachLoaiMonan (Giao diện danh sách loại món ăn)
- GDDanhsachMonan (Giao diện danh sách món ăn)
- GDChitietMonan (Giao diện chi tiết món ăn)
- GDChitietMonan (Giao diện chi tiết món ăn)
- GDTacdungGiavi (Giao diện tác dụng gia vị)
- GDDanhsachNhahang (Giao diện danh sách nhà hàng)
- GDThongtinChitietNhahang (Giao diện thông tin chi tiết nhà hàng)
- GDDanhsachvideoChebienMonan (Giao diện danh sách video chế biến món ăn)

Lớp thực thể:

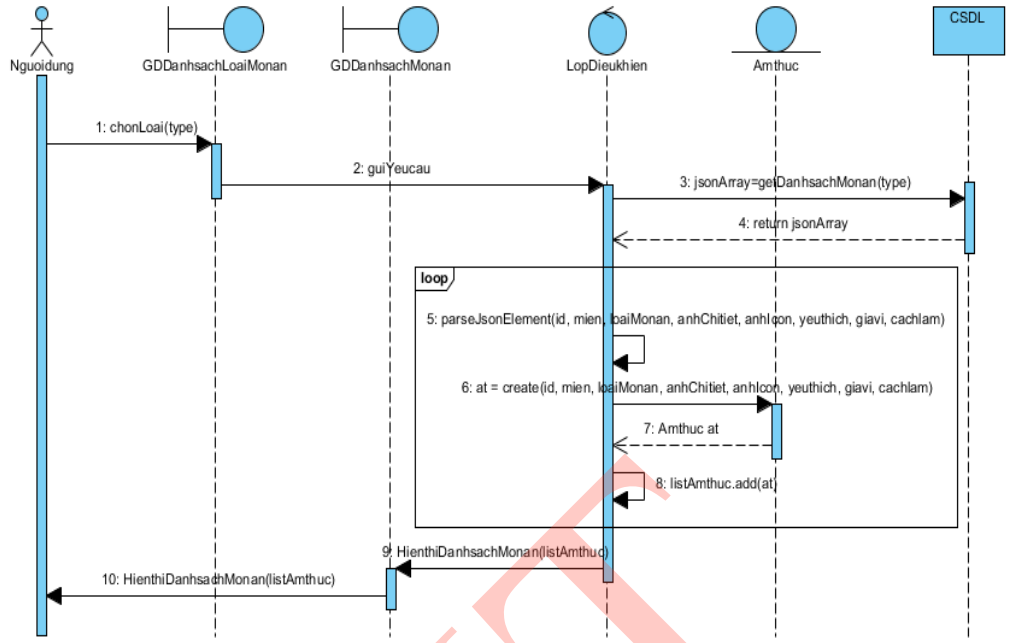
- Amthuc (Ẩm thực)
- Quangcao (Quảng cáo)
- Diadiem (Địa điểm)
- ChitietDiadiem (Chi tiết địa điểm)
- Giavi (Gia vị)
- Nhahang (Nhà hàng)

Lớp điều khiển:

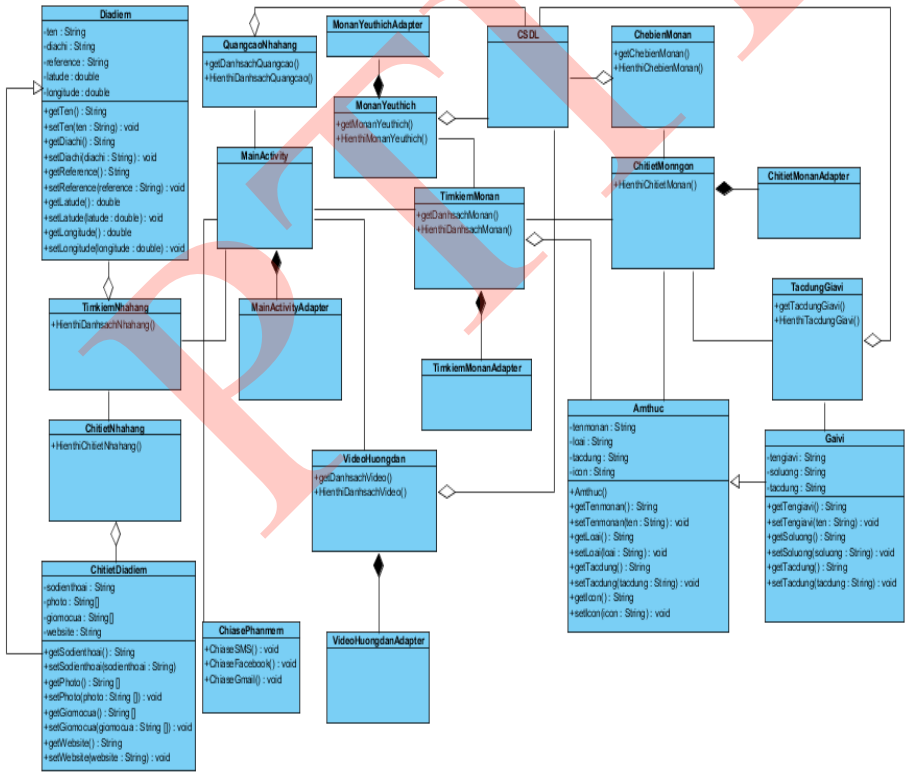
- LopDieukhien (Lớp điều khiển)

Thiết kế hệ thống:

Thiết kế cho từng chức năng.



Hình 1.6: Biểu đồ tuần tự cho chức năng TimkiemMonngon



Hình 1.7: Sơ đồ lớp chi tiết

1.2. Các khái niệm về CSDL

1.2.1. Các khái niệm chung

Hệ quản trị cơ sở dữ liệu (Database Management System - DBMS): Là một hệ thống phần mềm cho phép tạo lập cơ sở dữ liệu và điều khiển mọi truy nhập đối với cơ sở dữ liệu

đó. Trên thị trường phần mềm hiện nay ở Việt Nam đã xuất hiện khá nhiều phần mềm hệ quản trị cơ sở dữ liệu như: Microsoft Access, Foxpro, DB2, SQL Server, Oracle,.v.v…

Hệ quản trị cơ sở dữ liệu quan hệ (Relation Database Management System - RDBMS) là một hệ quản trị cơ sở dữ liệu theo mô hình quan hệ.

Một hệ quản trị cơ sở dữ liệu thường cung cấp hai kiểu ngôn ngữ khác nhau đó là: ngôn ngữ mô tả sơ đồ cơ sở dữ liệu và ngôn ngữ biểu diễn các truy vấn và các cập nhật cơ sở dữ liệu.

- Ngôn ngữ định nghĩa dữ liệu (Data Definition Language - DDL)

+ Một sơ đồ CSDL đặc tả bởi một tập các định nghĩa được biểu diễn bởi một ngôn ngữ đặc biệt được gọi là ngôn ngữ định nghĩa dữ liệu. Kết quả của việc dịch các ngôn ngữ này là một tập các bảng được lưu trữ trong một tệp đặc biệt được gọi là từ điển dữ liệu hay thư mục dữ liệu.

+ Một từ điển dữ liệu là một tệp chứa các siêu dữ liệu có nghĩa là các dữ liệu về dữ liệu. Tệp này được tra cứu trước khi dữ liệu thực sự được đọc hay được sửa đổi trong hệ CSDL.

+ Cấu trúc và các phương pháp truy nhập được sử dụng bởi hệ CSDL được đặc tả bởi một tập các định nghĩa trong một kiểu đặc biệt của DDL là ngôn ngữ định nghĩa và lưu trữ dữ liệu.

- Ngôn ngữ thao tác dữ liệu (Data Manipulation Language - DML):

+ Các yêu cầu về thao tác dữ liệu bao gồm:

• Tìm kiếm thông tin được lưu trữ trong CSDL.
• Thêm thông tin mới vào CSDL.
• Xoá thông tin từ CSDL.
• Thay đổi thông tin được lưu trữ trong CSDL.

+ Một ngôn ngữ thao tác dữ liệu (DML) là một ngôn ngữ cho phép người sử dụng truy nhập hay thao tác dữ liệu được tổ chức bởi mô hình dữ liệu thích hợp. Có hai kiểu ngôn ngữ thao tác dữ liệu cơ bản:

• Các DML thủ tục đòi hỏi người sử dụng phải đặc tả dữ liệu nào cần tìm kiếm và tìm kiếm những dữ liệu này như thế nào.

• Các DML phi thủ tục đòi hỏi người sử dụng đặc tả dữ liệu nào cần tìm kiếm mà không phải đặc tả tìm kiếm những dữ liệu này như thế nào.

1.2.2. Các phép toán của bài toán CSDL

Ngôn ngữ SQL (Structured Query Language) là ngôn ngữ truy vấn có cấu trúc, dùng để thao tác với dữ liệu trong cơ sở dữ liệu cũng như tạo và thay đổi cấu trúc của các cơ sở dữ liệu. Trong chương này ta sẽ trình bày một số câu lệnh SQL cơ bản.

Lệnh CREATE dùng để tạo các đối tượng cơ sở dữ liệu như các bảng, các view, các tệp chỉ số.v.v…

- Cú pháp:

+ CREATE TABLE <Tên bảng>(<Danh sách: Tên_cột Kiểu_cột> <Điều_kiện_kiểm_soát_dl >)

+ CREATE VIEW <Tên View>(<Danh sách: Tên_cột Kiểu_cột> <Điều_kiện_kiểm_soát_dl >) AS Q; với Q là một khối câu lệnh SELECT định nghĩa khung nhìn (view).

+ CREATE [UNIQUE] INDEX <tên chỉsố> ON <Ten bảng>(Tên cột [ASC|DESC])

- Một số kiểu dữ liệu: Integer - sốnguyên; float - dấu phảy động; char - ký tự, datetime - ngày tháng, boolean,…

Lệnh ALTER: dùng để thay đổi cấu trúc lược đồ của các đối tượng CSDL.

- Cú pháp:

+ ALTER TABLE <Tên bảng> <Thực hiện các lệnh trên cột>

Các lệnh trên cột có thể là:

• Xóa một cột: Delete <tên cột>

• Thêm một cột: Add <Tên cột>

• Thay đổi tên cột: Change column <Tên cột>To<Tên cột>

• Xóa khóa chính: Drop PRIMARY KEY

• Xóa khóa ngoại: Drop FOREIGN KEY

• Thiết lập khóa chính: PRIMARY KEY (Tên cột)

• Thiết lập khóa ngoại:

FOREIGN KEY (Tên cột) REFERENCES TO <tên bảng ngoài>

+ ALTER VIEW <Tên View>(<Danh sách: Tên_cột Kiểu_cột>

<Điều_kiện_kiểm_soát_dl >) AS Q; với Q là một khối câu lệnh

SELECT định nghĩa khung nhìn (view).

Lệnh DROP: dùng để xóa các đối tượng cơ sở dữ liệu như Table, View,

Index, .v.v…

- Cú pháp:

DROP TABLE <Tên bảng>

DROP VIEW <Tên view>

DROP INDEX <Tên index>

Lệnh INSERT INTO: dùng để chèn một hàng hoặc một số hàng cho bảng.

- Cú pháp:

+ INSERT INTO <Tên bảng> (Danh sách các cột) VALUES

(Danh sách các giá trị) hoặc

+ INSERT INTO <Tên bảng> (Danh sách các cột) (Các câu hỏi con);

Lệnh UPDATE: dùng để sửa đổi dữ liệu.

- Cú pháp:

UPDATE <Tên bảng>

SET <Tên_cột_1=Biểu_thức_1, Tên_cột_2=Biểu_thức_2,… >[WHERE <điều kiện>]

Lệnh DELETE FROM: Xoá một số hàng trong bảng.

- Cú pháp:

DELETE FROM <Tên bảng> WHERE <Điều kiện>

Khối câu lệnh phổ dụng: SELECT - FROM – WHERE. Ta có thể sử dụng theo cú pháp chung như sau:

SELECT [*| DISTINCT] <Danh sách các cột [AS <Bí danh>]>

FROM <Danh sách Tên bảng/Tên View>

[WHERE <Biểu thức điều kiện>]

[GROUP BY <Danh sách cột>]

[HAVING <Điều kiện>]

[ORDER BY <Tên cột/ Số thứ tự cột/Biểu thức> [ASC/DESC]]

CHƯƠNG 2: NHẬP MÔN VISUAL BASIC.NET

2.1 Bắt đầu với VB.NET

Ngôn ngữ **BASIC** (Beginner's All Purpose Symbolic Instruction Code) đã có từ năm 1964, BASIC rất dễ học và dễ dùng. Trong vòng 15 năm đầu, có rất nhiều chuyên gia Tin Học và công ty tạo các chương trình thông dịch (Interpreters) và biên dịch (Compilers) cho ngôn ngữ làm BASIC trở nên rất phổ thông.

Năm 1975, Microsft tung ra thị trường sản phẩm đầu tay Microsoft BASIC và tiếp đó Quick BASIC (còn gọi là **QBASIC**) thành công rực rỡ.

Quick BASIC phát triển trong nền Windows nhưng vẫn khó khăn khi tạo giao diện kiểu Windows. Sau đó nhiều năm, Microsoft bắt đầu tung ra một sản phẩm mới cho phép ta kết hợp ngôn ngữ dễ học BASIC và môi trường phát triển lập trình với giao diện bằng hình ảnh (Graphic User Interface - **GUI**) trong Windows. Đó là Visual Basic Version 1.0.

Sự chào đời của Visual Basic Version 1.0 vào năm 1991 thật sự thay đổi bộ mặt lập trình trong Công Nghệ Tin Học.

Trước đó, ta không có một giao diện bằng hình ảnh (GUI) với một **IDE (Integrated Development Environment)** giúp các chuyên gia lập trình tập trung công sức và thì giờ vào các khó khăn liên hệ đến doanh nghiệp của mình. Mỗi người phải tự thiết kế giao diện qua thư viện có sẵn **Windows API (Application Programming Interface)** trong nền Windows. Điều này tạo ra những trở ngại không cần thiết làm phức tạp việc lập trình.

Visual Basic giúp ta bỏ qua những hệ lụy đó, chuyên gia lập trình có thể tự **vẽ** cho mình giao diện cần thiết trong ứng dụng (application) một cách dễ dàng và như vậy, tập trung nổ lực giải đáp các vấn đề cần giải quyết trong doanh nghiệp hay kỹ thuật.

Ngoài ra, còn nhiều công ty phụ phát triển thêm các khuôn mẫu (modules), công cụ (tools, controls) hay ứng dụng (application) phụ giúp dưới hình thức **VBX** cộng thêm vào giao diện chính càng lúc càng thêm phong phú.

Khi Visual Basic phiên bản 3.0 được giới thiệu, thế giới lập trình lại thay đổi lần nữa. Kỳ này, ta có thể thiết kế các ứng dụng (application) liên hệ đến **Cơ Sở Dữ Liệu (Database)** trực tiếp tác động (interact) đến người dùng qua **DAO (Data Access Object)**. Ứng dụng này thường gọi là **ứng dụng tiền diện (front-end application) hay trực diện.**

Phiên bản 4.0 và 5.0 mở rộng khả năng VB nhắm đến Hệ Điều Hành Windows 95.

Phiên bản 6.0 cung ứng một phương pháp mới nối với Cơ Sở Dữ Liệu (Database) qua sự kết hợp của **ADO (Active Data Object)**. ADO còn giúp các chuyên gia phát triển mạng nối với Cơ Sở Dữ Liệu (Database) khi dùng Active Server Pages (ASP).

Tuy nhiên, VB phiên bản 6.0 (VB6) không cung ứng tất cả các đặc trưng của kiểu mẫu **ngôn ngữ lập trình khuynh hướng đối tượng** (**O**bject **O**riented **L**anguage - **OOL**) như các ngôn ngữ C++, Java.

Thay vì cải thiện hay vá víu thêm thắt vào VB phiên bản 6.0, Microsoft đã xoá bỏ tất cả làm lại từ đầu các ngôn ngữ lập trình mới theo kiểu OOL rất hùng mạnh cho khuôn nền .NET Framework. Đó là các ngôn ngữ lập trình **Visual Basic.NET và C# (gọi là C Sharp)**. Sau đó, nhiều ngôn ngữ lập trình khác cũng thay đổi theo ví dụ như smalltalk.NET, COBOL.NET, ... làm Công Nghệ Tin Học trở nên phong phú hơn, đa dạng hơn.

Tất cả những thay đổi này nhằm đáp ứng kịp thời sự đòi hỏi và nhu cầu phát triển cấp bách trong kỹ nghệ hiện nay.

Visual Basic.NET (VB.NET) là ngôn ngữ lập trình khuynh hướng đối tượng **(Object Oriented Programming Language)** do Microsoft thiết kế lại từ con số không. Visual Basic.NET (VB.NET) không kế thừa VB6 hay bổ sung, phát triển từ VB6 mà là một ngôn ngữ lập trình hoàn toàn mới trên nền Microsoft's .NET Framework. Do đó, nó cũng không phải là VB phiên bản 7. Thật sự, đây là ngôn ngữ lập trình mới và rất lợi hại, không những lập nền tảng vững chắc theo kiểu mẫu đối tượng như các ngôn ngữ lập trình hùng mạnh khác đã vang danh C++, Java mà còn dễ học, dễ phát triển và còn tạo mọi cơ hội hoàn hảo để giúp ta giải đáp những vấn đề khúc mắc khi lập trình. Hơn nữa, dù không khó khăn gì khi cần tham khảo, học hỏi hay đào sâu những gì xảy ra bên trong ... hậu trường OS, Visual Basic.NET (VB.NET) giúp ta đối phó với các phức tạp khi lập trình trên nền Windows và do đó, ta chỉ tập trung công sức vào các vấn đề liên quan đến dự án, công việc hay doanh nghiệp mà thôi.

Trong khóa học này, các bạn sẽ bắt đầu làm quen với kiểu lập trình dùng Visual Basic.NET (VB.NET) và dĩ nhiên, các khái niệm và thành phần cơ bản của .NET Framework.

Nếu ta để ý tên của Visual Basic**.NET** (VB.NET), ta thấy ngay ngôn ngữ lập trình này chuyên tạo ứng dụng (application) dùng trong mạng, liên mạng hay trong Internet. Do đó, ta sẽ tập trung vào việc lập trình các ứng dụng (applications) trên nền Windows và đó cũng là mục tiêu chính yếu khi học Visual Basic.NET cơ bản.

2.2 Cài đặt Microsoft Visual Studio.NET

Bộ Microsoft Visual Studio.NET bao gồm vừa mọi công cụ yểm trợ lập trình và ngôn ngữ lập trình .NET, tỷ như: **Visual Basic.NET (VB.NET), C# (C Sharp), Visual C++.NET và Visual J#.NET**

Tùy ý ta chọn loại ngôn ngữ lập trình nào thích hợp để cài vào máy vi tính. Không ai cấm ta cài đủ thứ vào máy nhưng dĩ nhiên cần phải có dư chỗ trong hard drive, Microsoft Visual Studio.NET sẽ tính toán và cho ta biết khả năng chứa như thế nào. Tuy nhiên, ta có thể chỉ chọn Visual Basic.NET (VB.NET) và các ứng dụng (application) liên hệ trước, nếu cần học thêm về C# hay Visual C++.NET, ta có thể cài sau cũng được vì nếu cài toàn bộ, ta sẽ cần khoãng trên dưới 1.5 GBytes trong hard drive.

Microsoft Visual Studio.NET có nhiều phiên bản khác nhau. Dưới đây, ta tạm dùng phiên bản **Enterprise Architecture 2003** làm thí dụ điển hình. Tùy theo phiên bản ta có, những bước cài đặt sẽ khác nhau 1 chút nhưng trên nguyên tắc, ta phải cài đầy đủ môi trường .NET yểm trợ lập trình trước khi cài Microsoft Visual Studio.NET, tỷ như:

Microsoft .NET Framework

Microsoft FrontPage Web Extensions Client

Microsoft Access trong bộ MS Office Professional

Microsoft SQL Server - sẽ hướng dẫn cài và bố trí MS SQL Server cho khóa học trong bài Cơ Sở Dữ Liệu (Database)

và các ứng dụng (application) liên hệ (Microsoft Visual Studio.NET cho biết ta cần những gì) như hình trong bước thứ 3.

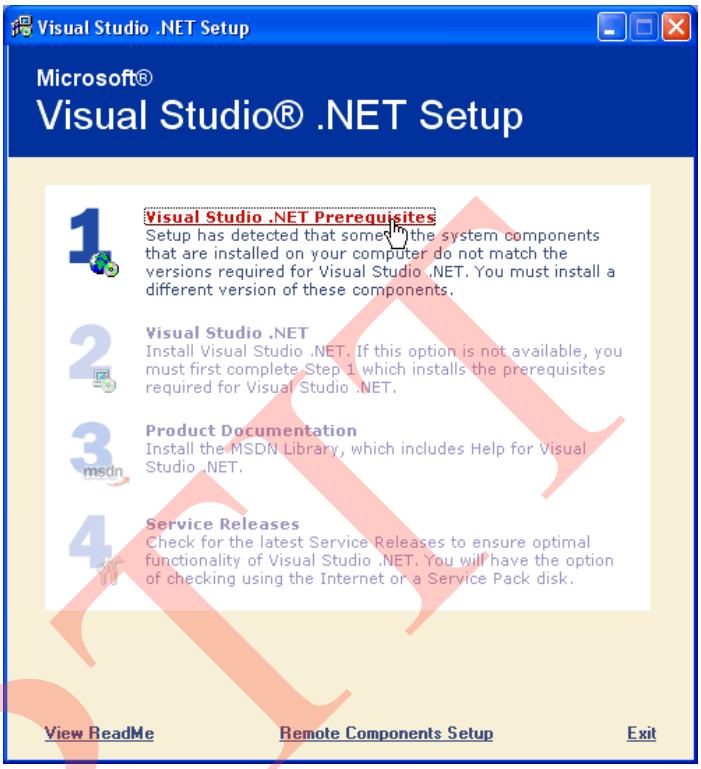
Bước 1:

Bắt đầu với dĩa 1 của bộ Microsoft Visual Studio.NET, dĩa này tự khởi động và hiển thị Windows hướng dẫn ta cài Microsoft Visual Studio.NET Setup. Nếu CD không tự khởi động được, ta cần chạy ứng dụng '**setup.exe**' trong vị trí gốc (root directory):

Chạy Windows Explorer, chọn dĩa cứng chứa Microsoft Visual Studio.NET Setup dĩa 1, nhấp đôi ứng dụng 'setup.exe' hay

Khởi động (Windows Start Menu) và chọn 'Run', gõ hàng chữ: 'e:\setup.exe' (nếu CD/DVD drive của ta là drive E).

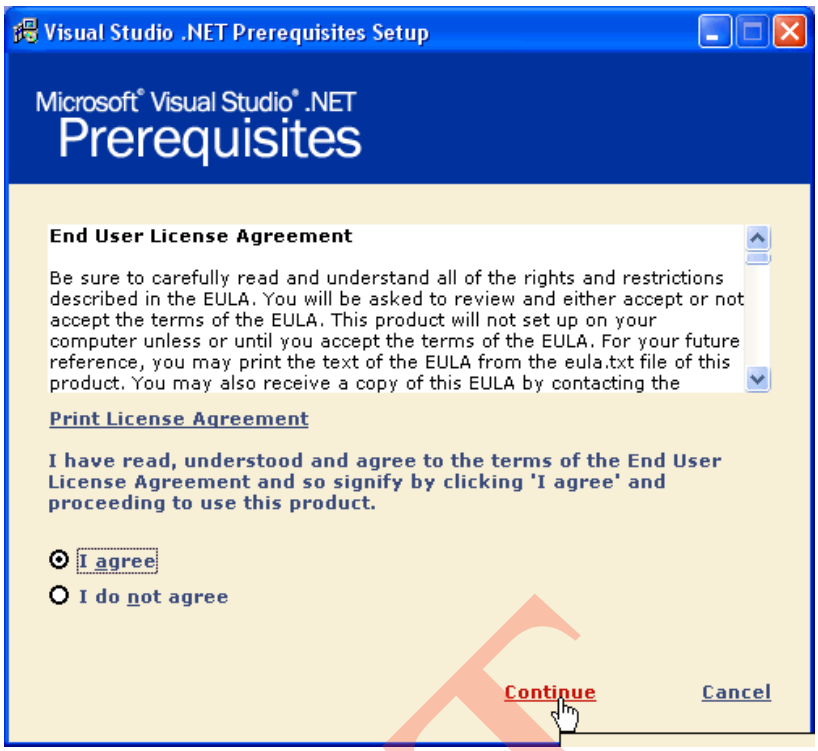
Microsoft Visual Studio.NET hiển thị 4 bước cài. Bước đầu tiên là chuẩn bị môi trường lập trình .NET với '**Visual Studio .NET Prerequisites**':



Hình 2.1: Màn hình bước 1

Bước 2:

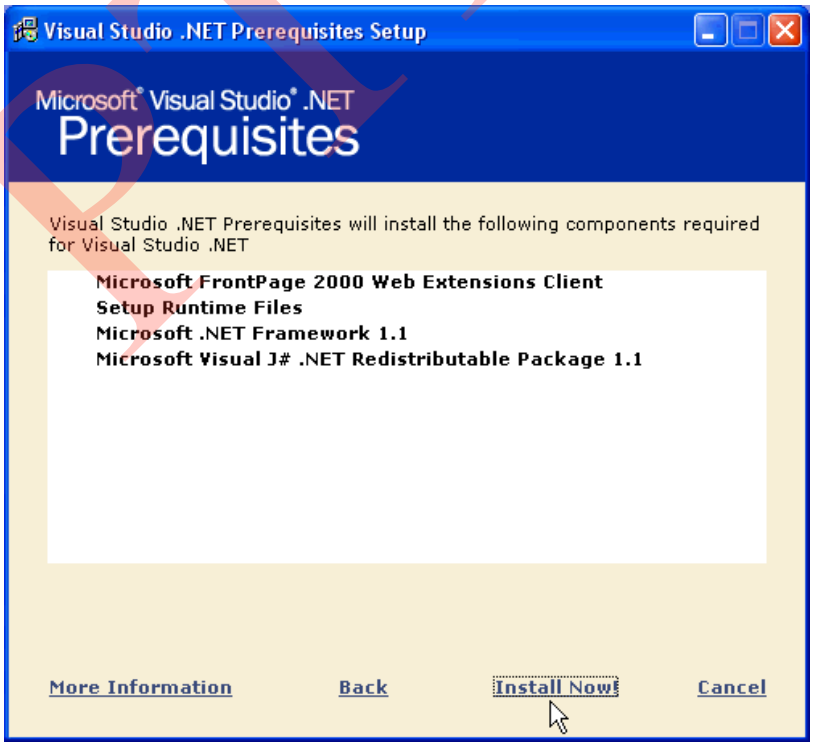
Nhập CD tên Microsoft Visual Studio.NET 2003 Prerequisites, chọn 'I agree' chấp nhận điều kiện dùng nhu liệu và nhấp Continue.



