Đĩa Microsoft Visual Studio 2008 Express Editions đính kèm

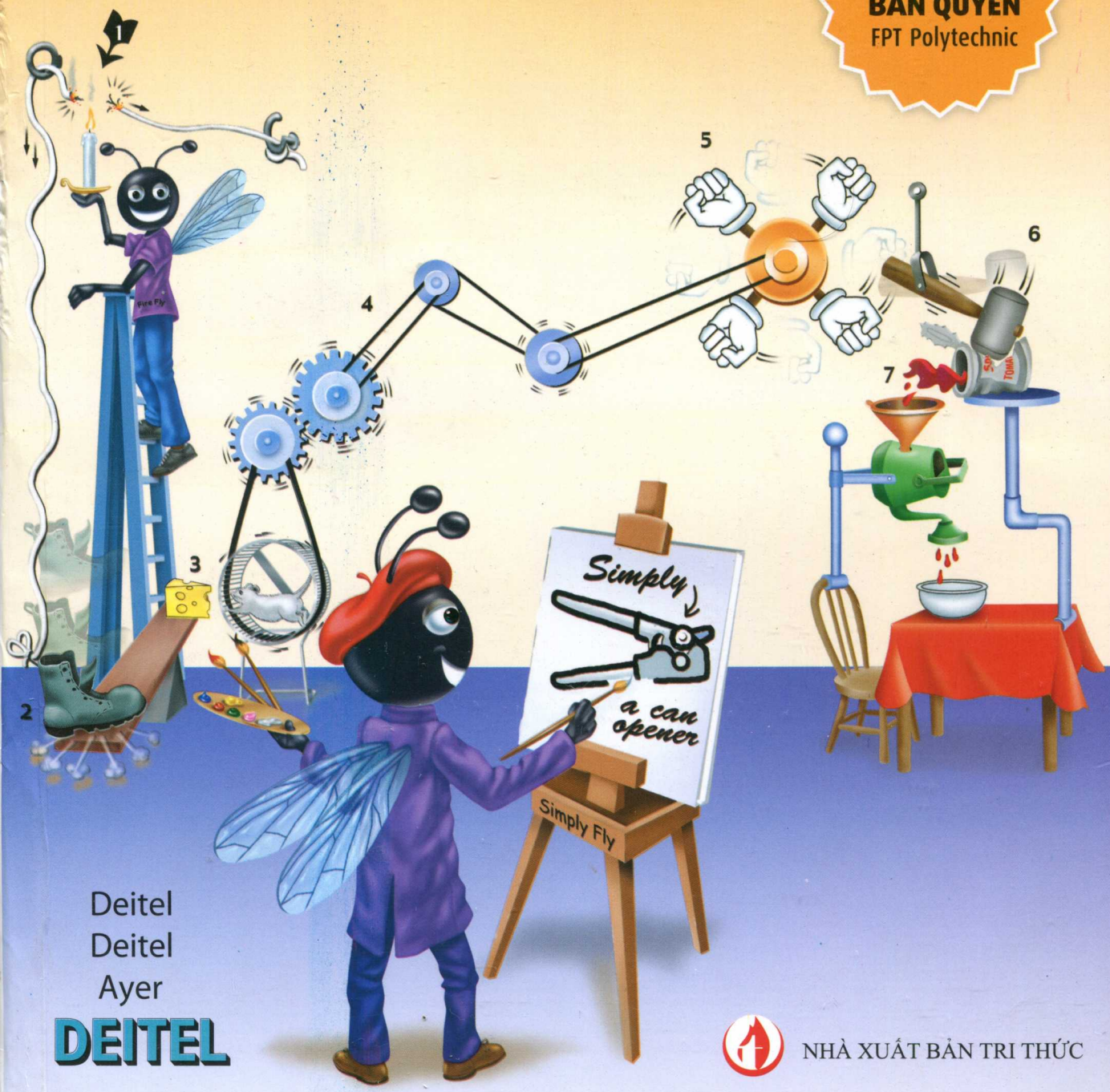
Lập trình trực quan
Visual Basic® 2008

PHIÊN BẢN LẦN 3

Simply Visual Basic 2008, Third Edition

Phương pháp tiếp cận dựa trên ứng dụng

TỦ SÁCH BẢN QUYỀN
FPT Polytechnic



Deitel
Deitel
Ayer
DEITEL

NHÀ XUẤT BẢN TRI THỨC

Lập trình trực quan
Visual Basic 2008
Phiên bản lần ba

PEARSON

Fpt University
FPT POLYTECHNIC

Giới thiệu sản phẩm

Dòng sách cơ bản

Simply C++: An Application-Driven Tutorial Approach

Simply C#: An Application-Driven Tutorial Approach

Simply Java Programming: An Application-Driven Tutorial Approach

Simply Visual Basic 2005, 2/E: An Application-Driven Tutorial Approach

Dòng sách hướng dẫn lập trình

Internet & World Wide Web How to Program, 4/E

Java How to Program, 7/E

C++ How to Program, 6/E

C How to Program, 5/E

Visual Basic 2005 How to Program, 3/E

Visual C# 2005 How to Program, 2/E

Small Java How to Program, 6/E

Small C++ How to Program, 5/E

Advanced Java 2 Platform How to Program

XML How to Program

Visual C++ 2008 How to Program, 2/E

Perl How to Program

Python How to Program

Sách hiện đang bán

Sách trên Website SafariX
`www.deitel.com/books/SafariX.html`

C++ How to Program, 5/E & 6/E

Java How to Program, 6/E & 7/E

Simply C++: An Application-Driven Tutorial Approach

Simply Visual Basic 2005: An Application-Driven Tutorial Approach, 2/E

Small C++ How to Program, 5/E

Small Java How to Program, 6/E

Visual Basic 2005 How to Program, 3/E

Visual C# 2005 How to Program, 2/E

Đăng ký nhận email tin tức DEITEL BUZZ ONLINE về kế hoạch xuất bản của Deitel tại:
`www.deitel.com/newsletter/subscribe.html`

Liên hệ với tác giả qua địa chỉ email:
`deitel@deitel.com`

Xem thông tin về các buổi hội thảo do công ty Deitel & Associates, Inc. worldwide tổ chức tại:
`www.deitel.com/training/`

hoặc gửi mail về địa chỉ:
`deitel@deitel.com`

Theo dõi các thông tin xuất bản cập nhật của Prentice Hall/Deitel tại:
`www.deitel.com`
`www.prenhall.com/deitel`

Nhận các tài nguyên web có sẵn từ "Trung tâm tài nguyên" giúp bạn trở thành chuyên gia Visual Basic, các ngôn ngữ lập trình quan trọng khác, phần mềm và Web 2.0 tại:
`www.deitel.com/ResourceCenters.html`

PEARSON

Lập trình trực quan
Visual Basic 2008
Phiên bản lần ba

Simply Visual Basic 2008, Third Edition

Bản dịch tiếng Việt
(Tái bản lần hai)

P. J. Deitel
Deitel & Associates, Inc.

H. M. Deitel
Deitel & Associates, Inc.

G. J. Ayer

Dịch thuật và Hiệu đính
Fpt Polytechnic

NHÀ XUẤT BẢN TRI THỨC

Fpt University
FPT POLYTECHNIC

PEARSON

Fpt University
FPT POLYTECHNIC

Title: Simply Visual Basic 2008, Third Edition
Author: P.J. Deitel, H.M. Deitel and G.J. Ayer

ISBN: 978-604-908-357-0

Tên sách: Lập trình trực quan Visual Basic 2008
Tác giả: P.J. Deitel, H.M. Deitel và G.J. Ayer

ISBN: 978-604-908-660-1

Liên hệ hợp tác về nội dung bản dịch tiếng Việt và phân phối tại Việt Nam:
Văn phòng FPT Polytechnic Việt Nam
Tòa nhà FPT Polytechnic, Đường Hàm Nghi, KĐT Mỹ Đình I, Từ Liêm, Hà Nội
Điện thoại: (04) 7 305 9886 - (04) 7 308 0898
Email: caodang@fpt.edu.vn
Website: http://www.poly.edu.vn

Lời đề tặng của tác giả

Dành tặng Carole Snyder, công ty Prentice Hall:

Xin cám ơn Carole vì những nỗ lực phi thường trong việc tìm kiếm, điều phối hỗ trợ đội ngũ giáo sư cũng như nhà chuyên môn đóng góp phê bình cho cuốn sách của chúng tôi không quản sức ép thời gian trong suốt nhiều năm qua.

Paul and Harvey

Dành tặng người cha đáng kính của tôi:

Cha đã hy sinh cho con nhiều hơn những gì con từng biết,
Và làm cho con nhiều hơn những gì cha từng nghĩ
Con yêu cha.

Greg

Thương hiệu:

DEITEL, hình ảnh con rệp với hai ngón tay cái giơ lên và khẩu hiệu thương mại (slogan) DIVE INTO được đăng ký thương hiệu bởi Deitel and Asociates, Inc.

Microsoft, Silverlight, Visual Studio, Visual Basic và Visual Web Developer được tập đoàn Microsoft đăng ký thương hiệu tại Mỹ và nhiều nước khác.

"Trung tâm tài nguyên" của Deitel

"Trung tâm tài nguyên" (Resource Centers) chứa một số lượng lớn nội dung trực tuyến bao gồm các tài nguyên, tài nguyên được phép tải về, những bài hướng dẫn, tài liệu, sách, sách điện tử, tạp chí, bài báo, blog, RSS feed và nhiều đề tài công nghệ và lập trình "nóng" nhất hiện nay. Hãy ghé thăm website dưới đây để truy cập các trung tâm tài nguyên được cập nhật thường xuyên.

```
www.deitel.com/ResourceCenters.html
```

Hãy đề xuất ý tưởng về những tài nguyên muốn sử dụng trên "Trung tâm tài nguyên"! và đăng ký nhận email thông báo miễn phí DEITEL BUZZ ONLINE tại:

```
www.deitel.com/newsletter/subscribe.html
```

Các nội dung trên "Trung tâm tài nguyên"

Dive Into Web 2.0 eBook

Computer Science
Regular Expressions
Games and Game Programing

Computer Game Programming
Computer Games
Mobile Gaming
Sudoku

Programming
ADO.NET
Adobe Flex
Ajax
Amazon Web Services
Apex
ASP.NET
ASP.NET 3.5
ASP.NET Ajax
C
C++
C++ Boost Libraries
C++ Game Programming
C#
Cloud Computing
Code Search Engines and Code Sites
Computer Game Programming
CSS 2.1
Dojo
Facebook Developer Platform
Flash 9
Java
Java Certification and Assessment Testing
Java Design Pattens
Java EE 5
Java SE 6
Java SE 7 (Dolphin) Resource Center
JavaFX
JavaScript
JSON
Microsoft LINQ
Microsoft Popfly
MySpace Developer Platform
.NET
.NET 3.0
.NET 3.5
OpenGL
OpenSocial
Perl
PHP
Programming Projects
Python
Regular Expressions
Refactoring
REST Web Services
Ruby
Ruby on Rails
Service-Oriented Architecture (SOA)
Silverlight
Visual Basic
Visual Basic 2008
Visual C++
Visual C# 2008 and C# 3.0
Visual Studio Team System
Web 3D Technologies
Web Services
Windows Presentation Foundation
XHTML
XML

Internet Business
Affiliate Programs
Competitve Analysis
Facebook Social Ads
Google AdSense
Google Analytics
Google Services
Internet Advertising
Internet Business Initiative
Internet Public Relations
Link Building
Location-Based Services
Online Lead Generation
Podcasting
Search Engine Optimization
Selling Digital Content
Sitemaps
Web Analytics
Website Monetization
YouTube and AdSense

Java
Java
Java Certification and Assessment Testing
Java Design Pattens
Java EE 5
Java SE 6
Java SE 7 (Dolphin) Resource Center
JavaFX

Microsoft
ADO.NET
ASP.NET
ASP.NET 3.5
ASP.NET Ajax
DotNetNuke (DNN)
Internet Explorer 7 (IE7)
Microsoft LINQ
Microsoft Popfly
.NET
.NET 3.0
.NET 3.5
SharePoint
Silverlight
Silverlight 2.0
SQL Server 2008
Visual Basic
Visual Basic 2008
Visual C ++
Visual C# 2008 and C # 3.0
Visual Studio Team System
Windows Communication Foundation
Windows Presentation Foundation
Windows Vista

Open Source and LAMP Stack
Apache
DotNetNuke (DNN)
Eclipse
Firefox
Linux
MySQL
Open Source
Perl
PHP
Python
Ruby

Software
Apache
DotNetNuke (DNN)
Eclipse
Firefox
Internet Explorer 7 (IE7)
Linux
MySQL
Open Source
Search Engines
SharePoint
Skype
Web Servers
Wikis
Windows Vista

Web 2.0
Avatars
Alert Services
KNOL
Attention Economy
Blogging
Building Web Communities
Community Generated Content
Facebook Developer Platform
Facebook Social Ads
Google Base
Google Video
Google Web Toolkit (GMT)
Internet Video
Joost
Location-Based Services
Mashups
Microformats
Recommender Systems
RSS
Social Graph
Socidal Media
Social Networking
Software as a Service (SaaS)
Virtural Worlds
Web 2.0
Web 3.0
Widgets

Other Topics
Computer Games
Computer Jobs
Gadgets and Gizmos
Ring Tones
Sudoku

Hướng dẫn tra cứu gói tài nguyên kèm theo sách

Khi tìm hiểu cuốn sách, các bạn lưu ý đọc hướng dẫn ở mục *Tải về các đoạn code ví dụ* trong phần ***Chuẩn bị*** để tải về gói tài nguyên kèm theo sách cho phần ví dụ và bài tập ở mỗi chương.

Trong phiên bản dịch, chúng tôi đã tiến hành lược bỏ một số chương, đồng thời thay đổi thứ tự các chương so với sách gốc. Do đó, trong quá trình sử dụng gói tài nguyên kèm theo sách, các bạn lưu ý chọn thư mục tài nguyên trong thư mục tải về theo đúng thứ tự tương ứng như sau:

Sách chuyển thể	Thư mục tài nguyên
Chương 1	Tutorial01
Chương 2	Tutorial02
Chương 3	Tutorial03
Chương 4	Tutorial04
Chương 5	Tutorial05
Chương 6	Tutorial06
Chương 7	Tutorial07
Chương 8	Tutorial08
Chương 9	Tutorial09
Chương 10	Tutorial10
Chương 11	Tutorial11
Chương 12	Tutorial12
Chương 13	Tutorial13
Chương 14	Tutorial14
Chương 15	Tutorial15
Chương 16	Tutorial16
Chương 17	Tutorial17
Chương 18	Tutorial18
Chương 19	Tutorial19
Chương 20	Tutorial21
Chương 21	Tutorial22
Chương 22	Tutorial23

Mục lục chung

MỤC LỤC

LỜI NÓI ĐẦU

Chào mừng đến với ngôn ngữ lập trình Visual Basic và thế giới của Microsoft Windows, lập trình web và lập trình mạng trên nền tảng .NET 3.5 của Microsoft! Deitel & Associates chúng tôi viết sách giáo khoa và sách chuyên môn về ngôn ngữ lập trình cho Prentice Hall, cung cấp các khóa học dành cho các doanh nghiệp trên toàn cầu và phát triển các ứng dụng Internet trên nền Web 2.0. Cuốn sách này nằm trong chuỗi sách *Simply*, đã được cập nhật theo phiên bản Visual Studio 2008 và .NET 3.5. Mục đích chính của chúng tôi là viết một cuốn sách tập trung vào khái niệm và tính năng nhưng với cách tiếp cận đơn giản nhất có thể. Cuốn sách hướng đến đối tượng độc giả sử dụng Windows 7 hoặc Windows XP.

Để đạt được mục tiêu đó, chúng tôi đã xây dựng phương pháp giảng dạy mới. Chúng tôi giới thiệu khái niệm cốt lõi của công nghệ hàng đầu với cách tiếp cận HƯỚNG ỨNG DỤNG dưới dạng bài hướng dẫn, kết hợp với cách tiếp cận CODE TRỰC TIẾP của riêng DEITEL để giảng dạy lập trình bằng các ứng dụng hoàn thiện, chạy được và thực tiễn. Chúng tôi đã kết hợp quan điểm của sách hướng dẫn thực hành với quan điểm của sách giáo khoa để tạo nên một cuốn sách vừa phù hợp đối với lớp học truyền thống lại vừa phù hợp đối với các sinh viên ngồi bên máy tính tự đọc sách và xây dựng ứng dụng ví dụ giống như đang đọc bài hướng dẫn. Ngoài ra, cuốn sách này cũng phù hợp cho các khóa học trực tuyến.

Trong quá trình đọc cuốn sách này, sinh viên sẽ học về những khái niệm cơ bản trong lập trình, thành phần giao diện người dùng đồ họa (GUI - graphical user interface) và cách xử lý với file. Hầu hết các phần đều có mục câu hỏi tự ôn tập và đáp án để sinh viên có thể kiểm tra lại kiến thức ngay lập tức.

Tất cả những nội dung kể trên đều được xem xét cẩn thận bởi các học giả, các nhà phát triển và các thành viên của nhóm Microsoft Visual Basic. Trong đó các thành viên của nhóm Microsoft Visual Basic đã làm việc cùng chúng tôi trong cuốn *Lập trình trực quan Visual Basic 2008, phiên bản lần ba* này.

Chúng tôi tin rằng cuốn sách này và những tài liệu hỗ trợ sẽ giúp sinh viên và giảng viên học tập Visual Basic một cách dễ dàng và thú vị. Chúng tôi cung cấp bộ tài liệu bổ sung giúp giảng viên có thể tối đa hóa chất lượng học tập cho sinh viên của mình.

Mọi thắc mắc trong quá trình đọc cuốn sách này xin gửi đến địa chỉ email

```
deitel@deitel.com
```

Chúng tôi sẽ hồi đáp cho bạn ngay khi có thể. Các sửa đổi, bổ sung của cuốn sách này và phần mềm Visual Basic hỗ trợ có thể được tìm thấy tại

```
www.deitel.com/books/SimplyVB20008/
```

Hãy đăng ký để nhận được email tin tức *DEITEL BUZZ ONLINE* miễn phí tại

```
www.deitel.com/newsletter/subscribe.html
```

Bạn cũng có thể theo dõi danh sách ngày càng tăng các trung tâm tài nguyên (Resource Center) Visual Basic và các chủ đề liên quan tại

```
www.deitel.com/ResourceCenters.html
```

Mỗi tuần chúng tôi sẽ thông báo những trung tâm tài nguyên mới nhất qua email cho bạn.

***Các điểm bổ sung, sửa đổi trong cuốn* Lập trình trực quan Visual Basic 2008, phiên bản lần ba**

Đây là một số bổ sung, sửa đổi chính trong phiên bản thứ 3 của cuốn *Lập trình trực quan Visual Basic 2008.*

- ***LINQ***. (Language Integrated Query - truy vấn tích hợp) là tính năng mới quan trọng bậc nhất trong Visual Basic 2008 và Visual C# 2008. LINQ cung cấp cú pháp thống nhất để truy vấn dữ liệu bao gồm các thao tác chèn, cập nhật, xóa. Định kiểu mạnh mẽ nên Visual Studio có thể cung cấp hỗ trợ *IntelliSense* cho các thao tác của LINQ và kết quả trả về của truy vấn. LINQ có thể truy vấn trên nhiều kiểu nguồn dữ liệu khác nhau như collection (bạn sẽ được học trong Chương 20 và 22), cơ sở dữ liệu và XML. Chúng tôi cũng giới thiệu thêm rất nhiều tính năng mới của ngôn ngữ Visual Basic hỗ trợ cho LINQ.
- ***Biểu thức điều kiện `If`***. Visual Basic cung cấp biểu thức điều kiện `If` mới (sẽ được trình bày ở Chương 7), cú pháp gồm điều kiện, biểu thức đúng và biểu thức sai. Điều kiện sẽ được kiểm tra, sau đó biểu thức đúng hoặc biểu thức sai sẽ được tính toán tùy thuộc vào kết quả kiểm tra điều kiện là đúng hay sai. Biểu thức này được sử dụng để viết tắt cho lệnh `If... Then... Else`.
- ***Tự động xác định kiểu cục bộ***. Nếu bạn khởi tạo biến cục bộ trong lúc khai báo, bạn có thể không cần khai báo kiểu biến - trình biên dịch sẽ tự động xác định kiểu của biến từ giá trị khởi tạo cho biến (được trình bày ở Chương 11).
- ***Tham số tùy chọn***. Bạn có thể chỉ ra giá trị mặc định cho tham số của phương thức - nếu tham số của phương thức không được cung cấp giá trị trong lời gọi phương thức, trình biên dịch sẽ tự động chèn vào giá trị mặc định của tham số tùy chọn trong lời gọi (được trình bày ở Chương 13).
- ***Khởi tạo đối tượng***. Khi tạo đối tượng mới, bạn có thể sử dụng cú pháp khởi tạo đối tượng mới để gán giá trị cho các thuộc tính của đối tượng mới tạo (được giới thiệu trong Chương 22).
- ***Cửa sổ "sửa lỗi nhanh"***. IDE đã cung cấp cửa sổ **Error Correction Options (tùy chọn sửa lỗi)** cho phép bạn có thể nhanh chóng khắc phục một số lỗi lập trình thông thường (được giới thiệu ở Chương 5).

***Tính sư phạm của cuốn sách* Lập trình trực quan Visual Basic 2008, phiên bản lần ba**

Tính sư phạm của cuốn sách được thể hiện qua những đặc điểm sau:

- ***Cách tiếp cận HƯỚNG ỨNG DỤNG dưới dạng hướng dẫn***. Mỗi chương sử dụng một ứng dụng thực tế để minh họa các khái niệm lập trình. Các ví dụ và bài tập là các ứng dụng desktop, Internet và Web được cập nhật mới nhất. Bảng liệt kê ứng dụng được trình bày trong Hình 1. Hầu hết các ví dụ đều tập trung vào doanh nghiệp, gia đình hoặc cá nhân. Đầu mỗi chương, sinh viên sẽ chạy thử ứng dụng đã hoàn chỉnh để quan sát cách thức làm việc của ứng dụng. Sau đó sinh viên sẽ tiến hành xây dựng ứng dụng theo hướng dẫn chi tiết từng bước. Cuốn sách tập trung vào các nguyên tắc trong công nghệ phần mềm và nhấn mạnh sự sáng sủa trong lập trình.
- ***Cách tiếp cận CODE TRỰC TIẾP***. Cuốn sách này tập trung vào ví dụ CODE TRỰC TIẾP. Mỗi chương kết thúc với một chương trình hoàn chỉnh, chạy được và sinh viên có thể chạy ứng dụng mà họ vừa tạo ra. Chúng tôi gọi phương pháp vừa giảng dạy vừa viết code này là cách tiếp cận ***CODE TRỰC TIẾP.***
- ***Công nghệ thực tiễn***. Cuốn sách sử dụng những công nghệ mới nhất trong thời điểm hiện tại để phát triển những ứng dụng hữu ích. Ví dụ, chúng tôi sử dụng Unified Modeling Language (UML) để thay thế flowchart - một chuẩn cũ. UML đã trở thành ngôn ngữ mô hình hóa bằng đồ họa được ưa thích để thiết kế ứng dụng hướng đối tượng. Trong cuốn sách *Lập trình trực quan Visual Basic 2008, phiên bản lần ba* chúng tôi sử dụng UML để biểu diễn luồng điều khiển cho một số lệnh điều khiển, để sinh viên có thể thực hành và biết cách đọc các loại biểu đồ được sử dụng trong ngành công nghiệp phần mềm.