Hình 2.2: Màn hình bước 2

Bước 3:

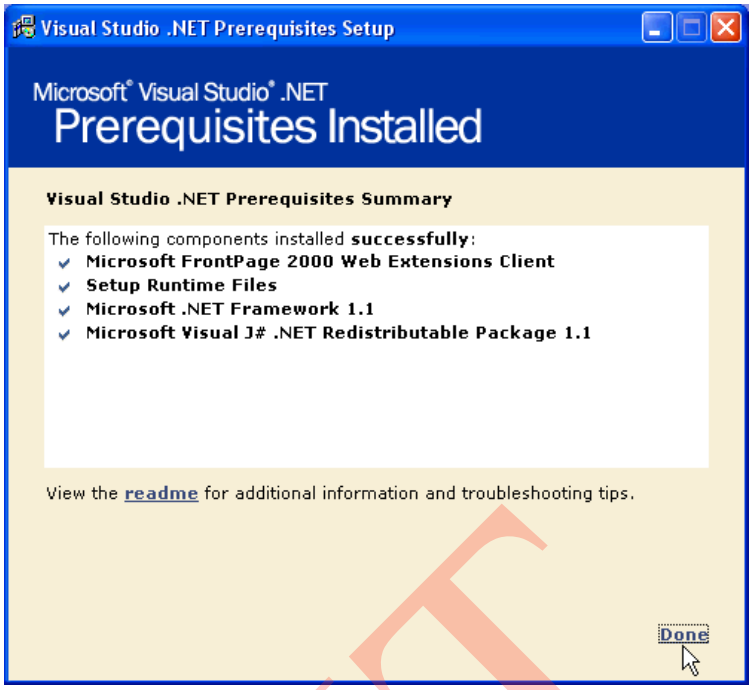
Nhấp Install Now! để cài các ứng dụng (application) liên hệ tạo môi trường .NET. Lưu ý ở đây, Microsoft Visual Studio.NET sẽ dò tìm những ứng dụng (application) cần thiết trong máy vi tính và tùy theo mỗi máy, bảng liệt kê ứng dụng có thể khác nhau. Thí dụ ở đây cho biết máy vi tính cần 4 ứng dụng phụ thuộc như hình sau:



Hình 2.3: Màn hình bước 3

Bước 4:

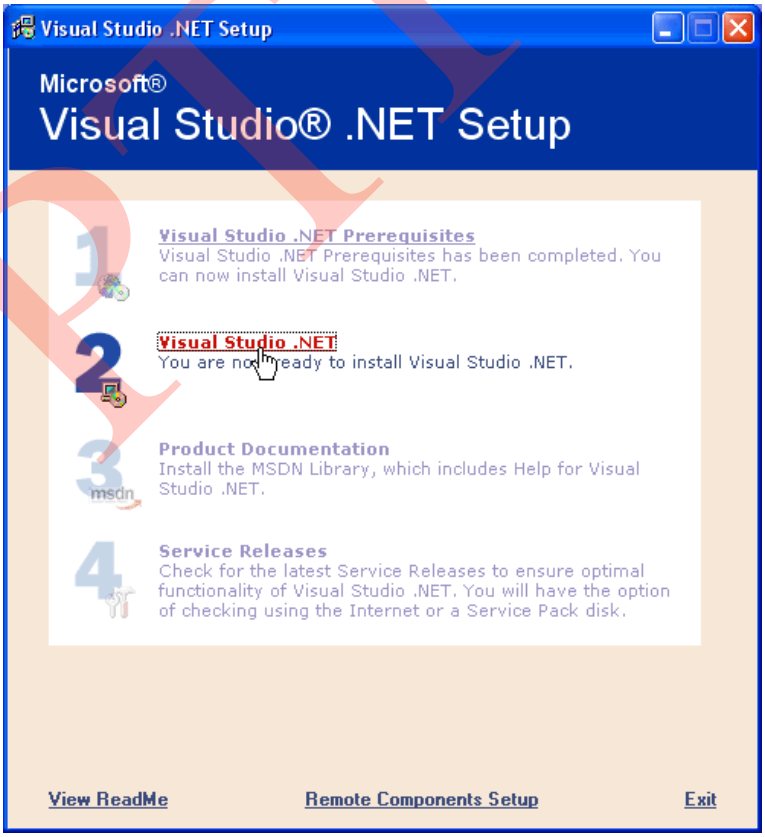
Chờ cho đến khi nào Microsoft Visual Studio.NET cài xong các ứng dụng phụ thuộc, nhấp nút Done.



Hình 2.4: Màn hình bước 4

Bước 5:

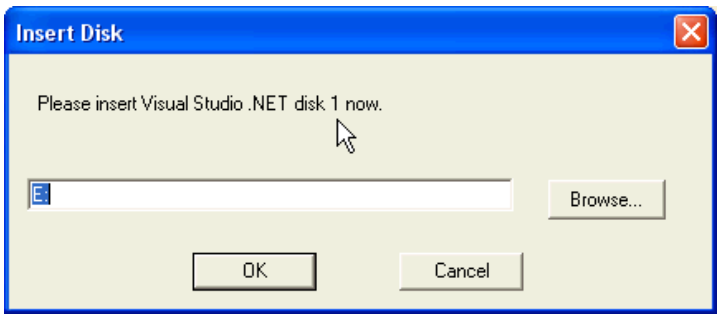
Tiếp tục chọn Visual Studio.NET



Hình 2.5: Màn hình bước 5

Bước 6:

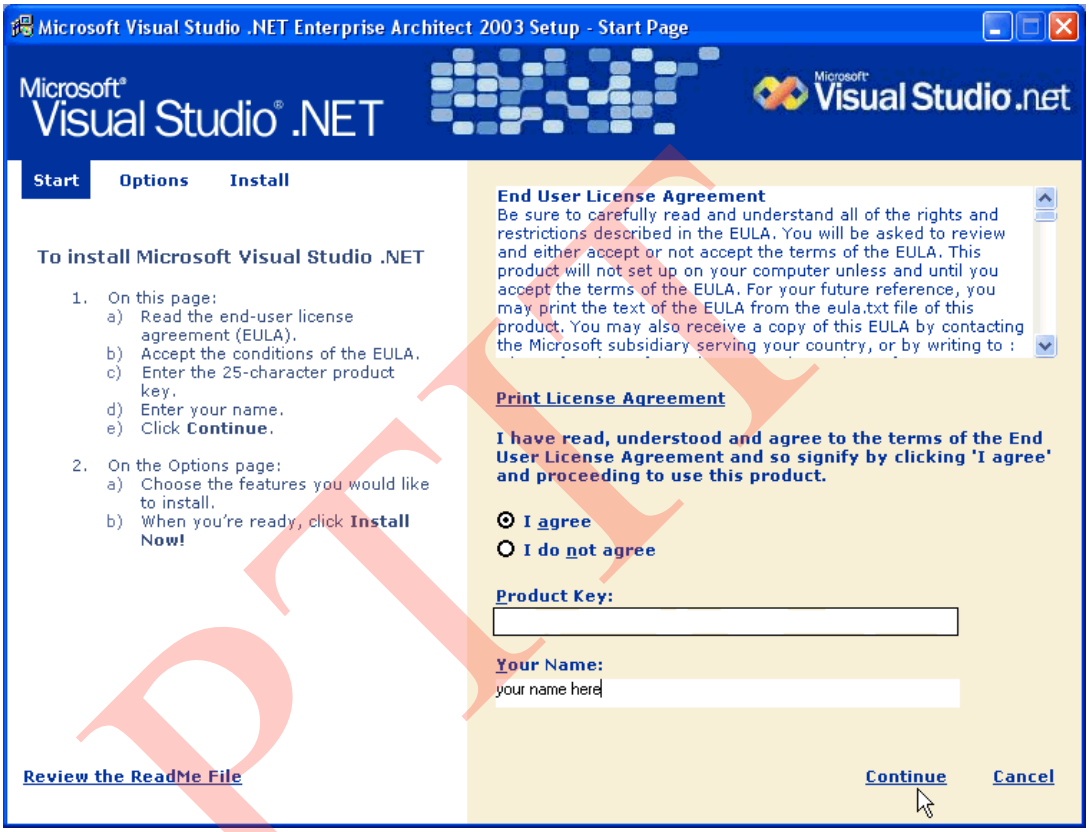
Nhập dĩa 1 vào máy và nhấp nút OK.



Hình 2.6: Màn hình bước 6

Bước 7:

Ta chọn 'I agree' và cung cấp Product Key trước khi nhấp nút Continue.

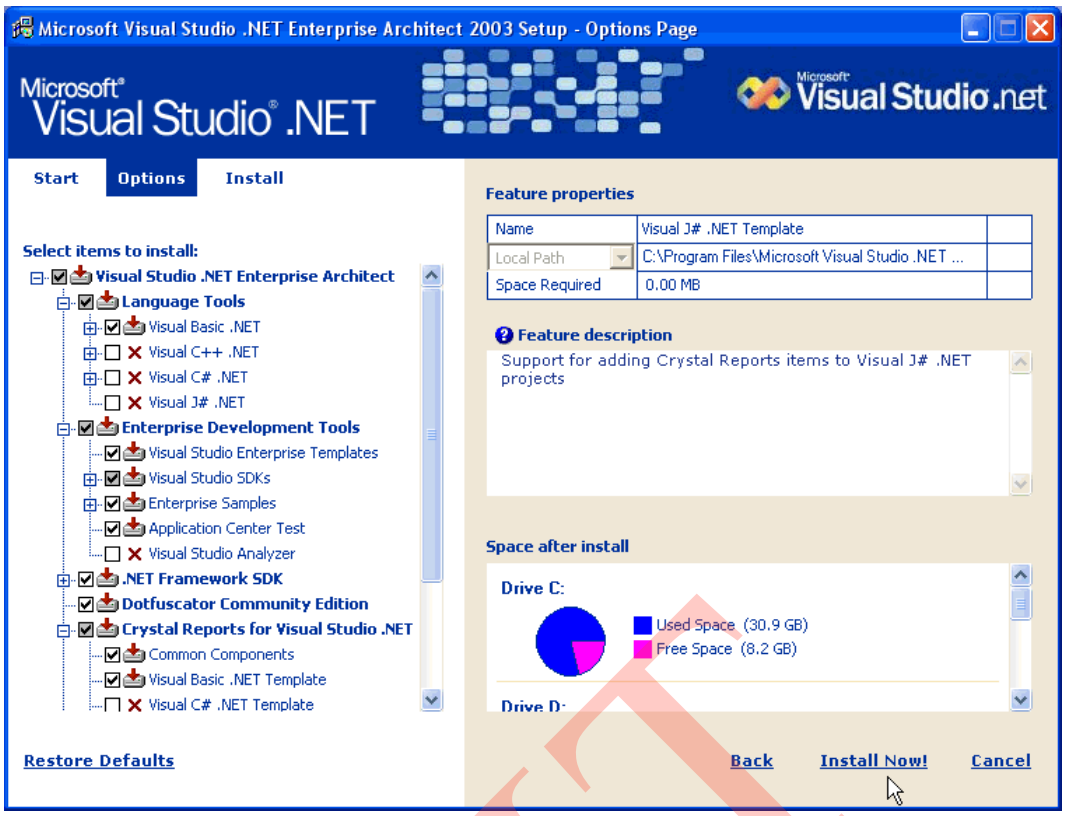


Hình 2.7: Màn hình bước 7

Bước 8:

Ta chỉ chọn những gì liên hệ đến Visual Basic.NET (VB.NET) cho khóa học Visual Basic.NET (VB.NET) Cơ Bản.

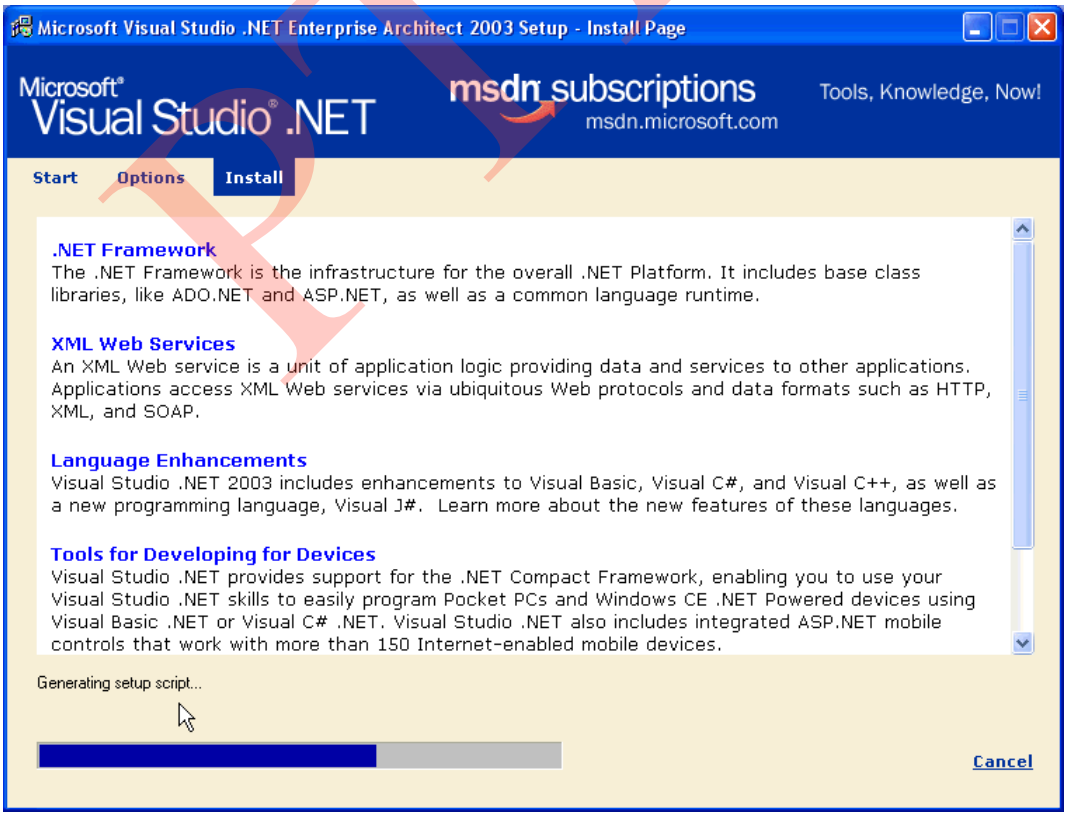
Xóa bỏ (uncheck) ngôn ngữ lập trình Visual C++.NET, Visual C#.NET, Visual J#.NET và các ứng dụng liên hệ, tỷ như: template, documetation, ...



Hình 2.8: Màn hình bước 8

Bước 9:

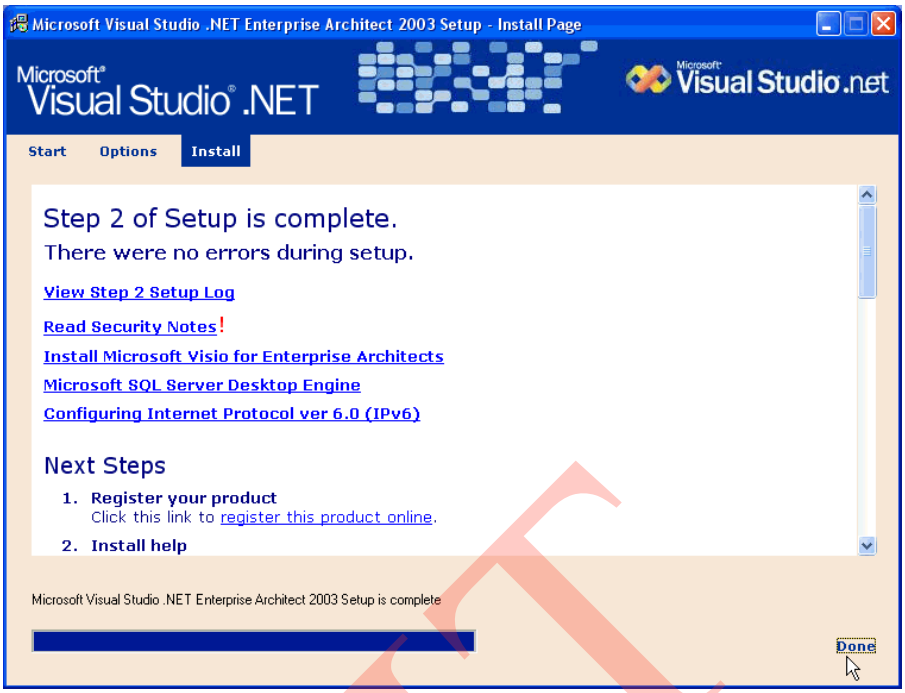
Nhấp Install Now. Microsoft Visual Studio.NET sẽ chạy ứng dụng cài và bố trí này khoãng trên dưới 1 tiếng đồng hồ tùy theo khả năng máy vi tính.



Hình 2.9: Màn hình bước 9

Bước 10:

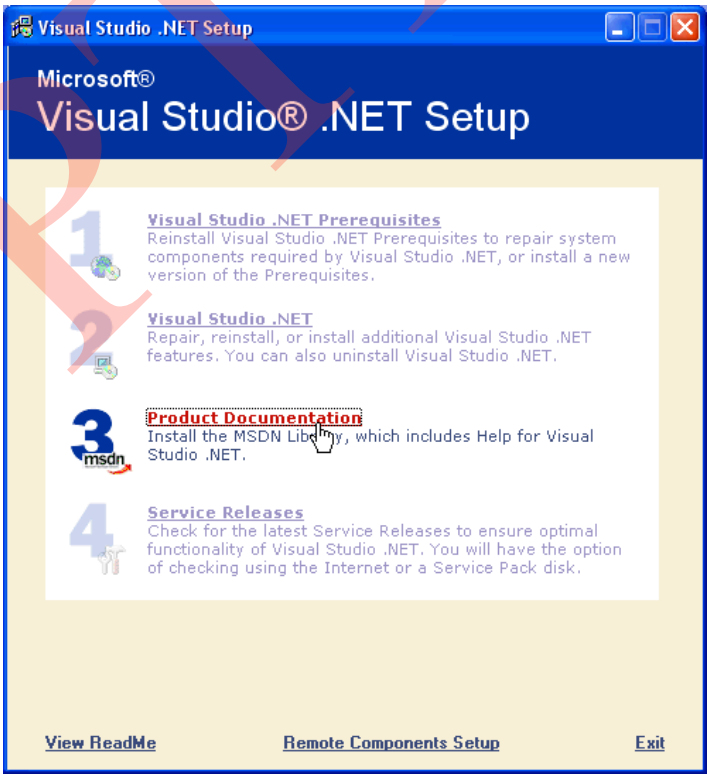
Nhấp **Done**. Microsoft Visual Studio.NET sẽ hiển thị Windows cài các thông tin phụ giúp lập trình và cả thư viện để ta tham khảo khi lập trình với Visual Basic.NET (VB.NET):

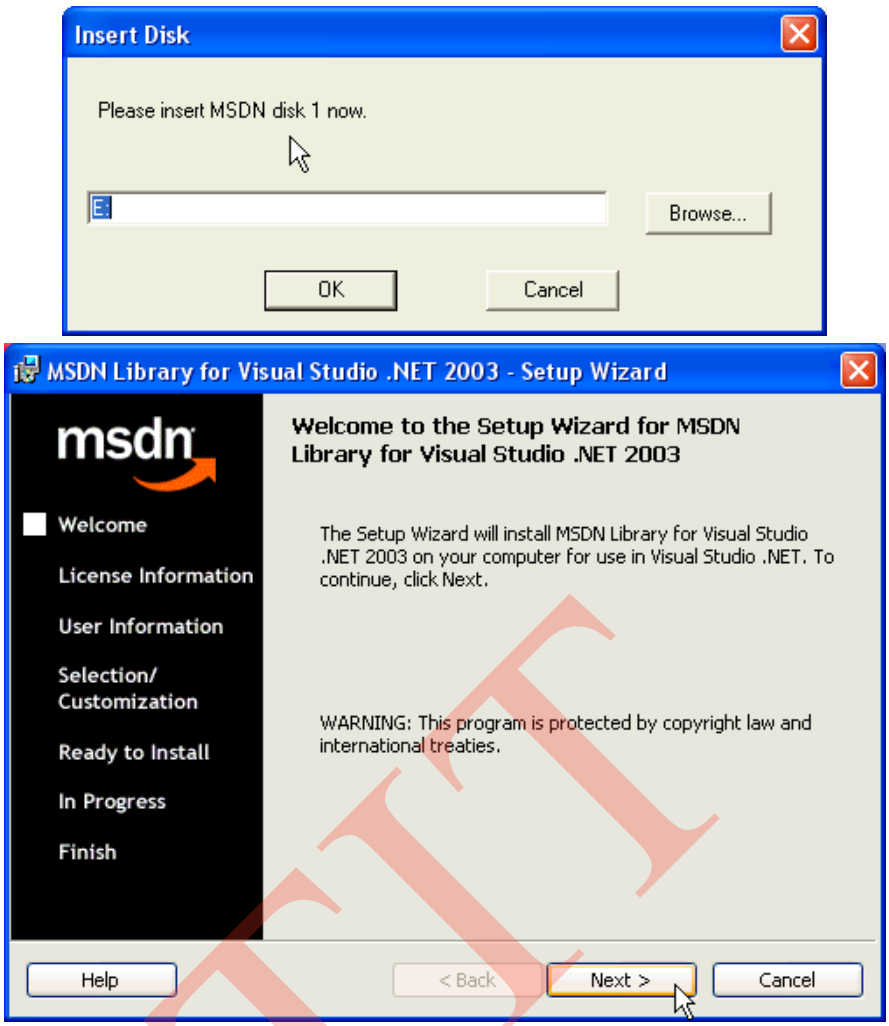


Hình 2.10: Màn hình bước 10

Bước 11:

Chọn Product Documetation và nhập dĩa 3 Microsoft Visual Studio.NET (tức dĩa 1 MSDN):

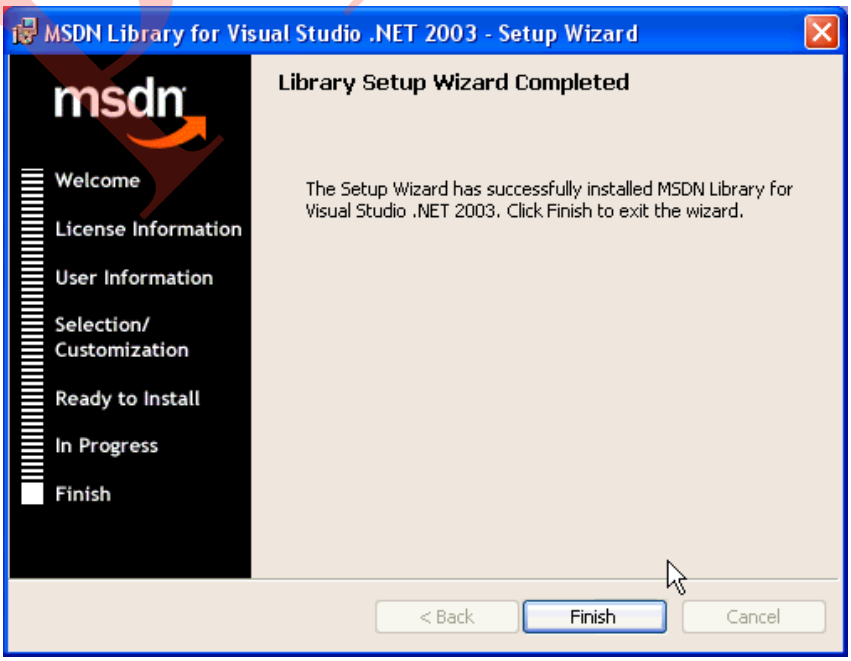




Hình 2.11: Một số màn hình bước 11

Bước 12:

Tiếp tục với các dĩa 2, 3 MSDN cho đến hết.



Hình 2.12: Màn hình bước 12

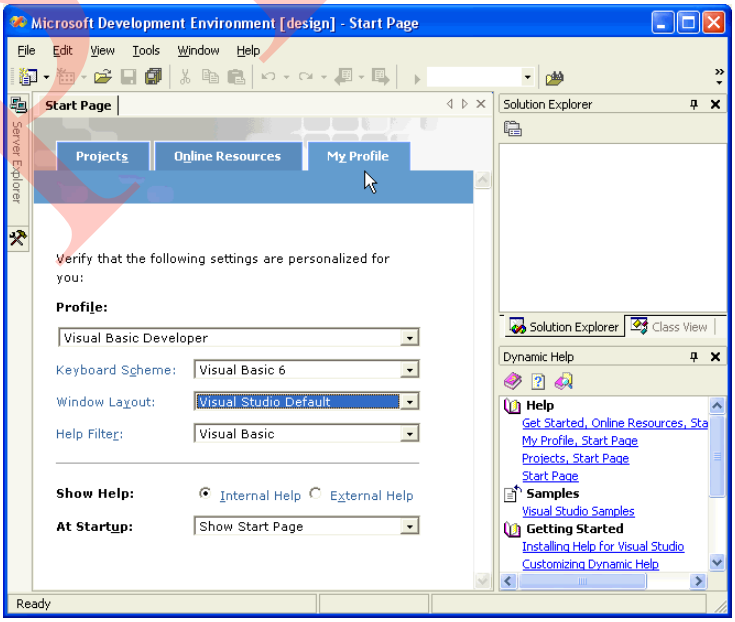
Như vậy, ta sẵn sàng cho việc lập trình với Visual Basic.NET (VB.NET). Bài kế hướng dẫn sơ lược cách dùng **Microsoft Visual Studio.NET Integrated Development Environment (gọi tắc là IDE)** cho việc tạo các ứng dụng (application) trong nền Windows.

Thật ra, ta có thể dùng Notepad để soạn mã nguồn (source code) và Visual Basic.NET compiler để chạy ứng dụng (application) mà không cần Microsoft Visual Studio.NET IDE tuy nhiên trong khóa học cơ bản, chúng tôi chọn Microsoft Visual Studio.NET để việc lập trình trở nên vui thích và hấp dẫn.

2.3 Giới thiệu MS Visual Studio.Net

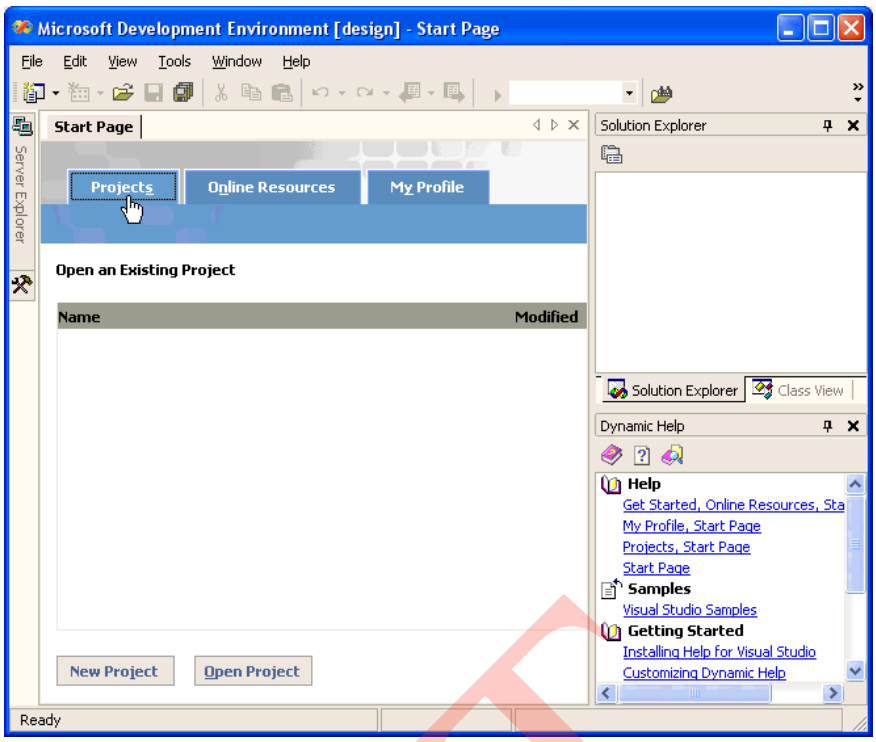
Microsoft Visual Studio.NET IDE là môi trường tập trung mọi công cụ cần thiết giúp việc lập trình dễ dàng.