- ***Lập trình trực quan và giao diện người dùng đồ họa (GUI- Graphical User Interfaces)***. Từ chương đầu tiên, chúng tôi đã giới thiệu về kỹ thuật lập trình trực quan - được sử dụng để tạo và hiệu chỉnh các chương trình có giao diện đồ họa nhanh chóng và dễ dàng. Các chương đầu cung cấp cho sinh viên nền tảng thiết kế giao diện người dùng đồ họa - khái niệm được sử dụng trong suốt cuốn sách khi chúng tôi dạy về khái niệm lập trình căn bản. Nhiều chương có Mẹo thiết kế giao diện được tóm tắt ở cuối mỗi chương để dễ dàng tham khảo. Phụ lục A tập hợp tất cả các thủ thuật thiết kế giao diện đồ họa.
- ***Windows Forms và Windows Presentation Foundation (WPF)***. Microsoft khuyến cáo rằng lập trình viên nên sử dụng Windows `Forms` thay vì WPF cho ứng dụng doanh nghiệp - là thị trường chính của sinh viên và lập trình viên đang đọc cuốn sách này. Chúng tôi thực hiện hầu hết các giao diện người dùng đồ họa bằng Windows `Forms`.

Những ứng dụng được trình bày trong cuốn sách này

Account Information	Enhanced Dental Payment	Photo Album
Address Book GUI	Factorial	Prime Numbers
Advanced Painter	Fee Calculator	Radio GUI
Alarm	File Scrape	Restaurant Bill
Alarm Clock GUI	Flag Quiz	Road Sign Test
Anagram Game	Food Survey	Quiz Average
Arithmetic Calculator	Form Painter	Salary Survey
Average Three Numbers	Fund Raiser	Sales Commission Calculator
Birthday Saver	Fuzzy Dice Order Form	Sales Report
Bouncing Ball Game	Gas Pump	Savings Calculator
Cafeteria Survey	Grade Calculator	Screen Scraping
Calculator GUI	Guess the Number	Security Panel
Car Payment Calculator	Income Tax Calculator	Shipping Time
Car Reservation	Interest Caculator	Sibling Survey
Cell Phone GUI	Inventory	Simple Calculator
Class Average	Inventory Enhancement	Simple Encryption
Compound Interest	Lottery Picker	Student Grades
Counter	Microwave Oven	Supply Cost Calculator
Craps Game	Microwave Oven GUI	Table of Power
Currency Converter	Miles Per Gallon	Task List
Customer Charge	Office Supplies	Vending Machine GUI
Account Analyzer	Monitor Invoice GUI	Triangle Creator
Dental Payment	Mortgage Calculator	Temperature Converter
Dice Simulator	Multiplication Teacher	Ticket Information
Digit Extraction	Odd Numbers	To-Do List
Discount Calculator	Office Supplies	Typing Tutor
Display Square	Password GUI	Vending Machine GUI
DVD Burner	Pay Raise Calculator	Wage Calculator
Dvorak Keyboard	Pig Latin	Welcome
Encryption	Present Value Calculator	View Name

Hình 1 Các ứng dụng trong Lập trình trực quan Visual Basic 2008.

- ***Font chữ trình bày***. Để bạn đọc tiện theo dõi, chúng tôi quy ước sử dụng font chữ trong sách như sau:

  Font **Time New Roman** đậm cho các thuật ngữ quan trọng

  Font **HP-Helve- Condense** đậm cho tên ứng dụng, các thành phần trên IDE,

  Font `Lucida Sans Typewriter` và font `Consolas` cho các đoạn mã minh họa trong chương.

  Font `Lucida Sans Typewriter` cho các từ chỉ định loại điều khiển (`TextBox`, `ListBox`...).
- ***Lập trình hướng đối tượng***. Lập trình hướng đối tượng là kỹ thuật được sử dụng rộng rãi nhất để phát triển phần mềm lớn, có thể tái sử dụng. Visual Basic 2008 cung cấp rất nhiều tính năng lập trình hướng đối tượng. Cuốn sách hướng dẫn sinh viên cách định nghĩa lớp và sử dụng đối tượng, đặt nền tảng cho những khóa học lập trình cao hơn.
- ***Trình gỡ lỗi của Visual Studio 2008***. Trình gỡ lỗi là công cụ phần mềm hỗ trợ lập trình viên tìm và sửa lỗi lôgic trong mã chương trình. Visual Studio 2008 chứa trình gỡ lỗi mạnh mẽ cho phép lập trình viên phân tích chương trình từng dòng một khi chương trình đang chạy. Trong cuốn sách này, chúng tôi sẽ trình bày chi tiết cách sử dụng những tính năng quan trọng của Visual Studio 2008 Debugger và cung cấp nhiều bài tập gỡ lỗi.

Dành cho giảng viên

Trọng tâm của cuốn sách

Lập trình trực quan Visual Basic 2008, phiên bản lần ba được thiết kế hướng đến những khóa học nhập môn và những khóa học lập trình dành cho học viên chưa có hoặc có ít kinh nghiệm về lập trình. Cuốn sách hướng dẫn các nguyên tắc lập trình và ngôn ngữ Visual Basic 2008 bao gồm kiểu dữ liệu, lệnh điều khiển, lập trình hướng đối tượng, các lớp thư viện của .NET Framework (.NET Framework Class Library), khái niệm GUI, lập trình hướng sự kiện... Sau khi nắm vững nội dung trong sách, sinh viên có thể lập trình Visual Basic 2008 và sử dụng được rất nhiều tính năng quan trọng của nền tảng .NET 3.5.

Cuốn sách này được cập nhật phiên bản Visual Studio mới nhất - Visual Studio 2008, trong đó bao gồm Visual Basic 2008. Chúng tôi đã xây dựng tất cả ứng dụng trong sách sử dụng phiên bản 2008. Tất cả các ứng dụng và solution hoàn toàn đã được kiểm thử và chạy trên nền tảng mới này.

Ghi chú về phần mềm dành cho cuốn sách này

Chúng tôi sử dụng công cụ phát triển Visual Studio 2008, bao gồm các phiên bản miễn phí Visual Basic 2008 Express và Visual Web Developer 2008 Express. Các phiên bản Microsoft Express đều nhẹ, dễ sử dụng và dễ học đối các học viên và những người mới. Các phiên bản Express cung cấp rất nhiều tính năng và có thể được sử dụng để xây dựng các ứng dụng .NET mạnh mẽ. Các phiên bản Express phù hợp cho khóa học mang tính học thuật dành cho những chuyên gia không có điều kiện sử dụng phiên bản đầy đủ của Visual Studio 2008.

Bạn có thể sử dụng các phiên bản Express để biên dịch và chạy tất cả các chương trình ví dụ và làm tất cả các bài tập trong cuốn sách này. Bạn cũng có thể sử dụng sản phẩm Visual Studio phiên bản đầy đủ để xây dựng và chạy ví dụ và bài tập. Tất cả những tính năng được hỗ trợ bởi phiên bản Express đều có trong phiên bản đầy đủ của Visual Studio 2008.

Cuốn sách có DVD đi kèm chứa tất cả các phiên bản Microsoft Visual Studio 2008 Express bao gồm Visual Basic 2008 Express, Visual Web Developer 2008 Express và SQL Server 2005 Express. (phiên bản SQL Server 2008 Express vẫn chưa được ra mắt tại thời điểm cuốn sách này được viết). Bạn có thể tải về các phần mềm trên từ địa chỉ:

`www.microsoft.com/express/`

Sau khi cài đặt phần mềm (sẽ được thảo luận trong mục Chuẩn bị nằm ngay sau mục Lời nói đầu này), bạn cũng nên cài tài liệu trợ giúp và SQL Server 2005 Express. Microsoft cung cấp diễn đàn dành riêng để hỗ trợ sử dụng các phiên bản Express tại địa chỉ:

```
forums.microsoft.com/msdn/ShowForum.
aspx?siteid=1&ForumID=24
```

Sau khi phiên bản SQL Server 2008 Express được ra mắt, chúng tôi sẽ bổ sung các thông tin về việc sử dụng phần mềm này cho cuốn sách tại địa chỉ `www.deitel.com/books/SimplyVB2008`.

Windows 7 và Windows XP

Độc giả của cuốn sách này có thể sử dụng Windows 7 hoặc Windows XP. Chúng tôi sử dụng Windows 7 trong quá trình xây dựng cuốn sách này, tuy nhiên các bước trong Windows XP cũng tương tự. Trong trường hợp khác nhau, chúng tôi sẽ chỉ ra các bước cụ thể dành riêng cho Windows XP. Trong cuốn sách này, chúng tôi sử dụng font chữ Windows 7 Segoe UI trên giao diện người dùng đồ họa. Trong trường hợp có những vấn đề phát sinh khi sử dụng Windows XP sau khi cuốn sách này được xuất bản, chúng tôi sẽ cung cấp những hướng dẫn cần thiết tại địa chỉ `www.deitel.com/books/SimplyVB2008`. Nếu gặp bất cứ vấn đề gì, hãy gửi email cho chúng tôi đến địa chỉ `deitel@deitel.com` và chúng tôi sẽ hồi âm tức thì cho bạn.

Ghi chú về thuật ngữ được sử dụng trong sách

Ở Chương 13, chúng tôi sử dụng thuật ngữ thủ tục `Sub` và thủ tục `Function` (đôi khi còn được gọi là hàm) thay vì thuật ngữ phương thức. Chúng tôi sử dụng thuật ngữ như vậy là vì hai lý do. Trước hết, các từ khóa `Sub` và `Function` được sử dụng trong định nghĩa thủ tục và phương thức vì vậy cách gọi này lôgíc đối với học viên. Thứ hai, các chuyên gia Visual Basic đã và đang sử dụng thuật ngữ này từ nhiều năm và sẽ tiếp tục dùng. Chúng tôi cũng sử dụng thuật ngữ "hàm" ở một số chỗ trong cuốn sách này để nhắc đến các thủ tục `Function` Visual Basic 6 hiện vẫn tồn tại trong Visual Basic 2008 (chẳng hạn `Val` và `Pmt`). Trong khi giới thiệu các khái niệm lập trình hướng đối tượng ở chương 19, chúng tôi sẽ trình bày sự khác nhau giữa các thủ tục và phương thức và chỉ ra rằng thực ra các thủ tục được định nghĩa trong cuốn sách này về bản chất đều là các phương thức.

Mục tiêu

Mỗi chương bắt đầu bởi phần mục tiêu để sinh viên biết những điều sẽ được học trong chương đó. Sau khi đã đọc xong chương này, sinh viên có cơ hội nhìn lại xem mình đã học được những gì, có đáp ứng được mục tiêu hay chưa.

Nội dung chính

Phần nội dung chính của chương giúp sinh viên tiếp cận nội dung sách theo kiểu từ trên xuống. Cùng với phần mục tiêu, phần nội dung chính giúp học viên biết trước được chủ đề của chương ngay từ đầu và tự chọn cho mình cách học phù hợp.

Các ứng dụng mẫu (cùng với kết quả đầu ra)

Chúng tôi giới thiệu những tính năng Visual Basic 2008 dưới dạng những chương trình hoàn chỉnh. Chúng tôi gọi cách tiếp cận này là CODE TRỰC TIẾP. Tất cả các ví dụ trong sách có thể được tải về từ địa chỉ:

```
www.deitel.com/books/SimplyVB2008
```

Minh họa/ hình ảnh/ bảng "ACE"

Sách bao gồm một lượng lớn các biểu đồ, đường kẻ và hình ảnh kết quả của ứng dụng. Các lệnh điều khiển đều được vẽ chi tiết dưới dạng các biểu đồ hoạt động UML. (*Ghi chú*: Chúng tôi không hướng dẫn về biểu đồ UML như một công cụ phát triển ứng dụng mà chỉ sử dụng các biểu đồ UML nhằm giải thích các thao tác

của các lệnh điều khiển trong Visual Basic 2008). Hầu hết các chương đều được bổ sung bảng "ACE" liệt kê các hành động động, điều khiển và các sự kiện quan trọng để có thể xây dựng được ứng dụng trong chương.

Thủ thuật lập trình

Hàng trăm các thủ thuật lập trình giúp sinh viên tập trung vào các khía cạnh quan trọng trong lập trình ứng dụng. Các thủ thuật và kinh nghiệm hay nhất được các tác giả tổng hợp lại sau hơn 70 năm lập trình và giảng dạy.

Thói quen lập trình tốt

Mục *Thói quen lập trình tốt* tập trung vào những kỹ thuật giúp tạo ra chương trình sáng sủa, dễ hiểu và dễ bảo trì hơn.

Lỗi lập trình thường gặp

Sinh viên thường mắc một số lỗi nào đó, cho nên chỉ ra *Lỗi lập trình thường gặp* sẽ giảm thiểu khả năng mắc phải những lỗi này.

Mẹo tránh lỗi

Những mẹo này chứa những gợi ý về việc tìm và loại bỏ các lỗi trong chương trình. Trong đó, có nhiều mẹo mô tả những tính năng của Visual Basic 2008 tránh gây ra lỗi trong chương trình ngay từ đầu.

Mẹo thiết kế phần mềm

Mẹo thiết kế phần mềm tập trung vào các vấn đề về kiến trúc và thiết kế ảnh hưởng đến việc xây dựng hệ thống phần mềm.

Mẹo thiết kế giao diện

Mẹo thiết kế giao diện tập trung vào những kỹ thuật giúp sinh viên tạo giao diện hấp dẫn, trực quan và thân thiện với người dùng. Phụ lục A sẽ tập hợp các mẹo thiết kế giao diện này.

Tổng hợp kỹ năng

Mỗi chương đều chứa bản tóm tắt dưới dạng danh sách tổng hợp các khái niệm lập trình được giới thiệu trong chương. Mục này nhấn mạnh những thao tác quan trọng cần thực hiện để xây dựng ứng dụng trong mỗi chương.

Thuật ngữ

Phần thuật ngữ là danh sách những thuật ngữ quan trọng được định nghĩa trong chương. Những thuật ngữ và định nghĩa này cũng có mặt trong bảng thuật ngữ của sách để sinh viên có thể nhanh chóng tra cứu thuật ngữ và ý nghĩa của chúng.

Câu hỏi tự ôn tập và đáp án

Các câu hỏi trắc nghiệm tự ôn tập và đáp án được thêm vào sau hầu hết các mục của mỗi chương giúp sinh viên củng cố lại kiến thức và chuẩn bị kiến thức trước khi làm bài tập. Sinh viên nên được khuyến khích thực hiện tất cả các bài tự ôn tập và kiểm tra đáp án.

Bài tập (Đáp án có trong tài nguyên hướng dẫn dành cho giảng viên)

Mỗi chương kết thúc bằng phần bài tập. Mỗi phần bài tập thường bao gồm mười câu hỏi trắc nghiệm, một bài tập dạng "Đoạn mã này làm gì?" và một bài tập dạng "Đoạn mã này có gì sai?", ba bài tập lập trình và một bài tập nâng cao. (*Chú ý*: Trong các bài tập dạng "Đoạn mã này làm gì?" và "Đoạn mã này có gì sai?" chúng tôi sẽ chỉ đưa một phần của đoạn mã vào).

Các câu hỏi trắc nghiệm liên quan đến thuật ngữ quan trọng và khái niệm, cách viết các lệnh Visual Basic 2008, viết một phần nhỏ của ứng dụng Visual Basic 2008 và viết phương thức, lớp, ứng dụng Visual Basic hoàn chỉnh. Tất cả các bài tập lập trình sử dụng phương pháp hướng dẫn theo từng bước để gợi ý cách thức giải quyết vấn đề. Chỉ có giáo viên được sự đồng ý của đại diện Prentice Hall mới có quyền truy cập đến đáp án của các bài tập.

[*CHÚ Ý:* Phần tài nguyên này chỉ dành cho giảng viên đã được cấp quyền, không dành cho sinh viên.]

Hướng dẫn thiết kế giao diện

Thiết kế giao diện người dùng đồ họa hợp lý và nhất quán rất quan trọng trong lập trình trực quan. Trong mỗi chương, chúng tôi sẽ tóm tắt những hướng dẫn thiết kế giao diện đã giới thiệu trong chương đó. Phụ lục A trình bày danh sách tổng hợp các hướng dẫn thiết kế giao diện để tiện cho việc tham khảo.

Tổng hợp điều khiển, sự kiện, thuộc tính & phương thức

Mỗi chương bao gồm một bản tóm tắt các điều khiển, sự kiện, thuộc tính và phương thức được nhắc tới trong chương. Bản tóm tắt bao gồm hình ảnh của mỗi điều khiển hiển thị ở trên Toolbox và hình ảnh hiển thị thực tế trên giao diện khi chạy ứng dụng và danh sách các thuộc tính, sự kiện và phương thức của điều khiển đó.

Microsoft Developer Network Academic Alliance (MSDNAA) và Microsoft DreamSpark

Microsoft Developer Network Academic Alliance (MSDNAA) - Phần mềm Microsoft miễn phí dành cho mục đích nghiên cứu và giáo dục

MSDNAA cung cấp phần mềm miễn phí cho mục đích nghiên cứu và giáo dục. Phần mềm cho giáo viên có thể được truy cập trực tiếp tại `www.microsoft.com/faculty`. Phần mềm cho các bộ phận khác xin truy cập địa chỉ `www.msdnaa.com`.

Microsoft DreamSpark - Công cụ thiết kế và phát triển chuyên nghiệp dành cho sinh viên

Microsoft cung cấp miễn phí rất nhiều công cụ phát triển cho sinh viên thông qua chương trình DreamSpark (`downloads.channel8.msdn.com/`). Tại thời điểm thực hiện cuốn sách này, website DreamSpark tuyên bố rằng các sinh viên ở mười một quốc gia (Mỹ, Vương quốc Anh, Canada, Trung Quốc, Đức, Pháp, Phần Lan, Tây Ban Nha, Thụy Điển, Thụy Sĩ và Bỉ) có thể nhận được phần mềm này sau khi đã được xác nhận là sinh viên.

Tài nguyên dành cho giáo viên của cuốn sách Lập trình trực quan Visual Basic 2008, phiên bản lần ba

Lập trình trực quan Visual Basic 2008, phiên bản lần ba có một lượng lớn các tài nguyên dành cho giáo viên. *Trung tâm tài nguyên dành cho giáo viên* của Prentice Hall gồm tài liệu *đáp án dành cho giáo viên* chứa đáp án của các bài tập cuối mỗi chương, một File đáp án của câu hỏi trắc nghiệm và các slide PowerPoint chứa mã và hình ảnh trong sách cùng nội dung chính trong sách được tóm tắt ngắn gọn. Các giáo viên có thể chỉnh sửa lại slide. Nếu bạn chưa đăng ký trở thành thành viên với tư cách giáo viên, hãy liên hệ với đại diện Prentice Hall của bạn.

[*CHÚ Ý:* Phần tài nguyên này chỉ dành cho giảng viên đã được cấp quyền, không dành cho sinh viên.]

Email tin tức miễn phí
DEITEL BUZZ ONLINE

Hàng tuần, *DEITEL BUZZ ONLINE* gửi thông tin về các trung tâm tài nguyên mới nhất bao gồm những bình luận về xu hướng công nghệ và sự phát triển, liên kết đến các bài viết miễn phí và các tài nguyên từ những cuốn sách đã và sắp xuất bản, kế hoạch ra mắt sản phẩm, đính chính, bài tập, kinh nghiệm cá nhân, thông tin về các khóa học dành cho doanh nghiệp... Đây cũng là một cách tốt để theo dõi bài viết về vấn đề liên quan đến cuốn sách *Lập trình trực quan Visual Basic 2008, phiên bản lần ba*. Để đăng ký nhận thư tin tức, bạn hãy truy cập vào địa chỉ

`www.deitel.com/newsletter/subscribe.html`

Các trung tâm tài nguyên trực tuyến Deitel

Website của chúng tôi `www.deitel.com` cung cấp hơn một trăm trung tâm tài nguyên liên quan đến rất nhiều các lĩnh vực bao gồm ngôn ngữ lập trình, phần mềm, Web 2.0, kinh doanh trên Internet và các dự án mã nguồn mở - xem danh sách đầy đủ các trung tâm tài nguyên trong trang đầu của cuốn sách. Các trung tâm tài nguyên được xây dựng từ những nghiên cứu mà chúng tôi đã thực hiện nhằm hỗ trợ cho những cuốn sách của chúng tôi. Chúng tôi đã tìm được rất nhiều các tài nguyên trực tuyến vô cùng hữu ích bao gồm các hướng dẫn, tài liệu, phần mềm để download, bài viết, blog, podcast, video, đoạn mã mẫu, sách, sách điện tử... - hầu hết chúng đều miễn phí. Hàng tuần, chúng tôi sẽ thông báo các tài nguyên trực tuyến mới nhất trong email tin tức *DEITEL BUZZ ONLINE* (`www.deitel.com/newsletter/subscribe.html`). Bạn có thể sẽ quan tâm đến các trung tâm tài nguyên sau đây khi nghiên cứu *Lập trình trực quan Visual Basic 2008, phiên bản lần ba:*

- LINQ
- Microsoft Popfly
- .NET 3.5
- Visual Basic 2008
- Visual Studio Team System
- Windows Communication Foundation
- Windows Presentation Foundation
- Windows Workflow Foundation
- Windows 7

Lời cảm ơn

Chúng tôi muốn gửi lời cảm ơn chân thành đến những người đã cống hiến cho cuốn sách này mà tên của họ không có mặt trên bìa của cuốn sách. Sự đóng góp, sự hợp tác, tình bạn và sự cảm thông của họ đóng vai trò vô cùng quan trọng trong việc xuất bản cuốn sách. Rất nhiều cá nhân ở Deitel & Associates đã dành hàng giờ làm việc cho dự án này - chúng tôi đặc biệt cảm ơn Abbey Deitel và Barbara Deitel.