- Để khởi động, chọn Start, Programs, thực đơn Microsoft Visual Studio.NET 2003 và ứng dụng (application) Microsoft Visual Studio.NET 2003.
- Chọn phần **My Profile**
- Chọn Profile là Visual Basic Developer vì khóa này chuyên trị Visual Basic.NET (VB.NET)
- Microsoft Visual Studio.NET sẽ hiển thị Visual Basic 6 trong hộp chữ **Keyboard Scheme** và ngay cả trong hộp **Windows Layout**. Bố trí này giúp tổ chức các cửa sổ trong IDE như các phiên bản trước của Microsoft Visual Studio. Trong khóa này, ta chọn Visual Studio Default.
- Bố trí gạn lọc giúp đỡ dành riêng cho ngôn ngữ lập trình Visual Basic.NET (VB.NET) trong hộp **Help Filter**.
- Internal Help hiển thị các thông tin ngay trong cùng một IDE window, trong khi External hiển thị thông tin trong 1 window riêng biệt.
- Ở phần **Startup**, chọn **Show Start Page**



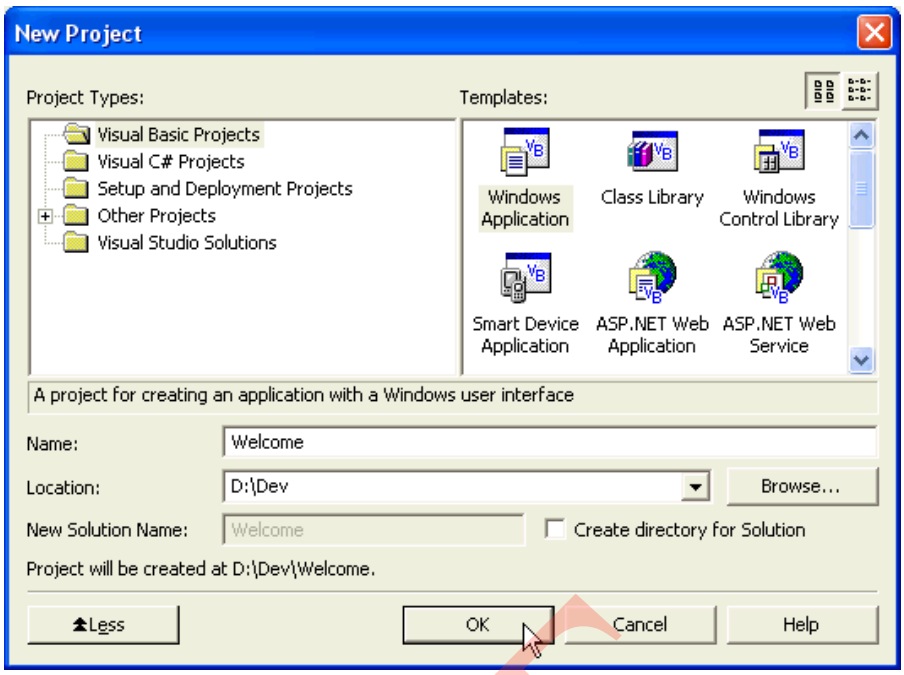
Hình 2.13: Màn hình bước 13

Đây là chổ tạo dự án mới hay mở dự án đã lập trình để sửa đổi. Ta chọn New Project để tìm hiểu thêm môi trường lập trình dùng Microsoft Visual Studio.NET



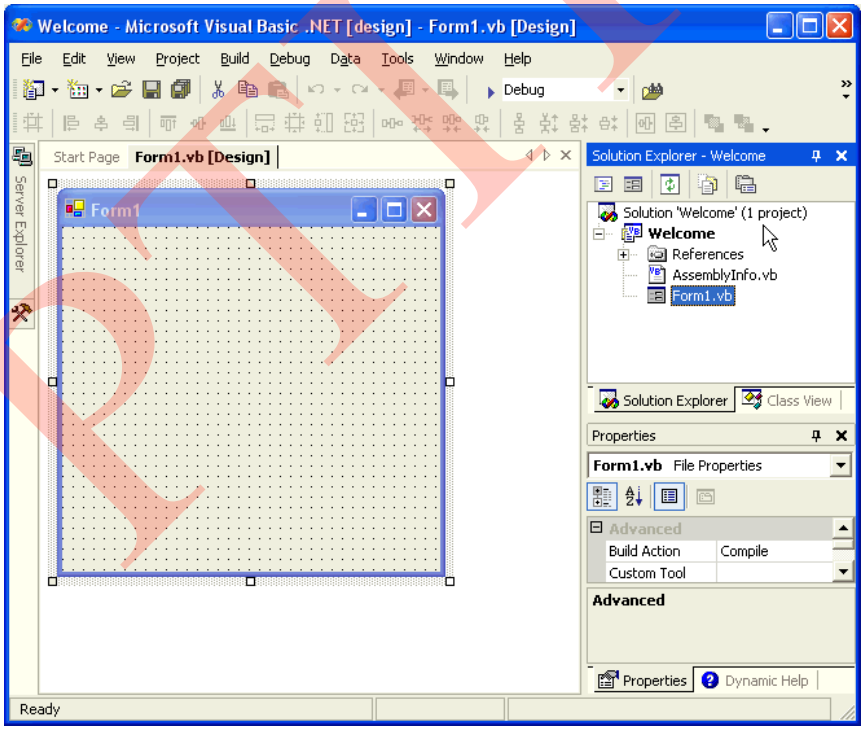
Hình 2.14: Màn hình bước 14

- Ta nhấp nút **New Project** để hiển thị bảng liệt kê các khuôn mẫu cho ứng dụng (application).
- Chọn **Visual Basic Project** trong window Project Types
- Chọn **Windows Application** trong bảng Template
- Đặt tên dự án là **Welcome**. Lưu ý ở đây, tên của dự án cũng là tên ngăn chứa (folder) chứa phụ dự trữ dự án. Thí dụ ta nhấp nút Browse để tạo 1 ngăn chứa (folder) tên Dev ở dĩa D, Microsoft Visual Studio.NET hiển thị D:\Dev ở hộp Location nhưng project sẽ được tạo và chứa ở ngăn chứa (folder) **D:\Dev\Welcome** (để ý hàng phía trên phần hiển thị các nút Less, OK, ... ta thấy hàng chữ: 'Project will be created at D:\Dev\Welcome)
- Nhấp OK



Hình 2.15: Mở dự án mới

Microsoft Visual Studio.NET IDE khởi động dự án mới trong phương thức thiết kế (Design Mode):



Hình 2.16: màn hình dự án mới

2.4 Thực đơn và thanh công cụ

Thực đơn (menu) của Microsoft Visual Studio.NET IDE ... **'biến hóa'** tùy theo công việc đang làm nhưng tổng quát, thực đơn (menu) chính hiển thị bao gồm:

File:

Tiêu chuẩn chung cho mọi ứng dụng (application) trong nền Windows. File dùng để mở (open) hay đóng (close) các tập tin (files) hay dự án (project).

Edit:

Edit cung cấp các chọn lựa khi soạn nguồn mã và dùng các công cụ lập trình, tỷ như: Undo, Redo, Cut, Copy, Paste và Delete

View:

View cung cấp sư chọn lựa hiển thị các Windows tạo môi trường của IDE, tỷ như: Solution Explorer, Properties, Output, Tool Box, Server Explorer. Nếu ta để ý sẽ thấy các Windows này thường nằm 2 bên hoặc bên dưới window thiết kế Form hay soạn nguồn mã.Các windows này cũng có thể hiển lộ hay thu kín lại nhường chổ cho window thiết kế được rộng rãi.

Project:

Dùng để quản lý dự án (project) bằng cách thêm vào hay xóa bỏ các tập tin liên hệ.

Build:

Một lựa chọn quan trọng trong thực đơn là Build cho phép ta xây dựng và chạy ứng dụng (application) 1 cách độc lập bên ngoài IDE.

Debug:

Debug không những giúp phương tiện rà tìm các lỗi lập trình trong môi trường IDE mà còn giúp kiểm tra từng bước một các nguồn mã trong dự án (project).

Data:

Giúp ta nối và sử dụng dữ kiện hay thông tin trong Cơ Sở Dữ Liệu (Database).

Tools:

Chứa các công cụ bố trí Microsoft Visual Studio.NET IDE.

Windows:

Tiêu chuẩn chung dùng quản lý mọi windows trong IDE.

Help:

Cung cấp nối yêu cầu giúp đỡ với Microsoft Visual Studio.NET documentation hay từ mạng Internet.

Cách dùng thanh công cụ sẽ được hướng dẫn tùy từng dự án (project). Tuy nhiên, 1 cách tổng quát, thanh công cụ mặc định (default) bao gồm như sau (theo thứ tự từ trái qua phải):

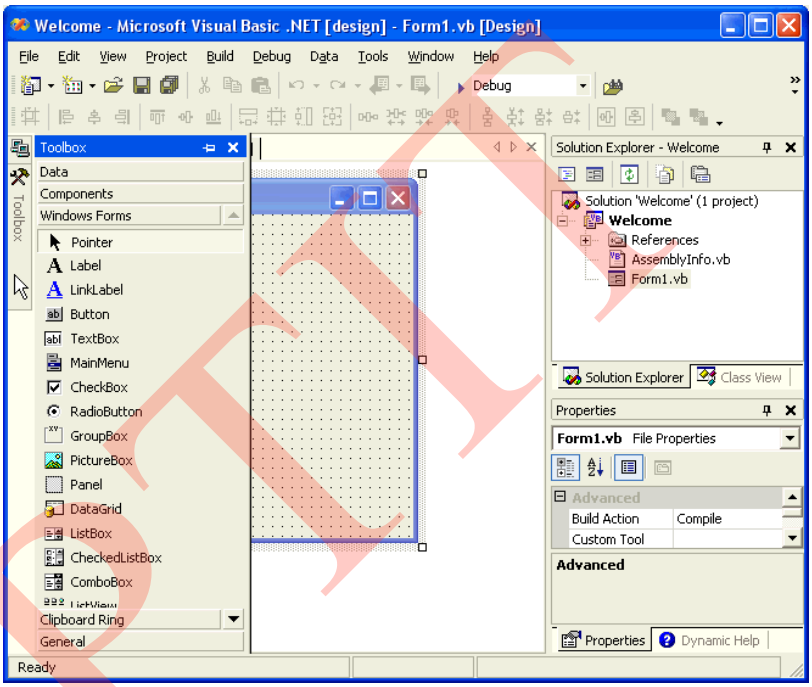
- New Project
- Add Item
- Open File
- Save (lưu trữ form hay module đang dùng)
- Save All (lưu trữ mọi forms, modules, ... đang dùng hay đang mở)
- Cut
- Copy
- Paste (sẽ hiển lộ sau khi ta nhấp nút Cut hay Copy)
- Undo
- Redo
- Navigate Backward (lướt lui)

- Navigate Forwards (lướt tới)
- Nút Start để chạy thử ứng dụng trong IDE
- Build Configuration (bố trí xây dựng ứng dụng) trong IDE. Ở đây, cho ta biết bố trí hiện dùng là Debug
- Truy tìm tập tin (Find in files)

và cuối cùng, nút Toolbar Options để hiển thị thêm các công cụ phụ thuộc khác.

Nhấp đơn hộp công cụ nằm phía bên tay trái window thiết kế như hình sau. Hộp công cụ bao gồm:

- Hộp Data
- Hộp Components
- Hộp Windows Forms
- Hộp Clipboard Ring
- Hộp Gerneral



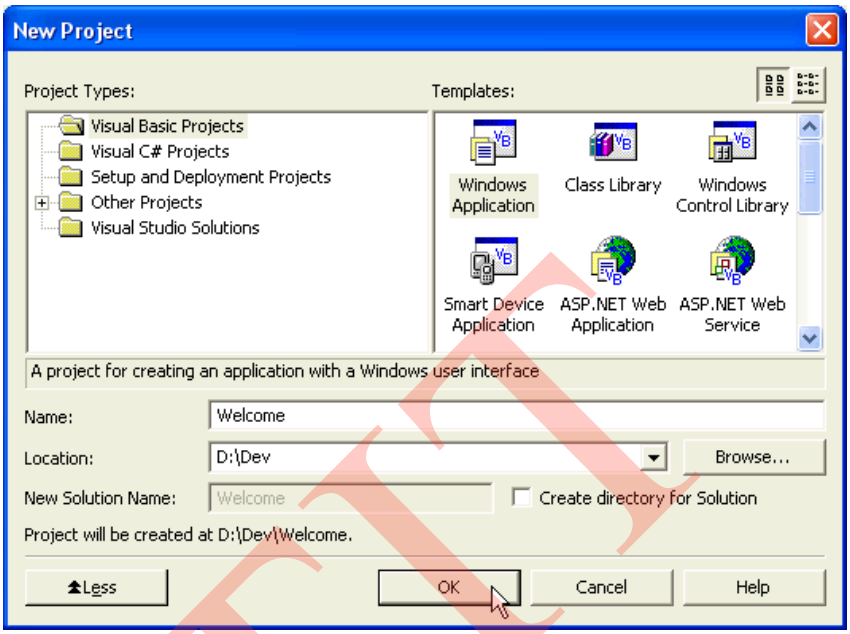
Hình 2.17: Màn hình soạn thảo

Bài kế tiếp, ta sẽ bắt đầu soạn dự án (project) đâu tiên với Microsoft Visual Studio.NET

CHƯƠNG 3: NGÔN NGỮ LẬP TRÌNH VISUAL BASIC.NET

3.1. Chương trình đầu tiên

Chúng ta bắt đầu với dự án (project) đầu tiên chào mừng các bạn đến với Visual Basic.NET. Như ta đã biết, dự án (project) Welcome được lưu trữ trong thư mục sau **D:\Dev\Welcome** như hình 2.1.



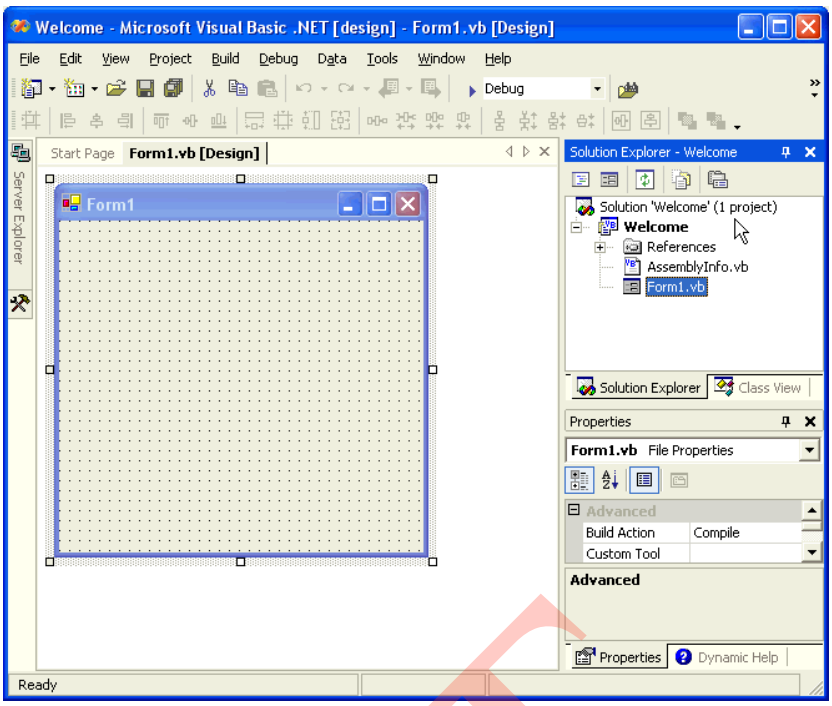
Hình 3.1. Mở dự án mới

Nhấp nút OK sẽ mở ra một cửa sổ dùng thiết kế một form trong nền Windows.

Dự án Welcome

Bước 1:

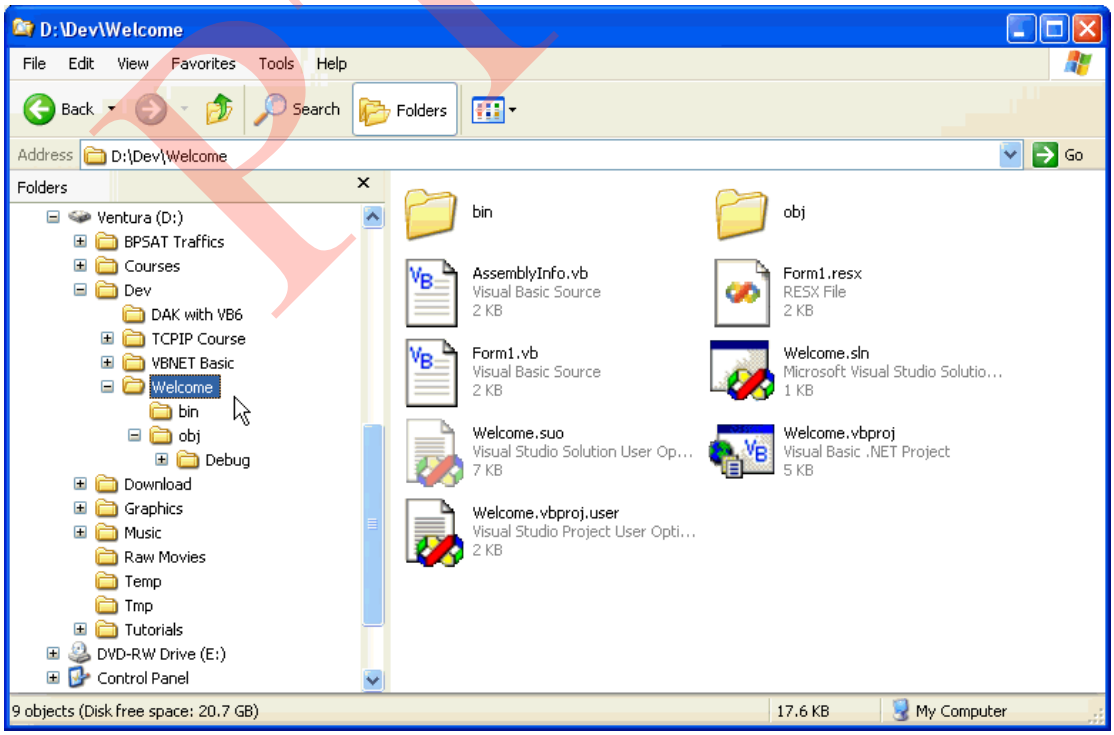
Microsoft Visual Studio.NET IDE khởi động dự án mới trong phương thức thiết kế (Design Mode) với 1 cửa sổ Form nằm ở giữa, tên mặc định là **Form1.vb**



Hình 3.2. Màn hình thiết kế

Nếu ta không lảm gì cả mà chỉ lưu trữ bằng cách chọn **File, Save All** và kiểm tra thư mục (folder) D:\Dev\Welcome, ta thấy Microsoft Visual Studio.NET tự động tạo ra và lưu trữ 1 số tập tin cần thiết trong đó có các tập tin **Welcome.sln** và **Welcome.vbproj** dùng đề quản lý dự án (project).

Thư mục (folder) **bin** là nơi lưu trữ dự án dưới hình thức ứng dụng (application) với phần đuôi là **.EXE** (ví dụ Welcome.exe) khi ta xây dựng dự án thành 1 ứng dụng (application) chạy ngoài IDE.



Hình 3.3. Các file tạo ra từ dự án

Bước 2:

- Đổi tên **Form1.vb** thành **Welcome.vb** bằng cách nhấp vào tên form ở **Solution Explorer Window** (nằm phía trên góc tay phải) hay ở hộp chữ **File Name** trong **Properties Windows** (phía dưới Solution Explorer)

Lưu ý: 1 solution có thể gồm nhiều dự án (project), 1 dự án (project) có thể gồm nhiều Forms khác nhau.

Lưu ý:

Khi đổi tên Form mặc định như vậy, ta phải bố trí **Startup Object** với tên **Welcome là object ta muốn khởi động đầu tiên** khi chạy dự án Welcome. Nếu không, dự án vẫn dùng Form1 và sẽ tạo lỗi vì Form1 đã đổi tên không còn hiện diện nữa.

- Đổi tên Form1 bằng cách chọn dự án Welcome trong **Solution Explorer** và chọn Properties.
- Chọn Welcome trong hộp chữ combo **Startup Object**.
- Nhấp nút Apply, OK

Bước 3:

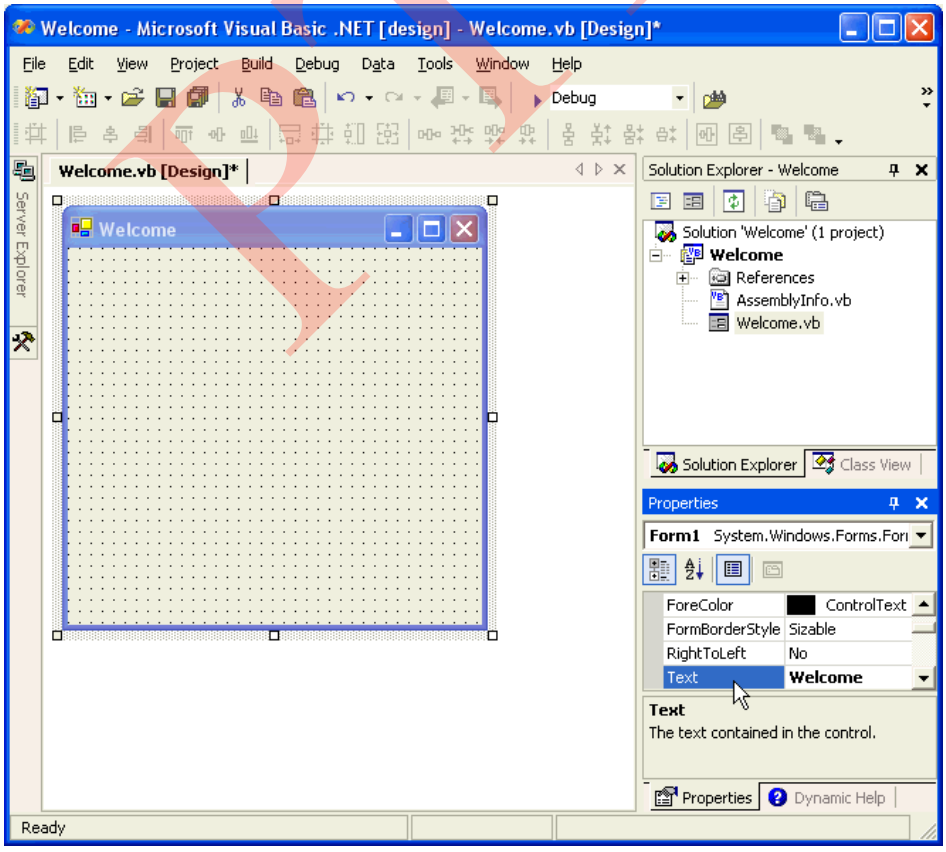
Nhấp vào Form hiển thị trong phần thiết kế.

Properties Window liên hệ thay đổi và hiển thị bảng đặc tính (properties) của Form. Bảng này sắp xếp và phân loại các đặc tính ra thành:

- **Accessibility**
- **Appearance**
- **Behaviour**

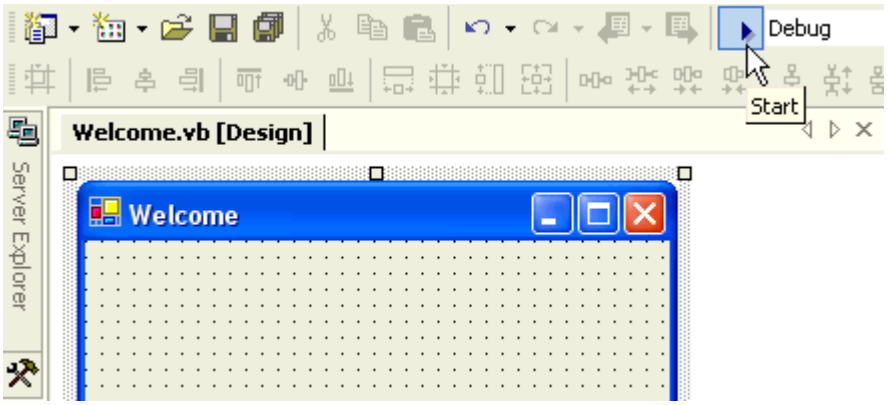
Nhằm giúp ta dễ dàng truy cập đặc tính cần đến.

Ta đổi tựa đề của Form từ Form1 ra Welcome bằng cách chọn đặc tính (property) **Text** và gỏ chữ Welcome.



Hình 3.4. Tiêu đề form

Ta có thể chọn nút Start để kiểm tra tựa đề của Form đã thay đổi theo ý hay không? Nút Start nằm ở Toolbar:



Hình 3.5. Chạy bằng nút start

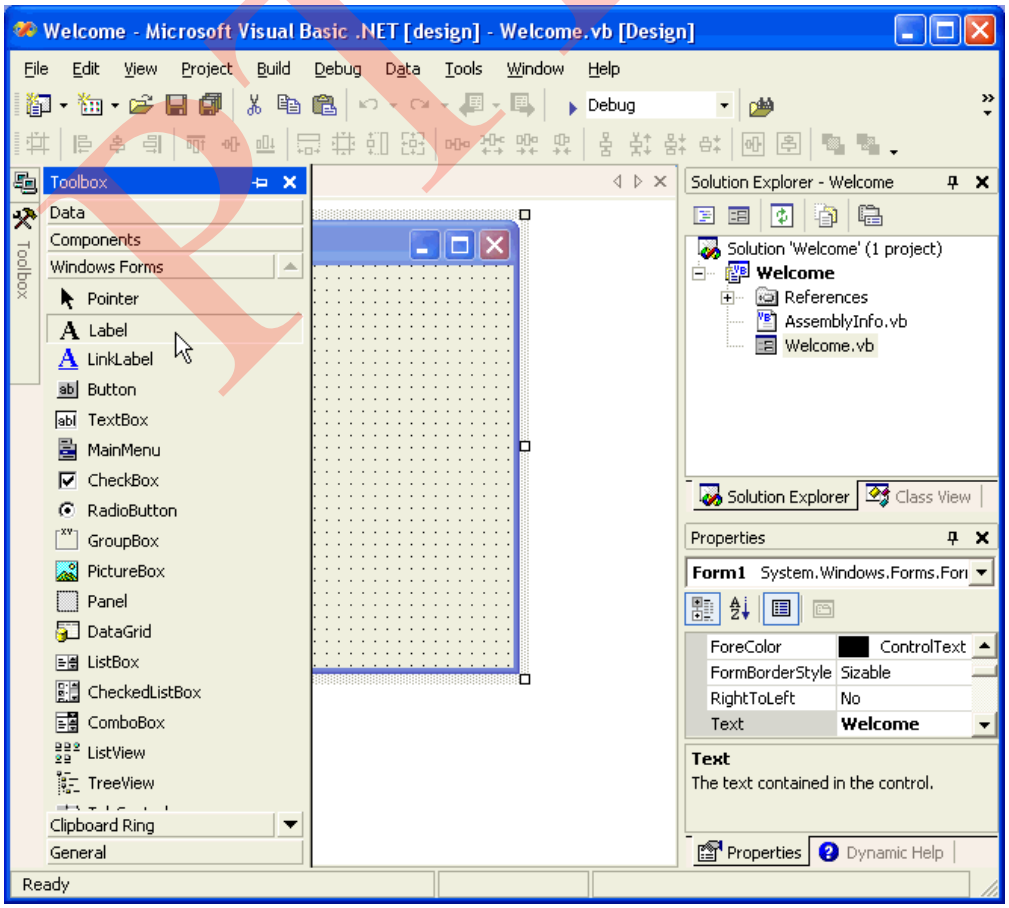
Như vậy, ta thấy Microsoft Visual Studio.NET IDE giúp ta tạo 1 Form dễ dàng như ...

Bước 4:

Từ Form 'Welcome' này, ta sẽ gắn:

- 1 nhãn hiệu (label) mang tựa 'Enter your name:"
- 1 hộp chữ để nhận dữ kiện từ user
- 1 nút mệnh lệnh 'Click Me' hiển thị hàng chữ 'Chào Mừng'
- 1 nút mệnh lệnh 'Exit' chấm dứt ứng dụng (application).

Mở Toolbox Window (nằm phía trái window thiết kế Form) và chọn công cụ **Label**. Hộp công cụ này chứa mọi đối tượng dùng tạo giao diện cũng như các công cụ phụ thuộc trong nền Windows.

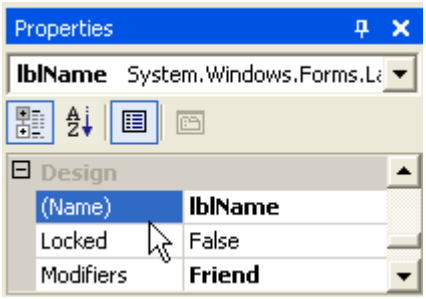


Hình 3.6. Công cụ thiết kế

Dùng mouse **kéo lê (click and drag)** 1 hình chữ nhật vừa đủ rộng nhằm chứa hàng chữ 'Enter your name:'. Nếu cần ta có thể điều chỉnh độ dài hay độ cao nhãn hiệu tùy ý.

Nhấp hộp chữ Text ở Properties Window và gỏ hàng chữ Enter your name:

Đặt tên nhãn hiệu này là **lblName** trong hộp chữ **(Name)** ở Properties Window:



Hình 3.7. Đặt tên

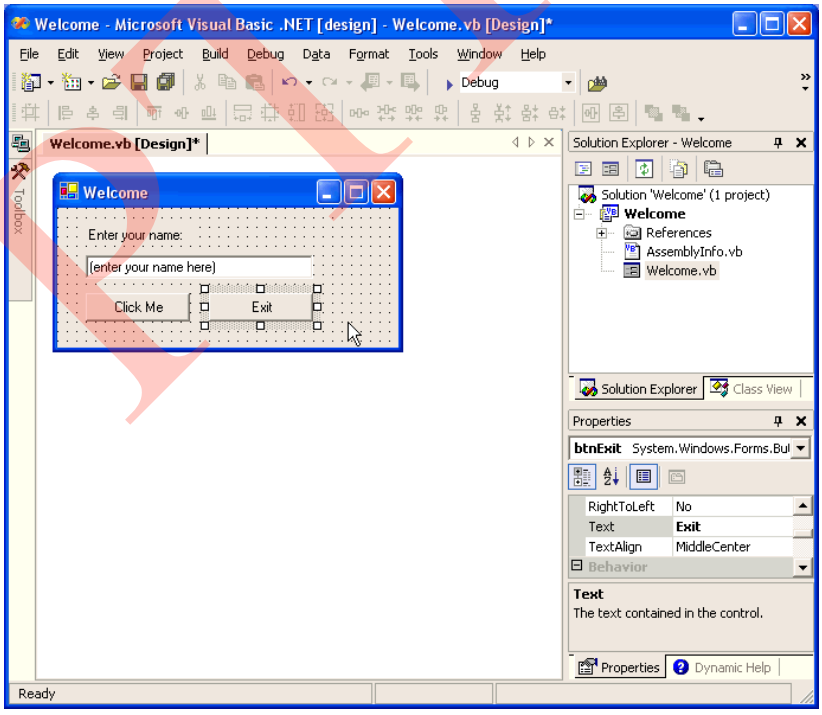
Bước 5:

Lập lại thao tác này cho các công cụ sau đây bằng cách chọn công cụ trong Toolbox và sau đó vẽ (click and drag) giao diện trên Form:

Bảng 3.1 Lựa chọn cho các đối tượng

Công cụ	Tên (Name)	Text
Textbox	tbxName	(enter your name here)
Button 1	btnClickMe	Click Me
Button 2	btnExit	Exit

Cuối cùng, ta sẽ có 1 giao diện cho ứng dụng (application) Welcome như sau:



Hình 3.8. Sau khi thiết kế

Thông thường, công ty nào cũng có tiêu chuẩn chung về danh pháp cho các hệ thống tin học, máy vi tính, thiết bị, công cụ hay nguồn mã. Để thống nhất lập trình với Visual Basic.NET (VB.NET) trong khóa học, ta có thể ấn định danh pháp cho các công cụ lập trình như sau:

Bảng 3.2: Các thuộc tính cho đối tượng form welcom

Công cụ	Tên đính kèm phía trước	Thí dụ:
Button	**btn** hoặc **cmd**	btnClickMe, cmdClickMe
ComboBox	**cbo**	cboContactName
CheckBox	**chk**	chkOver50
Label	**lbl**	lblTitle
ListBox	**lst**	lstProduct
MainMenu	**mnu**	mnuExtraOption
RadioButton	**rdb**	rdbYes
PictureBox	**pic**	picVovisoft
TextBox	**tbx**	tbxName

Như vậy, khi viết nguồn mã, mỗi lần gặp công cụ có tên đính kèm phiá trước là **tbx**, ta biết ngay đó là **Textbox**.

Bước 6:

Sau khi hoàn tất phần giao diện cho ứng dụng (application), ta cần thêm nguồn mã để xử lý các tình huống đặc biệt, tỷ như: nếu user nhấp vào nút Click Me thì chuyện gì sẽ xãy ra?

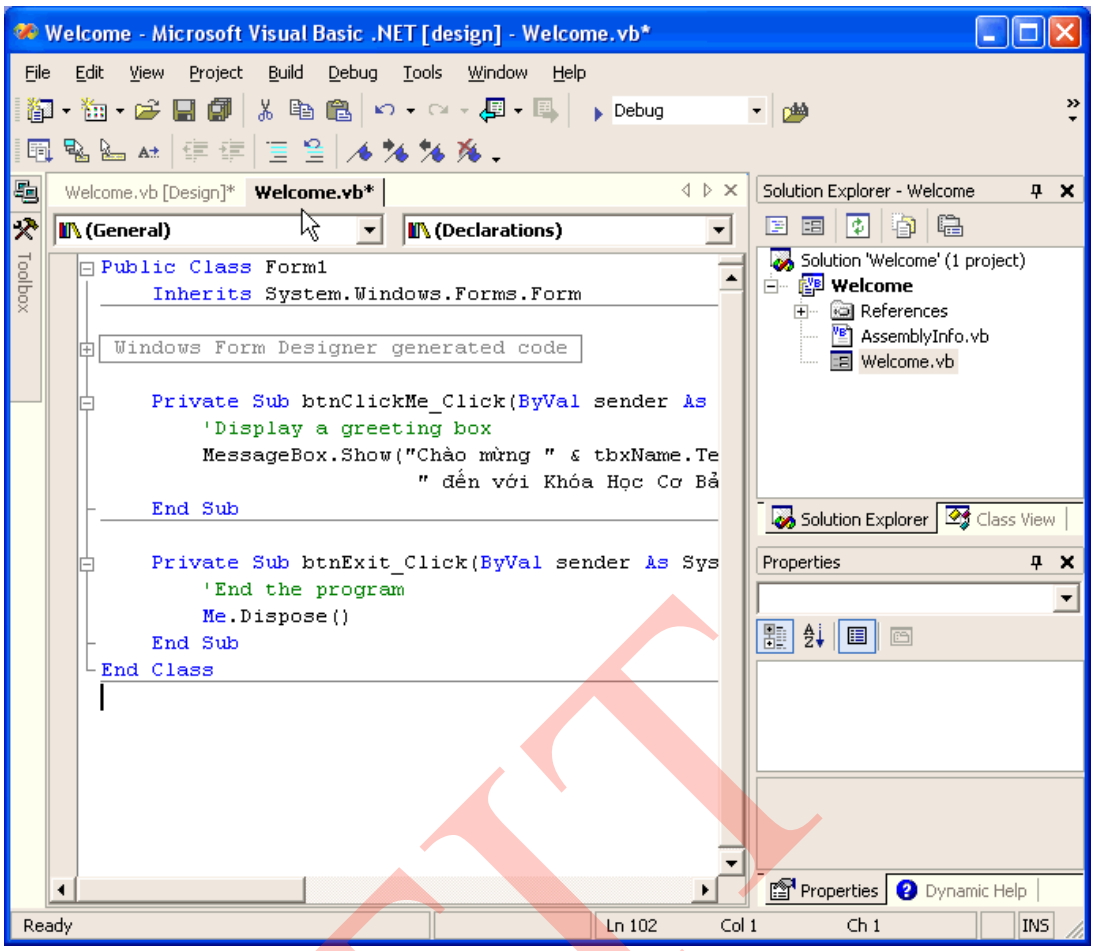
Code Editor sẽ giúp ta chuẩn bị nguồn mã. Thật vậy, khi ta nhấp đôi vào nút Click Me, Code Editor hiển thị nguồn mã tạo sẵn tổng quát cho mọi giao diện Windows và cho phép ta thêm mã vào phần **btnClickMe_Click**. Lưu ý ở đây, **Click là biến cố mặc định** khi user nhấp nút Click Me, Microsoft Visual Studio.NET chuẩn bị dùm ta 1 **Subroutine** để xử lý biến cố đó.

Lưu ý chỉ gỏ phần mã in đậm như sau:

```
Private Sub btnClickMe_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnClickMe.Click
'Display a greeting box
MessageBox.Show("Chào mừng " & tbxName.Text & _
                          " đến với Khóa Học Cơ Bản Visual
Basic.NET", "Welcome")
End Sub
```

Nhấp tab **Welcome.vb [Design]*** (kế bên tab Welcome.vb * có hình con trõ) để trở lại phần thiết kế Form, nhấp đôi nút Exit và gỏ mã:

```
'End the program
Me.Dispose()
```



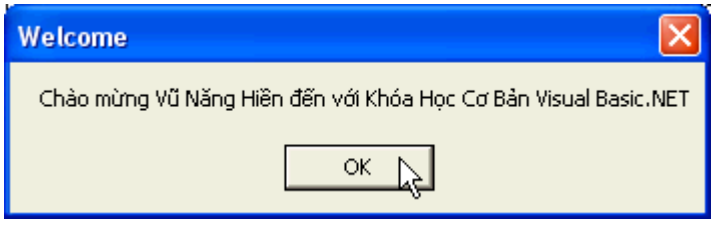
Hình 3.9. Màn hình code

Bước 7:

Nhấp nút Run để chạy thử ứng dụng trong môi trường IDE. Nhập tên vào hộp chữ dưới hàng 'Enter your name:'

Hình 3.10. Nhập liệu

Khi chọn nút Click Me, ứng dụng sẽ xử lý biến cố kích chuột đó và hiển thị 1 cửa sổ chào mừng:



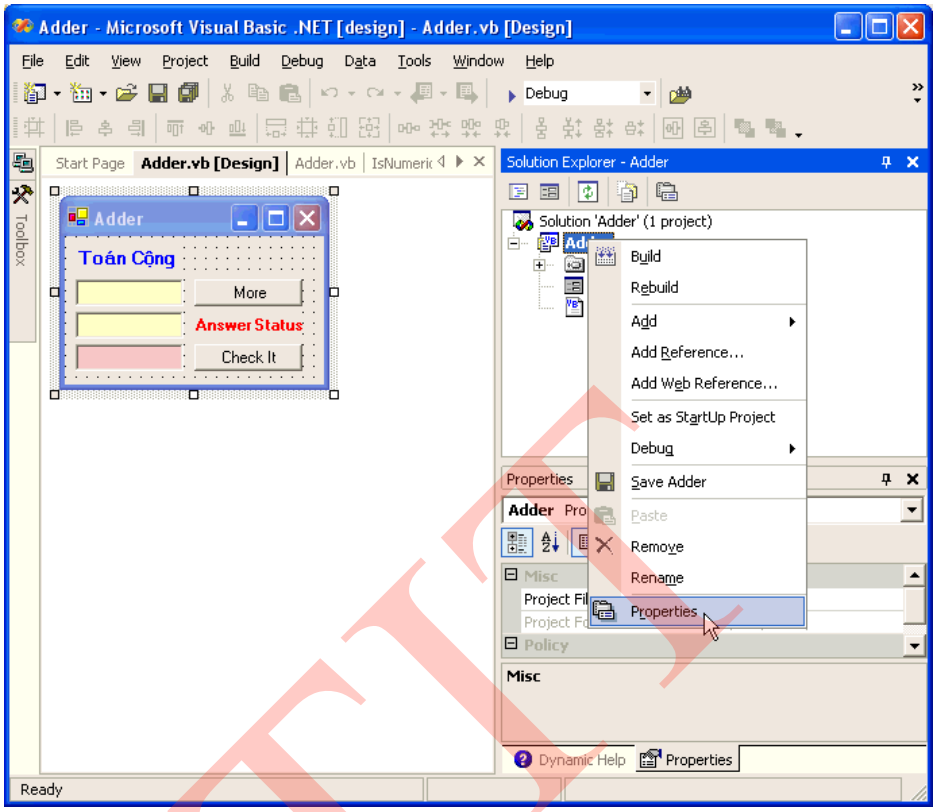
Hình 3.11. Kết quả

Như vậy, ta đã hoàn thành sứ mạng tạo 1 ứng dụng (application) đầu tiên dùng Microsoft Visual Studio.NET với ngôn ngữ lập trình Visual Basic.NET (VB.NET).

3.2. Mở rộng bài welcome

Bước 8:

Sau khi đi dạo một vòng làm quen với IDE của MS Visual Studio.NET, ta tiếp tục dự án Adder với giao diện sau:



Hình 3.12. Thiết kế chương trình

Dùng (bằng cách kéo thả - Click and Drag hay Click and Draw) các thiết bị trong hộp công cụ (Toolbox) vào Form1 và bố trí như sau:

Lưu ý, ở đây chỉ hướng dẫn và trình bày chi tiết phương pháp dùng và bố trí đặc tính (property) của 1 thiết bị trong hộp công cụ mà thôi. Sau đó, các bạn áp dụng tương tự như vậy với các thiết bị khác.

Thí dụ dùng và trình bày tiêu đề (lable) **Toán Cộng** như sau:

- Nhấp hộp công cụ (phía bên trái IDE) và nhấp đơn thiết bị **Label** (Click ...)
- Vẽ (... and Draw) 1 hình chữ nhật trong mặt trống của Form
- Chọn **Properties Window** của Label (để ý label có được chọn hay không, nếu không, ta có thể mở nhằm properties window của một thiết bị nào khác chứ không phải thiết bị ta muốn bố trí)
- Chọn **đặc tính (property) Text** va gõ hàng chữ Toán Cộng (có thể dùng ứng dụng **VPSKeys** với bố trí **Unicode** hoặc các ứng dụng gõ tiếng Việt tương đương)
- Chọn và mở rộng đặc tính (property) Fonts và thay đổi cở chữ và màu tùy ý.
- Chọn Name và đặt tên theo tiêu chuẩn định trước, tỷ như: **lblTitle** với **lbl** là chữ viết tắc của label cộng với tên của tiêu đề.
- Kéo lê (Click and Drag) thiết bị này đến vị trí tùy ý trong Form, tỷ như: vị trí phía trên bên trái như hình trình bày.

Áp dụng linh động hướng dẫn trên cho các thiết bị textbox, button, ... như sau:

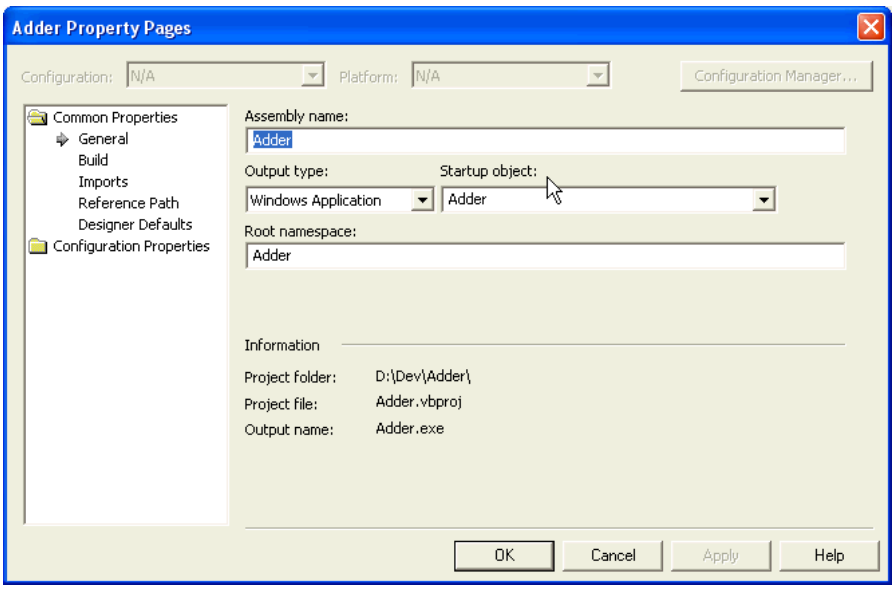
Bảng 3.3: Thuộc tính cho các đối tượng của form tính tổng

Công cụ	Bố trí đặc tính (property)
textbox1	Name = TbxNumber1 Text = (để trống ở đây) Text Align = Right BackColor = (tùy ý)
textbox2	Name = TbxNumber2 Text = (để trống ở đây) Text Align = Right BackColor = (tùy ý)
textbox3	Name = TbxNumber1 Text = (để trống ở đây) Text Align = Right BackColor = (tùy ý) ForeColor = Red
button1	Name = cmdMore Text = More
button2	Name = cmdCheckIt Text = Check It
label2	Name = lblResult Text = Answers Status TextAlign = MiddleCenter

Tuy ta có thể giữ tên mặc định Form1 trong dự án Adder nhưng có vẻ không chuyên nghiệp bằng đổi tên mặc định Form1 đó thành tên Adder thích hợp với dự án.

Lưu ý: khi đổi tên Form mặc định như vậy, ta phải bố trí **Startup Object** với tên **Adder là object ta muốn khởi động đầu tiên** khi chạy dự án Adder. Nếu không, dự án vẫn dùng Form1 và sẽ tạo lỗi vì Form1 đã đổi tên không còn hiện diện nữa.

- Đổi tên Form1 bằng cách chọn dự án Adder trong Solution Explorer và chọn Properties.
- Chọn Adder trong hộp chữ combo **Startup Object**.
- Nhấp nút Apply, OK



Hình 3.13. Thêm thuộc tính

Bước 10: Lập trình theo kiểu mẫu event - driven

Khi dùng MS Visual Studio.NET làm môi trường lập trình với Visual Basic.NET (VB.NET), thường thường ta tạo một giao diện (dưới hình thức Form) trước và sau đó gài nguồn mã vào, tỷ như: nhấp đôi nút **Check It** để mở tập tin chứa nguồn mã với tên mặc định là tên của dự án. Trước tiên, MS Visual Studio.NET sẽ tạo nguồn mã mặc định với các công dụng cơ bản yểm trợ giao diện ta vừa thiết kế (Form Adder) và ta sẽ cộng thêm mã để bố trí và kế hoạch sẵn mọi tình huống có thể xảy ra hầu hành động kịp thời tùy theo biến cố mà Form nhận được (thí dụ: người dùng nhấn vào nút Check It để kiểm tra bài toán cộng trong ứng dụng Adder). Kiểu chuẩn bị với nguồn mạ định trước như vậy được gọi là lập trình theo kiểu mẫu Event-Driven.

Bây giờ, ta bắt đầu thêm nguồn mã xử lý biến cố **Click** của nút **Check It** như sau:

- Nhấp đôi vào Form, IDE sẽ dùng **Designer Code Generator** tạo phần nguồn mã với tập tin **Adder.vb**
- Nguồn mã bắt đầu với Public Class Adder.
- Nhấp vào **tab** mang tên **Adder.vb [Design]** để trở về giao diện Form Adder. (Lưu ý hình con trỏ chỉ các tab trong IDE từ Satrt Page, Adder.vb [Design] và Adder.vb)
- Nhấp đôi vào nút **Check It** để mở phần nguồn mã của nút này với biến cố **Click**
- Gỏ nguồn mã sau đây phía dưới hàng **Private Sub cmdCheckIt_Click** (nhắc lại, **cmdCheckIt** là tên ta đặt cho nút **Check It** trong phần giao diện Form Adder): mã này kiểm tra xem ta đưa 1 giải đáp với con số hay chữ vào hộp chữ **tbxResult**? Nếu là con số, mã sẽ so sánh con số đó với kết quả bài toán cộng và báo cáo lại trong phần nhản hiệu **lblResult**.

```
Dim resultNumber As Integer
    If IsNumeric(tbxResult.Text) Then
    resultNumber = CInt(tbxNumber1.Text) + CInt(tbxNumber2.Text)
    If CInt(tbxResult.Text) = resultNumber Then
      lblResult.Text = "Correct"
    Else
      lblResult.Text = "Wrong"
    End If
```

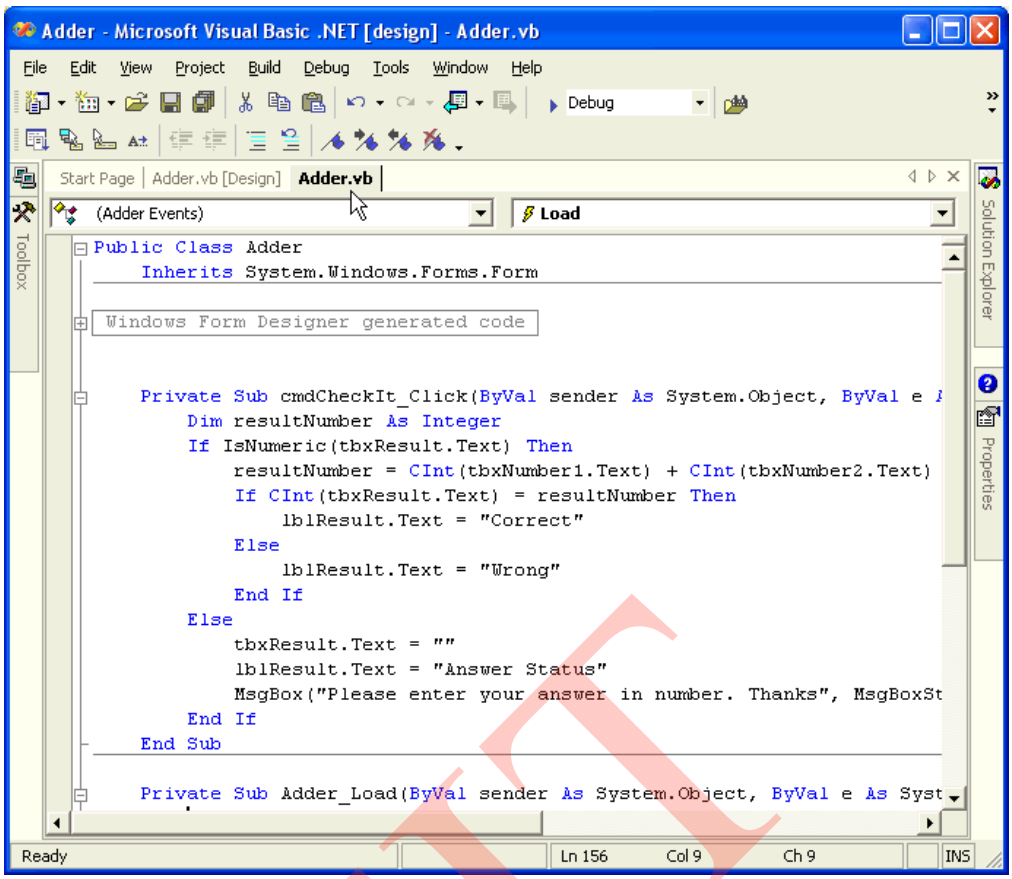
```
    Else
      tbxResult.Text = ""
      lblResult.Text = "Answer Status"
      MsgBox("Please enter your answer in number. Thanks",
MsgBoxStyle.Information, "Warning")
      End If
```

Tương tự, trở về phần thiết kế Form:

- Nhấp đôi vào chổ trống của Form cho nguồn mã Adder_Load
- Nhấp đôi vào nút More cho nguồn mã cmdMore_Click
- Gỏ nguồn mã cho Subroutine (sẽ học cách tạo Subroutine và Function ở các bài kế) SetRandomNumber. Mã ở đây tạo 2 con số ngẫu nhiên từ 1 đến 10000 cho bài toán cộng khi chạy ứng dụng Adder trong phần **Adder_Load** và trong nút **More**.

```
        Private Sub Adder_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
            SetRandomNumber()
        End Sub
        Private Sub cmdMore_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdMore.Click
            SetRandomNumber()
        End Sub
        Private Sub SetRandomNumber()
            Dim firstNumber, secondNumber As Integer
            Randomize()
            firstNumber = CInt(Int((10000 - 0 + 1) * Rnd() + 0))
            secondNumber = CInt(Int((10000 - 0 + 1) * Rnd() + 0))
            tbxNumber1.Text = firstNumber
            tbxNumber2.Text = secondNumber
        End Sub
```

Hình đặc trưng nguồn mã của dự án Adder:



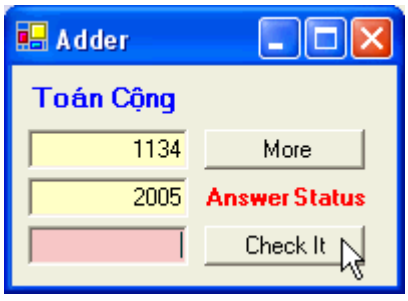
Hình 3.14. Viết code

Bước 11:

Nhấp nút Run (như hình dưới đây) để chạy ứng dụng (application) Adder trong môi trường IDE:

Hình 3.15. Thanh menu

Ta thấy bài toán cộng được hình thành với 2 số ngẫu nhiên và chờ ta gỏ vào giải đáp trong hộp chữ kế bên bút Check It. Sau đó, ta nhấp nút này để kiểm tra kết quả. Khi nào muốn làm lại bài toán cộng này, nhấp nút More:



Hình 3.16. Chạy chương trình

Lưu ý: MS Visual Studio.NET tạo một executable file mặc định là Adder.exe dưới một ngăn chứa cấp dưới (subfolder) **BIN.** Tập tin này là ứng dụng Adder tạo ra bởi dự án Adder.

Bước 12:

Lưu trữ mọi tập tin với thực đơn **File, Save All**

3.3. Dữ liệu và biến

Thông tin (Information) diễn tả một sự việc nào đó dưới nhiều hình thức khác nhau, tỷ như: tin tức trên báo, tin nhận được từ ký giả viết tay trên giấy, sự cố báo cáo trên TV, ... khác với dữ kiện (Data) dùng diễn tả thông tin đã được kiểm tra, đối chiếu, so sánh, xếp loại theo thứ tự và quan trọng hơn cả là được tổ chức để dùng trong một ứng dụng (application) điện toán. Do đó, thông tin được ghi chú ở các sổ tay không thể là dữ kiện mà một ứng dụng (application) nào đó có thể dùng được. Nếu muốn dùng thông tin như vậy, ta phải chuyển đổi qua hình thức dữ kiện, tỷ như: rà (scan) hay nhập (enter) vào 1 trang kế toán của MS Excel để có thể phân tích kết quả thu lượm.