Thật may mắn cho chúng tôi khi được làm việc với những chuyên gia về xuất bản đầy tài năng ở Prentice Hall. Chúng tôi biết ơn những đóng góp đặc biệt của Marcia Horton, Trưởng ban biên tập của Phòng kỹ nghệ và khoa học máy tính. Carole Snyder và Dolores Mars đã xây dựng một đội ngũ duyệt sách và quản lý quy trình duyệt sách. Francesco Santalucia (một nghệ sĩ tự do) và Kristine Carney của Prentice Hall đã thiết kế bìa sách - chúng tôi đã cung cấp ý tưởng và họ đã hiện thực hóa những ý tưởng ấy. Scott Disanno, Robert Engelhardt và Marta Samsel đã quản lý việc sản xuất sách. Quản lý marketing của chúng tôi - Chris Kelly - và sếp của ông - Margaret Waples - đã xây dựng chương trình marketing cho cuốn sách thông qua các kênh học thuật.

Những thành viên tham gia duyệt sách Lập trình trực quan Visual Basic 2008, phiên bản lần ba

Chúng tôi rất biết ơn công lao của đội ngũ duyệt sách. Tuân thủ tiến độ thời gian chặt chẽ, họ đã xem xét kỹ lưỡng toàn bộ nội dung và các chương trình trong sách, cung cấp vô số những gợi ý để cải thiện sự chính xác và hoàn thiện trong trình bày.

Kiểm duyệt viên của Microsoft: Adrian "Spotty" Bowles (Microsoft), Marcelo Guerra Hahn (Microsoft), Huanhui Hu (Microsoft), Timothy Ng (Microsoft), Akira Onishi (Microsoft), April Reagan (Microsoft), Steve Stein (Microsoft) và Scott Wisniewski (Microsoft). ***Kiểm duyệt viên từ các tổ chức về học thuật***: Douglas B. Bock (ĐH Nam Illnois Edwardsville), Edward Hunter (ĐH Chapman), Christopher J. Olson (ĐH Bang Dakota) và Josh Pauli (ĐH Bang Dakota). ***Kiểm duyệt viên từ các doanh nghiệp trong ngành CNTT***: Jeff Certain (Colorado CustomWare), Matthew Kleinwaks (Abby Rating Systems; Microsoft Visual Basic MVP), Éric Moreau (Moer; Microsoft Visual Basic MVP), José Antonio González Seco (Tòa Quốc hội Andalusia), Rod Stephens (Chủ tịch công ty tư vấn máy tính Rocky Mountain) và Chris Williams (Magenic; Microsoft Visual Basic MVP).

Visual Basic 2008 là một ngôn ngữ lập trình giúp bạn viết ra các chương trình một cách nhanh chóng và hiệu quả. Visual Basic 2008 rất thành công trong việc phát triển các hệ thống thông tin kinh doanh và các hệ thống thông tin tối quan trọng cho các tổ chức. Chúng tôi chân thành biết ơn những góp ý, phê bình, đính chính và đề nghị cải tiến của các bạn trong quá trình đọc cuốn sách này. Mọi ý kiến xin gửi về:

`deitel@deitel.com`

Chúng tôi sẽ hồi âm ngay tức thì và sẽ đăng những đính chính, lời giải thích trên website:

`www.deitel.com/books/SimplyVB2008/`

Chúng tôi hy vọng bạn sẽ dành tình cảm cho cuốn sách này nhiều như chúng tôi đã dành cho việc viết ra nó!

Paul J. Deitel

Tiến sĩ Harvey M. Deitel

Greg J. Ayer

Giới thiệu về tác giả

Paul J. Deitel - Giám đốc điều hành và Giám đốc kỹ thuật của Deitel & Associates tốt nghiệp ngành công nghệ công tin tại trường quản lý MIT Sloan. Ông có chứng chỉ Java Certified Programmer và Java Certified Developer và được Sun Microsystems công nhận là một Java Champion. Thông qua Deitel & Associates, ông đã cung cấp những khóa học Visual Basic, C#, C++, C và Java cho cộng đồng công nghệ thông tin bao gồm Cisco, IBM, Sun Microsystems, Dell, Lucent Technologies, Fidelity, NASA tại Trung tâm không gian Kennedy, White Sands Missile Range, Phòng nghiên cứu khí hậu khắc nghiệt quốc gia, Rogue Wave Software, Boeing, Stratus, Hyperion Software, Adra Systems, Entergy, CableData Systems, Nortel Networks, Puma, iRobot, Invensys... Ông đã có rất nhiều bài giảng/thuyết trình về Java và C++ tại Hiệp hội máy tính Boston. Ông cùng với cha ông - Tiến sĩ Harvey M. Deitel là những tác giả của những cuốn sách giáo khoa về ngôn ngữ lập trình bán chạy nhất thế giới.

Tiến sĩ Harvey M. Deitel - Chủ tịch và Giám đốc kỹ thuật của Deitel & Associates đã có 47 năm kinh nghiệm về lĩnh vực máy tính. Tiến sĩ Deitel đã nhận được bằng cử nhân khoa học và thạc sĩ khoa học của MIT và bằng tiến sĩ tại Đại học Boston. Ông có những kinh nghiệm uyên thâm về giảng dạy đại học và được công nhận là Chủ tịch hội đồng khoa học máy tính của trường Đại học Boston trước khi thành lập công ty Deitel & Associates cùng con trai, Paul J. Deitel. Ông và Paul là đồng tác giả của rất nhiều cuốn sách và tài liệu đa phương tiện. Hiện tại, họ vẫn đang tiếp tục xây dựng nhiều cuốn sách nữa. Những cuốn sách của Deitel

được toàn thế giới biết đến với các bản dịch được xuất bản bằng tiếng Nhật, Đức, Nga, Tây Ban Nha, Trung Quốc, Hàn Quốc, Pháp, Ba Lan, Ý, Bồ Đào Nha, Hy Lạp, Urdu và Thổ Nhĩ Kỳ. Tiến sĩ Deitel có hàng trăm bài phát biểu tại các hội thảo của giới chuyên môn tại các doanh nghiệp lớn, các viện nghiên cứu, các cơ quan chính phủ, các tổ chức và quân đội.

Greg Ayer là thạc sĩ khoa học máy tính của trường Đại học Đông Nam. Kinh nghiệm làm việc của ông bao gồm việc phát triển web, lập trình game, tư vấn và biên tập viên công nghệ. Thông qua các chương trình hợp tác của Đại học Đông Nam, ông đã hợp tác với Deitel & Associates 6 tháng. Ông tiếp tục làm cố vấn cho Deitel & Associate. Các mảng công việc của ông bao gồm phát triển cơ sở dữ liệu, lý thuyết tính toán, hệ điều hành, mạng và các ngôn ngữ lập trình.

Giới thiệu về Deitel & Associates

Deitel & Associate là một công ty quốc tế chuyên sản xuất các nội dung và giảng cho các doanh nghiệp đặc biệt là các ngôn ngữ lập trình máy tính, Internet và công nghệ phần mềm trên web, đào tạo về lập trình hướng đối tượng và phát triển kinh doanh trên Internet thông qua chiến lược kinh doanh Internet trên nền Web 2.0. Công ty cung cấp những khóa học có người hướng dẫn về lĩnh vực ngôn ngữ lập trình và nền tảng phổ biến như Visual Basic, C#, Visual C++, C++, Java, C, XML, Perl, lập trình hướng đối tượng, Internet và lập trình web và danh sách ngày càng tăng các khóa học về lập trình và phát trển phần mềm. Sáng lập viên của Deitel & Associate là Paul J. Deitel và Tiến sĩ Harvey M. Deitel. Khách hàng của công ty bao gồm các công ty lớn nhất thế giới, các cơ quan chính phủ, các tổ chức quân sự, các viện nghiên cứu. Trong vòng 32 năm hợp tác xuất bản với Prentice Hall, Deitel & Associates đã xuất bản các sách giáo khoa lập trình, sách chuyên môn và các khóa học về đa phương tiện qua Video trên web hoặc DVD như *Cyber Classrooms* hay *Live Lessons*, các nội dung trực tuyến cho các hệ thống quản lý khóa học phổ biến. Mọi liên hệ đến Deitel & Associate và các tác giả của cuốn sách xin gửi email tới địa chỉ:

```
deitel@deitel.com
```

Để tìm hiểu thêm về Deitel & Associates, các ấn bản và chương trình đào tạo cho doanh nghiệp *Dive Into* trên toàn thế giới, hãy ghé thăm:

```
www.deitel.com
www.deitel.com/books/
www.deitel.com/training/
```

Để đăng ký nhận email tin tức *DEITEL BUZZ ONLINE* miễn phí hãy truy cập:

```
www.deitel.com/newsletter/subscribe.html
```

Tham khảo danh sách các trung tâm tài nguyên Deitel tại địa chỉ:

```
www.deitel.com/resourcecenters.html
```

Các cá nhân muốn mua các ấn bản của Deitel thông qua `Amazon.com` hay `Informit.com` có thể ghé thăm website của chúng tôi:

```
www.deitel.com
```

Các công ty, cơ quan chính phủ, quân đội và các viện khoa học muốn mua sách số lượng lớn có thể đặt hàng trực tiếp tại Prentice Hall. Thông tin chi tiết xin truy cập:

```
www.prenhall.com/mischtm/support.html#order
```

CHUẨN BỊ

Mục này chứa các thông tin bạn nên xem qua trước khi sử dụng cuốn sách và các hướng dẫn để đảm bảo máy tính của bạn được cài đặt chính xác. Chúng tôi để phiên bản mới nhất của phần Chuẩn bị này trên website của cuốn sách `www.deitel.com/books/SimplyVB2008/`.

Quy ước về Font chữ

Chúng tôi sử dụng font khác nhau để phân biệt tên menu, mục nằm trong menu và các thành phần hiển thị trên IDE. Quy ước của chúng tôi là sử dụng font **HP-Helve-Condense** đậm cho các thành phần trên IDE (chẳng hạn cửa sổ **Properties**) và font `Lucida Sans Typewriter` cho các đoạn mã (chẳng hạn `Private x As Boolean = True`).

Phần mềm cho cuốn sách

Cuốn sách này bao gồm DVD chứa phần mềm Microsoft Visual Studio 2008 Express Edition - môi trường phát triển tích hợp của Visual Basic 2008, Visual C# 2008, Visual C++ 2008, Visual Web Developer 2008 và SQL Server 2005. Những công cụ này có thể được tải về từ địa chỉ `www.microsoft.com/express`. Phiên bản Express bao gồm đầy đủ chức năng và không bị hạn chế về thời gian sử dụng phần mềm. Chúng tôi sẽ giới thiệu cách cài đặt phần mềm này ngay sau đây. Bạn không cần đến Visual C# hoặc Visual ++ trong phạm vi cuốn sách này.

Yêu cầu về phần cứng và phần mềm cho Visual Studio 2008 Express Edition

Để cài đặt và chạy các phiên bản Visual Studio 2008 Express Edition, Microsoft khuyến cáo cấu hình tối thiểu như sau:

- **Hệ điều hành**: Windows XP Service Pack 2 (hoặc cao hơn), Windows Server 2003 Service Pack 1 (hoặc cao hơn), Windows Server 2003 R2 (hoặc cao hơn), Windows 7 hoặc Windows Server 2008.
- **Bộ vi xử lý**: Máy tính có bộ vi xử lý 1,6 GHz hoặc nhanh hơn (tốt nhất là 2,2 GHz hoặc cao hơn – 2,4 GHz đối với Windows 7).
- **RAM tối thiểu**: 192 MB, nhưng Microsoft khuyến cáo nên là 384 MB (768 MB đối với Windows 7).
- **Ổ cứng**: 1,3 GB đối với cài đặt đầy đủ.
- **Màn hình**: 1024 x 768 (khuyến cáo là 1280 x 1024).

Thiết lập hiển thị

Lập trình trực quan Visual Basic 2008, phiên bản lần 3 bao gồm hàng trăm ảnh chụp màn hình của ứng dụng. Bạn có thể sẽ phải điều chỉnh thiết lập màn hình trên máy tính để màn hình hiển thị giống như những ảnh chụp màn hình trong sách khi bạn phát triển ứng dụng. Thực hiện theo các bước sau để cấu hình chính xác cho màn hình:

- **Windows 7**
  1. Nhấn chuột phải vào màn hình, chọn **Personalize**.
  2. Nhấn chuột vào mục **Display** ở bên trái hộp thoại.

3. Chọn mục **Smaller - 100% (default)**. [*Lưu ý*: Nếu mục này đã được chọn thì bạn không cần phải làm gì cả].
4. Nhấn vào **Apply**.

- **Windows XP**
  1. Nhấn chuột phải màn hình, chọn **Properties**.
  2. Nhấn chuột vào tab **Settings**.
  3. Nhấn chuột vào nút **Advanced**.
  4. Ở tab **General**, chọn mục **Normal size (96 DPI)**. (*Chú ý*: Nếu mục này đã được chọn thì bạn không cần phải làm gì cả).
  5. Nhấn vào **OK**, sau đó nhấn tiếp vào **OK** để hoàn tất việc thay đổi.

Nếu bạn chọn các thiết lập khác, kích thước và vị trí của mỗi điều khiển GUI (chẳng hạn `Button` hay `Label`) theo thông số `Size` và `Location` chúng tôi chỉ định cho việc hiển thị trên màn hình sẽ có thể không giống như trong sách.

Thiết lập Theme cho desktop đối với Windows 7

Nếu bạn sử dụng Windows 7, hãy làm theo các bước sau để thiết lập theme máy bạn thành kiểu theme Windows 7.

1. Nhấn chuột phải vào desktop, sau đó nhấn chuột vào mục **Personalize**.
2. Trong mục **Aero Themes**, chọn **Windows 7** từ danh sách các kiểu Theme.

Thiết lập theme cho desktop đối với Windows XP

Nếu bạn đang sử dụng Windows XP, cửa sổ hiển thị trên màn hình trông hơi khác một chút so với các ảnh chụp màn hình trong sách. Hãy thực hiện theo hướng dẫn sau để thiết lập theme desktop của bạn thành theme Windows XP:

1. Nhấn chuột phải vào desktop, chọn **Properties**.
2. Nhấn chuột vào tab **Themes**. Chọn **Windows XP** trong danh sách thả xuống **Theme:**.
3. Nhấn chuột vào **OK** để lưu lại các thiết lập.

Hiển thị phần mở rộng của file

Một vài ảnh chụp màn hình trong *Lập trình trực quan Visual Basic 2008, phiên bản lần ba* hiển thị tên file có phần mở (ví dụ `.txt`, `.vb`, `.png`). Các thiết lập của bạn có thể sẽ cần phải điều chỉnh để có thể hiển thị được phần mở rộng trong tên file. Hãy thực hiện theo các bước sau để cấu hình cho máy tính:

1. Trong menu **Start**, chọn **All Programs**, sau đó chọn **Accessories**, chọn tiếp **Windows Explorer**.
2. Đối với Windows 7, chọn **Organize**, sau đó chọn **Folder and search options** từ menu đổ xuống. Đối với Windows XP, chỉ cần nhấn vào **Folder Options...** từ menu **Tools** của **Windows Explorer**.
3. Trong hộp thoại hiện ra, chọn tab **View**.
4. Trong khung **Advanced settings:**, bỏ chọn mục **Hide extensions for known file types**. (*Chú ý*: Nếu mục này hiện không được chọn thì không phải làm gì cả).

Chú ý về font Segoe UI đối với Windows XP

Ở Windows 7, Microsoft đã ra mắt font mới có tên Segoe UI. Toàn bộ ứng dụng từ Chương 1 đến 22 đều sử dụng font chữ này. Mặc định Windows XP không có font này, tuy nhiên bạn có thể cài đặt phần mềm Windows Live Mail để có font chữ này. Để tải về Windows Live Mail, bạn hãy truy cập vào địa chỉ `get.live.com/wlmail/overview`.

Bạn cũng cần phải kích hoạt chế độ ClearType trên máy tính, nếu không thì font chữ sẽ hiển thị không chính xác. ClearType là một công nghệ giúp hiển thị các font chữ trên màn hình sắc nét hơn. Để kích hoạt ClearType, thực hiện các bước sau:

1. Nhấn chuột phải lên màn hình desktop, chọn **Properties...** từ menu ngữ cảnh để hiển thị hộp thoại **Display Properties**.
2. Trong hộp thoại, nhấn chuột vào tab **Appearance**, sau đó nhấn chuột vào nút **Effects...** để hiển thị hộp thoại **Effects**.
3. Trong hộp thoại **Effects**, đảm bảo rằng mục **Use the following method to smooth edges of the screen fonts** đã được chọn, sau đó chọn **ClearType** từ combobox phía dưới mục này.
4. Nhấn chuột vào **OK** để đóng hộp thoại **Effects**, sau đó nhấn chuột vào **OK** để đóng hộp thoại **Display Properties**.

Tải về các đoạn code ví dụ

Các ví dụ của cuốn sách *Lập trình trực quan·Visual Basic 2008, phiên bản lần ba* có thể tải về từ địa chỉ:

```
www.deitel.com/books/SimplyVB2008
```

Thực hiện theo các bước sau ở phần sau để tải ví dụ về và để tạo thư mục `Examples` trên ổ cứng. Ảnh chụp màn hình trong phần này có thể hơi khác một chút so với những gì hiển thị trên máy tính của bạn do sử dụng các phiên bản Windows khác nhau. Chúng tôi đã sử dụng Windows 7 để chụp các hình ảnh minh họa trong sách.

Tải về các ví dụ trong sách từ website của Deitel

1. ***Đăng ký tại `www.deitel.com`.*** Nếu bạn chưa đăng ký, hãy truy cập vào địa chỉ `www.deitel.com` và nhấn chuột vào liên kết **Register** nằm bên dưới logo deitel góc trên bên trái của trang. Nếu bạn đã đăng ký thì hãy chuyển sang *Bước 2*. Điền thông tin của bạn vào các mục yêu cầu. Việc đăng ký là hoàn toàn miễn phí và chúng tôi cam đoan sẽ không chia sẻ thông tin của bạn với bất kỳ ai khác. Chúng tôi sẽ chỉ gửi mail thông tin về tài khoản cho bạn trừ phi bạn đăng ký thêm dịch vụ thư điện tử tin tức tại `www.deitel.com/newsletter/subscribe.html`. Sau khi đăng ký xong, bạn sẽ nhận được email xác nhận với mã xác nhận của bạn. Bạn cần mã xác nhận này đã đăng nhập vào `www.deitel.com` lần đầu tiên.
2. ***Tải ví dụ.*** Mở trình duyệt, truy cập vào địa `www.deitel.com` và đăng nhập bằng cách nhấn vào liên kết **Login** nằm bên dưới logo deitel góc trên bên trái của trang. Tiếp theo, truy cập đến địa chỉ `www.deitel.com/books/SimplyVB2008`. Nhấn chuột vào liên kết **Examples** để tải file `Examples.zip` về máy tính của bạn. Bạn hãy lưu file ở đâu đó trên máy tính. Tiếp theo, nếu là Windows 7, bạn chuyển sang *Bước 3*, nếu là Windows XP thì chuyển sang *Bước 4*.
3. ***Giải nén `Examples.zip` đối với Windows 7.*** Chúng tôi giả sử rằng các ví dụ nằm trong thư mục `C:\Examples` trên máy tính của bạn. Để đưa các ví dụ vào vị trí này, bạn hãy thực hiện theo các bước sau (các bước có thể hơi khác trong trường hợp bạn đã cài đặt công cụ giải nén file ZIP):
   - Tìm thư mục `Examples.zip` bằng Windows Explorer.
   - Nhấn chuột phải vào `Examples.zip` và chọn **Extract All...** để hiển thị hộp thoại **Extract Compressed (Zipped) Folders**.
   - Nhập `C:\` vào ô **Files will be extracted to this folder**.
   - Nhấn chuột vào nút **Extract**. Thư mục `C:\Examples` đã được tự động tạo cho bạn.

   [*Chú ý*: Trong một số trường hợp bạn không được phép lưu các file trực tiếp lên ổ `C:\` hoặc muốn lưu file file ở một nơi khác. Nếu chọn vị trí khác để giải nén, bạn cần phải thay thế vị trí này cho `C:\Examples` khi cuốn sách này nhắc đến `C:\Examples`].

4. ***Giải nén Examples.zip trên Windows XP.*** Chúng tôi giả sử rằng các ví dụ nằm trong thư mục `C:\Examples` trên máy tính. Để lưu các ví dụ vào vị trí này, thực hiện theo các bước sau đây:
   - Tìm thư mục `Examples.zip` bằng Windows Explorer.
   - Nhấn chuột phải vào file `Examples.zip` và chọn **Extract All...** để hiển thị hộp thoại **Extraction Wizard**.
   - Nhấn chuột vào nút **Next >**.
   - Gõ `C:\` vào ô **Files will be extracted to this directory**. Công cụ giải nén sẽ tự động tạo thư mục có tên `Examples` trên ổ `C:`.
   - Nhấn chuột vào nút **Next >**.
   - Nhấn chuột vào nút **Finish**. Thư mục `C:\Examples` sẽ được tự động tạo cho bạn.

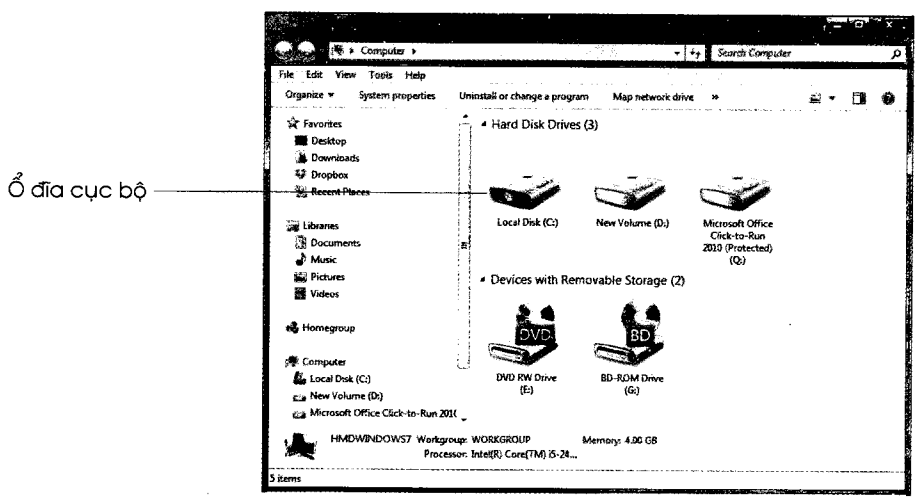
   [*Chú ý*: Trong một số trường hợp bạn không được phép lưu các file trực tiếp lên ổ `C:\` hoặc muốn lưu file ở một nơi khác. Nếu chọn vị trí khác để giải nén, bạn cần phải thay thế vị trí này cho `C:\Examples` khi cuốn sách này nhắc đến `C:\Examples`].