Mặc dù, Công Nghệ Tin Học đã phát triển và thay đổi nhanh chóng nhưng tiến trình xử lý và phát triển nhu liệu hầu như vẫn ... 'trước sau như một', nghĩa là không đổi gì cả. Ở đây, ta muốn nói đến **phương thức cơ bản** cho phát triển và giải đáp vấn đề cho việc lập trình. Anh Ngữ gọi là **Algorithm**. Algorithm đó là:

Trước khi ta viết nhu liệu giải quyết một vấn đề nào đó, ta phải phân ra (phân tích) thành những phần nhỏ hơn tùy từng trường hợp một để diễn tả cách giải quyết vấn đề và sau cùng tổng hợp lại. Tóm lại, đây là một **phương thức phân tích tổng hợp**. Nếu không áp dụng phương thức này, vấn đề xem có vẻ như ... 'rối tung lên' không thể giải quyết được.

Bây giờ, tưởng tượng bạn đang làm việc cho một ông ty viễn thông. Vấn đề đặt ra là làm sao cung cấp được hoá đơn tính tiền điện thoại mà khách hành đã dùng. Ta phải bắt đầu từ đâu? Làm gì trước, làm gì sau? Hoá đơn như thế nào? ...

Phương thức cần có là chia vấn đề thành những phần việc nhỏ và truy cập cách giải quyết phần việc đó, giả dụ như:

- Vào mỗi đầu tháng, ta sẽ cung cấp hoá đơn đến mỗi khách hàng.
- Cho mỗi khách hành, ta cần một bảng liệt kê các cú gọi đi trong tháng.
- Ta cần biết khoãng thời gian dùng cho mỗi cú điện thoại? lúc gọi? trong tuần hay cuối tuần? ban ngày hay ban đêm? để tính toán chi phí mỗi cú điện thoại.
- Trong từng hoá đơn một, ta tổng kết chi phí các cú điện thoại (dưới tiêu đề nội địa, ngoại quốc hay mobile, ...).
- Trong các dịp lễ lạc hay khuyến mãi, bao nhiêu phần trăm hạ giá?
- Ta cần cộng thêm tiền thuế bán dịch vụ cho mỗi hoá đơn.
- Sau khi tổng hợp lại, in ra và gởi hoá đơn đến khách hàng.

Như vậy, ta thấy phân tích để giải quyết vấn đề khi viết nhu liệu, ta hoàn toàn không để ý hay làm gì dính dáng tới ngôn ngữ lập trình. Thật sự, đây là mấu chốt quan trọng nhất của một chuyên gia lập trình chuyên nghiệp. Nếu không, ta chỉ là ... thiên lôi, ai sai đâu thì ... đánh đó, không thể tự mình đưa giải đáp cho các trở ngại nêu ra trong khi chuẩn bị thiết kế và phát triển một ứng dụng (application). Nên làm chuyên gia lập trình chứ đừng ngừng lại ở ... 'người viết mã' mà thôi.

Việc còn lại là chọn cho mình một ngôn ngữ lập trình hùng mạnh đủ khả năng phát triển các giải đáp cho mọi trở ngại: Visual Basic.NET (VB.NET).

Một cách tổng quát, ngôn ngữ lập trình chỉ gồm các **biến số (variables) và cách thức (methods)**. Ngôn ngữ lập trình dù phức tạp đến đâu thì cũng được xây dựng trên các biến số và cách thức mà thôi. Do đó, ta không thể so sánh ngôn ngữ lập trình này mạnh hơn hay yếu hơn, nhất là các ngôn ngữ lập trình .NET như Visual Basic.NET (VB.NET) hay C# hay C++.

Trên thực tế, các ngôn ngữ lập trình .NET đều được biên dịch ra một ngôn ngữ trung gian là MSIL (Microsoft Intermediate Language).

Ngôn ngữ lập trình chỉ là công cụ phụ giúp công việc của ta và chắc chắn sẽ thay đổi trong tương lai.

3.4. Biến số (Variable)

Biến số (Variable) dùng chứa một giá trị nào đó trong phương thức lập trình (algorithm). Ta có thể làm một quyết định dựa trên giá trị đó, ví dụ: giá trị đó bằng 9 không? hay nhỏ hơn 7? hay có thể thực hiện các thuật toán trên giá trị đó như cộng, trừ, nhân, chia, ...

Xét phương thức lập trình (algorithm) sau:

- Tạo 1 biến số đặt tên là 'count'
- Trong biến số count, chứa giá trị 35
- Cộng thêm 1 vào biến số count
- Hiển thị giá trị của biến số count trên màn hình (monitor)

Như vậy, ta phải tuyên bố biến số (variables) count, cho vào giá trị 35, cộng 1 thành 36 và hiển thị số 36 trên màn hình.

Trong Visual Basic.NET (VB.NET), dùng **Dim** và **Redim** tuyên bố biến số như sau:

```
Dim myVariable As Long
Dim myArray (5) As Integer
Dim yourArray ( ) As String = {"Dần", "Thân", "Tỵ", "Hợi",
"Tứ Hành Xung"}
Redim myArray (10) As Integer
```

Giải thích:

Dùng **Dim** tuyên bố (hay tuyên cáo) biến số myVariable thuộc loại dữ kiện **Long**. Redim để tuyên bố lại, nhất là khi thay đổi cở của **Array**. myArray (5) là một chuỗi biến số gồm 6 số bắt đầu từ số 0 với myARray (0), myArray (1), ... đến myArray (5) loại dữ kiện số nguyên (Integer). yourArray ( ) dùng giá trị bên trong dấu { } để xác định cở (ở đây, cở = 5, chỉ số hay 'index' **bắt đầu từ số 0**, 1, 2, 3, 4).

Array:

Array dùng chỉ số (index) để lưu trữ nhiều giá trị dưới cùng một tên biến số (variables), tỷ như:

```
Dim yourArray ( ) As String = {"Dần", "Thân", "Tỵ", "Hợi",
"Tứ Hành Xung"}
Dim strMonths ( ) As String = {"Giêng", "Hai", "Ba", "Tư",
"Năm", "Sáu", "Bảy", "Tám", "Chín", "Mười", "Mười Một",
"Chạp"}
Dim empRecords (100)
```

3.4.1. Chú thích

Trình biên dịch Visual Basic.NET (VB.NET Compiler) bỏ qua không biên dịch các phần **comments**, do đó ta có thể chú thích thêm phần dẫn giải hay phương thức giải quyết vấn đề cho từng nguồn mã. Chuyên nghiệp nhất là ghi lại algorithm của ta để các lập trình viên khác hay ... cả chính ta có thể hiểu mã ta đã viết .. từ nhiều tháng trước. Nhớ là con người

cũng ... 'mau quên lạ lùng'. Trên thực tế, chính ta cũng không biết ta viết .... cái gì nếu đọc lại mã sau ... chừng vài tháng.

Trong Visual Basic.NET (VB.NET), đánh dấu nơi ghi chú thích với **dấu ' (dấu apostrophe)** , tỷ như:

```
        'tạo biến số count và chứa giá trị 35
    Dim count As Integer
    count = 35
         'cộng thêm 1 vào count
    count = count + 1
         'hiển thị giá trị của count
    MessageBox.Show ("Value of count is now " & count)
```

Whitespace cũng quan trọng không kém. Việc chừa các khoãng trống như vậy nhằm cho nguồn mã được đọc dễ dàng. Thường thường, ta nên chừa một hàng trống giữa các bước trong phươn gthức lập trình (algorithm) như thí dụ trình bày ở trên, ta thấy có hàng trống sau hàng count = 35.

3.4.2. Loại dữ liệu (Data Types)

Khi dùng biến số (variables), ta cần biết và bố trí trước biến số đó lưu trữ loại dữ kiện (data types) nào, điều này giúp ích máy vi tính xử lý tài nguyên dễ dàng hơn trong lúc chạy ứng dụng (application).

Tổng quát, các loại dữ kiện (data types) bao gồm:

Bảng 3.4: Số nguyên (Number)

Loại dữ kiện	**Cở (Size)**	**Range**	**Chú thích**
Byte	1 byte	0 tới 255	Byte = 8 bits trong hệ thống nhị phân. Byte không yểm trợ số âm (negative number).
Short	2 bytes	-32,768 tới 32,768	Rất tiện lợi cho các biến số (variables) lưu trữ số nguyên cở nhỏ.
Integer	4 bytes	-2,147,483,648 tới 2,147,483,647	Tiêu chuẩn số nguyên. Loại dữ kiện này được máy vi tính xử lý nhanh nhất và ít tài nguyên nhất.
Long	8 bytes	9,223,372,036,854,775,808 tới 9,223,372,036,854,775,808	Đây là số nguyên lớn từ -9 quintillion tới 9 quintillion (-9 x 10^{18} tới +9 x 10^{18})

Bảng 3.5: Số thực (Decimal Number)

Loại dữ kiện	**Cở (Size)**	**Range**	**Chú thích**
Single	4 bytes	Cho số âm: -3.402823 x 10^{-38} tới -1.401298 x 10^{-45}. Cho số dương: 1.401298 x 10^{-45} tới 3.402823 x 10^{38}.	Đây là số thực vô cùng nhỏ hay vô cùng lớn.

Double	8 bytes	Cho số âm: -1.79769313486231 x 10^{308} tới -4.94065645841247 x 10^{-324}. Cho số dương: 4.94065645841247 x 10^{-324} tới 1.79769313486231 x 10^{308}.	Double còn gọi là loại dữ kiện 'double prescision floating point' do có thể lưu trữ số lẻ gấp đôi loại 'single', tức là 15 số lẻ sau '**decimal point**'.

Bảng 3.6: Chữ và hàng chữ (hay câu)

Loại dữ kiện	Cở (Size)	Range	Chú thích
Char	2 bytes	Một chữ	Dùng lưu trữ từng chữ một.
String	10 bytes + 2 bytes cho mỗi chữ (character)	Hàng chữ có thể kéo dài tới 2 tỷ (billion) chữ	Dùng lưu trữ một hàng chữ hay cả nguyên một cuốn sách.

Bảng 3.7: Các loại đơn giản khác

Loại dữ kiện	Cở (Size)	Range	Chú thích
Boolean	2 bytes	True hoặc False	VB.NET dùng 2 bytes cho số 0 (False) và 1 (True).
Date	8 bytes	Từ ngày 1 tháng Giêng năm 100 tới ngày 31 tháng Chạp năm 9999	Loại dữ kiện có khả năng tính toán năm nhuần. Nếu ta cộng 1 ngày vào biến số lưu trữ ngày 28/02/2000, ta sẽ có 29/02/2000 nhưng nếu cộng cho ngày 28/02/2001, ta lại có 01/03/2001.

3.4.3. Hằng số (Contants)

Trái với biến số (variables), hằng số không thay đổi giá trị trong suốt đời sống của ứng dụng (application). Ta dùng **Const** để tuyên bố hằng số, tỷ như:

```
Const PI = 3.1416 As Double
Const DSN As String = "MyDatabaseName"
```

3.5. Tên

Thông thường, ta thoả thuận một danh pháp chung khi đặt tên các biến số (variables) hay hằng số, nếu không, chính ta sau này có thể mất công tìm hiểu loại các biến số hay hằng số trong ứng dụng (application). Quy ước tổng quát khi đặt tên bao gồm 2 phần:

- Tiền tố (Prefix): thường dùng chữ in thường chỉ loại biến số (variables) hay hằng số (constant).
- Tên: chữ đầu tiên dùng chữ Hoa và tên phải đầy đủ ý nghĩa để khỏi mất công tham khảo sau này.

Đề nghị tên quy ước như sau:

Bảng 3.8: Qui ước đặt tên

Loại dữ kiện	Tiền Tố (Prefix)	Thí dụ
Byte	Byt	bytAge
Short	Sht	shtCounter

Integer	Int	intCounter
Long	Lng	lngResolution
Single	Sng	sngInterestRate
Double	Dbl	dblTotalSalesInYear
Char	Chr	chrMiddleName
String	Str	strAddress
Boolean	Bol	bolIsCompleted
Date	Dte	dteHireDate
(User-defined types)	(vài chữ - 2 hay 3 chữ trong tên của structure)	empRecord
Constant	(no prefix, chữ nối nhau bằng dấu _ **underscores**)	TAX_RATE
Enumerations	(vài chữ - 2 hay 3 chữ trong tên của Enumerations)	dteWeekday, dowWeekday, colBackGroundColor

3.6. Phương thức (method)

Method là nguồn mã độc lập (self-contained) dùng để thực hiện công việc ta muốn làm trong ứng dụng (application). Method rất quan trọng vì:

- Phân giải (break-up) chương trình thành các phần tử nhỏ hơn, có trách nhiệm rõ ràng, đơn giản hơn và dễ hiểu.
- Khuyến khích dùng lại nguồn mã (reusable code)

Ta phân biệt 2 loại mehods:

- `Subroutine: với Sub ... End Sub`
- `Function: với Function ... End Function`

Dùng Subroutine khi ta muốn thực hiện công việc gì đó và Function khi muốn nhận kết quả trả về.

Thí dụ dùng subroutine 'SetRandomNumber' trong bài 5:

Bố trí 2 số ngẫu nhiên vào hộp chữ nhưng không trả về giá trị nào.

```
Sub SetRandomNumber()
  Dim firstNumber, secondNumber As Integer
  Randomize()
  firstNumber = CInt(Int((10000 - 0 + 1) * Rnd() + 0))
  secondNumber = CInt(Int((10000 - 0 + 1) * Rnd() + 0))
  tbxNumber1.Text = firstNumber
  tbxNumber2.Text = secondNumber
End Sub
```

Thí dụ dùng Function:

Cộng 2 số và trả về kết quả bài toán cộng.

```
Function Addition (ByVal Number1 As Integer, ByVal Number2 As Integer)
    Addition = Number1 + Number2
End Function
```

Tên methods:

Thường thường, ta đặt tên quy ước cho methods bằng động từ chỉ công việc thực hiện và tên phải đầy đủ ý nghĩa, Ví dụ:

```
GetCustomerName
OpenCustomerRecord
CalculateRepaymentPerMonth
ReadXMLFile
GetEnvironementVariables
SaveMyNetworkConfiguration
```

3.7. Phạm vi (scope)

Phương thức (method) là nguồn mã chạy độc lập, do đó các biến số (variables) được khai báo trong một method chỉ có ý nghĩa khi dùng trong method đó mà thôi. Ta gọi là trong phạm vi method (**scope)**. Biến số (variables) dùng trong method này không có ảnh hưởng gì đến biến số (variables) trong method khác, ví dụ:

```
Sub DisplayMyName
      Dim strName
      strName = "Vũ Năng Hiền"
      MessageBox.Show(strName)
  End Sub
  Sub DisplayYourName
      Dim strName
      strName = "Đặng Quang Lương"
      MessageBox.Show(strName)
  End Sub
```

Ta nhận thấy 2 subroutine có cùng 1 biến số (variables) **strName** nhưng giá trị 2 biến số (variables) này khác nhau. Thay đổi giá trị biến số **strName** trong subroutine **DisplayMyName** không làm thay đổi giá trị biến số strName trong subroutine **DisplayYourName**.

CHƯƠNG 4: CẤU TRÚC LỆNH

4.1. Lệnh điều kiện

Quyết định kiểu **Conditional Logic** trong nguồn mã gồm 2 loại:

- Loại để tìm hiểu ta đang xử lý phần nào trong giải thuật hoặc đối phó với các trở ngại định trước hay bất ngờ. Thí dụ như: ta muốn mở 10 tập tin để đọc, trước hết ta cần kiểm tra xem tập tin có hiện diện hay không? Nếu không, ta sẽ phải làm gì? Nếu có, ta cần kiểm tra xem khi nào tập tin đó được đọc hết? Sau đó, ta sẽ lập lại các bước kiểm tra đó với tập tin kế tiếp.
- Loại để thi hành các phần khác nhau trong giải thuật dựa trên dữ kiện nào đó. Thí dụ như: ta muốn gởi email đến 10 khách hàng trong danh sách với điều kiện khách hàng đó có máy vi tính, nếu không ta phải điện thoại hay in thư gởi đến khách hàng.

Các cú pháp (syntax) khi dùng lệnh if như sau:

Cú pháp 1:

Cách đơn giản nhất khi làm một quyết định.

```
If (điều kiện) Then
    (mã thi hành nếu điều kiện thỏa mãn, nghĩa là = True)
End If
```

Thí dụ:

```
'Tuyên bố biến số và giá trị cho intYourAge
  Dim intYourAge
  intYourAge = 70
  'Quyết định và công bố kết quả
  If intYourAge => 65 Then
      MessageBox.Show ("You should retire")
  End If
```

Cú pháp 2:

Quyết định với 2 tình trạng trái ngược nhau.

```
If (điều kiện) Then
   (mã thi hành nếu điều kiện thỏa mãn, nghĩa là = True)
Else
    (mã thi hành nếu điều kiện không được thỏa mãn, nghĩa
là = False)
End If
```

Thí dụ:

```
    'Tuyên bố biến số và giá trị cho intYourAge
    Dim intYourAge
    intYourAge = 40
    'Quyết định và công bố kết quả
    If intYourAge => 65 Then
        MessageBox.Show ("You should retire.")
    Else
       MessageBox.Show ("You are too young to retire.")
```

```
        End If
```

Cú pháp 3:

Quyết định với nhiều tình trạng thay đổi hay khác nhau.

```
 If (điều kiện 1) Then
      (mã thi hành nếu điều kiện 1 thỏa mãn, nghĩa là điều
 kiện 1 = True)
 ElseIf (điều kiện 2) Then
     (mã thi hành nếu điều kiện 2 được thỏa mãn, nghĩa là
 điều kiện 2 = True)
 Else
     (mã thi hành khi không điều kiện nào thỏa mãn, nghĩa
 là điều kiện 1 và điều kiện 2 đều = False)
 End If
```

Thí dụ:

```
    'Tuyên bố biến số và giá trị cho intYourAge
    Dim intYourAge
    intYourAge = 50
    'Quyết định và công bố kết quả
    If intYourAge => 65 Then
      MessageBox.Show ("You should retire.")
      ElseIf intYourAge => 45 Then
      MessageBox.Show ("You should pay more to your current
scheme.")
     Else
        MessageBox.Show ("You are too young to worry about.")
     End If
```

Cú pháp 4:

Quyết định trong quyết định (**Nested If**).

```
 If (điều kiện 1) Then
      (mã thi hành nếu điều kiện 1 thỏa mãn, nghĩa là điều
 kiện 1 = True)
      If (điều kiện 2) Then
      (mã thi hành nếu điều kiện 1 và 2 được thỏa mãn,
 nghĩa là điều kiện 1 = true và điều kiện 2 = True)
      End If
 End If
```

Thí dụ:

```
        'Tuyên bố biến số và giá trị cho intYourAge
        Dim intYourAge
        intYourAge = 65
        'Quyết định và công bố kết quả
        If intYourAge > 55 Then
            MessageBox.Show ("You should retire.")
```

```
            If intYourAge = 65 Then
                MessageBox.Show ("You must retire.")
            End If
        End If
```

Chú thích:

Nếu ta trên 55 tuổi, MessageBox sẽ hiển thị hàng chữ 'You should retire', nhưng nếu ta đúng 65, MessageBox sẽ hiển thị hàng chữ 'You must retire'. Sở dỉ có điều kiện trong điều kiện trên 55 tuổi như vậy là vì tuổi ta có thể là 60 hay 70 miễn sao trên 55 là được, do đó ta kèm thêm một điều kiện nữa để có thể hiển thị hàng chữ 'You must retire' khi nào số tuổi vừa đúng 65.

Cú pháp 5:

Dùng If chỉ trong một hàng mã có thể gọn gàng hơn nhưng không cung cấp cấu trúc giải thuật rõ ràng và thường khó đọc hơn.

```
If (điều kiện) Then (mã thi hành nếu điều kiện = True)
Else (mã thi hành nếu điều kiện = False)
```

hoặc

```
If (điều kiện) Then
  (nếu điều kiện =True, thi hành mã 1: mã 2 : mã 3: ...)
```

Lưu ý: kiểu cú pháp 5 không cần phải có **End If** ở cuối hàng.

Thí dụ:

```
    Dim count As Integer
    If count = 8 Then MessageBox.Show ("Count = 8")
       Else MessageBox.Show ("Count is not 8")
    If count = 8 Then
      MessageBox.Show ("Count = 8") : count = count + 1 :
MessageBox.Show ("Count is now " & count)
```

Chú thích:

Ở hàng If đầu tiên, ta tuyên bố biến số (variables) count , so sánh với giá trị 8 và hiển thị kết quả.

Hàng If thứ nhì, ta hiển thị kết quả, sau đó cộng thêm 1 vào cùng 1 biến số (variables) count, như vậy count (cuối cùng) = count (với giá trị là 8) + 1 sẽ bằng 9 và hiển thị kết quả **Count is now 9**.

Cách viết mã từ trên xuống dưới, ta thấy mã dễ đọc, dễ hiểu hơn:

```
    Dim count As Integer
  If count = 8 Then
     MessageBox.Show ("Count = 8")
     count = count + 1
     MessageBox.Show ("Count is now " & count)
  End If
```

4.2. Toán tử so sánh

Khi dùng If để kiểm tra các diều kiện, không những ta chỉ so sánh **bằng (=)** không thôi, mà còn so sánh nhiều kiểu khác nhau nữa. Gọi chung là các dấu so sánh (**Comparision Operators**) gồm có:

- < > : là dấu không bằng (Not Equal To)
- < : dấu nhỏ hơn
- > : dấu lớn hơn
- < = : dấu nhỏ hơn hoặc bằng
- > = : dấu lớn hơn hoặc bằng
- AND : dùng kiểm tra hơn 1 điều kiện
- OR : dùng kiểm tra điều kiện này hoặc điều kiện nọ

Ví dụ:

```
        If yourSurName < > "Vu" Then
           MessageBox.Show("You are not my relative")
        End If
        If yourAge < 18 Then
           MessageBox.Show("You can not drive a car")
        End If
        If yourSurName > 21 Then
           MessageBox.Show("You can marry")
        End If
        If yourAge <= 64 Then
           MessageBox.Show("You can not retire")
        End If
        If yourAge >= 65 Then
           MessageBox.Show("You must retire")
        End If
        If yourAge >= 18 AND yourHeigth >= 1.60 Then
           MessageBox.Show("You can be a movie star")
        End If
       If yourPreferredDrink="Rượu" OR yourPreferredDrink="Bia"
Then
           MessageBox.Show("You are ... dân nhậu")
        End If
```

Chú thích:

Nhắc thêm ở đây về cách dùng **AND** và **OR** trong điều kiện. Ta biết khi điều kiện được kiểm tra sẽ cho biết giá trị là True (hay 1, hay 'ON', hay là Tắt hoặc Đóng) hoặc False (hay 0, hay 'OFF', hay là Mở hoặc Khoá). Để dễ nhớ, ta thiết lập bảng sau:

Bảng 4.1: Toán tử AND

Điều kiện A	Điều kiện B	Kết quả	Chú thích
0	0	0	Nếu A False và B False, kết quả sau cùng là False
0	1	0	Nếu A False và B True, kết quả sau cùng là False
1	0	0	Nếu A True và B False, kết quả sau cùng là False
1	**1**	**1**	Nếu A True và B True, kết quả sau cùng là True. Ta

			thấy chỉ có một trường hợp bằng True với dấu AND khi nào cả 2 đều True.

Bảng 4.2: Toán tử OR

Điều kiện A	Điều kiện B	Kết quả	Chú thích
0	**0**	**0**	Nếu A Fasle và B False, kết quả sau cùng là False. Ta thấy chỉ có một trường hợp bằng False với dấu OR khi nào cả 2 đều False.
0	1	1	Nếu A False và B True, kết quả sau cùng là True
1	0	1	Nếu A True và B False, kết quả sau cùng là True
1	1	1	Nếu A True và B True, kết quả sau cùng là True.

4.3. So sánh xâu

Khi so sánh chữ hay câu, ta thường gặp trở ngại khi không lưu ý đến các **chữ thường hay chữ Hoa** (**case sensitive**). Nhớ là đối với máy vi tính, khi so sánh như vậy, chữ **a** thường khác với chữ **A** Hoa vì chúng có giá trị khác nhau.

Thí dụ 1:

```
Dim mySociety As String
mySociety = "VOVISOFT"
If mySociety = "Vovisoft" Then
   MessageBox.Show("You are a Vovisoft's member")
Else
   MessageBox.Show("You are not a Vovisoft's member")
End IF
```

Thí dụ 2: dùng **Compare** method của **String** object so sánh 2 chữ hay câu như sau:

```
Dim mySociety As String
mySociety = "VOVISOFT"
If String.Compare (mySociety, "Vovisoft", True) = 0 Then
   MessageBox.Show("You are a Vovisoft's member")
Else
   MessageBox.Show("You are not a Vovisoft's member")
End IF
```

Chú thích:

String.Compare dùng để so sánh 2 giá trị của String và trả về 1 số nguyên (Integer) sau khi so sánh. Nếu method trả về số 0, nghĩa là 2 chữ hay câu giống nhau về giá trị, ngoài ra sẽ trả về số khác số 0.

4.4. Lệnh lựa chọn

Đây là loại thứ hai trong **Conditional Logic** để thi hành các phần khác nhau trong giải thuật dựa trên những điều kiện khác nhau nào đó. Thí dụ như:

- Nếu là khách hàng A, gởi email đến địa chỉ khách hàng A.
- Nếu là khách hàng B, gởi email đến địa chỉ khách hàng B.

- Nếu là khách hàng C, gởi email đến địa chỉ khách hàng C.
- Nếu là khách hàng D, gởi email đến địa chỉ khách hàng D.
- Nếu là khách hàng E, gởi email đến địa chỉ khách hàng E.

Ta có thể dùng If ... Then ... ElseIf ... End If như sau:

```
     If khách hàng = "A" Then
    gởi email đến địa chỉ khách hàng A
 ElseIf khách hàng = "B" Then
    gởi email đến địa chỉ khách hàng B
 ElseIf khách hàng = "C" Then
    gởi email đến địa chỉ khách hàng C
 ElseIf khách hàng = "D" Then
    gởi email đến địa chỉ khách hàng D
 ElseIf khách hàng = "E" Then
    gởi email đến địa chỉ khách hàng E
 End If
```

Tuy nhiên, nếu ta muốn đổi **khách hàng** thành **công ty** chẳng hạn, ta phải thay đổi chữ **khách hàng** ở từng câu If một, như vậy quả là phiền phức và không đạt năng suất cao như cách dùng cú pháp **Select Case** :

Cú pháp 1 (Syntax 1):

```
Select Case công ty
Case "A"
    gởi email đến địa chỉ công ty A
Case "B"
    gởi email đến địa chỉ công ty B
Case "C"
    gởi email đến địa chỉ công ty C
Case "D"
    gởi email đến địa chỉ công ty D
Case "E"
    gởi email đến địa chỉ công ty E
End Select
```

Lưu ý:

Khi dùng Select Case, có phân biệt chữ thường và chữ Hoa, ví dụ: công ty **A** khác với công ty **a**.

Cú pháp 2:

Dùng Select Case để chọn trường hợp gồm nhiều điều kiện có một giải đáp chung:

```
Select Case strMyContactName
  Case "A", "B", "E"
   MessageBox.Show ("Chào các bạn học!!", "Greeting")
  Case "C", "D"
    MessageBox.Show("Hay tham gia nhe!", "Greeting")
End Select
```

Trong đó, ta thấy trường hợp A, B và E có chung một giải đáp nhưng khác với trường hợp C và D.

Cú pháp 3:

Dùng Select Case cho các trường hợp ngoại lệ **Case Else**:

```
Select Case strMyContactName
    Case "A", "B", "E", "C", D"
        MessageBox.Show ("Chào các bạn học!", "Greeting")
    Case Else
        MessageBox.Show ("Hay tham gia!", "Greeting")
End Select
```

4.5. Vòng lặp

Looping Logic dùng trong trường hợp cần lập đi lập lại nhiều lần (hay đúng hơn nữa, một số lần nhất định) việc thi hành một công tác nào đó, ví dụ: cộng thêm 10 sản phẩm vào bảng liệt kê sản phẩm của công ty, hiển thị (display) 5 CD nhạc tuyệt phẩm hàng đầu trong năm.

2 loại cơ bản của Looping Logic - **For** loop và **Do** loops bao gồm:

- `For ... Next`
- `For Each ... In ... Next`
- `Do Until ... Loop`
- `Do While ... Loop`
- Các trường hợp đặc biệt

Cú pháp 1:

```
For số lần đếm từ số ... đến số ...
    (thi hành công việc nào đó)
Next
```

Thí dụ 1:

```
    'Tuyên bố biến số dùng làm counter
    Dim intCounter
For intCounter = 1 To 10
    MessageBox.Show ("Vovisoft", "Greeting")
Next
```

Thí dụ 2:

```
    'Tuyên bố biến số dùng làm counter
    Dim intCounter
For intCounter = 10 To 100 Step 10
    MessageBox.Show ("Vovisoft", "Greeting")
Next
```

Chú thích:

Thí dụ 1, tạo biến số (variables) intCounter để đếm từ 1 đến 10, mỗi lần đếm như vậy trong For ... Next loop, ta hiển thị 1 cửa sổ với hàng chữ Vovisoft.

Thí dụ 2, mỗi lần đếm ta nhảy 10 bước (hay cộng thêm 10 vào số lần đếm) bắt đầu với inCounter = 10 là lần đầu tiên, kế là 20, 30, ... đến 100.

Cú pháp 2:

```
For Each ... In ...
    (thi hành công việc nào đó)
Next
```

Thí dụ: Liệt kê tất cả các thư mục phụ (subfolders) trong dĩa C (root directory trong drive C)

```
'Biến array dùng lưu trữ các ngăn chứa phụ (subfolders)
Dim subFolders( ) As DirectoryInfo
subFolders = New DirectoryInfo("C:\").GetDirectories
'Loop liệt kê các ngăn chứa phụ (subfolders) trong dĩa C
Dim subFolder As DirectoryInfo
For Each subFolder In subFolders
     lstData.Items.Add (subFolder.FullName)
Next
```

Chú thích:

Khai báo và tạo biến số (variables) loại Array trực thuộc đối tượng **DirectoryInfo**. Dùng method **GetDirectories** của object DirectoryInfo để lấy và lưu trữ các thư mục phụ trong dĩa C.

Sau đó, dùng **For Each ... Next** loop kiểm tra từng phần một trong array subFolders và cộng tên của phần vào bảng liệt kê tên **lstData**.

Cú pháp 3:

```
Do Until (điều kiện)
     (thi hành công việc nào đó)
Loop
```

Thí dụ: Liệt kê từng số ngẫu nhiên và chấm dứt loop khi nào số đó là số 10

```
'Bố trí object tạo số ngẫu nhiên
Dim random As New Random( )
'Khai baos 1 biến số chứa số ngẫu nhiên mặc định là 0
Dim intRandomNumber As Integer = 0
'Loop cho đến khi nào số intRandomNumber = 10
Do Until intRandomNumber = 10
    'Tạo 1 số ngẫu nhiên
    intRandomNumber = random.Next (25)
    'cộng vào bảng liệt kê tên lstData
    lstData.Items.Add (intRandomNumber)
Loop
```

Chú thích:

Ta dùng random là 1 thể hiện (instance) của object Random để tạo số ngẫu nhiên (random number generator) trong Do Until ... Loop và lưu trữ giá trị đó vào biến số (variables) intRandomNumber. Khi nào giá trị số này bằng 10, ta chấm dứt việc cộng số vào bảng liệt kê tên lstData.

Cú pháp 4:

Ngược lại với Do Until ... Loop là Do While ... Loop. **Do While ... Loop chỉ thi hành khi nào điều kiện bằng True, ngược lại với Do Until ... Loop sẽ chấm dứt khi nào điều kiện bằng True.**

```
Do While (điều kiện)
    (thi hành công việc nào đó)
Loop
```

Thí dụ: Liệt kê từng số ngẫu nhiên và chấm dứt loop khi nào số đó = 10 hay lớn hơn 10

```
'Bố trí object tạo số ngẫu nhiên
Dim random As New Random( )
'Khai bao 1 biến số chứa số ngẫu nhiên mặc định là 0
Dim intRandomNumber As Integer = 0
'Loop khi số intRandomNumber < 10
Do While intRandomNumber < 10
    'Tạo 1 số ngẫu nhiên
    intRandomNumber = random.Next (25)
    'cộng vào bảng liệt kê tên lstData
    lstData.Items.Add (intRandomNumber)
Loop
```

Chú thích:

Ta dùng random là 1 thể hiện của object Random để tạo số ngẫu nhiên (random number generator) trong Do While ... Loop và lưu trữ giá trị đó vào biến số (variables) intRandomNumber. Khi nào giá trị số này nhỏ hơn 10, ta cộng số đó vào bảng liệt kê tên lstData, nếu không, ta chấm dứt loop.

Cú pháp 5:

Đây là phiên bản khác của Do Until và Do While:

```
Do
    (thi hành công việc nào đó)
Loop While (điều kiện)
Do
    (thi hành công việc nào đó)
Loop Until (điều kiện)
```

Lưu ý:

Phiên bản này khác phiên bản trước ở chỗ:

- **Thi hành công việc trước**
- Sau đó mới kiểm tra điều kiện để tiếp tục hay chấm dứt loop, như vậy tối thiểu, công việc được thi hành 1 lần.

Các trường hợp đặc biệt:

1. Nested Loops:

Nhiều trường hợp cần đến 2 hay nhiều loop trong algorithm, tỷ như:

```
'Tuyên bố và bố trí hàng và cột
Dim intRow, intColume As Integer
'Loop hàng từ hàng thứ nhất đến hàng 10
For intRow = 1 To 10
```

```
        'Mỗi hàng, loop từ cột thứ nhất đến cột 5
        For intColume = 1 To 5
         'hiển thị hàng và cột bảng liệt kê tên lstData
         lstData.Items.Add ("Hàng" & intRow & " và cột " &
           intColume)
        Next
    Next
```

Chú thích:

Ta dùng 2 lần For ... Next (nested loop) để hiển thị hàng trước cột sau trong bảng liệt kê tên lstData theo thứ tự sau:

Hàng	**1**	và	cột	1
Hàng	1	và	cột	2
Hàng	1	và	cột	3
Hàng	1	và	cột	4
Hàng	1	và	cột	5
Hàng	**2**	và	cột	1
Hàng	2	và	cột	2
Hàng	2	và	cột	3
Hàng	2	và	cột	4
Hàng	2	và	cột	5
...				
...				
Hàng	**10**	và	cột	1
Hàng	10	và	cột	2
Hàng	10	và	cột	3
Hàng	10	và	cột	4
Hàng	10	và	cột	5

Dùng **Exit For** hay **Exit Do** để chấm dứt loop (quit) vô điều kiện.

Chú ý khi tạo và loop, ta có thể gặp vòng lặp không thoát, vú dụ như:

```
    'Tuyên bố biến số
     Dim counter As Integer = 0
     'Loop bế tắc không lối thoát
     Do
         counter += 1
     Loop Until counter = 0
```

Chú thích:

Mặc dù trước khi vào loop, counter = 0 nhưng với **counter += 1** được thi hành trước khi kiểm tra điều kiện counter = 0, ở đây có nghĩa là counter (hiện tại) bằng counter (trước đó là 0) cộng thêm 1 như vậy giá trị của counter bây giờ bằng 1 (vì 0 + 1 = 1).

Đến khi kiểm tra điều kiện counter = 0 ở câu Loop Until counter = 0, điều kiện này sẽ là False, do đó loop bắt đầu lập lại với counter = 2, 3, 4,… không thoát.

Cách giải quyết khi Infinite Loop:

- Trường hợp chạy ứng dụng (application) trong MS Visual Studio.NET, chọn **Debug | Stop Debugging** để chấm dứt.
- Trường hợp chạy ứng dụng (application) bên ngoài MS Visual Studio.NET, nhấp các nút **Ctrl + Alt + Delete** và chọn **Task Manager**, sau đó chọn ứng dụng (application) có hàng chữ kèm '**Not Responding**' trong phần mục **Status** và nhấp nút **End** để chấm dứt.

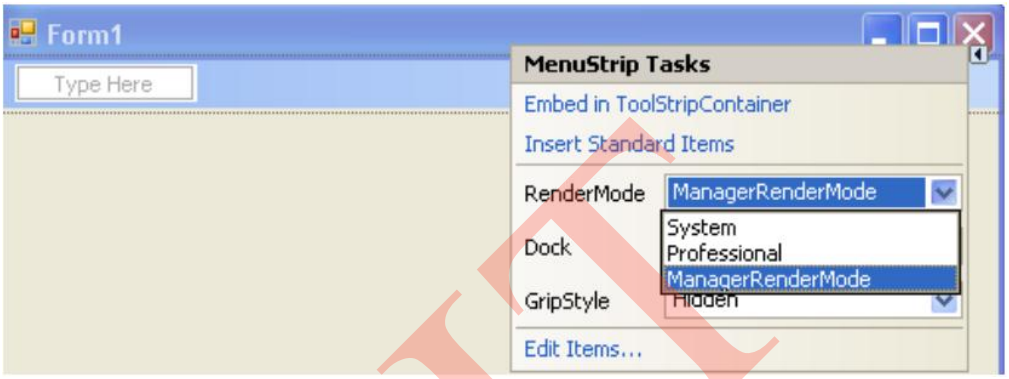
Cuối cùng, kiểm tra và điều chỉnh lại điều kiện để chấm dứt loop.

CHƯƠNG 5: THIẾT KẾ BIỂU MẪU DÙNG CÁC ĐIỀU KHIỂN

5.1. Cửa sổ form

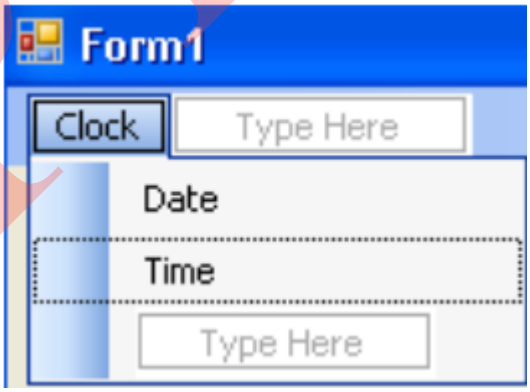
Tạo mới một giải pháp mang tên MyMenu và thêm vào đó một dự án mới cùng tên như đã biết trong các ví dụ trước. Tại giao diện thiết kế, ta đưa điều khiển MenuStrip vào trong Form bằng cách tích đúp chuột hay kéo thả như đã biết.

Chúng ta không cần quan tâm đến vị trí của menu trên form vì Visual Studio sẽ tự động đặt nó sao cho phù hợp. Chúng ta có thể thay đổi các thuộc tính sao cho phù hợp bằng cách chọn mở Smart Tags (nút mũi tên tam giác màu đen bên góc phải điều khiển Menu).



Hình 5.1. Tạo menu

Khi được đặt vào form thì điều khiển menu sẽ được đặt tại một vùng như trên hình gọi là khay công cụ - Component tray và VS sẽ hiển thị trực quan menu trên đầu cửa sổ Form. Chuỗi Type Here là nơi bạn có thể chọn và nhập vào các mục chọn cho menu. Chúng ta sẽ tạo ra menu ngay sau đây. Nhắp chuột vào chuỗi Type Here và gõ vào chuỗi "Clock" và ấn enter. Nhắp chuột vào chuỗi Type Here con ở dưới rồi gõ Date, Time như hình 5.2.



Hình 5.2. Menu con

Để đóng phần thiết kế menu, ta chọn một vùng nào đó trên form, để hiển thị ta lại chọn vào menu Clock như trên. Bây giờ chúng ta sẽ tạo một số tùy biến cho Menu. Trong một số phần mềm hay ngay trình duyệt Windows Explorer của hệ điều hành chúng ta có thể ấn tổ hợp Alt + phím tắt để mở nhanh một thực đơn nào đó. Các phím tắt ấy được gọi là phím truy cập (Access Key). Phím này có dấu gạch chân ở dưới. Trong VS, để tạo phím này ở ta chỉ việc gõ thêm dấu „&" trước ký tự nào muốn hiển thị gạch chân trong phần Type Here. Tạo ra các phím tắt cho các mục chọn của menu Clock như hình 5.3.

Hình 5.3. Phím tắt

Việc thay đổi thứ tự các mục chọn bằng cách mở chế độ thiết kế menu rồi nhắp chọn mục chọn nào đó, kéo nó đến vị trí mong muốn. ví dụ ta kéo mục chọn Time lên thay cho vị trí mục chọn Date. Bây giờ chúng ta tạo ra sự kiện kích chuột cho các mục chọn của menu. Khi bạn tích chuột vào Date hay Time thì một nhãn Label sẽ xuất hiện và hiển thị thông tin ngày hay giờ tương ứng. Để làm được điều này, trước hết ta tạo ra một Label vào trong form.Tạo thuộc tính cho đối tượng Label1: BorderStyle – FixedSingle; Font – Bold 14; Text – rỗng; TextAlign – MiddleCenter. Cài đặt thủ tục sự kiện cho mục chọn menu. Bây giờ chúng ta sẽ tạo sự kiện tích chuột cho các mục con trong menu Clock.

Nhắp vào menu Clock trên form1 để hiển thị menu con. Nhắp đôi chuột vào mục chọn Time để mở cửa sổ Code Editor và tạo ra một thủ tục có tên TimeToolStripMenuItem_Click. Trong VS.NET 2005 thì khi bạn gõ tên mục chọn là gì thì mặc định khi tích đúp chuột để viết mã thì VS sẽ tạo ra một thủ tục có phần đầu tên trùng với tên mục chọn (phần tên chưa có dấu cách trống phân cách tên mục chọn) menu (ở trên là TimeToolStripMenuItem_Click). Tất nhiên đây là mặc định, ta có thể thay đổi tên nhờ thuộc tính Name ở cửa sổ Properties.

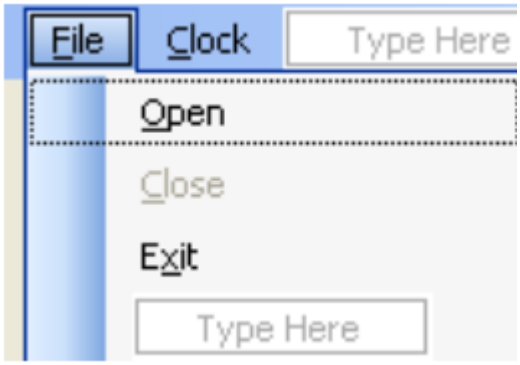
Nhập dòng mã sau:

```
Label1.Text = TimeString
Tương tự với thủ tục DateToolStripMenuItem_Click của mục chọn Date
Label1.Text = DateString
```

Thêm mục File vào menu chương trình

Tạo thêm một mục con Color vào trong menu Clock. Mục này sẽ kích hoạt hộp thoại ColorDialog1 chọn màu cho Label1.

Tạo một Menu File bên cạnh menu Clock như hình 5.4. Đồng thời tạo thêm các mục con Open, Close, Exit trong menu này.



Hình 5.4. Menu file

Tiếp theo ta thay đổi tên bằng thuộc tính Name trong cửa sổ Properties cho các mục chọn: mục Open thành mnuOpenItem, Close thành mnuCloseItem, Exit thành mnuExitItem. Ta cũng đặt thuộc tính Enable của mục Close (giờ là mnuCloseItem) thành False. Thuộc tính này vô hiệu hóa hay làm mờ mục Close như hình. Nó chỉ được sáng lên để người dùng chọn khi mã thực thi chương trình cho phép.

Viết mã chương trình

Ttạo thủ tục mnuOpenItem_Click bằng cách nhấp đúp chuột vào mục Open trên menu File và nhập đoạn mã sau:

```
OpenFileDialog1.Filter = "Bitmaps (*.bmp) | *.bmp"
If OpenFileDialog1.ShowDialog()=Windows.Forms.DialogResult.OK
    Then
       PictureBox1.Image = System.Drawing.Image.FromFile _
       (OpenFileDialog1.FileName)
       mnuCloseItem.Enabled = True
End If
```

Chú thích mã:

- Đoạn mã thứ nhất giúp lọc ra loại file để mở là file ảnh dạng Bitmap (*.bmp). Bạn có thể mở nhiều loại file bằng câu lệnh:

```
OpenFileDialog1.Filter = _ "Bitmaps(*.bmp)|*.bmp|JPEG(*.jpg)|
    *.jpg|All Files(*.*)|*.*"
```

- Phương thức ShowDialog() là phương thức mới trong VS.NET, nó có thể dùng được với mọi hộp thoại và cửa sổ Windows Forms. Phương thức này trả về kết quả mang tên DialogResult cho biết người dùng đã chọn vào hộp thoại. Và nếu nút OK được chọn thì kết quả trả về sẽ bằng với DialogResult.OK.

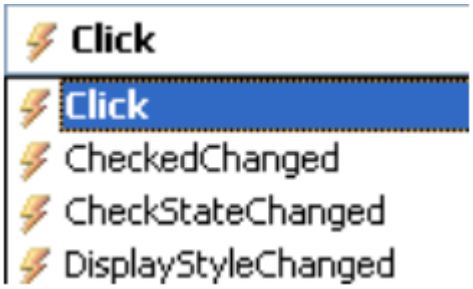
- Khi nút Open được nhấn, nếu hợp lệ thì thuộc tính FileName của OpenFileDialog sẽ mang đầy đủ đường dẫn và tên file của file đã mở vì thế mà dòng mã thứ 3 sẽ nạp chính xác ảnh vào PictureBox1.

Tương tự ta cũng nhấn đúp chuột vào mục Close để tạo thủ tục chọn cho nó và nhập đoạn mã sau:

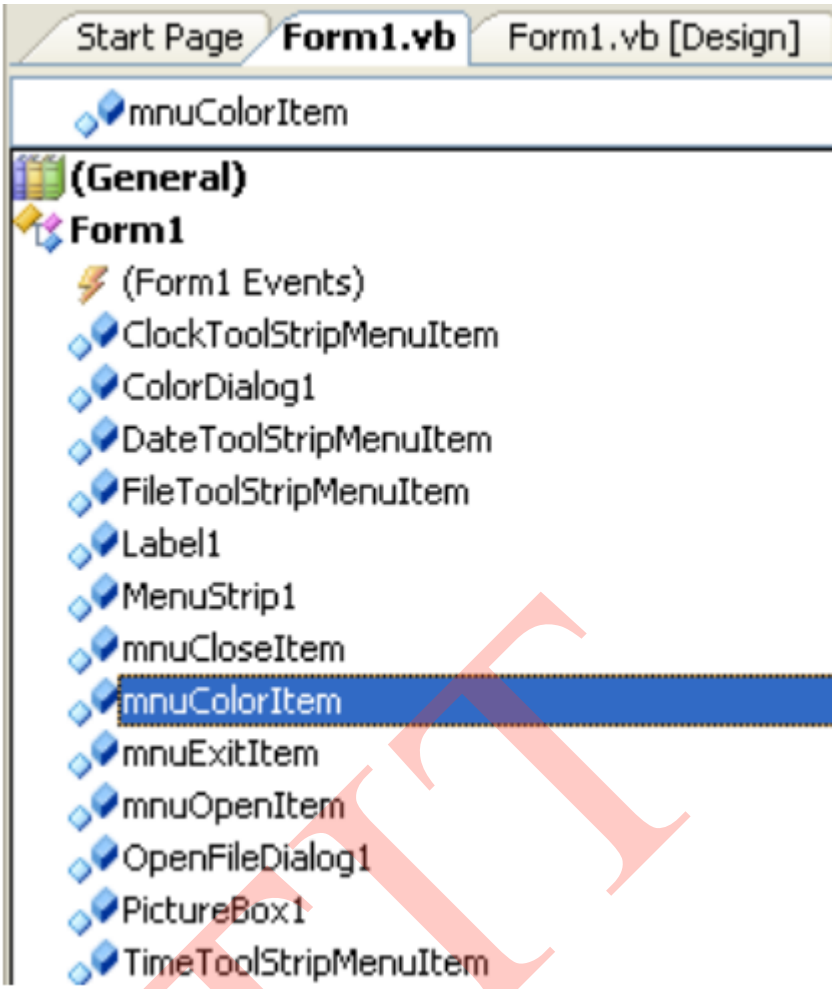
```
PictureBox1.Image = Nothing
        mnuCloseItem.Enabled = False
```

Khi mở ảnh rồi thì mục Close sáng lên, khi chọn vào mục này thì PictureBox1 không còn ảnh nữa và mục này lại bị vô hiệu hóa. Nhắp đôi vào mục Exit và nhập dòng mã: End

Tạo thủ tục mnuColorItem_Click bằng cách nhấn đúp hay chọn từ danh sách xổ xuống như hình 5.5.



Hình 5.5. Sự kiện tích chuột



Hình 5.6. Sự kiện

Nhập vào đoạn mã:

```
ColorDialog1.ShowDialog()
        Label1.ForeColor = ColorDialog1.Color
```

Chú thích mã:

- Phát biểu đầu tiên gọi ShowDialog() để hiển thị hộp thoại ColorDialog.
- Phát biểu thứ hai nhận giá trị màu trả về từ hộp thoại ColorDialog và gán cho màu chữ Text – ForeColor của điều khiển Label1. Ta có thể gán màu cho bất cứ thuộc tính nào như BackColor. Ngoài ra, ta cũng có thể thêm các thuộc tính khác cho hộp thoại ColorDialog trước khi gọi đến phương thức ShowDialog(). Một số thuộc tính và cách gọi được liệt kê như sau:

'ColorDialog1.FullOpen = True :Hiển thị khung tùy biến màu mở rộng

'ColorDialog1.AllowFullOpen = True: hiển thị nút định nghĩa màu tùy biến

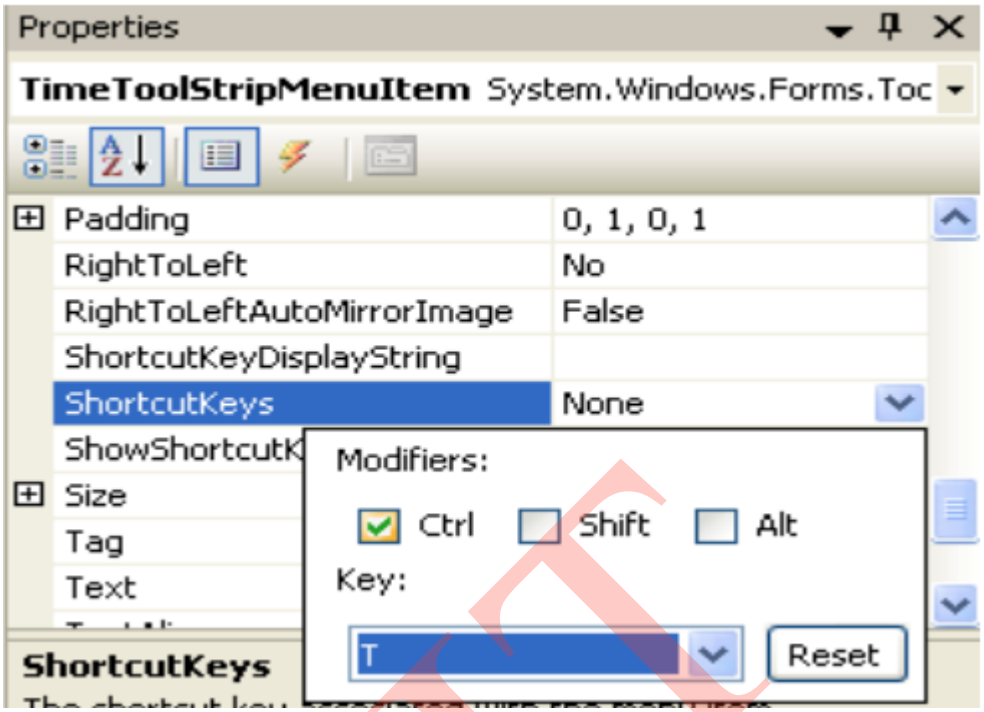
'ColorDialog1.AnyColor = True: cho phép chọn tất cả các loại màu

'ColorDialog1.ShowHelp = True: Hiển thị nút nhấn trợ giúp

'ColorDialog1.SolidColorOnly = True: Hiển thị chỉ những màu đặc

Khi chạy chương trình bằng cách nhấn phím F5 hay Start trên Standard Bar và thử tất cả các tính năng của chương trình. Phím tắt cho phép ta ấn tổ hợp phím để thực hiện lệnh mà không cần chọn menu. Ví dụ như Ctrl+C để sap chép một đoạn text trong Word. Chúng ta chọn gán các phím tắt cho menu trong chương trình MyMenu.

Trước hết mở giải pháp MyMenu ở chế độ thiết kế tích vào menu Clock trên Form, chọn mục Time và chọn chuột phải, chọn Properties. Thiết lập thuộc tính ShortCutKeys như hình 5.7.

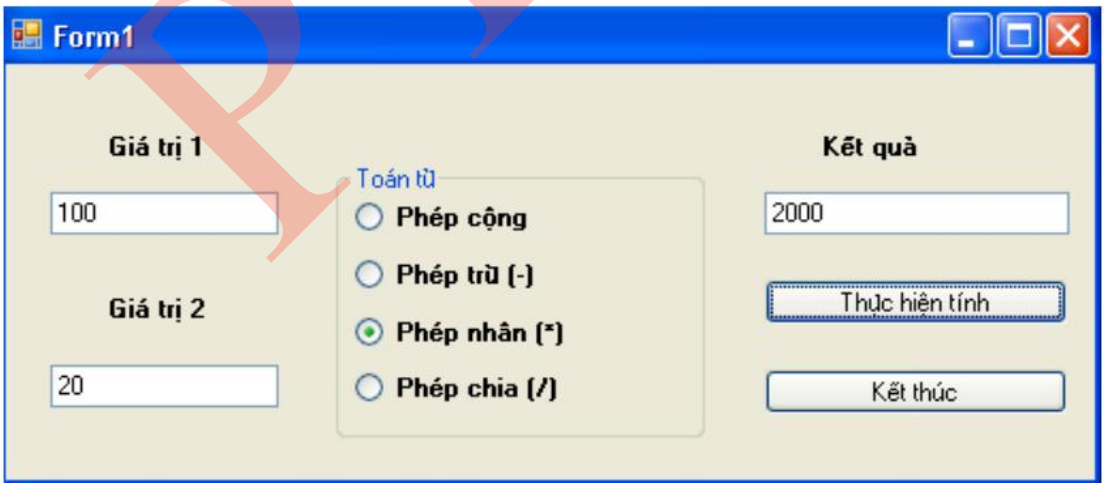


Hình 5.7. đặt phím tắt

Tương tự chúng ta chọn các mục còn lại theo ý thích sao cho các phím nóng không trùng nhau.

5.2. Các thành phần giao diện cơ bản trong window

Giao diện:



Hình 5.8. Giao diện các phép toán cơ sở

Chương trình gồm hai textbox cho phép nhập hai giá trị để gán cho hai biến value1 và value2, bốn radiobutton cho phép chọn bốn toán tử khác nhau, khi đã nhập đầy đủ hai giá trị thì có thể thực hiện tính bằng cách nhấp chọn nút „thực hiện tính" và kết quả hiển thị trong ô textbox3 – kết quả. Xây dựng giao diện:

Tạo một giải pháp và thêm một dự án cùng tên BasicMath đồng thời thiết kế giao diện như hình 5.8. Viết mã:

- Khai báo biến: ta khai báo 2 biến value1, value2 ở đầu lớp form1 như sau:

Dim value1, value2 As Double

- Tạo thủ tục Button1_Click bằng cách tích đúp chuột vào nút „thực hiện tính" và nhập đoạn mã sau:

```
If TextBox1.Text = "" Or TextBox2.Text = "" Then
          MsgBox("Bạn cần nhập đầy đủ hai giá trị")
      Else
          value1 = CDbl(TextBox1.Text)
          value2 = CDbl(TextBox2.Text)
          If RadioButton1.Checked = True Then
              TextBox3.Text = value1 + value2
          End If
          If RadioButton2.Checked = True Then
              TextBox3.Text = value1 - value2
          End If
          If RadioButton3.Checked = True Then
              TextBox3.Text = value1 * value2
          End If
          If RadioButton4.Checked = True Then
              TextBox3.Text = value1 / value2
          End If
      End If
```

Chú thích mã:

- Hàm CDbl là hàm chuyển kiểu sang kiểu Double. Chọn phím F5 hay nút start để chạy chương trình.