Tạo thư mục làm việc của bạn

Trong mục sau đây, bạn tạo thư mục trong ổ `C:` để lưu những ứng dụng do bạn tạo ra. Trong cuốn sách này, chúng tôi giả sử rằng thư mục bạn sử dụng là `C:\SimplyVB2008`. Nếu bạn chọn một vị trí khác làm thư mục làm việc của bạn, bạn cần phải thay thế vị trí này cho `C:\SimplyVB2008` khi cuốn sách này nhắc đến `C:\SimplyVB2008`.

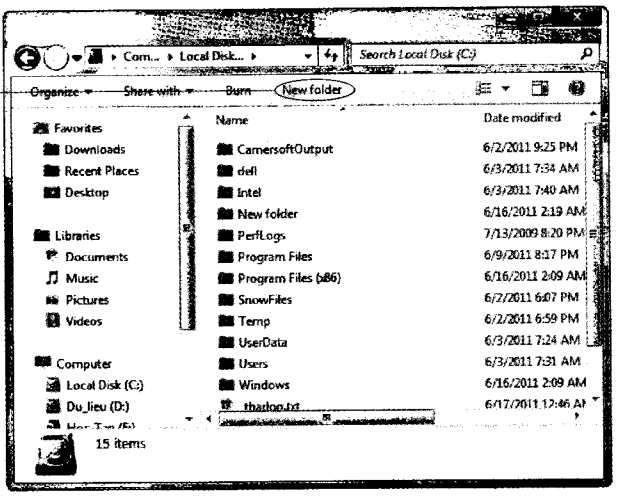
Tạo thư mục làm việc trên Windows 7

1. ***Chọn ổ đĩa.*** Mở menu **Start** và chọn **Computer** để truy cập vào danh sách ổ đĩa trên máy tính (Hình 1). Nhấn đúp chuột vào **Local Disk (C:)**. Nội dung của ổ `C:` được hiển thị như sau trên cửa sổ.

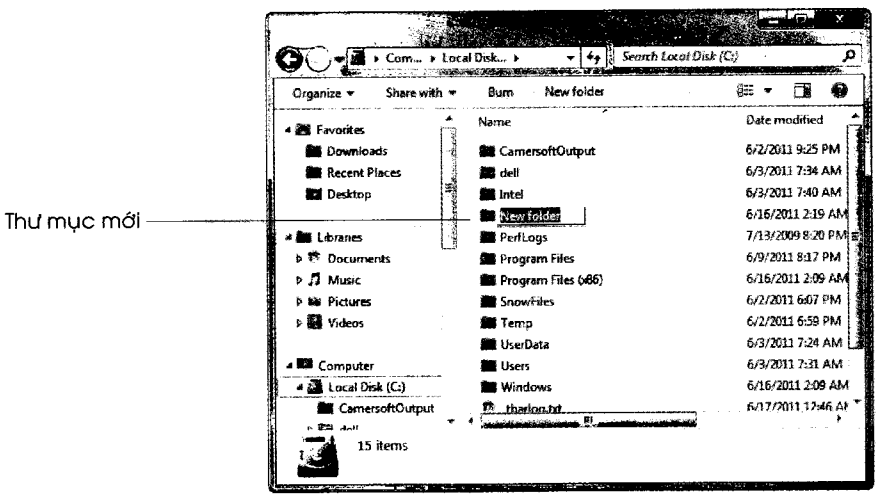


Hình 1 Các ổ đĩa máy tính được liệt kê trong **Computer**.

2. ***Tạo thư mục mới***. Nhấn chuột vào nút **New Folder** (Hình 2). Một thư mục rỗng mới hiện ra trong ổ C: (Hình 3).

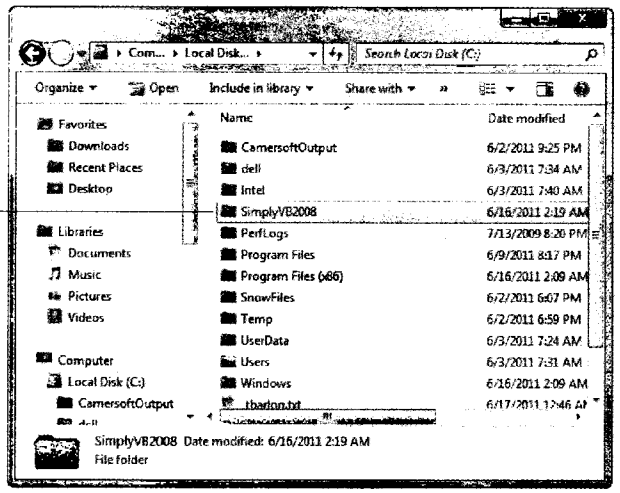


Hình 2 Tạo thư mục mới.



Hình 3 Thư mục mới hiện ra trong ổ C:.

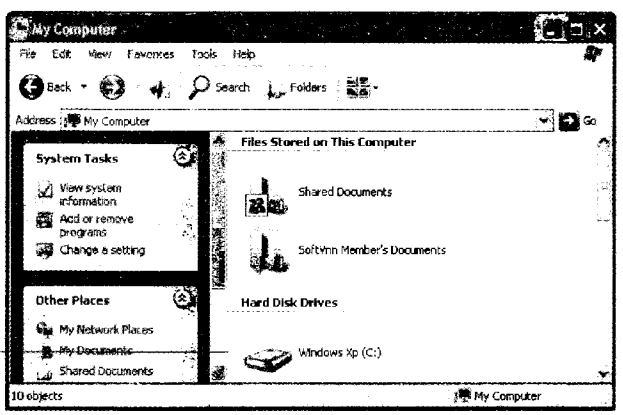
3. ***Đặt tên thư mục***. Nhập tên cho thư mục. Tốt nhất bạn nên chọn một tên dễ hiểu và dễ nhớ. Chúng tôi đã đặt tên thư mục là `SimplyVB2008` (Hình 4) và sử dụng tên này trong các thao tác trong cuốn sách này.



Hình 4 Thư mục làm việc mới trên ổ `C:`.

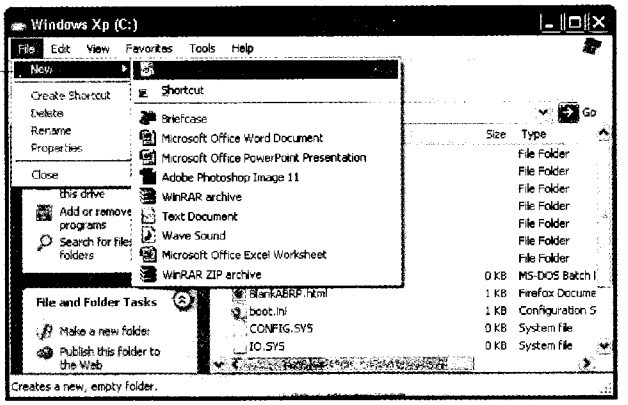
Tạo thư mục mới trên Windows XP

1. ***Chọn ổ đĩa***. Mở menu **Start** và chọn **My Computer** để truy cập vào danh sách ổ đĩa trên máy tính (Hình 5). Nhấn đúp chuột vào ổ `C:`. Nội dung của ổ `C:` được hiển thị như sau trên cửa sổ.

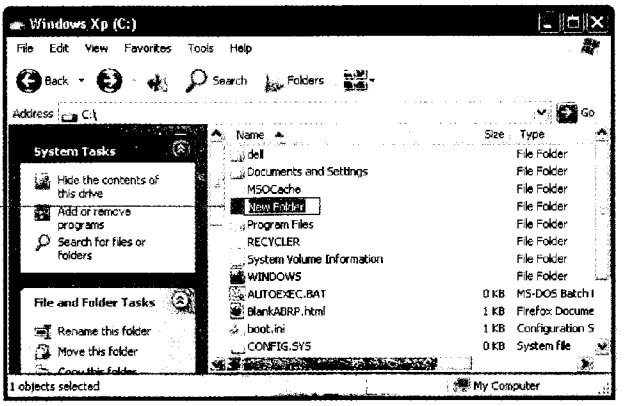


Hình 5 Các ổ đĩa được liệt kê trong **My Computer**.

2. ***Tạo thư mục mới***. Chọn menu **File**. Dưới menu con **New**, chọn **Folder** (Hình 6). Thư mục mới sẽ hiển thị trong ổ `C:` (Hình 7). [*Ghi chú*: Từ thời điểm này, chúng tôi sử dụng ký tự **>** để minh họa việc chọn menu. Ví dụ, để thực hiện bước này, chúng tôi ký hiệu **File > New > Folder**].

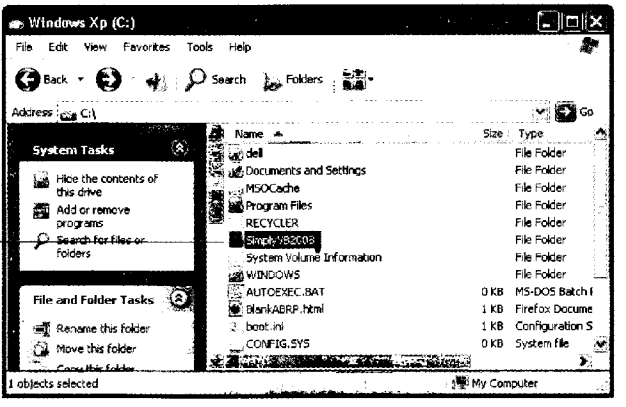


Hình 6 Tạo thư mục mới.



Hình 7 Thư mục mới hiện ra trên ổ C:.

3. ***Đặt tên thư mục.*** Nhập tên cho thư mục mới tạo. Tốt nhất bạn nên đặt một cái tên dễ nhớ và dễ hiểu. Chúng tôi chọn `SimplyVB2008` (Hình 8). Bạn có thể sử dụng thư mục này để lưu các ứng dụng do bạn tạo ra và các bài tập của bạn.



Hình 8 Thư mục mới tạo trên ổ C:.

Cài đặt phần mềm

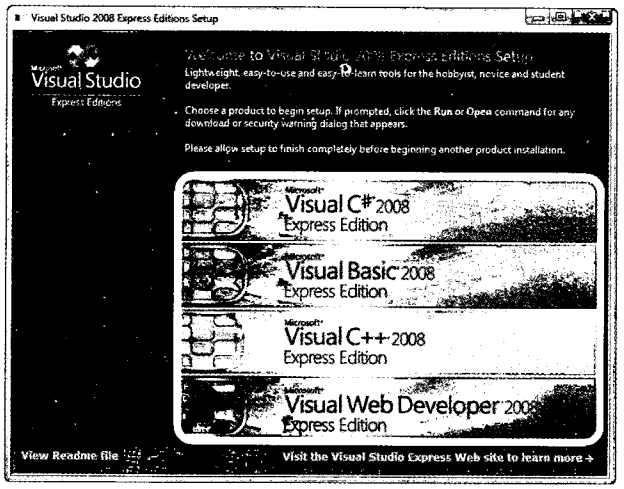
Trước khi bạn có thể chạy ứng dụng trong *Lập trình trực quan Visual Basic 2008, phiên bản lần ba* hoặc xây dựng các ứng dụng của mình, bạn cần phải cài đặt môi trường phát triển. Chúng tôi sử dụng phiên bản Visual Basic 2008 Express của Microsoft trong các ví dụ từ Chương 1 đến Chương 22. Tất cả các phiên bản Visual Studio Express nằm trong DVD đi kèm sách hoặc thể được tải về từ địa chỉ:

```
www.microsoft.com/express/
```

Trong mục sau đây, bạn sẽ tiến hành cài đặt phần mềm Express Edition.

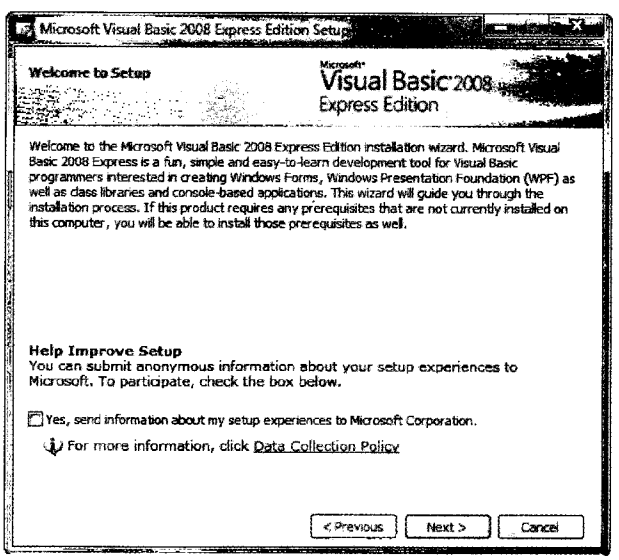
Cài đặt Visual Basic 2008 Express Edition

1. ***Chạy file cài đặt phiên bản Express***. Đưa đĩa DVD đi kèm sách vào trong ổ DVD của máy tính để chạy file cài đặt phần mềm (Hình 9). Nếu cửa sổ **Visual Studio 2008 Express Editions Setup** không hiện ra, sử dụng Windows Explorer để hiển thị nội dung của ổ DVD và nhấn đúp vào `Setup.hta` để chạy file cài đặt.



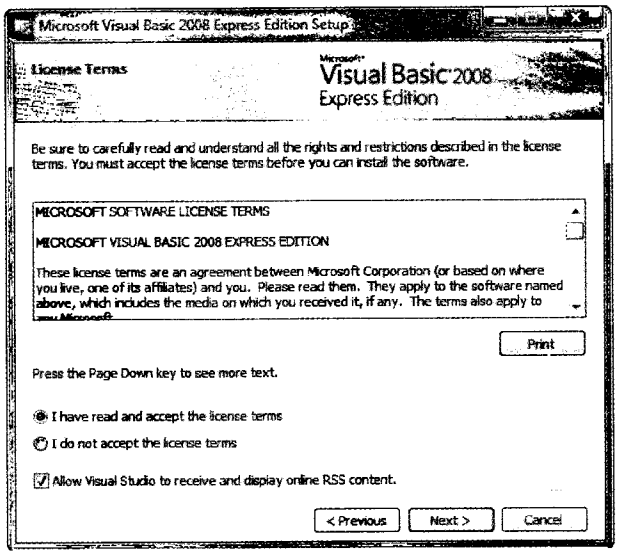
Hình 9 Cửa sổ **Visual Studio 2008 Express Editions Setup**.

2. ***Chạy file cài đặt Visual Studio 2008 Express Edition***. Trong cửa sổ **Visual Studio 2008 Express Editions Setup**, nhấn chuột vào mục **Visual Basic 2008 Express Edition** để hiển thị cửa sổ **Visual Basic 2008 Express Edition Setup** (Hình 10), sau đó nhấn **Next >**.



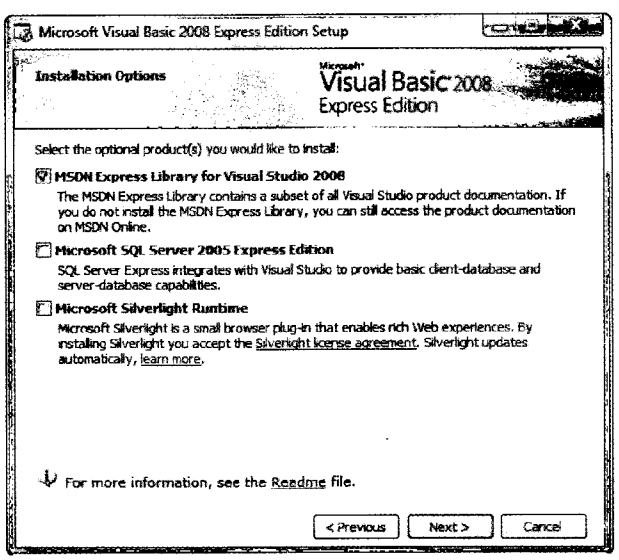
Hình 10 Cửa sổ **Visual Studio 2008 Express Editions Setup**.

3. ***Chấp nhận các điều khoản***. Hãy đọc kỹ các điều khoản trong bản thỏa thuận giấy phép sử dụng (Hình 11). Nhấn chuột vào **I have read and accept the license terms** để đồng ý với các điều khoản, sau đó nhấn **Next >**. [*Chú ý*: Nếu bạn chọn không đồng ý với các điều khoản, phần mềm sẽ không cài đặt và bạn sẽ không thể tạo và chạy ứng dụng Visual Basic].

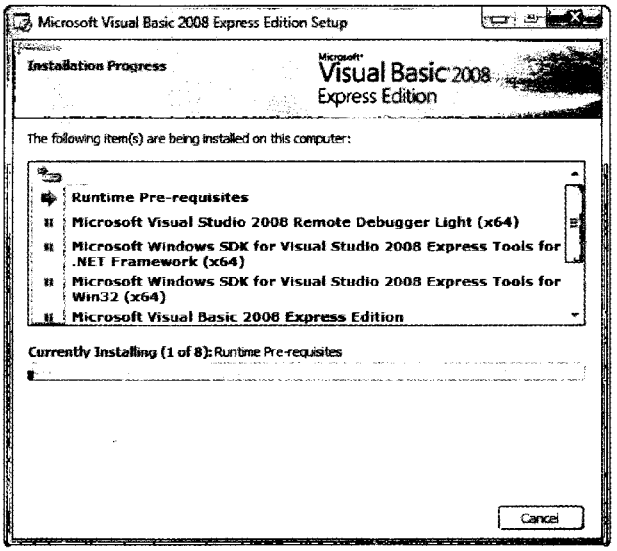


Hình 11 Chấp nhận thỏa thuận giấy phép sử dụng.

4. ***Chọn các tùy chọn cài đặt.*** Chọn **MSDN Express Library for Visual Studio 2008**, bỏ chọn **Microsoft SQL Server 2005 Express Edition (x86)** và bỏ chọn **Microsoft Silverlight Runtime** để cài đặt (Hình 12). Nhấn **Next >**. [*Chú ý*: Việc cài đặt tài liệu MSDN là không bắt buộc nhưng được khuyến khích].
5. ***Tiếp tục và hoàn tất cài đặt.*** Nhấn **Next >**, sau đó nhấn **Finish >** để tiếp tục cài đặt. Trình cài đặt sẽ bắt đầu sao chép các file cần thiết để cài đặt Visual Basic 2008 Express Edition. Chờ đợi quá trình cài đặt hoàn tất - quá trình cài đặt có thể hơi lâu. Sau khi hoàn tất cài đặt, nhấn chuột vào **Exit**.



Hình 12 Hộp thoại các tùy chọn cài đặt.

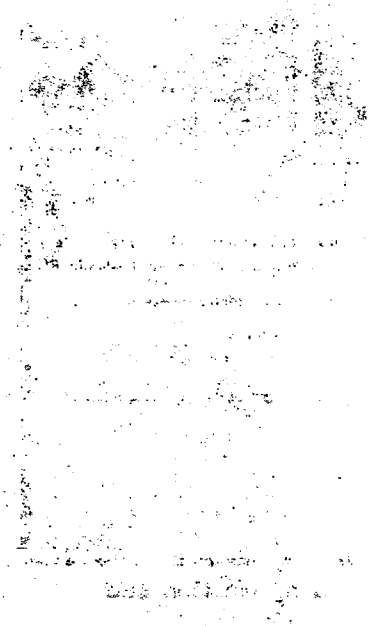


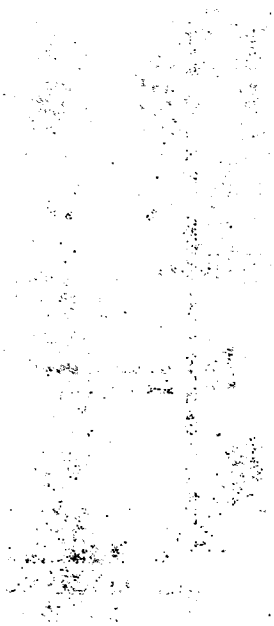
Hình 13 Quá trình cài đặt (đang tiến hành) của Visual Basic 2008 Express Edition.

Ghi chú khác

- Đối với ảnh chụp các đoạn mã trong sách, những đoạn mã được chọn sẽ được tô màu nền xám.
- Một số bạn có thể muốn thay đổi bố cục không gian làm việc (workspace) trong công cụ phát triển. Trong trường hợp muốn quay lại bố cục không gian làm việc mặc định, hãy chọn **Windows > Reset Window Layout.**
- Có một số khác biệt giữa sản phẩm Visual Studio 2008 phiên bản đầy đủ và phiên bản Express chúng tôi sử dụng trong cuốn sách này, chẳng hạn có thêm một số mục menu.
- Rất nhiều các mục menu chúng tôi sử dụng trong sách có các biểu tượng tương ứng (được hiển thị cạnh mỗi mục trên các menu) trên một trong những thanh công cụ ở trên cùng của môi trường phát triển. Vì đã quen với các biểu tượng này, bạn có thể sử dụng các thanh công cụ này để tăng tốc cho quá trình phát triển. Tương tự, nhiều mục menu có các phím tắt (cũng được hiển thị cạnh mỗi mục trên các menu) để truy cập nhanh đến các lệnh.

Bây giờ, bạn đã sẵn sàng để bắt đầu nghiên cứu Visual Basic với cuốn sách *Lập trình trực quan Visual Basic 2008, phiên bản lần ba.* Chúng tôi hy vọng bạn sẽ yêu thích cuốn sách này!





CHƯƠNG 1

Mục tiêu

Trong chương này, bạn sẽ tìm hiểu để:

- Hiểu đặc điểm của ngôn ngữ lập trình bậc thấp và bậc cao.
- Hiểu cơ bản về lập trình hướng đối tượng.
- Chạy ứng dụng Visual Basic đầu tiên.
- Hiểu thêm về .NET và Visual Basic.

Nội dung chính

Ứng dụng Advanced Painter

Giới thiệu về máy tính, Internet và Visual Basic

Chào mừng đến với Visual Basic 2008! Cuốn sách này sử dụng cách tiếp cận đơn giản, hướng dẫn từng bước để dạy những kiến thức căn bản về lập trình Visual Basic. Chúng tôi hi vọng rằng bạn sẽ nhận được những thông tin bổ ích và nhiều niềm vui khi học Visual Basic.

Đặc trưng của cuốn sách này là dạy Visual Basic sử dụng **cách tiếp cận hướng ứng dụng**, cung cấp hướng dẫn từng bước để tạo và tương tác với những ứng dụng máy tính hữu ích và thực tiễn. Cách tiếp cận này, cùng với **cách tiếp cận code trực tiếp** giới thiệu rất nhiều ứng dụng Visual Basic hoàn chỉnh và minh họa kết quả khi chạy ứng dụng, bạn sẽ được học những kỹ năng cơ bản làm nền tảng cho khả năng lập trình tốt sau này. Tất cả các chương trình ví dụ đều có thể được tải về từ website `www.deitel.com/books/simplyVB2008`.