CHƯƠNG 6: MẢNG

6.1. Làm việc với mảng

Mảng giúp quản lý các dữ liệu lớn hết sức dễ dàng. Việc truy cập các phần tử của mảng thông qua chỉ số. Việc khai báo mảng tương tự như khai báo biến. Việc khai báo thường chứa các thông tin như:

- Tên mảng: Tên đại diện cho mảng, việc truy cập một phần tử mảng gồm tên mảng và chỉ số mảng.
- Kiểu dữ liệu: Tất cả các phần tử trong mảng phải có cùng kiểu.
- Kích thước mảng: Là số chiều của mảng.
- Số phần tử của mảng: Số phần tử tối đa của mảng

Cú pháp chung khai báo mảng có kích thước là:

Dim ArrayName(Dim1Index, Dim2Index) As DataType

Trong đó:

- ArrayName: tên mảng
- Dim1Index và Dim2Index: là hai chiều của mảng
- Datatype: kiểu dữ liệu của mảng. Khi chưa xác định kiểu cụ thể, có thể dùng kiểu Object.

Ví dụ:

Khai báo Dim Employee(4) As String: khai báo mảng một chiều chứa 5 phần tử có tên là Employee có kiểu String.

Ta cũng có thể khai báo mảng một cách toàn cục trong module bằng từ khóa Public như sau: Public Employee(4) As String. Để khai báo mảng hai chiều mang tên ScoreBoard ta có thể khai báo như sau:

Dim ScoreBoard(1, 4) As Short

Mảng này gồm 2*5 = 10 phần tử tương ứng với 10 ô vuông gồm hai dòng và 5 cột đánh số từ 0.

6.2. Làm việc với các phần tử trong mảng

Sau khi khai báo, bạn có thể sử dụng mảng. Việc truy cập vào một phần tử của mảng nhờ tên mảng và chỉ số của mảng đặt trong ngoặc đơn, chỉ số là số nguyên, là biến nguyên hay biểu thức có giá trị. Để duyệt qua tất cả các phần tử trong mảng, dùng vòng lặp For…Next.

Ví dụ:

```
employee(3) = "Thanh Van"
```

Khai báo trên gán cho phần tử có chỉ số thứ 3 (tại ô thứ 4) tên là “Thanh Van”.

```
ScoreBoard(0, 2) = 12
```

Khai báo trên gán cho phần tử ở dòng 0, cột 2 giá trị là 12.

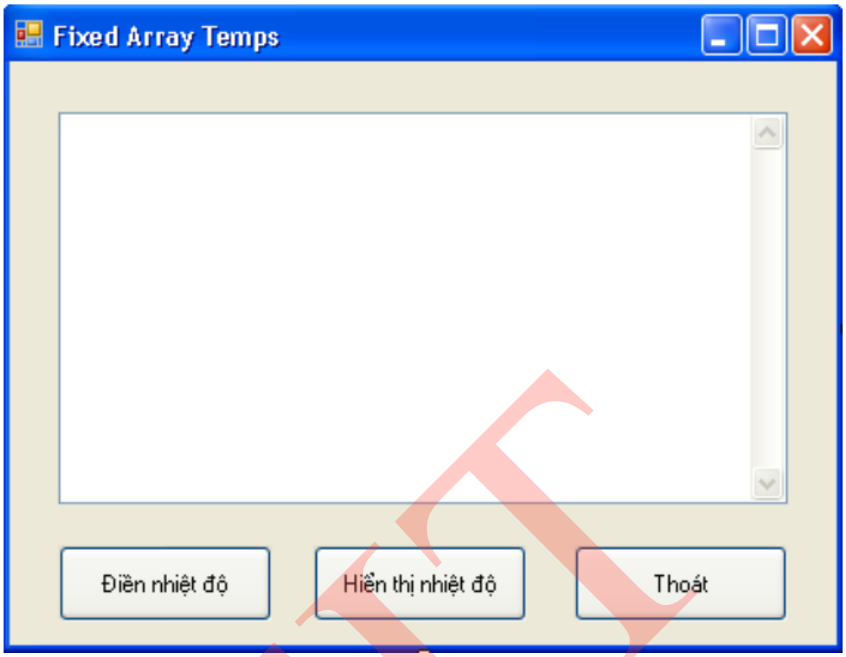
6.3. Sử dụng mảng có kích thước cố định

Bây giờ ta tạo ví dụ MyFixedArray sử dụng mảng một chiều có tên nhietdo để ghi lại giá trị nhiệt độ cao thấp hàng ngày trong tuần. Mảng này được khai báo ở đầu form và được

gán giá trị bằng hàm InputBox nhờ vòng lặp For…Next. Toàn bộ nội dung của mảng sau đó lại được hiển thị lại vào một textbox cũng nhờ vòng lặp For…Next.

Thiết kế giao diện:

Tạo mới một giải pháp và thêm vào một dự án có cùng tên là MyFixedArray. Thiết kế giao diện như hình 6.1.



Hình 6.1. Màn hình thiết kế

Trong đó: nút button1 có text là "Điền nhiệt độ", button2 là "Hiển thị nhiệt độ", button3 là "Thoát". Viết mã:

Trước hết ta khai báo mảng nhietdo ở ngay dưới dòng Public Class Form1 như sau:

```
Dim nhietdo(6) As Single
```

Khai báo như trên nghĩa là tất cả các thủ tục, các hàm đều có thể sử dụng mảng này. Tiếp theo ta tạo ra sự kiện nhập vào các giá trị nhiệt độ trong tuần bằng cách tạo thủ tục Button1_Click và nhập mã như sau:

```
Private Sub Button1_Click(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles Button1.Click
      Dim Prompt, tieude As String
      Dim i As Short
      Prompt = "Điền vào nhiệt độ của ngày."
      For i = 0 To UBound(nhietdo)
          tieude = "Ngày " & (i + 1)
          nhietdo(i) = CInt(InputBox(Prompt, tieude))
      Next
  End Sub
```

Trong đó, hàm Ubound(nhietdo) là hàm lấy về chỉ số trên của mảng nhietdo, trong trường hợp này là 6. Sau đó ta cho hiển thị các giá trị nhiệt độ trong bảy ngày trong tuần cũng như giá trị nhiệt độ trung bình bằng thủ tục Button2_Click khi người dùng click vào nút "Hiển thị nhiệt độ" như sau:

```
Private Sub Button2_Click(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles Button2.Click
```

```
        Dim ketqua As String
        Dim i As Short
        Dim tong As Single = 0
        ketqua = "Nhiệt độ của tuần: " & vbCrLf & vbCrLf
        For i = 0 To UBound(nhietdo)
            ketqua = ketqua & "Ngày " & (i + 1) & _
            vbTab & nhietdo(i) & vbCrLf
            tong = tong + nhietdo(i)
        Next
        ketqua = ketqua & vbCrLf & _
        "Nhiệt độ trung bình: " & _
        Format(tong / 7, "0.0")
        TextBox1.Text = ketqua
    End Sub
```

Thủ tục này lại sử dụng vòng lặp For…Next để duyệt lại các phần tử trong mảng sau khi đã được gán giá trị ở thủ tục button1_Click. Biến ketqua được dùng để làm chuỗi kết xuất gộp các giá trị phần tử mảng. Sau mỗi lần gộp ta sử dụng hằng số vbCrLf điều khiển dấu ngắt dòng và dấu về đầu dòng (tương đương với hai hàm Chr(13) và Chr(10)). Hằng vbTab để phân cách giữa phần ghi ngày và ghi nhiệt độ.

Ta tạo thủ thục Button3_Click và nhập phát biểu End để kết thúc chương trình. Chạy chương trình: Khi chạy chương trình và nhập đủ giá trị nhiệt độ cho 7 ngày.

6.4. Tạo mảng động

Việc dùng mảng là rất thuận tiện. Tuy nhiên khi ta chưa biết chính xác số phần tử của mảng là bao nhiêu thì như thế nào? Ví dụ khi ta muốn để người dùng nhập vào bao nhiêu nhiệt độ tùy thích, nhập càng nhiều thì độ chính xác càng cao. VB giải quyết việc này bằng mảng động. Kích thước mảng động chỉ được chỉ định khi chương trình thực thi chứ không định trong lúc viết mã. Việc khai báo trước kích thước mảng là không cần thiết nhưng cũng cần dành chỗ trước cho mảng đó. Các bước tạo mảng động:

- Chỉ định tên và kiểu cho mảng khi thiết kế form, ví dụ Dim nhietdo() As Single
- Thêm mã xác định kích thước mảng khi chương trình thực thi. Ví dụ khi chương trình chạy ta hỏi xem người dùng muốn nhập bao nhiêu ngày, ví dụ:

```
Dim songay As Integer
songay = InputBox("Ban muon nhap bao nhieu ngay?", "Tao mang dong")
```

- Dùng biến songay để định lại kích thước mảng (trừ đi 1 vì mảng tính từ 0). Ví dụ

```
If songay > 0 Then ReDim nhietdo(songay - 1)
```

- Tiếp theo ta dùng hàm Ubound(nhietdo) để xác định số phần tử của mảng.

Bây giờ chúng ta sẽ làm lại ví dụ trên sử dụng mảng động:

- Trước hết, ta khai báo lại mảng động và khai báo biến songay chứa số ngày người dùng muốn nhập bằng đoạn mã ngay dưới dòng khai báo lớp form1:

```
Dim nhietdo() As Single
Dim songay As Integer
```

- Sau đó sửa lại mã của thủ tục Button1_Click như sau:

```
Dim Prompt, tieude As String
    Dim i As Short
    Prompt = "Điền vào nhiệt độ của ngày."
    'Nhap so ngay muon ghi nhiet do
    songay = InputBox("Ban muon nhap bao so ngay?","")
    If songay > 0 Then ReDim nhietdo(songay - 1)
    For i = 0 To UBound(nhietdo)
        tieude = "Ngày " & (i + 1)
        nhietdo(i) = CInt(InputBox(Prompt, tieude))
Next
```

- Tiếp theo thay số 7 trong thủ tục Button2_Click bằng biến songay:

```
ketqua = ketqua & vbCrLf & _
   "Nhiệt độ trung bình: " & _
Format(tong / songay, "0.0")
```

- Ta có thể dùng phát biểu Try…Catch để bắt lỗi nếu người dùng nhập vào một số nhỏ hơn 0.
- Chạy lại chương trình và kết quả rõ ràng linh động hơn.

CHƯƠNG 7: VB.NET KẾT NỐI CƠ SỞ DỮ LIỆU DÙNG ADO.NET

7.1 ADO.NET là gì

ActiveX Data Object.NET (ADO.NET) là một thư viện phần mềm .NET Framework. Bao gồm các thành phần phần mềm cung cấp dịch vụ truy cập dữ liệu. ADO.NET được thiết kế để cho phép các nhà phát triển viết mã được quản lý cho việc tiếp cận các nguồn dữ liệu bị ngắt kết nối, có thể có quan hệ hoặc không quan hệ (như XML hoặc dữ liệu ứng dụng). Tính năng này của ADO.NET giúp ứng dụng có thể chia sẻ dữ liệu hay các ứng dụng phân tán.

ADO.NET cung cấp kết nối truy cập với CSDL bằng cách sử dụng .NET Managed Provinder (*mặc định .Net Framework có hai Managed Providers là SQL Managed Provider và OleDB Managed Provider*) và truy cập bị ngắt kết nối bằng cách sử dụng **Datasets**, đó là các ứng dụng sử dụng kết nối cơ sở dữ liệu chỉ trong thời gian truy xuất dữ liệu hoặc cập nhật dữ liệu. Datasets là thành phần giúp đỡ để lưu trữ các dữ liệu liên tục trong bộ nhớ để cung cấp truy cập khi bị ngắt kết nối sử dụng các nguồn tài nguyên cơ sở dữ liệu một cách hiệu quả và khả năng mở rộng tốt hơn.

ADO.NET được phát triển từ ADO, cũng là một công nghệ tương tự như ADO.NET với một vài thay đổi cấu trúc cơ bản. Mặc dù có một vài trường hợp để làm việc trong chế độ bị ngắt kết nối bằng cách sử dụng ADO, nhưng dữ liệu được chuyển đến CSDL trong ADO.NET hiệu quả hơn bằng cách sử dụng **Data Adapters**. Các đại diện trong bộ nhớ của dữ liệu giữa ADO và ADO.NET là khác nhau. ADO.NET có thể giữ các dữ liệu trong một bảng kết quả duy nhất, nhưng ADO giữ nhiều bảng cùng với các chi tiết của mối quan hệ. Không giống như ADO, việc truyền dữ liệu giữa các ứng dụng bằng cách sử dụng ADO.NET không sử dụng COM (Component Object Model) để sắp xếp mà dùng datasets, nó truyền dữ liệu như là một dòng XML.

Kiến trúc ADO.NET dựa trên hai yếu tố chính là : Dataset và .NET Framework data provider

Data provides gồm các thành phần sau :

Datasets

- Một tập hợp đầy đủ của CSDL bao gồm các bảng có liên quan , các ràng buộc và mối quan hệ của chúng
- Chức năng giống như truy cập dữ liệu từ xa từ dịch vụ Web XML
- Thao tác với dữ liệu động
- Xử lý dữ liệu theo kiểu không kết nối
- Cung cấp cho xem thứ bậc XML của dữ liệu quan hệ
- Xử dụng các công cụ XSLT và XPath Query để hoạt động trên dữ liệu

.NET Framework Data Provider bao gồm các thành phần sau đây để thao tác dữ liệu :

- Connection: Cung cấp kết nối với nguồn dữ liệu (database). Hay nói cách khác nó là đối tượng có nhiệm vụ thực hiện kết nối tới CSDL
- Command: Thực hiện các thao tác cần thiết với CSDL để lấy dữ liệu, sửa đổi dữ liệu hoặc thực hiện các thủ tục được lưu trữ. Hiểu đơn giản là nó đưa ra các mệnh lệnh đối với CSDL

- DataReader: Cung cấp việc đọc dữ liệu và chỉ đọc 1 dòng dữ liệu tại một thời điểm và nó đọc theo chiều tiến từ đầu đến cuối.

- DataApdater: Nó đóng vài trò như là cầu nối giữa Dataset và CSDL, tải dữ liệu lên dataset hoặc đồng bộ các thay đổi ở dataset về lại CSDL

- Datasets: Để lưu trữ, truy cập từ xa và lập trình với dữ liệu phẳng (Flat), dữ liệu XML và dữ liệu quan hệ

ADO.NET cho phép:

- Thao tác với CSDL trong cả hai môi trường là Connected data & Disconnected data.
- Làm việc tốt với XML
- Tương tác được với nhiều nguồn dữ liệu
- Làm việc trên môi trường internet

7.2 Lập trình với ADO.NET

Cơ sở dữ liệu rất quan trọng trong việc lưu trữ thông tin. Dữ liệu có rất nhiều nguồn và đa dạng. VB.NET được thiết kế với mục đích truy xuất, hiển thị, phân tích cơ sở dữ liệu. ADO.NET là mô hình lập trình truy xuất dữ liệu chung cho tất cả các ngôn ngữ và chương trình Windows. Với ADO.NET, chúng ta có thể truy xuất đến mọi hệ cơ sở dữ liệu theo cùng cách thức và mã chương trình như nhau.

7.2.1 Thuật ngữ về cơ sở dữ liệu

Một số thuật ngữ về cơ sở dữ liệu:

- Cơ sở dữ liệu là một file tổ chức thông tin thành các bảng gọi là bảng (Table).
- Mỗi bảng bao gồm nhiều hàng và cột, cột thường được gọi là trường (field) và dòng được gọi là bản tin (record).

Mô hình truy xuất cơ sở dữ liệu trong ADO.NET: thiết lập kết nối đến cơ sở dữ liệu. Tiếp theo đối tượng điều phối (data adapter) được tạo ra để truy vấn dl từ các bảng. Sau đó tạo các đối tượng DataSet chứa bảng dữ liệu mà chúng ta muốn trích dữ liệu.

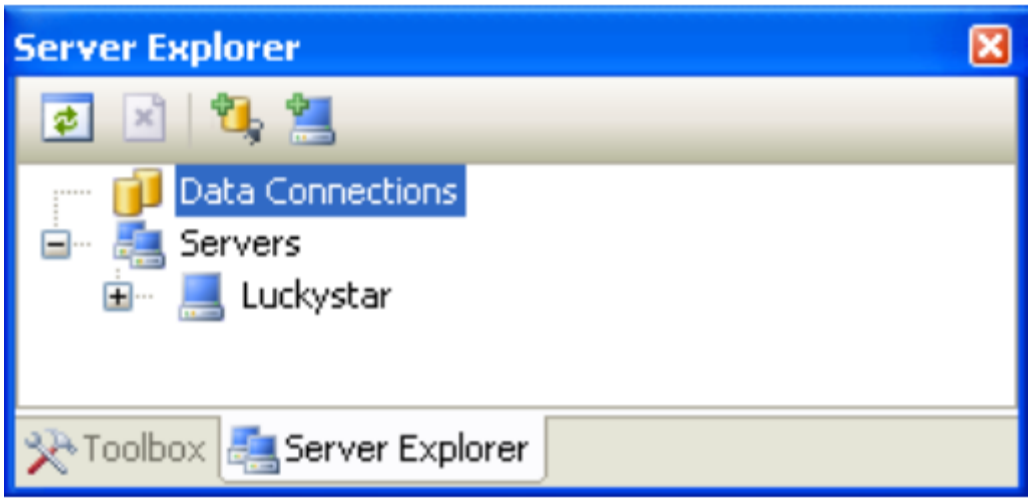
DataSet chỉ tạo bản sao của bảng dữ liệu mà thôi. Cuối cùng là gán thông tin trong DataSet vào các đối tượng hiển thị trên Form như TextBox, Label, Button, DataGrid,…

7.2.2 Làm việc với cơ sở dữ liệu Access

MS Access có tên Students.mdb. Sau khi đã biết cách kết nối và đưa dữ liệu vào dataset, chúng ta xây dựng và tích hợp chúng vào giao diện của form.

Chúng ta tạo mới một Solution có tên MyADOForm và thêm vào một dự án cùng tên.

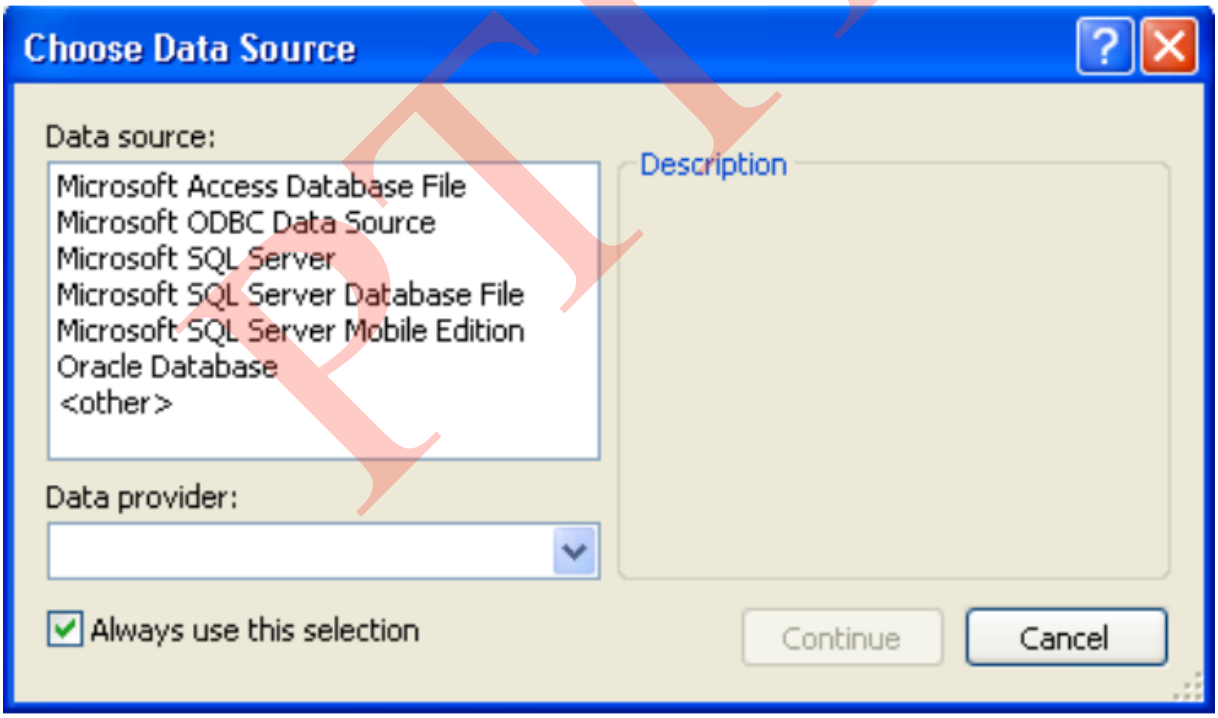
Chọn View | Server Explorer từ menu để hiện cửa sổ Server Explorer như hình sau:



Hình 7.1 Giao diện kết nối cơ sở dữ liệu cục bộ

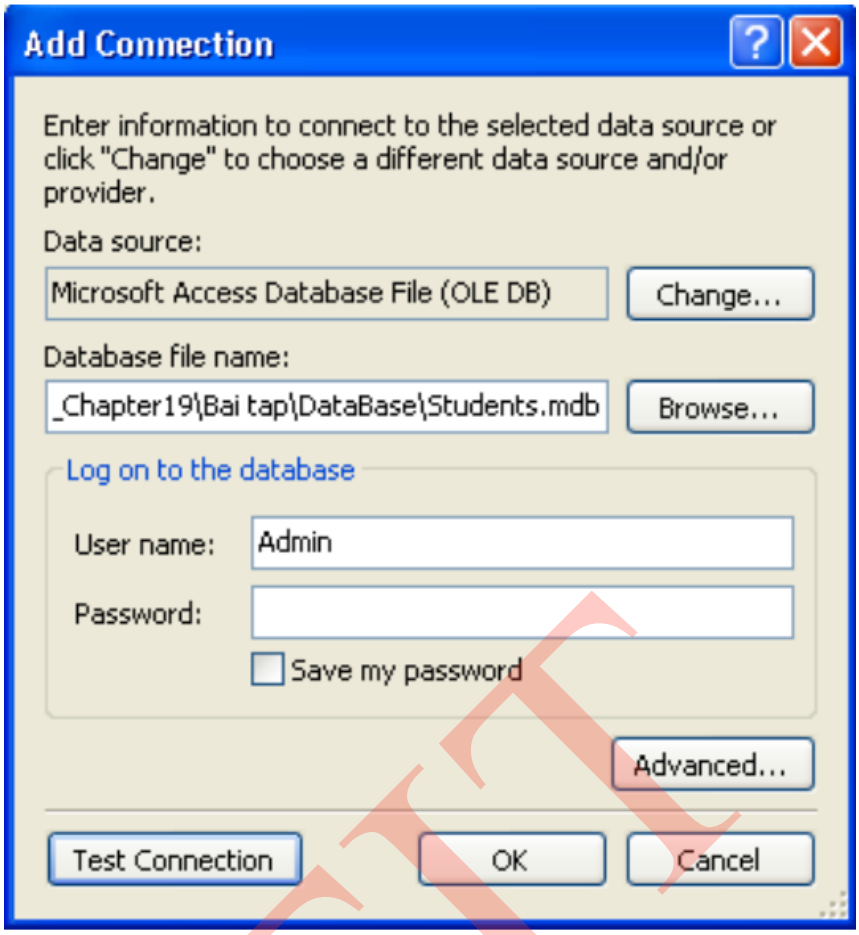
Đây là công cụ đồ họa cho phép kết nối đến cơ sở dữ liệu cục bộ, trên server theo mô hình client server. Ta có thể dùng nó để xem cấu trúc trong cơ sở dữ liệu, xem thuộc tính của bảng, kiểu dữ liệu của trường và bản tin trong cơ sở dữ liệu. Chúng ta có thể kéo các kết nối và bảng dữ liệu trong cửa sổ này để tạo ra đối tượng dữ liệu cho chương trình.

Chúng ta có thể dùng kéo thả, bằng cách tích chuột vào nút Connect To DataBase trong cửa sổ Server Explorer. Một hộp thoại Choose Data Source hiện ra cho phép ta chọn nguồn dữ liệu.



Hình 7.2 Giao diện chọn hệ cơ sở dữ liệu

Chúng ta chọn Microsoft Access DataBase File và chọn nút Continue, xuất hiện hộp thoại Add Connection như hình sau:



Hình 7.3 Giao diện chọn đường dẫn tệp cơ sở dữ liệu

Ta chọn đường dẫn đến cơ sở dữ liệu bằng cách chọn nút Browse, sau đó chọn cơ sở dữ liệu Students.mdb. Chúng ta có thể kiểm tra xem kết nối có thành công không bằng cách chọn nút Test Connection, và cũng có thể tùy chỉnh kết nối bằng cách chọn nút Advanced.

Chúng ta thấy dòng mã kết nối ở ô cuối cùng, dòng mã có nội dung như sau: "Provider=Microsoft.Jet.OLEDB.4.0;DataSource="D:\Data\Studying\VS.Net05\Bai tap\DataBase\Students.mdb"”

Nhấn OK để thêm kết nối vào Server Explorer.

7.2.3 Tạo bộ điều phối dữ liệu Data Adapter

Bước hai trong thao tác cơ sở dữ liệu là tạo bộ điều phối Data Adapter. Data Adapter sẽ định nghĩa chính xác những thông tin mà ta muốn lấy trong cơ sở dữ liệu, là nền tảng để tạo DataSet.

VB.NET cung cấp rất nhiều cách tạo bộ điều phối. Cách đơn giản nhất là ta kéo các biểu tượng bảng trong Server Explorer vào cửa sổ form trong chế độ thiết kế. Ta cũng có cách thứ hai dùng công cụ Data Adapter Configuration Winzard. Ta gọi đến công cụ này bằng cách chọn đối tượng OledbDataAdapter trên tab Data của ToolBox và đặt nó lên form.

7.2.4 Sử dụng đối tượng điều khiển OleDbDataAdapter

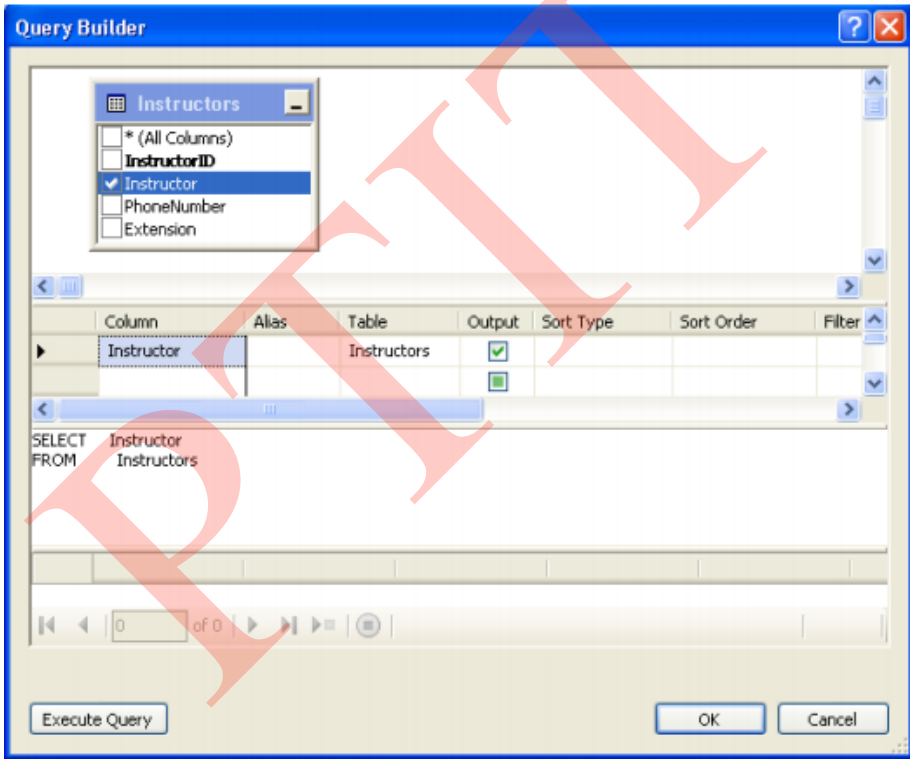
Chọn tab Data trong cửa sổ ToolBox. Tab này chứa các điều khiển để thao tác với cơ sở dữ liệu. Trong tab này có hai đối tượng OleDbConnection và sqlConnection đều cho phép tạo kết nối đến cơ sở dữ liệu. Nhưng chúng ta đã kết nối bằng Server Explorer nên không cần hai đối tượng này nữa.

Kéo đối tượng OleDbDataAdapter vào trong form. Nếu đối tượng này không xuất hiện, ta có thể thêm nó vào bằng cách tích chuột phải vào tab Data chọn Choose Item, xuất hiện cửa sổ Choose ToolBox Items. Chọn tab .Net Framework Components và chọn OleDbAdapter. Chọn OK để hoàn thiện việc thêm Item này cho ToolBox. Chúng ta cũng có thể làm tương tự với các đối tượng khác.

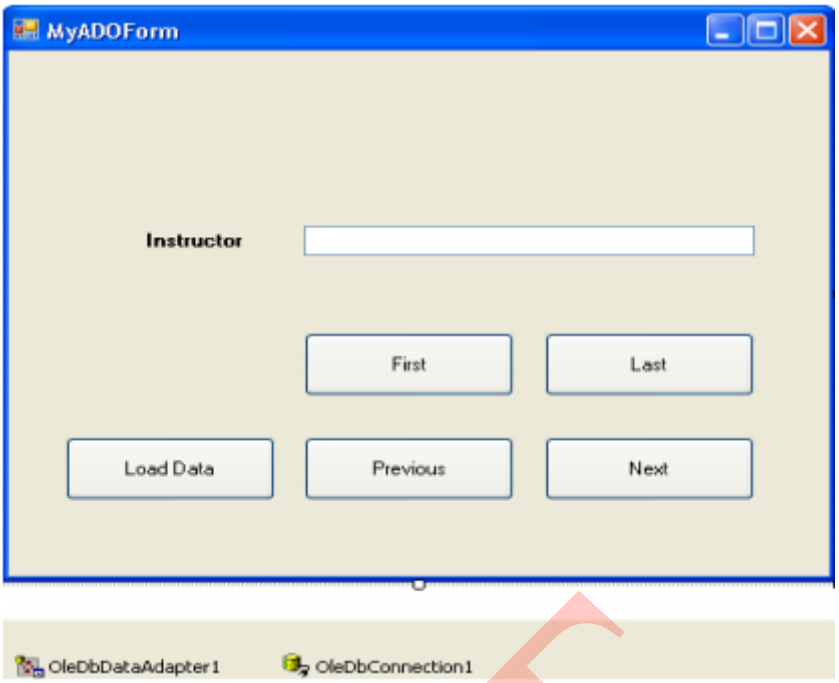
OleDbAdapter được thiết kế để kết nối đến cơ sở dữ liệu Access. Khi kéo thả đối tượng này vào form thì VS.NET sẽ tao trình Data Adapter Configuration Winzard. Một màn hình khởi đầu, chọn Next để chuyển sang màn hình thứ hai.

Ta chọn Next hai lần để xuất hiện màn hình soạn thảo câu lệnh SQL. Nếu chúng takhông dùng câu lệnh SQL, có thể nhấn vào nút Query Builder, để dùng kéo thả mà Visual Basic hỗ trợ. Sau đó chọn bảng Instructors, chọn Add, chọn Close để đóng cửa sổ này lại.

Ta thấy trong bảng Instructors có các ô CheckBox tương ứng với các trường, Query sẽ tạo câu lệnh tương ứng để rút thông tin của bảng. Trong ví dụ sau chúng ta chỉ rút thông tin từ một cột trong bảng, ta chọn cột Instructor như hình H.7.4, chọn OK. (câu lệnh SQL để trích rút dữ liệu).



Hình 7.4 Chọn trường để xây dựng câu lệnh SQL



Hình 7.5 Giao diện Form

Sau khi nhấn OK, một cửa sổ Generate The SQL Statement hiện ra hiển thị câu lệnh SQL ta vừa tạo. Chọn Finish đê hoàn thành việc tạo đối tượng điều phối.

7.2.5 Làm việc với DataSet

Chúng ta tạo ra đối tượng trình diễn dữ liệu cho người dùng thao tác, đối tượng DataSet. Nó là ảnh của cơ sở dữ liệu nên mọi thao tác của người dùng sẽ chưa ảnh hưởng đến cơ sở dữ liệu cho đến khi có yêu cầu cập nhật. Trong ví dụ ta tạo đối tượng DataSet trình diễn thông tin trong cột Instructor của bảng Instructors trong cơ sở dữ liệu Students.mdb.

Ta chọn phải chọn form1 nếu không chọn thì các lệnh tạo DataSet sẽ không hiển thị trên menu. Chọn Data | Generate DataSet từ menu để làm xuất hiện hộp thoại Generate DataSet. Chúng ta đặt tên nào tùy thích tại ô New, ví dụ là DsInstructors. Chọn ô checkBox Add this dataset to the designer để Visual Basic đưa dataser vào khay công cụ. Chọn OK và đối tượng DataSet DsINstructors được tạo ra trên khay công cụ. VB.NET sẽ tự thêm vào một file có tên DsInstructors.xsd trong cửa sổ Solution Explorer. File này chứa các thông tin về dữ liệu theo khuôn dạng XML.

7.3 Sử dụng các điều khiển ràng buộc dữ liệu

Chúng ta sử dụng các thành phần điều khiển như Textbox, Label, Button để đưa cơ sở dữ liệu lên form. Để trình bày được ta tạo mối ràng buộc dữ liệu (data binding), tức là dữ liệu hiển thị lên trong các điều khiển sẽ phụ thuộc vào nguồn dữ liệu có trong DataSet hay DataAdapter.

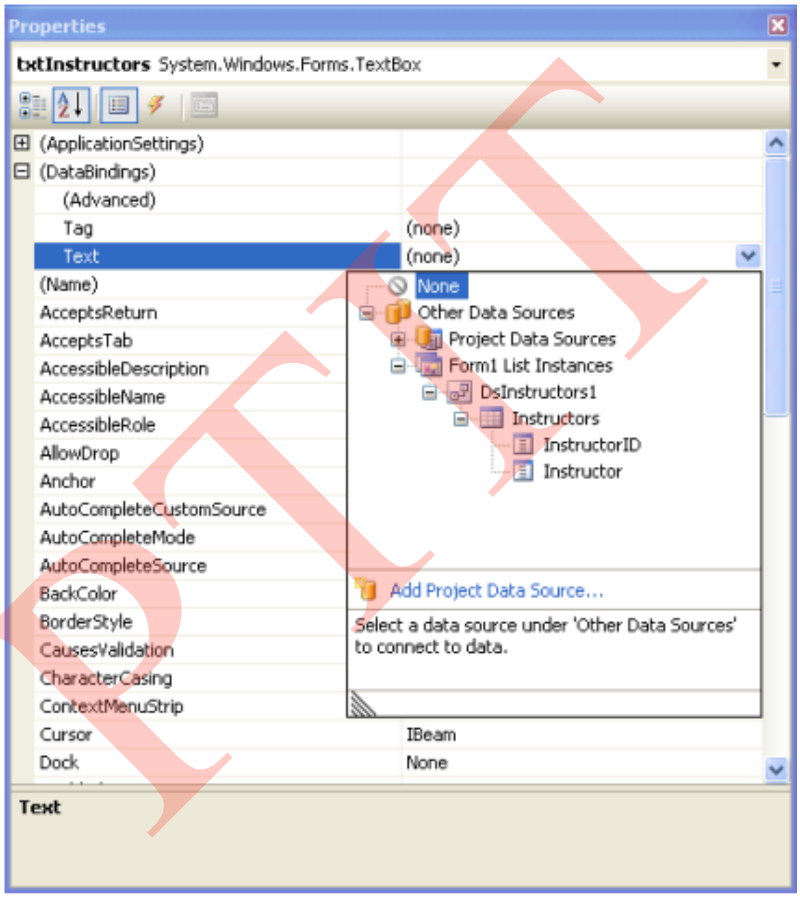
Chúng ta ràng buộc dữ liệu với các điều khiển sau: TextBox, Label, ListBox, ComboBox, RadioButon, DataGrid và PictureBox. Trong đó đặc biệt là DataGrid, nó cho phép ta hiển thị toàn bộ nội dung của DataSet.

Trong ví dụ ta ràng buộc dữ liệu vào TextBox để hiển thị thông tin trong bảng Instructors của cở dữ liệu Students.mdb. Chúng ta thiết kế giao diện form như hình 8.5 trên. Trong đó thuộc tính của các điều khiển như sau:

- Button First: Name – btnFirst, enable – False
- Button Last: Name – btnLast, enable – False
- Button Next: Name – btnNext, enable – False
- Button Previous: Name – btnPrevious, enable – False
- Button Load Data: Name – btnLoadData
- TextBox1: Name - txtInstructors

Các điều khiển còn lại có thuộc tính như hình.

Bây giờ ta sẽ tiến hành ràng buộc dữ liệu là các trường (cột dữ liệu – field) vào textbox txtInstructors. Để làm điều này, ta chọn ô textbox và mở Properties của nó ra. Tích chuột vào dấu (+) bên cạnh nhánh thuộc tính DataBindings và chọn ô text, tích chuột vào nút mũi tên đi xuống và ta có thể nhìn thấy nguồn dữ liệu DsInstructors1 hiển thị trong danh sách như hình sau:



Hình 7.6 Giao diện chọn nguồn dữ liệu và thuộc tính cho từng thành phần

Chọn cột Instructor để chỉ định trường này sẽ hiển thị trong ô textbox txtInstructor. Ta đã thực hiện ràng buộc xong, sau đó viết mã để xuất dữ liệu khi chương trình thực thi.

Chúng ta tạo thủ tục btnLoadData_Click bằng cách từ cửa sổ thiết kế form và nhấn đúp chuột vào nút Load Data rồi nhập đoạn mã chương trình :

```
DsInstructors1.Clear()
OleDbDataAdapter1.Fill(DsInstructors1)
btnFirst.Enabled = True
btnLast.Enabled = True
btnNext.Enabled = True
btnPrevious.Enabled = True
```

Đây là mã để xóa sạch dữ liệu trong DataSet DsInstructors1 ban đầu. Sau đó bộ điều phối DataAdapter1 cập nhật dữ liệu vào đối tượng DataSet DsInstructors1 mà chúng ta đã tạo ra ở bước 3 bằng phương thức Fill().

Chạy thử chương trình: chọn F5 để kiểm thử chương trình. Khi chương trình chạy, chọn nút Load Data để chương trình hiển thị bản ghi đầu tiên của trường Instructor trong bảng dữ liệu. Sau đó chúng ta mở rộng một số chức năng khác của ứng dụng cơ sở dữ liệu như duyệt các bản ghi, đếm và hiển thị số bản ghi hiện hành.

Các thao tác cơ sở dữ liệu gồm:

- Tạo kết nối đến cơ sở dữ liệu cần truy xuất.
- Tạo đối tượng điều phối DataAdapter.
- Tạo đối tượng trình diễn DataSet.
- Tạo ràng buộc dữ liệu vào các điều khiển hiển thị dữ liêu.

Ta có thể thêm các yêu cầu: cập nhật, thống kê, tìm kiếm, …

7.4 Tạo các điều khiển duyệt xem dữ liệu

Ở đây chúng ta chỉ dừng lại ở việc ràng buộc dữ liệu và hiển thị được bản ghi đầu tiên vào ô textbox. Trong phần tiếp theo chúng ta sẽ tạo ra các nút cho phép duyệt qua các bản ghi khác nhau, xem từ bản ghi đầu tiên đến cuối cùng.

ADO.NET cho phép quản lý và duyệt qua các bản ghi (record) bằng đối tượng CurrentManager. Với đối tượng này ta có thể biết được vị trí hiện hành của bản ghi, đi đến bản ghi sau cùng, trở về bản ghi đầu tiên, đến bản ghi kế, đến bản ghi trước. Mỗi DataSet đều có sẵn đối tượng CurrentManager và mỗi đối tượng form đều có thuộc tính BindingContext theo dõi tất cả đối tượng CurrentManager trên form.

Trong phần trên chúng ta đã tạo ra bốn nút chọn lần lượt First, Last, Next, Previous. Giờ chúng ta sẽ viết mã cho chúng. BindingContext, CurrentManager để duyệt qua các bản ghi.

Tạo thủ tục btnFirst_Click với nội dung như sau:

```
Me.BindingContext(DsInstructors1,"Instructors").Position = 0
btnFirst.Enabled = False
btnNext.Enabled = True
btnLast.Enabled = True
```

Mã này hiển thị bản ghi đầu tiên của DsInstructors1 sử dụng đối tượng BindingContext. Nó gán giá trị 0 cho thuộc tính Position (vị trí) để con trỏ hiện hành của dữ liệu chuyển đến bản ghi đầu tiên.

Tạo thủ tục btnLast_Click và nhập đoạn mã sau:

```
'Đếm tổng số bản ghi
Dim tongsobanghi As Integer = Me.BindingContext
(DsInstructors1, "Instructors").Count
'Chuyển con trỏ đến bản ghi cuối cùng
Me.BindingContext(DsInstructors1, "Instructors").Position =
tongsobanghi - 1
btnLast.Enabled = False
btnFirst.Enabled = True
btnPrevious.Enabled = True
```

```
btnNext.Enabled = False
```

Tạo thủ tục btnNext_Click và nhập vào đoạn mã sau:

```
'Đếm số bản ghi hiện hành
Dim tongsobanghi As Integer = Me.BindingContext
(DsInstructors1, "Instructors").Count
'Nếu chưa phải là bản ghi cuối thì next lên 1
If Me.BindingContext(DsInstructors1, "Instructors").Position <
tongsobanghi - 1 Then
Me.BindingContext(DsInstructors1, _
"Instructors").Position += 1
btnFirst.Enabled = True
btnPrevious.Enabled = True
btnLast.Enabled = True
Else
btnNext.Enabled = False
btnLast.Enabled = False
btnFirst.Enabled = True
btnPrevious.Enabled = True
End If
```

Thủ tục btnPrevious_Click:

```
'Nếu chưa phải là bản ghi đầu thì lùi lại 1
If Me.BindingContext(DsInstructors1,"Instructors").Position >
0 Then
Me.BindingContext(DsInstructors1,"Instructors").Position -= 1
btnFirst.Enabled = True
btnLast.Enabled = True
btnNext.Enabled = True
Else
btnFirst.Enabled = False
btnPrevious.Enabled = False
End If
```

Vậy là chúng ta đã tạo xong các nút cho phép duyệt qua các bản ghi. Chọn F5 để chạy chương trình. Ấn nút Load Data để hiển thị dữ liệu vào textbox. Ấn các phím để duyệt qua các bản ghi trong cơ sở dữ liệu. Chọn nút Close ở góc phải trên của form để đóng chương trình lại. Chúng ta sẽ tạo điều khiển label cho hiển thị vị trí bản ghi hiện hành để người dùng tiện quan sát.

7.5 Hiển thị vị trí của bản ghi hiện hành

Ngoài cơ chế duyệt xem các bản ghi, ta cũng cần cho người dùng biết đó là bản ghi thứ mấy. Chúng ta sẽ thêm một nhãn Label để hiển thị thứ tự của bản ghi. Từ thiết kế form và thêm vào một nhãn label1 có thuộc tính Name là lblIndexOfRecord, thuộc tính Text của nhãn là "Record 0 of 0". Ta tạo một thủ tục có tên count() ở ngay dưới phát biểu khai báo form1 như sau:

```
Private Sub Count()
```

```
Dim tongsobanghi, banghihienhanh As Integer
tongsobanghi = Me.BindingContext _
(DsInstructors1, "Instructors").Count
banghihienhanh = Me.BindingContext _
(DsInstructors1, "Instructors").Position + 1
lblIndexOfRecord.Text = "Record " & _
banghihienhanh.ToString & "Of " & tongsobanghi.ToString
End Sub
```

Thủ tục này sẽ gán thuộc tính count của đối tượng BindingContext vào biến tongsobanghi và thuộc tính Position của nó cho biến banghihienhanh nhưng cộng thêm 1 vì thứ tự bản ghi trong bảng dữ liệu được tính từ 0. Sau đó hai giá trị của hai biến trên được gán cho thuộc tính Text của điều khiển Label lblIndexOfRecord.

Để thủ tục này phát huy tác dụng thì bạn sẽ thêm lời gọi thủ tục này trong các thủ tục khác như btnFirst_Click, btnLast_Click, btnPrevious_Click, btnNext_Click như sau: Count()

Chương trình của chúng ta đến đây là hoàn thiện. Chọn F5 để chạy chương trình.

7.6 Sử dụng DataGrid để hiển thị dữ liệu trong bảng

Trong phần này chúng ta dùng DataGrid để hiển thị dữ liệu của bảng trong cơ sở dữ liệu Students.mdb. Chúng ta điền đầy đủ nội dung khung lưới bằng dữ liệu của bảng ở dạng chuỗi sau đó thực hiện một số thao tác định dạng, sắp xếp và ghi lại những thay đổi trong DataGrid trở lại cơ sở dữ liệu. Giống như TextBox, Ta có thể ràng buộc dữ liệu trong DataSet vào DataGrid. Ràng buộc này thông qua hai thuộc tính là DataSource và DataMember. Trong MyDataGridBinding chúng ta đưa toàn bộ nội dung của bảng Instructors có trong DsInstructors1 hiển thị trong khung lưới DataGrid.

Ví dụ MyDataGridBinding:

Tạo mới một Solution và thêm vào một dự án cùng tên là MyDataGridBinding.

Kết nối cơ sở dữ liệu:

Nếu trong ví dụ trước chúng ta đã kết nối với cơ sở dữ liệu rồi thì trong cửa sổ Server Explorer sẽ có một kết nối đến cơ sở dữ liệu nhưng có thêm một gạch đỏ ở kết nối đó. Nếu muốn sử dụng lại kết nối này ta chỉ việc chọn nút Refresh. Trong Ví dụ này, giả sử ta copy file cơ sở dữ liệu Students.mdb vào cùng thư mục với dự án để tiện thao tác. Ta chọn nút để thực hiện kết nối đến cơ sở dữ liệu như đã biết. Chọn cơ sở dữ liệu mà chúng ta vừa copy vào thư mục chứa dự án. Chọn OK để hoàn thành kết nối.

Tạo đối tượng điều phối DataAdapter:

Tạo thêm đối tượng OleDbDataAdapter vào trong form bằng cách kéo nó từ ToolBox ở tab data vào trong form. Khi đó một cửa sổ Data Adapter Configuration xuất hiện.

Chọn Next hai lần để hiện cửa sổ Generate SQL Statements. Tại đây Ta có thể tự gõ câu lệnh SQL hay sử dụng Query Builder. Ví dụ ta chọn nhập trực tiếp câu lệnh SQL.

```
SELECT Extension, PhoneNumber, Instructor, InstructorID
FROM Instructors
```

Câu lệnh này lấy dữ liệu ở cả bốn trường trong bảng Instructors. Chọn Next để xem kết quả của Winzard. Lúc này, trình Winzard tự tạo ra các câu lệnh là Update (cập nhật), Select,

Insert (chèn), Delete (xóa). Chọn Finish để kết thúc quá trình xây dựng tạo đối tượng điều phối DataAdapter có tên OleDbDataAdapter1.

Tạo đối tượng trình diễn DataSet:

Từ Form: Chọn Data | Generate DataSet từ menu làm hiện hộp thoại Generate DataSet như đã biết. Tại ô New bạn nhập vào tên DsInstructors và đánh dấu vào ô checkBox Add this DataSet To The Designer để Visual studio tạo ra đối tượng DataSet và đưa nó vào khay hệ thống. Chọn OK để Visual studio tạo đối tượng DataSet cho bảng Instructors trong cơ sở dữ liệu Students.mdb. Lúc này cửa sổ form có thêm các đối tượng vừa chọn.

Chúng ta đã hoàn thành ba bước đầu của thao tác với cơ sở dữ liệu Tiếp sử dụng DataGrid để trình bày dữ liệu.

Tạo đối tượng DataGrid:

Kéo form cho kích thước rộng hơn để chứa đủ khung lưới DataGrid với 4 cột và 10 dòng. Đưa điều khiển DataGrid trên ToolBox vào trong form. Kéo chiều dài của nó cho phù hợp với chiều kích thước của form.

Tạo thêm một nút nhấn nữa vào form. Đặt thuộc tính Name là btnLoad và text là "Load Data". Mở Properties của DataGrid và đặt thuộc tính Anchor của nó là cả Left, Right, Top, Bottom.

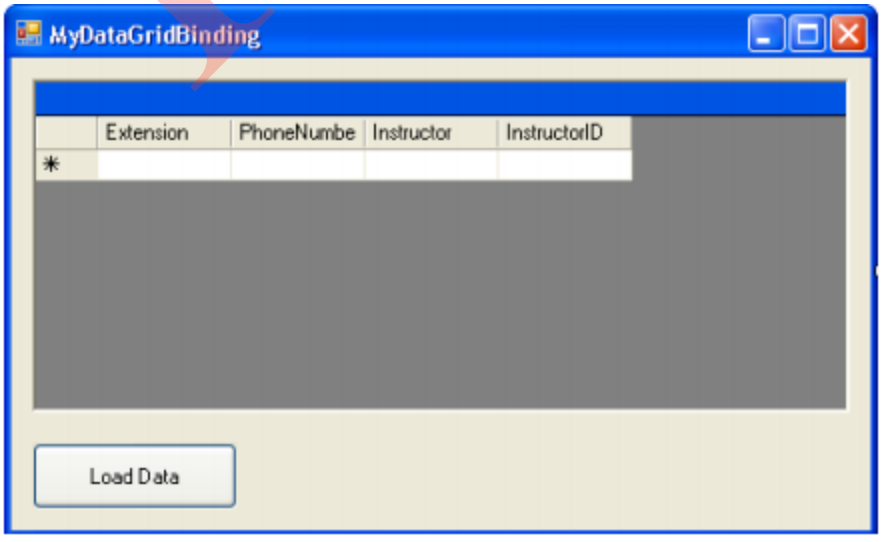
Tiếp theo ta sẽ dùng thuộc tính DataSource và DataMember để ràng buộc dữ liệu trong DsInstructors1 vào khung lưới DataGrid. Chúng ta hiển thị các tùy chọn của thuộc tính DataSource trong cửa sổ Properties. Một chương trình có thể có rất nhiều DataSet nhưng tại một thời điểm khung lưới chỉ có thể thể hiện một DataSet mà thôi, ta chọn DsInstructors1.

Tiếp chọn thuộc tính DataMember là Instructors.

Ngay sau khi ta chọn xong hai thuộc tính DataSource và DataMember thì khung lưới sẽ hiển thị các cột dữ liệu dù chưa có dòng dữ liệu nào hiển thị. Dữ liệu sẽ được đưa vào khung lưới khi chương trình thực thi.

Chọn nút Load Data và đặt thuộc tính Anchor của nó là Bottom, Left. Lúc này giao diện form thiết kế sẽ như hình H. 7.7.

Sau đó, chúng ta cần viết mã để đổ dữ liệu vào khung lưới bằng phương thức Fill như đã viết trong mục 7.5.



Hình 7.7 Cửa sổ form khi thiết kế xong

Tạo thủ tục btnLoad_Click bằng cách double click vào nút LoadData và nhập mã sau:

```
DsInstructors1.Clear()
OleDbDataAdapter1.Fill(DsInstructors1)
```

Nhấn nút Save All để lưu lại các thay đổi và chạy thử chương trình. Chọn F5 để chạy chương trình. Chọn nút Load Data để nạp dữ liệu vào trong khung lưới DataGrid: Ta có thể kéo để thay đổi kích thước form sao cho các thông tin về cơ sở dữ liệu xuất hiện đầy đủ. Chọn nút Close để đóng chương trình.

7.7 Định dạng các ô lưới trong DataGrid

Chúng ta định dạng các thành phần trong DataGrid thông qua thuộc tính của nó lúc thiết kế hay khi thực thi chương trình.. Với ví dụ trên ta làm như sau: từ cửa sổ thiết kế form và mở thuộc tính Properties của khung lưới DataGrid.

- Đặt thuộc tính PreferredColumnWidth là 110 (rộng 110 đơn vị đo Pixel).
- Đặt thuộc tính ColumnHeadersVisible là False. Với thiết lập này thì phần tiêu đề của các cột sẽ không hiển thị.
- Chọn thuộc tính BackColor, chọn màu vàng nhạt hiển thị cho nội dung chuỗi chứa trong ô lưới tạo các dòng xen kẽ nhau.
- Đặt thuộc tính GridLineColor, chọn màu xanh.

Còn rất nhiều thuộc tính khác ta ta có thể lựa chọn.

7.8 Cập nhật cơ sở dữ liệu trở lại bảng

DataSet chỉ tiến hành sao chép bảng của cơ sở dữ liệu, chứ không làm thay đổi nội dung cơ sở dữ liệu cho đến khi có yêu cầu cập nhật bằng phương thức UpDate. Cùng với thuộc tính ReadOnly của DataSet sẽ cho phép có thay đổi hay không với cơ sở dữ liệu.

Từ cửa sổ thiết kế form mở thuộc tính properties của DataGrid và thiết lập giá trị TRUE đối với thuộc tính ReadOnly cho phép có những thay đổi dữ liệu trong khung lưới.

Tạo một nút nữa trên form. Thuộc tính: Name – btnUpdate, Text – "Update". Nút Update sẽ hiển thị khi có những thay đổi trong DataGrid và khi cập nhật trở lại cơ sở dữ liệu khi người dùng tích chuột vào nó.

Tạo thủ tục btnUpdate_Click và nhập mã như sau:

```
Try
OleDbDataAdapter1.Update(DsInstructors1)
Catch ex As Exception
MsgBox(ex.ToString)
End Try
```

Thủ tục này sử dụng phương thức Update của OleDbDataAdapter1 để yêu cầu các thay đổi trong tập DataSet DsInstructors1 trở lại bảng cở dữ liệu.

Khi chạy chương trình, ta thay đổi nội dung một cột nào đó hay có thể thêm một bản ghi nữa và chọn nút Update để cập nhật vào cơ sở dữ liệu. Sau đó lại chọn nút Load Data để xem cơ sở dữ liệu có thay đổi gì không.