Máy tính ngày càng được sử dụng nhiều hơn trong mọi lĩnh vực cuộc sống. Trong thời kỳ giá cả leo thang, chi phí để có được một chiếc máy tính lại đang sụt giảm đáng kể bởi sự phát triển nhanh của cả công nghệ phần cứng lẫn phần mềm. Công nghệ chíp silicon đã làm cho máy tính có giá rẻ hơn nhiều và khiến hàng tỉ chiếc máy tính phổ thông đang được sử dụng trên toàn thế giới, giúp con người trong công việc và đời sống hàng ngày, hỗ trợ các ngành công nghiệp và hoạt động của các cơ quan chính phủ.

Cuốn sách giúp bạn khởi đầu con đường học tập đầy thử thách nhưng sẽ có phần thưởng cho những người xứng đáng. Nếu bạn muốn liên lạc với chúng tôi, hãy gửi email đến địa chỉ `deitel@deitel.com`, chúng tôi sẽ phản hồi lại nhanh chóng. Để biết thêm thông tin chi tiết, hãy ghé thăm website `www.deitel.com`.

1.1 Máy tính là gì?

Máy tính là một thiết bị có thể thực hiện phép tính toán và thực hiện quyết định lôgic nhanh gấp hàng triệu, hàng tỉ thậm chí hàng nghìn tỉ lần so với con người. Ví dụ, nhiều máy tính cá nhân ngày nay có thể thực hiện hàng tỉ phép cộng mỗi giây. Một người thực hiện tính toán bằng máy tính cầm tay sẽ phải mất cả đời để hoàn thành số phép tính mà một máy tính cá nhân thực hiện trong một giây. **Siêu máy tính (supercomputer)** nhanh nhất hiện nay thậm chí còn thể thực hiện cả nghìn tỉ phép cộng trong một giây!

Máy tính xử lý dữ liệu, sử dụng một tập các chỉ lệnh được gọi là **chương trình máy tính (computer program).**

Những chương trình này hướng dẫn máy tính thông qua một chuỗi hành động được sắp xếp theo thứ tự do lập trình viên tạo ra. Trong cuốn sách này, chúng ta sử dụng thuật ngữ "ứng dụng" thay vì "chương trình". Ứng dụng là một chương trình làm một công việc hữu ích. Mỗi chương trong cuốn sách này trung bình sẽ chứa năm ứng dụng - một ứng dụng trong ví dụ và bốn ứng dụng trong các bài tập - như vậy, có khoảng 90 ứng dụng trong cuốn sách này.

Máy tính là một tập hợp bao gồm nhiều thiết bị khác nhau (như bàn phím, màn hình, chuột, ổ cứng, bộ nhớ, ổ DVD, máy in và bộ xử lý) được gọi là **phần cứng**. Những chương trình chạy trên máy tính được gọi là **phần mềm**. Lập trình hướng đối tượng (phương pháp lập trình mô hình hóa các đối tượng trên thực tế bằng các bản sao phần mềm) trong Visual Basic cũng như các ngôn ngữ lập trình khác là bước đột phá đáng kể có thể cải tiến hiệu suất lập trình.

TỰ ÔN TẬP

1. Máy tính xử lý dữ liệu, sử dụng các tập chỉ lệnh gọi là _____.
   a) phần cứng
   b) chương trình máy tính
   c) bộ xử lý
   d) lập trình viên
2. Các thiết bị tạo nên máy tính được gọi là _____.
   a) phần cứng
   b) phần mềm
   c) chương trình
   d) lập trình viên

Đáp án: **1)** b. **2)** a.

1.2 Tổ chức máy tính

Máy tính có thể được chia làm 6 khối chức năng chính:

1. **Khối nhập (Input unit)**. Bộ phận "nhận" này của máy tính nhận thông tin (dữ liệu và các chương trình máy tính) từ các **thiết bị nhập** khác nhau như bàn phím và chuột. Những thiết bị nhập khác bao gồm microphone (để ghi âm), máy scanner (để quét hình ảnh) và máy ảnh số (để chụp ảnh và tạo video).
2. **Khối xuất (Output unit)**. Bộ phận "vận chuyển" này của máy tính mang thông tin đã được xử lý và lưu trữ vào những **thiết bị xuất** khác nhau để thông tin có thể tồn tại ở bên ngoài máy tính. Kết quả xuất ra có thể được hiển thị trên màn hình, được in ra giấy, được chạy trên các thiết bị âm thanh/video, được truyền ra Internet, v.v... Kết quả xuất ra cũng có thể được sử dụng để điều khiển các thiết bị khác như rô-bốt trong sản xuất.
3. **Khối nhớ (Memory unit)**. Bộ phận "nhà kho" có sức chứa tương đối ít và có tốc độ truy cập cao này của máy tính có nhiệm vụ lưu trữ dữ liệu tạm thời khi ứng dụng đang chạy. Khối nhớ lưu lại thông tin được nhập vào thông qua các thiết bị nhập để thông tin sẵn có cho việc xử lý. Để chạy được, các chương trình cần phải nằm trong bộ nhớ. Khối nhớ cũng chứa thông tin đã qua xử lý cho đến khi nó được gửi đến người dùng thông qua thiết bị xuất. Thông thường, khối nhớ còn được gọi là **khối nhớ (memory)** hay bộ nhớ chính **(primary memory)**. Bộ nhớ truy cập ngẫu nhiên **(RAM - Random Access Memory)** là một ví dụ của bộ nhớ chính. Thông tin lưu trên bộ nhớ chính thường chỉ là tạm thời, nghĩa là dữ liệu bên trong bộ nhớ sẽ bị xóa khi máy tính bị tắt.
4. **Khối xử lý trung tâm (CPU)**. CPU là bộ phận đóng vai trò quản lý của máy tính, giám sát hoạt động của các thành phần khác. CPU thông báo cho khối nhập khi thông tin sắp được đọc vào khối nhớ, hướng dẫn cho ALU thời điểm sử dụng thông tin từ khối nhớ để tính toán và báo khối xuất thời điểm gửi thông tin từ khối nhớ đến thiết bị xuất nhất định. Nhiều máy tính để bàn cấu hình cao ngày nay có nhiều CPU.

5. **Khối số học và lôgic (ALU)**. ALU (một thành phần của CPU) là bộ phận "sản xuất" của máy tính. ALU thực hiện các phép tính như cộng, trừ, nhân, chia. ALU cũng đưa ra các quyết định cho phép máy tính thực hiện các quyết định như xác định xem hai đối tượng được lưu trong bộ nhớ có bằng nhau không.
6. **Khối lưu trữ thứ cấp (Secondary storage unit)**. Khối này là bộ phận "nhà kho" có sức chứa lớn, lâu dài của máy tính và còn được gọi là bộ nhớ thứ cấp. Các thiết bị lưu trữ thứ cấp như ổ cứng, ổ CD-ROM, ổ DVD và ổ USB thường lưu chương trình và dữ liệu mà khối khác không thể sử dụng ngay được. Máy tính cần lấy thông tin này khi cần - có thể lấy ngay lập tức hoặc có thể mất đến hàng giờ, ngày, tháng, thậm chí nhiều năm sau. Thông tin trong bộ nhớ thứ cấp có thời gian truy cập lâu hơn so với bộ nhớ chính. Tuy nhiên, bộ nhớ thứ cấp rẻ hơn rất nhiều so với bộ nhớ chính. Thông tin lưu trên bộ nhớ thứ cấp là lâu dài, tức vẫn được lưu lại ngay cả khi máy tính đã bị tắt.

TỰ ÔN TẬP

1. _____chịu trách nhiệm thực hiện tính toán và đưa ra quyết định.
   a) Khối xử lý trung tâm b) Khối nhớ
   c) Khối số học và lôgíc d) Khối xuất
2. Thông tin được lưu trữ trong_____thường bị mất khi máy tính bị tắt.
   a) bộ nhớ chính b) bộ nhớ thứ cấp
   c) ổ CD-ROM d) ổ cứng

Đáp án: **1)** c. **2)** a.

1.3 Ngôn ngữ máy, ngôn ngữ assembly và ngôn ngữ bậc cao

Lập trình viên viết chỉ lệnh bằng những ngôn ngữ lập trình khác nhau. Trong đó, một số ngôn ngữ máy tính có thể hiểu ngay được, một số ngôn ngữ khác thì phải cần có một bước dịch trung gian. Mặc dù hiện nay có hàng trăm ngôn ngữ máy tính đang được sử dụng, nhưng chúng có thể được chia thành ba kiểu cơ bản như sau:

1. Ngôn ngữ máy
2. Ngôn ngữ assembly
3. Ngôn ngữ bậc cao

Máy tính chỉ có thể hiểu trực tiếp **ngôn ngữ máy (machine language)** của riêng nó. Là "ngôn ngữ tự nhiên" của một máy tính cụ thể, ngôn ngữ máy được định nghĩa bởi thiết kế phần cứng của máy tính đó. Các ngôn ngữ máy thường bao gồm các chuỗi số (gồm các chữ số 0 và 1) hướng dẫn máy tính thực hiện các phép tính toán phổ thông. Thông thường bạn sử dụng hệ số thập phân với các chữ số từ 0 đến 9. Hệ số với chỉ hai chữ số 0 và 1 được gọi là hệ số nhị phân. Chương trình ngôn ngữ máy được gọi là các "mã nhị phân" là vì thế. Ngôn ngữ máy là loại ngôn ngữ phụ thuộc máy, nghĩa là một ngôn ngữ máy cụ thể chỉ có thể được sử dụng trên một kiểu máy tính tương ứng. Đoạn ngôn ngữ máy sau đây sẽ cộng *số tiền làm thêm giờ* vào *tiền công cơ bản* và lưu trữ kết quả vào *tổng thù lao phải trả* được trình bày ở dạng mà con người không thể hiểu được.

```
+1300042774
+1400593419
+1200274027
```

Trong khi máy tính ngày càng trở nên phổ biến, lập trình trên ngôn ngữ máy tỏ ra rất chậm chạp và dễ gây ra lỗi. Thay vì sử dụng các chuỗi số mà máy tính có thể trực tiếp hiểu được, lập trình viên sử dụng từ tiếng Anh viết tắt để đại diện cho các phép tính cơ bản của máy tính. Từ viết tắt chính là cơ sở của **ngôn ngữ assembly.** Các **chương trình dịch (translator program)** được gọi là các

assembler (bộ hợp ngữ) giúp chuyển chương trình ngôn ngữ assembly sang ngôn ngữ máy với tốc độ máy tính. Đoạn mã sau đây của chương trình viết bằng ngôn ngữ **assembly** cũng cộng *tiền công làm thêm giờ* vào *tiền công cơ bản* và lưu trữ *tổng thù lao phải trả*, nhưng lần này người ta có vẻ dễ hiểu hơn một chút so với ví dụ ngôn ngữ máy.

```
LOAD    BASEPAY
ADD     OVERPAY
STORE   GROSSPAY
```

Mặc dù đã trở nên rõ ràng hơn cho con người, nhưng máy tính lại không thể hiểu được đoạn mã viết bằng ngôn ngữ assembly cho đến khi chúng được dịch sang ngôn ngữ máy bằng chương trình dịch .

Tốc độ viết chương trình của lập trình viên sẽ được cải thiện với sự ra đời của ngôn ngữ assembly, tuy nhiên ngôn ngữ này vẫn yêu cầu nhiều chỉ lệnh để thực hiện thậm chí một tác vụ đơn giản. Để tăng tốc độ lập trình, các **ngôn ngữ bậc cao (high-level language)** - ngôn ngữ mà một lệnh đơn thực hiện một lượng lớn các tác vụ - đã được phát triển. Các chương trình dịch, được gọi là **trình biên dịch (compiler)** chuyển ngôn ngữ bậc cao thành ngôn ngữ máy. Ngôn ngữ bậc cao cho phép lập trình viên viết chỉ lệnh trông như những câu nói tiếng Anh hàng ngày và chứa các ký hiệu toán học thông thường. Chẳng hạn ứng dụng tính lương được viết bằng ngôn ngữ bậc cao có thể chứa lệnh như sau:

```
grossPay = basePay + overTimePay
```

Từ các ví dụ trên, ta có thể thấy vì sao lập trình viên lại thích ngôn ngữ bậc cao hơn so với ngôn ngữ máy và ngôn ngữ assembly. Visual Basic là một trong những ngôn ngữ bậc cao phổ biến trên thế giới. Trong phần tiếp theo, bạn sẽ học về phiên bản mới nhất của Microsoft đó là Visual Basic 2008.

TỰ ÔN TẬP

1. Ngôn ngữ lập trình duy nhất mà máy tính có thể hiểu được trực tiếp đó là_____của riêng nó.
   a) ngôn ngữ bậc cao
   b) ngôn ngữ assembly
   c) ngôn ngữ máy
   d) tiếng Anh
2. Chương trình dịch ngôn ngữ bậc cao sang ngôn ngữ máy được gọi là_____.
   a) assembler
   b) trình biên dịch
   c) lập trình viên
   d) bộ chuyển đổi

Đáp án: **1)** c. **2)** b.

1.4 Visual Basic

Visual Basic được phát triển từ **BASIC** (Beginner's All-purpose Symbolic Instruction Code). BASIC là ngôn ngữ để viết ra chương trình đơn giản nhanh chóng và dễ dàng, ngôn ngữ này được phát triển vào những năm 1960 bởi giáo sư John Kemeny và Thomas Kurtz của Đại học Dartmouth. Mục đích chính của BASIC là để hướng dẫn những người mới học về kỹ thuật lập trình cơ bản.

Khi Bill Gates thành lập Microsoft vào những năm 1970, ông đã sử dụng BASIC trên một số máy tính cá nhân. Vào cuối những năm 1980 và đầu những năm 1990, Microsoft đã phát triển **giao diện người dùng đồ họa (GUI)** Microsoft Windows - thành phần trực quan của hệ điều hành mà người dùng có thể tương tác được. Với việc tạo ra Windows GUI, BASIC phát triển thành **Visual Basic** và được giới thiệu bởi Microsoft vào năm 1991 để việc lập trình ứng dụng Windows trở nên dễ dàng hơn.

Cho đến khi Visual Basic xuất hiện, phát triển ứng dụng Microsoft Windows là công việc khá khó khăn. Visual Basic hiện nay là ngôn ngữ hướng đối tượng,

hướng sự kiện trong đó chương trình được tạo bằng cách sử dụng công cụ phần mềm **môi trường phát triển tích hợp (Integrated Development Environment IDE)**. Với **Visual Studio** IDE của Microsoft, bạn có thể viết, chạy, kiểm thử và gỡ lỗi cho ứng dụng Visual Basic nhanh chóng và thuận tiện hơn.

Phiên bản mới nhất của Visual Basic là phiên bản hướng đối tượng hoàn toàn - bạn sẽ học những kiến thức cơ bản về kỹ thuật hướng đối tượng ngay sau đây và sẽ nghiên cứu kỹ hơn trong phần còn lại của cuốn sách này. Visual Basic là ngôn ngữ hướng sự kiện - bạn sẽ viết chương trình đáp lại những **sự kiện (event)** của người dùng như nhấn chuột, nhấn vào phím. Visual Basic là ngôn ngữ lập trình trực quan, ngoài việc viết mã để tạo ứng dụng, bạn cũng sẽ sử dụng giao diện người dùng đồ họa của Visual Studio để kéo và thả các đối tượng được định nghĩa trước như `Button` (nút) và `Textbox` (hộp văn bản) lên giao diện của ứng dụng và đồng thời dễ dàng thay đổi kích thước của chúng, còn Visual Studio sẽ viết mã để tạo giao diện cho bạn.

Microsoft giới thiệu chiến lược .NET (phát âm là "dot-net") vào năm 2000. **Nền tảng .NET (.NET platform)** - là tập các thành phần phần mềm giúp chương trình .NET có thể chạy được - cho phép ứng dụng được triển khai trên nhiều thiết bị khác nhau (như điện thoại di động) cũng như máy tính để bàn. Nền tảng .NET cung cấp mô hình lập trình cho phép các thành phần phần mềm được tạo ra bằng các ngôn ngữ lập trình khác nhau (như Visual Basic và C#) có thể giao tiếp với nhau. Chúng ta sẽ tìm hiểu thêm về .NET chi tiết hơn trong Mục 1.9.

TỰ ÔN TẬP

1. Microsoft tạo_____vào năm 1991 để lập trình ứng dụng Windows trở nên dễ dàng hơn.
   a) Windows b) BASIC
   c) Visual Basic d) C#
2. Visual Basic được phát triển từ_____- ngôn ngữ được tạo ra để viết chương trình nhanh chóng và đơn giản.
   a) .NET b) Windows
   c) Java d) BASIC

Đáp án: **1)** c. **2)** d.

1.5 Các ngôn ngữ bậc cao khác

Mặc dù hàng trăm ngôn ngữ bậc cao khác đã được phát triển, chỉ một số là được sử dụng rộng rãi. IBM đã phát triển **Fortran** (Formula Translator) vào giữa những năm 1950 để tạo ứng dụng khoa học công nghệ yêu cầu tính toán toán học phức tạp. Fortran hiện vẫn được sử dụng rộng rãi.

COBOL được phát triển vào cuối những năm 1950 bởi một nhóm các nhà sản xuất máy tính hợp tác cùng chính phủ và những người dùng máy tính. COBOL được sử dụng chủ yếu cho ứng dụng kinh doanh yêu cầu thao tác với một lượng lớn dữ liệu. Phần lớn các phần mềm kinh doanh ngày nay vẫn sử dụng COBOL.

Ngôn ngữ C, được phát triển bởi Dennis Ritchie ở Bell Laboratories vào những năm 1970 đã nhận được sự công nhận rộng rãi như ngôn ngữ lập trình cho hệ điều hành UNIX, C++ (là mở rộng của ngôn ngữ C, được phát triển bởi Bjarne Stroustrup vào đầu những năm 1980 ở Bell Laboratories). C++ cung cấp khả năng **lập trình hướng đối tượng (OOP)**. Nhiều hệ điều hành lớn hiện nay (như Microsoft Windows) được viết bằng C hoặc C++.

Đối tượng (object) là **thành phần (component)** phần mềm có thể tái sử dụng để mô hình hóa đối tượng trong thế giới thực. Chương trình hướng đối tượng thường dễ hiểu, chính xác và dễ thay đổi hơn chương trình được viết bằng những kỹ thuật khác. Visual Basic 2008 cung cấp đầy đủ tính năng cho lập trình hướng đối tượng.

Vào đầu những năm 1990, nhiều tổ chức trong đó có Sun Microsystems dự đoán thiết bị điện tử tiêu dùng thông minh sẽ có thị trường lớn nên sẽ tác động mạnh mẽ lên **vi xử lý (microprocessor)** - là con chíp giúp máy tính có thể hoạt động. Tuy nhiên thị trường này đã không phát triển nhanh chóng như Sun mong đợi. Trong khi World Wide Web bùng nổ vào năm 1993, Sun đã nhìn ra tiềm năng sử dụng ngôn ngữ lập trình mới của mình là **Java** để tạo ra trang web có nội dung động có thể tương tác. Sun đã giới thiệu **Java** vào năm 1995 và đã thu hút sự chú ý của cộng đồng bởi web hiện tại đang được quan tâm. Lập trình viên ngày nay sử dụng Java để tạo trang web có nội dung động (nội dung được tạo ra để đáp lại những tương tác của người dùng), để tạo ứng dụng doanh nghiệp có quy mô lớn, để cải thiện khả năng của web server (máy tính cung cấp nội dung cho trình duyệt web của bạn khi bạn lướt web), để cung cấp ứng dụng cho các thiết bị tiêu dùng (điện thoại di động, máy nhắn tin và PDA) và cho nhiều mục đích khác.

Trong năm 2000, Microsoft đã giới thiệu **C#** (phát âm là "C-Sharp") vào thời điểm giới thiệu chiến lược .NET. Ngôn ngữ lập trình C# được thiết kế dành riêng cho nền tảng .NET. Nó có nguồn gốc từ C, C++ và Java, và tiếp thu những ưu điểm của những ngôn ngữ này. Giống như Visual Basic, C# là ngôn ngữ lập trình hướng đối tượng và có khả năng truy cập đầy đủ đến thư viện các thành phần dựng sẵn của .NET, giúp bạn lập trình nhanh hơn. C#, Java và Visual Basic có những khả năng tương tự nhau nên học Visual Basic có thể mang lại cho bạn nhiều cơ hội nghề nghiệp.

TỰ ÔN TẬP

1. _____ là mở rộng của C và cung cấp khả năng hướng đối tượng.
   a) Visual Basic b) C++
   b) Ngôn ngữ assembly d) Windows
2. _____là ngôn ngữ lập trình ban đầu được phát triển cho nền tảng .NET của Microsoft.
   a) C# b) Java
   c) C++ d) Visual Basic
3. _____được phát triển vào cuối những năm 1950 và được sử dụng để lập trình phần lớn các phần mềm kinh doanh ngày nay.
   a) COBOL b) Fortran
   c) Java d) C
4. _____được phát triển vào những năm 1950, được sử dụng để tạo ứng dụng khoa học kỹ thuật yêu cầu tính toán toán học phức tạp.
   a) Visual Basic b) Fortran
   c) COBOL d) C#

Đáp án: **1)** b. **2)** a. **3)** a. **4)** b.

1.6 Lập trình cấu trúc

Trong suốt những năm 1960, việc phát triển phần mềm thường tốn công sức vượt ngoài dự kiến, chi phí lớn hơn ngân sách dự án trong khi sản phẩm lại chưa đủ độ tin cậy. Mọi người bắt đầu nhận ra rằng việc phát triển phần mềm phức tạp hơn tưởng tượng. Hoạt động nghiên cứu nhằm giải quyết các vấn đề xảy ra đã dẫn tới sự ra đời **lập trình cấu trúc (structured programming)** - là hướng tiếp cận có quy tắc để tạo chương trình rõ ràng, chính xác và dễ chỉnh sửa.

Kết quả của nghiên cứu này là sự phát triển ngôn ngữ lập trình **Pascal** vào năm 1971. Pascal được đặt theo tên của nhà toán học và nhà triết học thế kỷ 17 Blaise Pascal. Pascal được thiết kế để giảng dạy lập trình cấu trúc và nhanh chóng trở ngôn ngữ lập trình được chọn để giới thiệu cho phần lớn sinh viên. Đáng tiếc là ngôn ngữ này lại thiếu một số tính năng cần thiết cho ứng dụng thương mại, công

nghiệp và chính phủ. Ngược lại, C - ngôn ngữ cũng ra đời từ việc nghiên cứu lập trình cấu trúc, khắc phục được những hạn chế của Pascal. Do đó, các lập trình viên chuyên nghiệp đã nhanh chóng chọn C.

Ngôn ngữ lập trình **Ada**, được xây dựng dựa trên Pascal và được tài trợ bởi Bộ quốc phòng Mỹ trong suốt những năm 1970 và đầu những năm 1980. Ngôn ngữ này được đặt theo tên của Ada Byron, quý bà Lovelace, con gái của nhà thơ Lord Byron. Lovelace được công nhận là lập trình viên đầu tiên của thế giới. Bà đã viết ứng dụng vào đầu những năm 1800 cho thiết bị máy tính cơ khí mang tên Analytical Engine của Charles Babbage.

TỰ ÔN TẬP

1. Trong suốt những năm 1960 và 1970, việc nghiên cứu để giải quyết vấn đề về phát triển phần mềm như chậm tiến độ, vượt ngân sách và sản phẩm tạo ra không đáng tin cậy dẫn đến sự ra đời của_____.
   a) đa luồng
   b) lập trình hướng đối tượng
   c) Ada
   d) lập trình cấu trúc
2. _____ được thiết kế để dạy lập trình cấu trúc.
   a) C++
   b) C
   c) Java
   d) Pascal

Đáp án: **1)** d. **2)** d.

1.7 Xu hướng phần mềm chủ đạo: Kỹ thuật đối tượng

Khi lợi ích của lập trình cấu trúc được công nhận vào những năm 1970, kỹ thuật phần mềm được cải tiến hơn bắt đầu xuất hiện. Cho đến những năm 1980 và 1990, lập trình hướng đối tượng mới được sử dụng rộng rãi, tuy nhiên, ngay từ những năm trước lập trình viên đã cảm thấy họ có trong tay công cụ để cải tiến mạnh mẽ quy trình phát triển phần mềm.

Đối tượng là gì và tại sao chúng lại đặc biệt? **Kỹ thuật đối tượng (object technology)** là cách thức đóng gói nhằm tạo những đơn vị phần mềm hữu ích. Có đối tượng ngày tháng, đối tượng thời gian, đối tượng phiếu lương, đối tượng hóa đơn, đối tượng con người, đối tượng âm thanh, đối tượng video, đối tượng file, đối tượng bản ghi, v.v... Thực ra thì hầu như bất kỳ danh từ nào cũng có thể thể hiện bằng đối tượng phần mềm. Đối tượng có **thuộc tính (property)** - còn gọi là **đặc tính (attribute)** chẳng hạn như màu sắc, kích thước, trọng lượng và thực hiện **hành động (action)** - còn gọi là **hành vi (behavior)** hay **phương thức (method)** - như di chuyển, ngủ hay vẽ. **Lớp (Class)**[1] là kiểu của nhóm đối tượng có liên quan. Ví dụ tất cả những ô tô đều thuộc lớp "ô tô" mặc dù mỗi ô tô có thể khác nhau về hãng sản xuất, kiểu dáng, màu sắc và các thành phần bổ sung. Lớp chỉ ra định dạng chung cho các đối tượng của lớp và các thuộc tính, hành động có sẵn cho đối tượng thuộc lớp đó. Đối tượng liên quan đến lớp cũng tương tự như ngôi nhà và bản thiết kế của ngôi nhà vậy. Người thợ có thể xây dựng nhiều ngôi nhà từ cùng một bản thiết kế, tương tự như vậy, lập trình viên cũng có thể tạo nhiều đối tượng từ cùng một lớp.

Trước khi ngôn ngữ hướng đối tượng xuất hiện, **ngôn ngữ lập trình thủ tục (procedural programming language)** - như Fortran, Pascal, BASIC và C - tập trung vào hành động (động từ) hơn là vật thể hoặc đối tượng (danh từ). Điều này khiến cho công việc lập trình trở nên rắc rối. Tuy nhiên, sử dụng ngôn ngữ hướng đối tượng phổ biến ngày nay như Visual Basic, C++, Java và C#, bạn có thể lập trình theo hướng đối tượng, cách tiếp cận này phản ánh đúng với cách bạn nhận thức thế giới. Điều này nâng cao hiệu suất lập trình lên đáng kể.

1: Có thể xem như một bản thiết kế hay một mẫu ban đầu của những đối tượng thuộc cùng một loại nào đó

Với kỹ thuật đối tượng, lớp được thiết kế hợp lý có thể được tái sử dụng trong những dự án sau này. Sử dụng lớp có sẵn trong thư viện có thể giảm công sức đáng kể khi xây dựng những hệ thống mới. Một số tổ chức chia sẻ rằng lợi ích lớn nhất mà họ nhận được từ lập trình hướng đối tượng thực ra không phải là tái sử dụng phần mềm, mà là tạo ra các phần mềm dễ hiểu hơn vì được tổ chức tốt hơn và dễ bảo trì hơn.

Hướng đối tượng cho phép bạn tập trung vào "bức tranh tổng thể". Bạn không mất thời gian bận tâm về chi tiết cách đối tượng tái sử dụng thực thi như thế nào. Thay vào đó, bạn tập trung vào hành vi và tương tác giữa các đối tượng. Bản đồ chỉ đường thể hiện tất cả cây cối, nhà cửa sẽ rất khó đọc. Khi loại bỏ những chi tiết như vậy và chỉ để lại những chi tiết cần thiết, bản đồ sẽ trở nên dễ hiểu hơn. Tương tự như vậy, khi ứng dụng được chia thành các đối tượng sẽ dễ hiểu, dễ chỉnh sửa và dễ cập nhật bởi vì chúng che giấu đi nhiều chi tiết phức tạp.

Rõ ràng lập trình hướng đối tượng sẽ trở thành phương thức lập trình chủ đạo trong một vài thập kỷ tới. Visual Basic là một trong những ngôn ngữ hướng đối tượng được sử dụng rộng rãi nhất trên thế giới.

TỰ ÔN TẬP

1. _____chú trọng vào các hành động (động từ) hơn là vật thể (danh từ).
   a) C# b) Lập trình hướng đối tượng
   c) Visual Basic d) Lập trình thủ tục
2. Trong lập trình hướng đối tượng,_____, tương tự như bản thiết kế, là kiểu của những đối tượng liên quan.
   a) lớp b) đặc tính
   c) hành vi d) thuộc tính

Đáp án: **1)** d. **2)** a.

1.8 Internet và World Wide Web

Vào cuối những năm 1960, ARPA - cơ quan dự án nghiên cứu cao cấp thuộc Bộ Quốc phòng Mỹ - đề ra kế hoạch nối mạng hệ thống máy tính trung tâm của hơn mười trường đại học được đầu tư bởi ARPA và các viện nghiên cứu. Các máy tính đã được kết nối với đường truyền thông có tốc độ 56 Kb/s (1 Kb/s bằng 1024 bit một giây). Vào thời điểm đó, hầu hết những người có kết nối mạng đều kết nối thông qua đường dây điện thoại có tốc độ 110 bit mỗi giây. Dự án là một bước tiến khổng lồ. Ngay sau đó, ARPA đã tiến hành triển khai mạng tên là **ARPAnet**, tiền thân của mạng **Internet** ngày nay.

Mọi thứ trên thực tế dần khác so với kế hoạch ban đầu. Mặc dù ARPAnet giúp các nhà nghiên cứu có thể kết nối máy tính của họ, nhưng mục đích chính của việc này là khả năng truyền thông nhanh chóng và dễ dàng qua **thư điện tử (email)**. Điều này vẫn đúng đối với Internet ngày nay, thông qua email, thông điệp tức thời (instant messaging) và truyền nhận file (file transfer) cho phép hàng tỉ người trên toàn cầu có thể giao tiếp với nhau.

Giao thức (hay tập các quy tắc) để truyền thông thông qua ARPAnet là **Transmission Control Protocol (TCP)**. TCP đảm bảo rằng các thông điệp, còn được gọi là các gói (packet), được định tuyến hợp lý từ người gửi đến người nhận, đến nơi nguyên vẹn và được ghép lại theo đúng thứ tự.

Song song với sự phát triển của Internet, các tổ chức trên toàn cầu đã xây dựng nên các mạng riêng cho mục đích truyền thông nội bộ trong tổ chức và giữa các tổ chức với nhau. Rất nhiều phần cứng và phần mềm mạng xuất hiện. Khó khăn lúc này là làm sao cho những mạng khác nhau này có thể giao tiếp với nhau. ARPA giải quyết vấn đề này bằng cách phát triển **Internet Protocol (IP)**. Internet Protocol (IP) tạo ra "mạng của mạng" thực sự và là kiến trúc ngày nay của Internet. Tập các giao thức kết hợp ngày nay gọi là **TCP/IP**.

Doanh nghiệp nhanh chóng nhận ra rằng bằng cách sử dụng Internet, họ có thể cải tiến hoạt động và cung cấp những dịch vụ mới và tốt hơn cho khách hàng. Các công ty bắt đầu đầu tư một lượng lớn ngân sách cho việc phát triển và cải thiện bộ mặt của họ trên Internet. Điều này tạo ra cuộc cạnh tranh khốc liệt giữa các hãng truyền thông, các nhà cung cấp phần cứng và phần mềm để đáp ứng được nhu cầu về hạ tầng. Kết quả của sự cạnh tranh này là **băng thông (bandwidth)** - khả năng mang thông tin của đường truyền thông - trên Internet gia tăng đáng kể trong khi chi phí về phần cứng ngày càng thấp.

World Wide Web là một tập các phần cứng và phần mềm hỗ trợ Internet và cho phép người dùng máy tính có thể tìm kiếm và xem tài liệu đa phương tiện (tài liệu chứa văn bản, đồ họa, hoạt hình, âm thanh và video) trên hầu hết các lĩnh vực. Mặc dù Internet được phát triển cách đây hơn ba thập kỷ nhưng World Wide Web (WWW) lại mới được giới thiệu gần đây. Vào năm 1989, Tim Berners-Lee làm việc cho Tổ chức nghiên cứu hạt nhân châu Âu (CERN - European Organization for Nuclear Research) đã bắt đầu phát triển công nghệ để chia sẻ thông tin thông qua các tài liệu văn bản được "siêu liên kết" với nhau. Berners-Lee đã gọi phát minh của ông là **ngôn ngữ đánh dấu siêu văn bản (Hypertext Markup Language - HTML)**. Ông cũng viết ra các giao thức truyền thông như **giao thức truyền tải siêu văn bản (HyperText Transfer Protocol - HTTP)** để hình thành nên nền tảng cho hệ thống thông tin siêu văn bản của ông mà sau này trở thành World Wide Web.

Vào tháng 10 năm 1994, Berners-Lee đã thành lập tổ chức - được gọi là **World Wide Web Consortium (W3C, `www.w3.org`)** - để phát triển các công nghệ cho World Wide Web. Một trong những mục tiêu chính của W3C là tất cả mọi người có thể truy cập web bao gồm cả những người khuyết tật, hay những người có ngôn ngữ và văn hóa khác nhau.

Internet và web chắc chắn sẽ được xem là một trong những sáng tạo quan trọng nhất của loài người. Trong quá khứ, hầu hết ứng dụng máy tính chỉ chạy trên máy tính "độc lập" (máy tính không kết nối với các máy tính khác). Ứng dụng ngày nay có thể được viết với mục đích truyền thông giữa các máy tính trên thế giới. Như bạn có thể thấy, thực ra mục đích này cũng chính là mục đích chính mà chiến lược .NET của Microsoft hướng đến. Internet và World Wide Web khiến thông tin có thể được truy cập ngay tức thì và thuận tiện cho mọi đối tượng, làm cho cá nhân hay doanh nghiệp có thể được toàn thế giới biết đến. Internet và World Wide Web cũng thay đổi sâu sắc đến cách làm việc và cách sống của con người.

TỰ ÔN TẬP

1. Internet ngày nay đã phát triển từ_____- một dự án của Bộ Quốc phòng Mỹ.
   a) ARPAnet b) HTML
   c) CERN d) WWW
2. Tập các giao thức để truyền thông thông qua mạng Internet được gọi là _____.
   a) HTML b) TCP/IP
   c) ARPA d) TCP

Đáp án: 1) a. 2) b.

1.9 Giới thiệu về Microsoft .NET

Vào tháng 6 năm 2000, Microsoft đã giới thiệu chiến lược **.NET (.NET initiative)** - `www.microsoft.com/net`, một tầm nhìn rộng và mới về sử dụng Internet và web trong phát triển, phân phối và sử dụng phần mềm. Thay vì ép buộc lập trình viên sử dụng một ngôn ngữ lập trình duy nhất, chiến lược .NET cho phép lập trình viên tạo ứng dụng .NET bằng bất cứ ngôn ngữ nào tương thích với .NET (như Visual Basic,

Visual C++, Visual C# và nhiều ngôn ngữ khác). Một phần của chiến lược .NET là công nghệ **ASP.NET** của Microsoft cho phép bạn tạo ứng dụng web.

Chiến lược .NET mở rộng ý tưởng **tái sử dụng phần mềm (software reuse)** cho Internet bằng cách cho phép lập trình viên chỉ tập trung vào thành phần đặc trưng riêng cho ứng dụng mà không phải xây dựng mọi thành phần của ứng dụng. Lập trình trực quan (sẽ được trình bày ở phần sau của cuốn sách) đã trở nên phổ biến bởi nó giúp lập trình viên có thể tạo ứng dụng Windows và web dễ dàng bằng cách sử dụng thành phần đồ họa được đóng gói sẵn như **Button, Textbox** và **Scrollbar.**

.NET Framework là trái tim trong chiến lược .NET của Microsoft. Framework này thực thi ứng dụng và dịch vụ web, chứa thư viện lớp (còn gọi là **Framework Class Library**) và cung cấp các khả năng lập trình khác mà bạn sẽ sử dụng để xây dựng nên các ứng dụng Visual Basic. Trong cuốn sách này, bạn sẽ phát triển phần mềm .NET bằng Visual Basic. Steve Ballmer, CEO của Microsoft, đã tuyên bố rằng Microsoft đã "đánh cược cả công ty" vào .NET. Cam kết nghiêm túc như vậy hứa hẹn tương lai tươi sáng cho lập trình viên Visual Basic 2008.

TỰ ÔN TẬP

1. _____là công nghệ thiết kế dành riêng cho nền tảng .NET và hướng đến các lập trình viên phát triển ứng dụng web.
   a) Visual Basic b) C++
   c) HTML d) ASP.NET
2. Lập trình viên sử dụng_____- là một phần của .NET Framework - để xây dựng ứng dụng Visual Basic 2008.
   a) Visual Basic Library b) Framework Class Library
   c) Microsoft Class Library d) Visual Basic Framework

Đáp án: 1) d. 2) b.

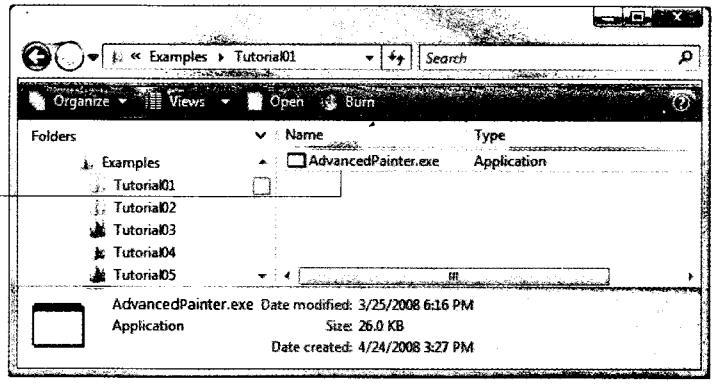
1.10 Chạy thử ứng dụng Advanced Painter của Visual Basic

Bạn sẽ có cơ hội "thử" những chức năng của ứng dụng trong mỗi chương. Bạn sẽ thực sự được chạy và tương tác với ứng dụng đã hoàn chỉnh. Sau đó, bạn sẽ được học các tính năng Visual Basic cần thiết để xây dựng ứng dụng. Cuối cùng, bạn sẽ tổng hợp những gì đã học được lại để tự xây dựng cho mình một phiên bản của ứng dụng. Bây giờ bạn sẽ bắt đầu Chương 1 bằng việc chạy ứng dụng có sẵn cho phép người dùng có thể vẽ bằng "cọ vẽ" với bốn màu sắc và ba kích thước khác nhau.

Mục *Chạy thử ứng dụng* ***Advanced Painter*** sau đây sẽ cho bạn biết người dùng có thể sử dụng phần mềm để vẽ bằng các kiểu cọ vẽ khác nhau như thế nào. Các thành phần và chức năng bạn thấy trong ứng dụng này là các thành phần và chức năng tiêu biểu khi bạn học về lập trình trong cuốn sách này. [*Lưu ý*: Chúng tôi sẽ sử dụng font chữ khác nhau để phân biệt giữa các tính năng của IDE (như tên menu và các mục trên menu) và các thành phần khác hiển thị trên IDE. Quy ước của chúng tôi nhấn mạnh các thành phần trên IDE (như menu **File**) bằng font chữ **HP-Helve-Condense** và nhấn mạnh các thành phần khác như tên file (ví dụ `Form1.cs`) bằng font `Lucida Sans Typewriter`. Các định nghĩa được **in đậm.**]

Chạy thử ứng dụng Advanced Painter

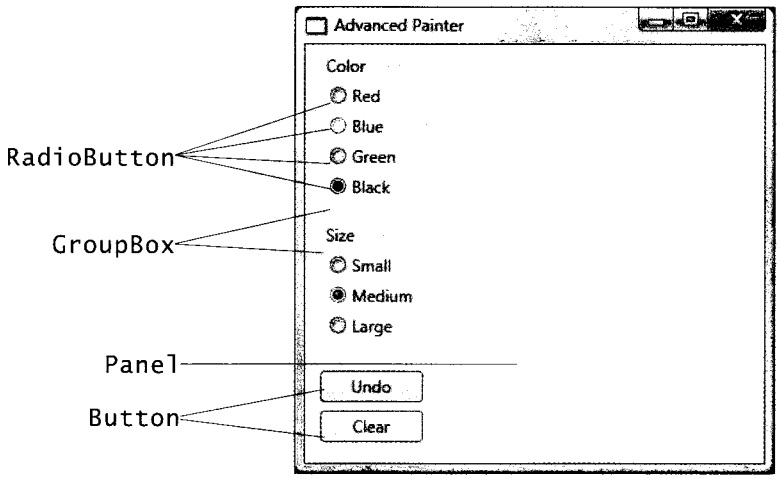
1. ***Kiểm tra cài đặt.*** Đảm bảo rằng bạn đã cài đặt máy tính đúng bằng cách xem lại mục *Chuẩn bị* đằng sau mục *Lời nói đầu.*
2. ***Tìm thư mục chứa ứng dụng.*** Mở một cửa sổ Windows Explorer và di chuyển đến thư mục `C:\Examples\Tutorial01` (Hình 1.1).



Nhấn đúp vào file này để chạy ứng dụng

Hình 1.1 Nội dung của `C:\Examples\Tutorial01`.

3. ***Chạy ứng dụng Advanced Painter.*** Lúc này bạn đang ở thư mục chứa ứng dụng, nhấn đúp vào tên file `AdvancedPainter.exe` (Hình 1.1) để chạy ứng dụng (Hình 1.2).



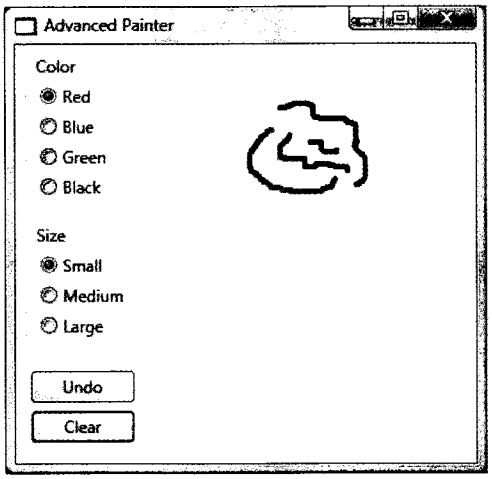
Hình 1.2 Ứng dụng Visual Basic **Advanced Painter**.

Trong Hình 1.2, một số đối tượng đồ họa - còn gọi là **điều khiển (control)** được gán nhãn bên cạnh. Các điều khiển bao gồm `GroupBox` (hộp nhóm), `RadioButton` (hộp chọn Radio), `Panel` (khung) và các `Button` (các điều khiển này sẽ được trình bày chi tiết hơn ở phần sau của cuốn sách này). Ứng dụng này cho phép bạn vẽ bằng cọ vẽ màu đỏ, xanh da trời, xanh lá cây hoặc đen với các nét vẽ nhỏ, trung bình, lớn. Bạn sẽ khám phá các khả năng trên khi chạy thử ứng dụng. Ngoài ra bạn có thể hủy bỏ các thao tác vừa thực hiện hoặc xóa toàn bộ bản vẽ và vẽ lại từ đầu.

Bằng cách sử dụng điều khiển có sẵn (thực chất là đối tượng), bạn có thể tạo ứng dụng bằng Visual Basic nhanh hơn rất nhiều so với việc tự viết mã. Trong cuốn sách này, bạn sẽ học cách sử dụng điều khiển có sẵn và cách tự viết mã để hoàn thiện ứng dụng của mình.

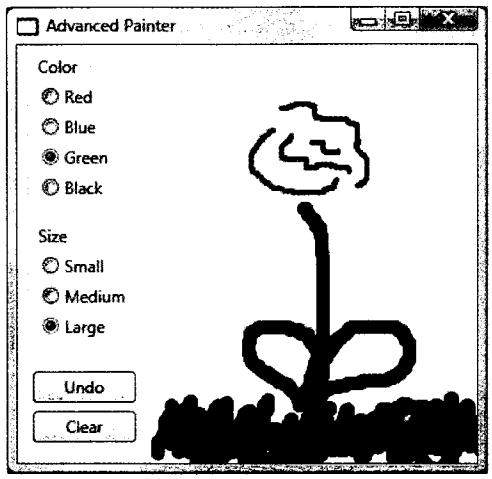
Thuộc tính của cọ vẽ được thiết lập mặc định (thiết lập ban đầu mà bạn thấy ngay khi ứng dụng chạy) với các RadioButton có nhãn **Black** và **Medium** được chọn. Bạn nên đưa ra thiết lập mặc định để gợi ý cho người dùng để họ có thể chọn thiết lập của riêng họ. Bây giờ bạn hãy thử chọn thiết lập cho mình.

4. ***Thay đổi màu sắc của cọ vẽ.*** Nhấn chuột vào các `RadioButton` được gắn nhãn **Red** để thay đổi màu sắc của cọ vẽ và **Small** để thay đổi kích thước của cọ vẽ. Trỏ chuột vào `Panel` trắng, nhấn và giữ phím chuột trái để vẽ bằng cọ vẽ. Vẽ các cánh hoa như trong Hình 1.3. Sau đó nhấn chuột vào `RadioButton` có nhãn là **Green** để thay đổi màu sắc của cọ vẽ một lần nữa.



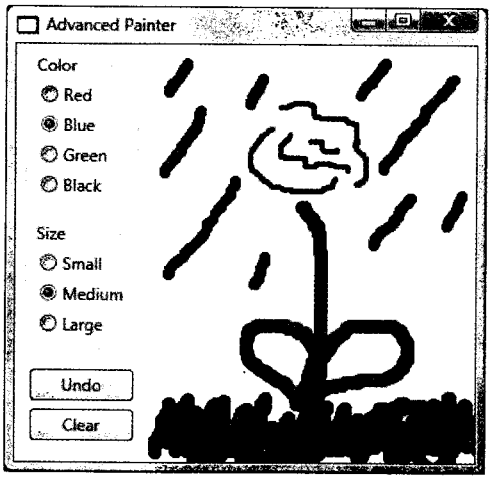
Hình 1.3 Thử vẽ bằng màu cọ vẽ mới.

5. ***Thay đổi kích thước của cọ vẽ.*** Nhấn chuột vào `RadioButton` được gắn nhãn **Large** để thay đổi kích thước của cọ vẽ. Vẽ cỏ và thân cây như Hình 1.4.



Hình 1.4 Thử vẽ với kích thước cọ vẽ mới.

6. ***Hoàn tất bức vẽ.*** Nhấn chuột vào `RadioButton` có nhãn là **Blue**. Sau đó nhấn chuột vào `RadioButton` có nhãn là **Medium**. Vẽ những hạt mưa như trong Hình 1.5 để hoàn tất bức vẽ.



Hình 1.5 Hoàn thành bức vẽ.

7. ***Đóng ứng dụng.*** Đóng ứng dụng đang chạy bằng cách nhấn chuột vào nút x ở trên cùng bên phải cửa sổ (Hình 1.5).

1.11 Tài nguyên web

Internet và web là các tài nguyên đặc biệt. Mục này bao gồm liên kết đến những website chứa nhiều thông tin hữu ích. Các mục tham khảo như thế này sẽ được đưa vào cuốn sách nếu cần thiết.

`www.deitel.com/visualbasic2008/`

Trung tâm tài nguyên Visual Basic của chúng tôi tập trung một lượng khổng lồ các tài liệu Visual Basic trực tuyến. Tìm kiếm các tài nguyên, phần mềm tải về, các hướng dẫn, tài liệu, sách, sách điện tử, bài báo, bài viết, blog, v.v... sẽ giúp bạn phát triển ứng dụng Visual Basic.

`www.deitel.com`

Ghé thăm site này để tải về các đoạn mã, bản cập nhật, bài đính chính và tài nguyên bổ sung cho ấn bản của Deitel & Associates, bao gồm đính chính, những câu hỏi thường gặp, các liên kết hữu ích và các đoạn mã có thể tải về của cuốn *Simply Visual Basic 2008.*

`www.prenhall.com/deitel`

Trang web của Deitel & Associates trên website của Prentice Hall chứa thông tin về các ấn bản và đoạn mã có thể tải về của cuốn sách này.

`msdn.microsoft.com/vbasic`

Đây là website về Visual Basic của Microsoft với các liên kết đến các đoạn mã mẫu, các hướng dẫn, blog, webcast và các tài nguyên hữu ích khác.

`www.softlord.com/comp`

Ghé thăm site này để tìm hiểu thêm về lịch sử máy tính.

`www.elsop.com/wrc/h_comput.htm`

Site này giới thiệu lịch sử ngành điện toán. Site chứa nội dung về các danh nhân trong lĩnh vực máy tính, sự phát triển của các ngôn ngữ lập trình và sự phát triển của các hệ điều hành.

`www.w3.org/History.html`

Bạn hãy ghé thăm site này để tìm hiểu lịch sử của web.

`www.netvalley.com/intval/07262/main.htm?sdf=1`

Site này giới thiệu lịch sử của Internet.

1.12 Tổng kết

Trong chương này, bạn đã được tìm hiểu về cách tổ chức của máy tính. Bạn đã được học về các loại ngôn ngữ lập trình và loại nào cần có chương trình dịch. Bạn đã được làm quen với một vài ngôn ngữ lập trình phổ biến nhất. Bạn đã tìm hiểu về tầm quan trọng của lập trình cấu trúc và lập trình hướng đối tượng. Bạn cũng được giới thiệu sơ qua về lịch sử của Internet và web cùng chiến lược .NET và một số tính năng đặc trưng của .NET.

Bạn đã chạy thử ứng dụng Visual Basic. Trong quá trình chạy thử, bạn biết rằng Visual Basic cung cấp rất nhiều điều khiển đã được xây dựng sẵn để thực hiện các chức năng hữu ích và sau khi làm quen với chức năng của những điều khiển này, bạn có thể phát triển ứng dụng nhanh hơn rất nhiều so với việc tự viết mã hoàn toàn. Bạn cũng được khuyến khích để khám phá một số website với nhiều thông tin về máy tính, Internet, web, .NET và Visual Basic bổ sung cho cuốn sách này.

Trong chương tiếp theo, bạn sẽ tìm hiểu về môi trường phát triển tích hợp (IDE - Integrated Development Environment) Visual Basic 2008. IDE sẽ giúp bạn tạo ứng dụng của riêng mình. Bạn sẽ được học bằng cách tiếp cận hướng ứng dụng, qua đó bạn sẽ hiểu về những tính năng của Visual Basic. Cách tiếp cận hướng ứng dụng được thể hiện trong việc bố trí nội dung trong mỗi chương như sau:

1. nghiên cứu yêu cầu của người dùng đối với ứng dụng;
2. chạy thử phiên bản hoàn chỉnh của ứng dụng;
3. nghiên cứu công nghệ cần thiết để tự xây dựng ứng dụng và
4. tự xây dựng ứng dụng của riêng bạn.

Trong quá trình đọc cuốn sách này, nếu bạn có bất kỳ câu hỏi nào về Visual Basic 2008, xin vui lòng gửi một email đến địa chỉ `deitel@deitel.com`, chúng tôi sẽ hồi đáp cho bạn ngay khi có thể. Chúng tôi thành thật hi vọng bạn sẽ thích thú việc nghiên cứu phiên bản mới nhất của ngôn ngữ Visual Basic của Microsoft - một trong những ngôn ngữ lập trình được sử dụng rộng rãi trên thế giới với cuốn sách *Simply Visual Basic 2008, ấn bản lần ba*. Chúc may mắn!

THUẬT NGỮ

.NET Framework - Phần mềm được xây dựng bởi Microsoft giúp chạy ứng dụng, cung cấp Framework Class Library và nhiều tính năng lập trình khác.

.NET Framework Class Library - Tập hợp các lớp và phương thức được xây dựng và đóng gói sẵn hỗ trợ việc tính toán toán học, thao tác với chuỗi, thao tác với ký tự, các hoạt động nhập/xuất, kiểm tra lỗi và các công việc khác.

Ada - Là ngôn ngữ lập trình được đặt theo tên quý bà Ada Lovelace, phát triển dưới sự tài trợ của Bộ Quốc phòng Mỹ vào những năm 1970 và đầu những năm 1980.

ARPAnet - Tiền thân của mạng Internet ngày nay.

ASP.NET - Phần mềm .NET giúp bạn xây dựng các ứng dụng web.

assembler - Chương trình dịch giúp chuyển chương trình được viết bằng ngôn ngữ assembly thành ngôn ngữ máy.

BASIC (Beginner's All-perpose Symbolic Instruction Code) - Là ngôn ngữ lập trình để viết chương trình đơn giản. Được phát triển vào giữa những năm 60 bởi giáo sư John Kemeny và Thomas Kurtz của trường Đại học Dartmouth. Mục đích chính của BASIC là dành cho những người mới học làm quen với các kỹ thuật lập trình.

băng thông (bandwidth) - Khả năng mang thông tin của đường truyền thông.

bộ nhớ (memory) - Là tên gọi khác cho khối nhớ máy tính.

bộ nhớ chính (primary memory) - Là tên gọi khác của khối nhớ.

bộ nhớ khả biến (volatile memory) - Bộ nhớ có thể bị xóa toàn bộ dữ liệu khi máy tính bị tắt.

bộ nhớ truy cập ngẫu nhiên (RAM) - Một trong những ví dụ của bộ nhớ chính.

bộ vi xử lý (microprocessor) - Chip khiến máy tính hoạt động (còn gọi là bộ não của máy tính).

C# - Là ngôn ngữ lập trình được thiết kế riêng cho nền tảng .NET. Ngôn ngữ này được phát triển dựa vào các ưu điểm của C, C++, Java. Tương tự như Visual Basic, C# là ngôn ngữ hướng đối tượng và có khả năng truy cập đến thư viện các thành phần được xây dựng sẵn của .NET giúp bạn có thể phát triển ứng dụng nhanh chóng.

chiến lược .NET (.NET Initiative) - Tầm nhìn của Microsoft về việc sử dụng Internet và web trong việc phát triển, kiến thiết, phân phối và sử dụng phần mềm.

chương trình dịch (translator program) - Chương trình chuyển các chương trình ngôn ngữ assembly sang ngôn ngữ máy.

chương trình hướng sự kiện (event-driven program) - Là chương trình đáp lại các sự kiện được tạo ra bởi người dùng như nhấn chuột hoặc nhấn phím.

chương trình máy tính (computer program) - Là một tập các chỉ lệnh hướng dẫn cho máy tính thông qua chuỗi hành động có thứ tự.

COBOL (COmmon Business Oriented Language) - Là ngôn ngữ lập trình được phát triển vào cuối những năm 1950 bởi một nhóm các nhà sản xuất máy tính hợp tác cùng với chính phủ và những người dùng máy tính. Ngôn ngữ này được sử dụng chủ yếu trong ứng dụng kinh doanh cần thao tác với một lượng lớn dữ liệu.

đặc tính (attribute) - Là tên gọi khác của thuộc tính (property) đối tượng.

điều khiển (control) - Là thành phần giao diện đồ họa có thể tái sử dụng như `GroupBox`, `RadioButton`, `Button` hay `Label`.

đối tượng (object) - Thành phần phần mềm mô hình hóa các đối tượng trong thế giới thực.

Fortran (Formula Translator) - Là ngôn ngữ lập trình được phát triển bởi IBM vào giữa những năm 50 (hiện vẫn được sử dụng rộng rãi) để tạo ứng dụng khoa học kỹ thuật yêu cầu tính toán toán học phức tạp.

Framework Class Library - Là tập các lớp và phương thức đã được đóng gói sẵn của .NET dùng để thực hiện tính toán toán học, xử lý chuỗi, xử lý ký tự, các hoạt động nhập/xuất, kiểm tra lỗi và nhiều hoạt động khác.

giao diện người dùng đồ họa (GUI)[2] - Là phần trực quan của ứng dụng giúp người dùng có thể tương tác với ứng dụng.

giao thức truyền tải siêu văn bản (HTTP) - Giao thức giúp file HTML có thể được truyền đi thông qua web.

IDE (Integrated Development Environment - môi trường phát triển tích hợp) - Công cụ phần mềm giúp bạn viết, chạy, kiểm thử và gỡ lỗi cho chương trình nhanh chóng và thuận tiện.

Internet - Là mạng máy tính toàn cầu. Hầu hết con người ngày nay truy cập vào Internet thông qua web.

Java - Là ngôn ngữ lập trình phổ biến được sử dụng để tạo trang web có nội dung động, để xây dựng các ứng dụng doanh nghiệp quy mô lớn, để cải tiến chức năng cho server web, để cung cấp ứng dụng cho các thiết bị điện tử tiêu dùng và nhiều mục đích khác.

khối lưu trữ thứ cấp (secondary storage unit) - Là bộ phận lưu trữ dung lượng lớn, lâu dài của máy tính.

2: Còn được gọi là giao diện đồ họa người dùng.

khối nhập (input unit) - Là thành phần nhận thông tin (dữ liệu và các chương trình máy tính) từ những thiết bị nhập khác nhau như bàn phím, chuột, microphone, máy quét và camera số.

khối nhớ (memory unit) - Vùng nhớ có dung lượng tương đối ít và có khả năng truy cập nhanh của máy tính, có nhiệm vụ lưu trữ dữ liệu tạm thời trong khi ứng dụng đang chạy.

khối số học và lôgic (ALU)[3] - Là bộ phận "sản xuất" của máy tính. ALU thực hiện công việc tính toán và ra quyết định.

khối xử lý trung tâm (CPU)[4] - Là thành phần trong phần cứng máy tính chịu trách nhiệm giám sát hoạt động của các thành phần khác.

khối xuất (output unit) - Thành phần của máy tính lấy thông tin đã xử lý và đưa đến các thiết bị xuất để thông tin có thể được sử dụng bên ngoài máy tính.

kỹ thuật đối tượng (object technology) - Là cách thức đóng gói để tạo ra các đơn vị phần mềm có ích. Mỗi đơn vị này tập trung vào một mục đích cụ thể của ứng dụng. Có các đối tượng ngày tháng, đối tượng thời gian, đối tượng tiền lương, đối tượng file và những đối tượng tương tự.

lập trình cấu trúc (structured programming) - Là cách tiếp cận có quy tắc để tạo chương trình sáng sủa, chính xác và dễ chỉnh sửa.

lập trình hướng đối tượng (OOP - Object oriented programming) - Mô hình hóa các đối tượng trong thế giới thực bằng các bản sao phần mềm.

lập trình trực quan với Visual Basic - Bạn sử dụng giao diện người dùng đồ họa (GUI) của Visual Studio để kéo và thả điều khiển lên màn hình sau đó gán nhãn và thay đổi kích thước cho chúng. Visual Studio viết sẵn khá nhiều mã, giúp bạn tiết kiệm công sức đáng kể.

lập trình viên máy tính (computer programmer) - Là người viết các chương trình máy tính.

lớp (class) - Là kiểu của nhóm đối tượng liên quan với nhau. Lớp mô tả định dạng chung cho các đối tượng thuộc lớp, thuộc tính và hành động sẵn có cho đối tượng phụ thuộc lớp này. Có thể tưởng tượng mối tương quan giữa đối tượng với lớp của đối tượng tương tự như ngôi nhà với bản thiết kế của ngôi nhà.

máy tính (computer) - Là một thiết bị có thể thực hiện tính toán và đưa ra các quyết định lôgic nhanh hơn hàng triệu, hàng tỉ thậm chí hàng nghìn tỉ lần so với con người.

nền tảng .NET (.NET Platform) - Tập hợp các thành phần phần mềm khiến chương trình .NET có thể chạy được, cho phép ứng dụng được phân phối tới nhiều thiết bị và máy tính khác. Cung cấp mô hình lập trình cho phép các thành phần phần mềm được viết bằng các ngôn ngữ khác nhau (như Visual Basic và C#) có thể giao tiếp với nhau.

ngôn ngữ assembly (hợp ngữ) - Là loại ngôn ngữ lập trình sử dụng các từ viết tắt tiếng Anh cho các phép tính cơ bản trên máy tính.

ngôn ngữ bậc cao (high-level language) - Là loại ngôn ngữ lập trình trong đó mỗi lệnh trong chương trình có thể thực hiện rất nhiều tác vụ. Ngôn ngữ bậc cao sử dụng các chỉ lệnh trông khá giống với tiếng Anh hàng ngày và chứa các ký hiệu toán học thông thường.

ngôn ngữ đánh dấu siêu văn bản (HTML) - Là ngôn ngữ dùng để đánh dấu thông tin để chia sẻ trên web thông qua những tài liệu văn bản được liên kết với nhau bằng siêu liên kết.

3: Còn gọi là đơn vị xử lý số học - lôgíc
4: Còn gọi là đơn vị xử lý trung tâm.v

ngôn ngữ lập trình thủ tục (procedural programming language) - Là ngôn ngữ lập trình (như Fortran, Pascal, BASIC và C) tập trung vào hành động (động từ) hơn là vật thể và đối tượng (danh từ).

ngôn ngữ máy (machine language) - Là ngôn ngữ tự nhiên của máy tính, về cơ bản bao gồm những chuỗi số hướng dẫn máy tính cách thực hiện hầu hết hoạt động của máy tính.

Pascal - Là ngôn ngữ lập trình được thiết kế cho mục đích dạy lập trình có cấu trúc, được đặt theo tên của nhà toán học và triết học nổi tiếng ở thế kỷ 17 - Blaise Pascal.

phần cứng (hardware) - Nhiều thiết bị khác nhau tạo nên máy tính bao gồm bàn phím, màn hình, chuột, ổ đĩa cứng, bộ nhớ, CD-ROM, DVD, máy in và bộ xử lý.

phần mềm (software) - Ứng dụng chạy trên máy tính.

phương thức (method) - Một phần của lớp thực hiện tác vụ và có thể trả về thông tin khi hoàn thành tác vụ đó.

tái sử dụng phần mềm (software reuse) - Là sử dụng lại các thành phần phần mềm có sẵn, một cách tiếp cận giúp bạn tránh khỏi việc "phát minh lại bánh xe", giúp bạn phát triển ứng dụng nhanh hơn.

TCP/IP (Transmission Control Protocol/Internet Protocol) - Tập hợp các giao thức truyền thông dành cho Internet.

thiết bị nhập (input device) - Thiết bị được dùng để tương tác với máy tính như bàn phím, chuột, microphone, máy quét và camera số.

thiết bị xuất (output device) - Thiết bị của máy tính sẽ tiếp nhận thông tin sau khi được máy tính xử lý.

thuộc tính (property) - Là đặc tính của đối tượng như kích thước, màu sắc, chiều dài.

trình biên dịch (compiler) - Là chương trình dịch chuyển chương trình được viết bằng ngôn ngữ bậc cao thành ngôn ngữ máy.

Visual Basic - Ngôn ngữ lập trình được giới thiệu bởi Microsoft giúp cho việc lập trình Windows trở nên dễ dàng hơn.

Visual Studio - Môi trường phát triển tích hợp (IDE) để phát triển ứng dụng sử dụng Visual Basic (và các ngôn ngữ khác).

World Wide Web (WWW) - Một hệ thống truyền thông cho phép người dùng máy tính có thể tìm và xem tài liệu đa phương tiện (như các tài liệu văn bản, đồ họa, hoạt hình, âm thanh và video).

World Wide Web Consortium (W3C) - Một diễn đàn mà thông qua đó cá nhân và các công ty được chấp thuận có thể phối hợp phát triển và chuẩn hóa các công nghệ cho web.

CÂU HỎI TRẮC NGHIỆM

1.1 Web được phát triển_____.

a) bởi ARPA
b) tại CERN bởi Tim Berners-Lee
c) trước Internet
d) để thay thế cho Internet

1.2 Chiến lược_____của Microsoft kết hợp Internet và web vào trong việc phát triển phần mềm.

a) .NET
b) BASIC
c) Windows
d) W3C

1.3 `TextBox`, `Button` và `RadioButton` là những ví dụ về_____.

a) nền tảng
b) ngôn ngữ bậc cao
c) IDE
d) điều khiển

1.4 _____là một ví dụ về bộ nhớ chính.

a) TCP b) RAM

c) ALU d) CD-ROM

1.5 Visual Basic là một ví dụ về một ngôn ngữ_____, trong đó một lệnh chương trình chứa rất nhiều tác vụ.

a) máy b) bậc trung

c) bậc cao d) assembly

1.6 Giao thức nào có mục đích chính là tạo một "mạng của mạng"?

a) TCP b) IP

c) OOP d) Các lựa chọn trên đều sai

1.7 Lợi ích chính của lập trình_____là phần mềm tạo ra bằng kỹ thuật này dễ hiểu hơn và và được tổ chức tốt hơn phần mềm được tạo ra bằng kỹ thuật trước đây.

a) hướng đối tượng b) tập trung

c) thủ tục d) HTML

1.8 Tập hợp các lớp và phương thức được đóng gói sẵn của .NET được gọi là _____.

a) NCL b) WCL

c) .NET Framework Class Library d) PPCM

1.9 Khả năng mang thông tin của đường truyền thông được gọi là_____.

a) nối mạng b) bộ nhớ thứ cấp

c) lưu lượng d) băng thông

1.10 Ngôn ngữ lập trình nào được tạo ra dành riêng cho .NET?

a) C# b) C++

c) BASIC d) Visual Basic

BÀI TẬP

1.11 Phân loại các mục sau đây vào hai nhóm phần cứng và phần mềm.

a) CPU b) Trình biên dịch

c) Khối nhập liệu d) Chương trình xử lý văn bản

e) Chương trình Visual Basic

1.12 Các chương trình dịch (ví dụ assembler và các trình biên dịch) chuyển các chương trình từ ngôn ngữ này (ngôn ngữ nguồn) sang một ngôn ngữ khác (ngôn ngữ đích). Những phát biểu sau đây là *đúng* hay *sai*?

a) Trình biên dịch dịch các chương trình được viết bằng ngôn ngữ bậc cao sang các chương trình được viết bằng ngôn ngữ đích.

b) Assembler dịch các chương trình được viết bằng ngôn ngữ nguồn thành các chương trình được viết bằng ngôn ngữ đích.

c) Một trình biên dịch dịch các chương trình được viết bằng ngôn ngữ nguồn thành các chương trình được viết bằng ngôn ngữ máy.

d) Ngôn ngữ bậc cao nói chung đều phụ thuộc vào phần cứng máy tính.

e) Chương trình được viết bằng ngôn ngữ máy thì phải dịch mới có thể chạy trên máy tính.

1.13 Các máy tính có thể được chia thành sáu khối.

a) Khối nào có thể được coi là "chỉ huy" của các khối khác?

b) Khối nào là "nhà kho" có dung lượng lưu trữ lớn lưu trữ thông tin ngay cả khi máy tính đã tắt?

c) Khối nào có thể xác định xem hai phần tử được lưu trong bộ nhớ có giống nhau hay không?

d) Khối nào nhận thông tin từ các thiết bị như bàn phím và chuột?

1.14 Viết dạng đầy đủ các từ viết tắt sau đây:

a) W3C
b) TCP/IP
c) OOP
d) HTML

1.15 Lợi ích của việc sử dụng kỹ thuật lập trình hướng đối tượng là gì?

2 CHƯƠNG

Mục tiêu

Trong chương này, bạn sẽ tìm hiểu để:

- Điều hướng từ **Start Page** của Visual Studio 2008.
- Tạo project Visual Basic.
- Sử dụng menu và toolbar của IDE.
- Thao tác với cửa sổ trong IDE Visual Basic 2008 Express Edition.
- Sử dụng tính năng tự động ẩn.
- Sử dụng các tính năng trợ giúp của IDE Visual Basic 2008 Express Edition.
- Đóng project Visual Basic.

Nội dung chính

Ứng dụng Welcome

Giới thiệu IDE Visual Basic 2008 Express Edition

Visual Studio 2008 là **môi trường phát triển tích hợp (Intergrate Development Environment - IDE)** của Microsoft dùng để tạo, chạy và gỡ lỗi ứng dụng được viết bằng các ngôn ngữ lập trình .NET khác nhau. IDE cho phép bạn tạo ứng dụng bằng cách kéo thả các thành phần có sẵn vào vị trí phù hợp. Đây còn gọi là **lập trình trực quan (visual programming)**, kỹ thuật đã giúp đơn giản hóa rất nhiều cho công việc phát triển ứng dụng. Trong chương này, bạn sẽ tìm hiểu các tính năng của IDE Visual Studio 2008 cần thiết để bắt đầu tạo ứng dụng Visual Basic.

2.1 Chạy thử ứng dụng Welcome

Trong mục này, bạn sẽ được học theo cách tiếp cận hướng ứng dụng để chuẩn bị xây dựng ứng dụng hiển thị thông điệp chào mừng và hiển thị hình ảnh. Ứng dụng cần phải đáp ứng những yêu cầu sau:

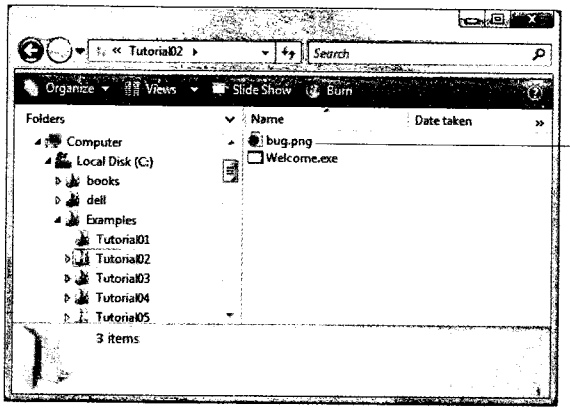
Yêu cầu đối với ứng dụng

Công ty phần mềm (Deitel & Associates) đã đề nghị bạn phát triển ứng dụng Visual Basic hiển thị thông điệp "Welcome to Visual Basic 2008" và hình ảnh biểu tượng của công ty.

Trong chương này, bạn sẽ làm quen với Visual Basic 2008 Express Edition và bắt đầu phát triển ứng dụng **Welcome**. Sau đó, ở Chương 3, bạn sẽ tổng hợp lại những gì đã học được và tạo ứng dụng **Welcome** bằng cách làm theo hướng dẫn từng bước. [*Lưu ý*: Quy ước của chúng tôi là sử dụng font **HP-Helve-Condense** cho tên ứng dụng]. Bạn bắt đầu bằng việc chạy thử ứng dụng đã hoàn thiện. Sau đó, bạn tìm hiểu những tính năng của Visual Basic cần thiết để xây dựng cho mình một phiên bản của ứng dụng này.

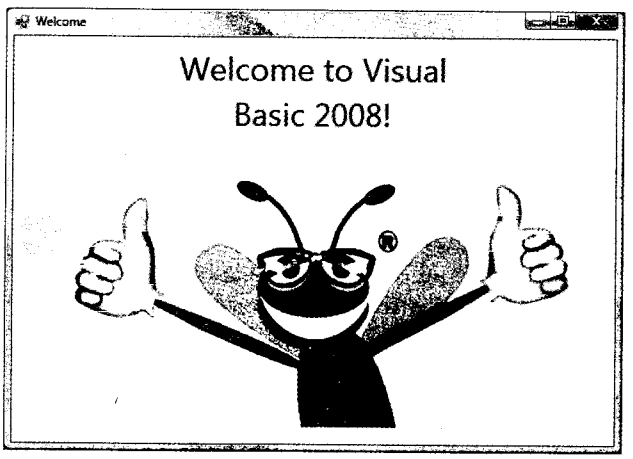
Chạy thử ứng dụng Welcome

1. ***Kiểm tra cài đặt máy tính***. Hãy đảm bảo rằng bạn đã cài đặt máy tính chính xác bằng cách đọc phần *Chuẩn bị* ở đầu cuốn sách này ngay sau *Lời nói đầu*.
2. ***Tìm thư mục ứng dụng***. Mở Windows Explorer và di chuyển đến thư mục `C:\Examples\Tutorial02` (Hình 2.1).



Hình 2.1 Nội dung của `C:\Examples\Tutorial02`.

3. ***Chạy ứng dụng Welcome.*** Nhấn đúp chuột vào file `Welcome.exe` (Hình 2.1) để chạy ứng dụng (Hình 2.2).



Hình 2.2 Ứng dụng **Welcome** đang chạy.

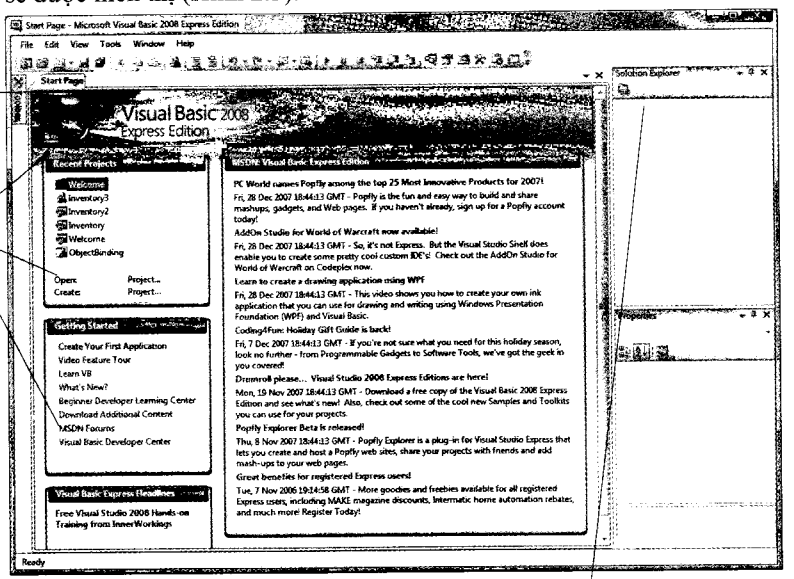
4. ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng,

2.2 Tổng quan về IDE Visual Basic 2008 Express Edition

Có rất nhiều phiên bản Visual Studio khác nhau. Những ví dụ trong cuốn sách này đều được xây dựng dựa trên *Microsoft Visual Basic 2008 Express Edition* - công cụ chỉ hỗ trợ ngôn ngữ lập trình Visual Basic. Bạn có thể mua phiên bản đầy đủ của Visual Studio 2008. Ngoài Visual Basic, công cụ này còn hỗ trợ các ngôn ngữ khác như Visual C#, Visual C++. Ảnh chụp màn hình và nội dung trình bày trong sách tập trung vào IDE của Visual Basic 2008 Express Edition.

Mục này sẽ giới thiệu về Visual Basic 2008 Express Edition IDE. Để khởi động IDE, chọn **Start > All Programs > Microsoft Visual Basic 2008 Express Edition**. Chúng tôi sử dụng ký tự > để minh họa việc chọn menu từ một menu khác. Ví dụ,

chúng tôi viết **File > Open File** để minh họa rằng bạn chọn lệnh **Open File** từ menu File. Sau khi Express Edition được khởi động xong, **Start Page (trang khởi động)** sẽ được hiển thị (Hình 2.3).



Hình 2.3 Start Page trong Visual Basic 2008 Express Edition.

Tùy thuộc vào phiên bản Visual Studio, trang **Start Page** có thể hiển thị hơi khác so với hình ảnh trên Hình 2.3. **Start Page** chứa danh sách những liên kết đến tài nguyên của IDE Visual Basic 2008 Express Edition và trên Internet, rất hữu ích cho lập trình viên chưa quen với Visual Basic. Ngoài ra, **Start Page** còn cung cấp các liên kết đến các thông tin mới nhất về Visual Basic (như các cập nhật và các bản vá lỗi) và thông tin về những chủ đề lập trình chuyên sâu, rất hữu ích cho lập trình viên có kinh nghiệm. Từ bây giờ trở đi, chúng tôi sẽ gọi IDE Visual Basic 2008 Express Edition đơn giản là "Visual Basic" hoặc "IDE". Sau khi khám phá IDE, bạn có thể trở về **Start Page** bằng cách chọn **View > Other Windows > Start Page**. [*Lưu ý*: Nếu bạn thay đổi cách bố trí cửa sổ trong IDE, bạn có thể thiết lập lại về mặc định bằng cách chọn **Window > Reset Window Layout**].

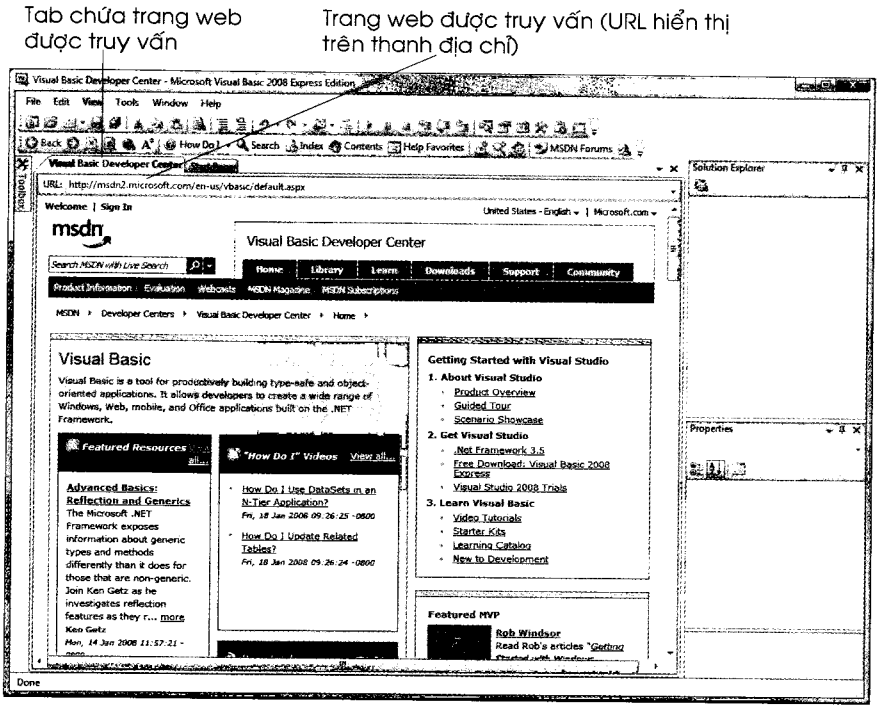
Liên kết trên Start Page

Liên kết trên **Start Page** được tổ chức thành các mục - **Recent Projects, Getting Started, Visual Basic Express Headlines** và **MSDN: Visual Basic Express Edition.** Hãy nhấn chuột vào bất cứ liên kết nào trên **Start Page** để hiển thị thông tin liên quan gắn với liên kết đó. Chúng tôi gọi việc nhấn chuột trái một lần là chọn hoặc nhấn chuột và nhấn chuột trái hai lần là nhấn đúp chuột.

Mục **Recent Projects** chứa thông tin về những project được tạo hoặc chỉnh sửa gần đây. Bạn cũng có thể mở những project có sẵn hoặc tạo project mới bằng cách nhấn chuột vào các liên kết trong mục này. Mục **Getting Started** tập trung vào việc sử dụng IDE để xây dựng chương trình, học Visual Basic, kết nối đến cộng đồng lập trình viên Visual Basic (ví dụ kết nối với lập trình viên thông qua các nhóm tin tức hay website) và cung cấp nhiều công cụ phát triển khác.

Nếu bạn có kết nối Internet, **Visual Basic Express Headlines** và **MSDN: Visual Basic Express Edition** chứa liên kết đến tài nguyên về lập trình bao gồm những khóa học, tin tức mới nhất về Visual Basic. Để truy cập kho thông tin lớn hơn về Visual Studio, bạn có thể truy cập đến thư viện trực tuyến MSDN (Microsoft Developer Network) tại địa chỉ `msdn2.microsoft.com/library`. Site MSDN chứa nhiều bài viết, dữ liệu để tải về và các bài hướng dẫn về những công nghệ liên quan đến Visual Basic. Bạn có thể truy cập trang web từ IDE bằng Internet

Explorer (**trình duyệt web tích hợp** bên trong IDE). Để truy cập trang web, nhập URL của trang web vào **thanh địa chỉ** (Hình 2.4) và nhấn phím *Enter* - tất nhiên máy tính của bạn phải có kết nối Internet. (Nếu thanh địa chỉ không được hiển thị, chọn **View > Other Windows > Web Browser**). Trang web bạn muốn xem sẽ được hiển thị trên một tab khác trong Visual Basic IDE (Hình 2.4). Chúng tôi sẽ trình bày về những cửa sổ khác hiển thị trong IDE ngoài **Start Page** và trình duyệt web tích hợp ở phần sau của chương này.



Hình 2.4 Mở một trang web bằng IDE Visual Basic 2008 Express Edition.

TỰ ÔN TẬP

1. Khi mở Visual Basic 2008 Express Edition lần đầu tiên,_____sẽ được hiển thị.
   a) **What's New Page** b) **Start Page**
   c) **Welcome Page** d) Các lựa chọn trên đều sai
2. Mục_____trong **Start Page** chứa danh sách các project được mở hoặc được tạo trong Visual Basic.
   a) **MSDN** b) **Gettting Started**
   c) **Recent Projects** d) **Visual Basic Express Headlines**

Đáp án: **1)** b. **2)** c.

2.3 Tạo project cho ứng dụng Welcome

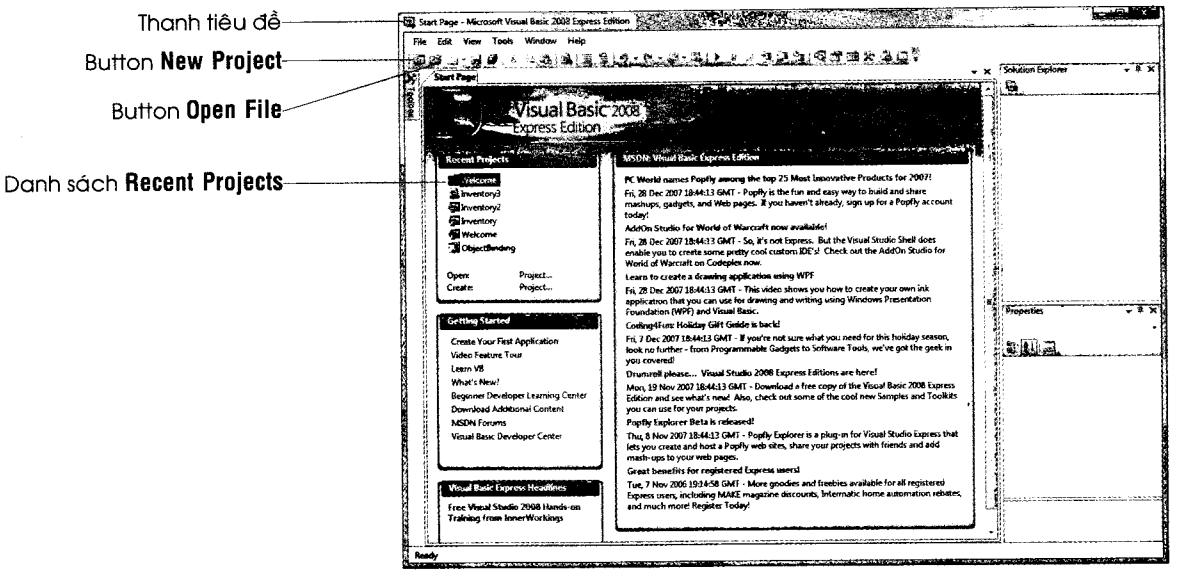
Trong mục này, bạn tạo **ứng dụng Windows Forms (Windows Forms application)** đơn giản bằng Visual Basic. Visual Basic tổ chức ứng dụng theo kiểu **project** và **solution**. Project là một nhóm các file liên quan với nhau, chẳng hạn file mã Visual Basic và bất kỳ hình ảnh nào tạo nên chương trình. Solution chứa một hoặc nhiều project. Trong cuốn sách này, mọi ứng dụng đều chứa một solution. Ứng dụng lớn có thể chứa nhiều project, trong đó mỗi project thực hiện một nhiệm vụ cụ thể. Trong cuốn sách này, mỗi ứng dụng bạn tạo ra sẽ chỉ gồm một project.

Tạo project cho ứng dụng Welcome

1. ***Tạo project mới***. Khởi động Visual Basic. Có một số cách để tạo project mới hoặc mở project có sẵn:

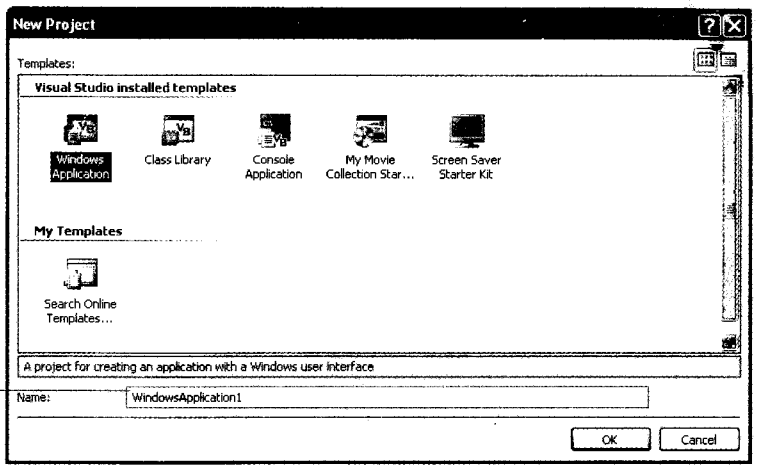
- Chọn **File > New Project...** để tạo project mới hoặc chọn **File > Open Project...** để mở project có sẵn.
- Trên **Start Page**, dưới mục **Recent Projects**, nhấn chuột vào liên kết **Create: Project** hoặc **Open: Project...**
- Nhấn chuột vào `Button` **New Project** (Hình 2.5) để hiển thị hộp thoại **New Project** (Hình 2.6) hoặc nhấn chuột vào `Button` **Open File** (Hình 2.5) để hiển thị hộp thoại **Open File**.

Hộp thoại (dialog) là cửa sổ có thể hiển thị thông tin và thu thập thông tin từ người dùng. Giống như cửa sổ, hộp thoại có dòng văn bản hiển thị trên **thanh tiêu đề (title bar).**



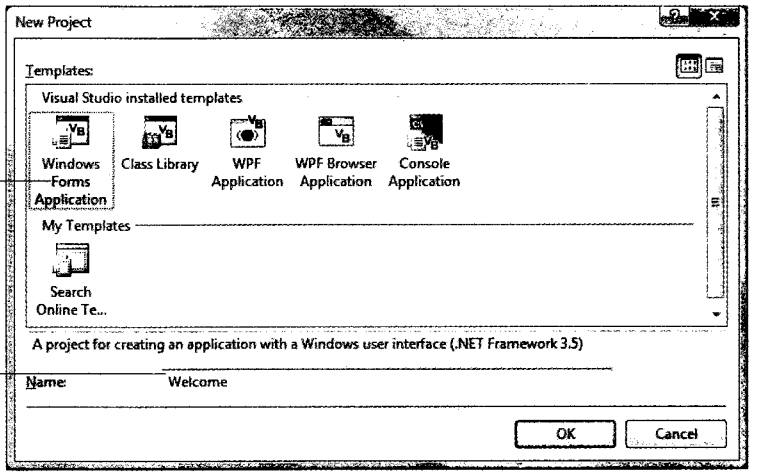
Hình 2.5 Button **New Project** và danh sách **Recent Projects**.

2. ***Chọn kiểu project***. Visual Basic cung cấp **template (khuôn mẫu)** cho nhiều kiểu project khác nhau (Hình 2.6). **Template** là các kiểu project có thể tạo trong Visual Basic - bao gồm ứng dụng Windows Forms, ứng dụng console và ứng dụng khác (trong phạm vi cuốn sách này bạn sẽ sử dụng ứng dụng Windows Forms). Bạn cũng có thể tạo các template cho ứng dụng của riêng bạn. [*Lưu ý*: Tùy thuộc vào phiên bản của Visual Studio, tên và số lượng các mục hiển thị trong khung **Templates:** có thể khác so với hình minh họa].
3. ***Chọn template***. Chọn **Windows Forms Application** (Hình 2.7), là ứng dụng chạy trên hệ điều hành Windows (ví dụ Windows XP hay Windows Vista) và có **giao diện người dùng đồ họa (graphical user interface - GUI)** - phần có thể nhìn thấy được của ứng dụng mà người dùng có thể tương tác. Ứng dụng Windows bao gồm các sản phẩm phần mềm do Microsoft phát triển như Microsoft Word, Internet Explorer và Visual Studio hoặc sản phẩm phần mềm được tạo bởi các hãng phần mềm khác và các phần mềm do bạn và những lập trình viên khác tạo ra.



Tên project mặc định (được cung cấp bởi Visual Basic)

Hình 2.6 Hộp thoại **New Project**.



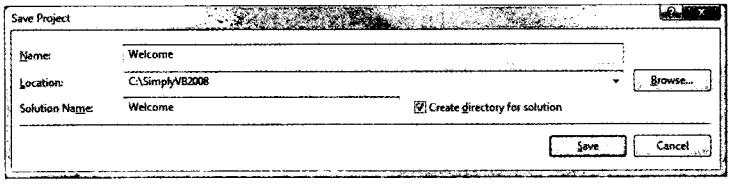
Windows Forms Application Visual Basic (đang được chọn)

Thay đổi tên project

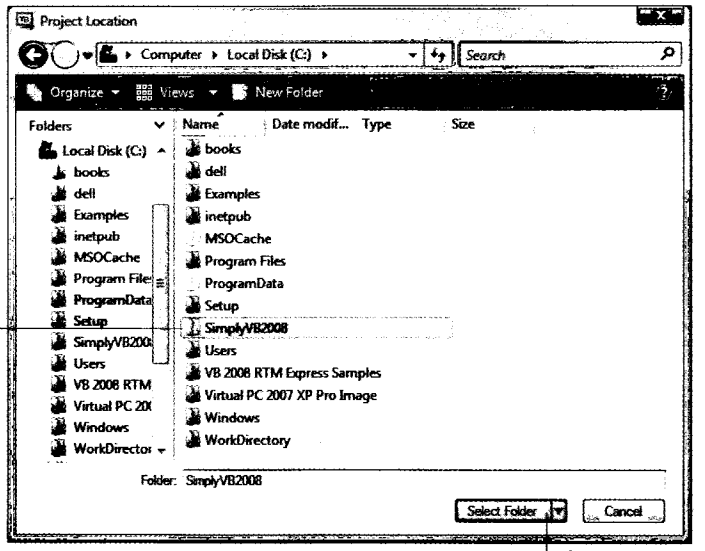
Hình 2.7 Hộp thoại **New Project** với thông tin về project đã được thay đổi.

4. ***Thay đổi tên project.*** Theo mặc định, Visual Basic đặt tên cho project là **WindowsApplication1** (Hình 2.6) và đặt các file của project vào trong thư mục có tên là `WindowsApplication1`. Để đổi tên của project, nhập `Welcome` vào trong trường **Name:** (Hình 2.7), sau đó nhấn **OK**. Thay đổi tên project thành `Welcome` và tên thư mục cũng sẽ tự động được đổi thành `Welcome`.

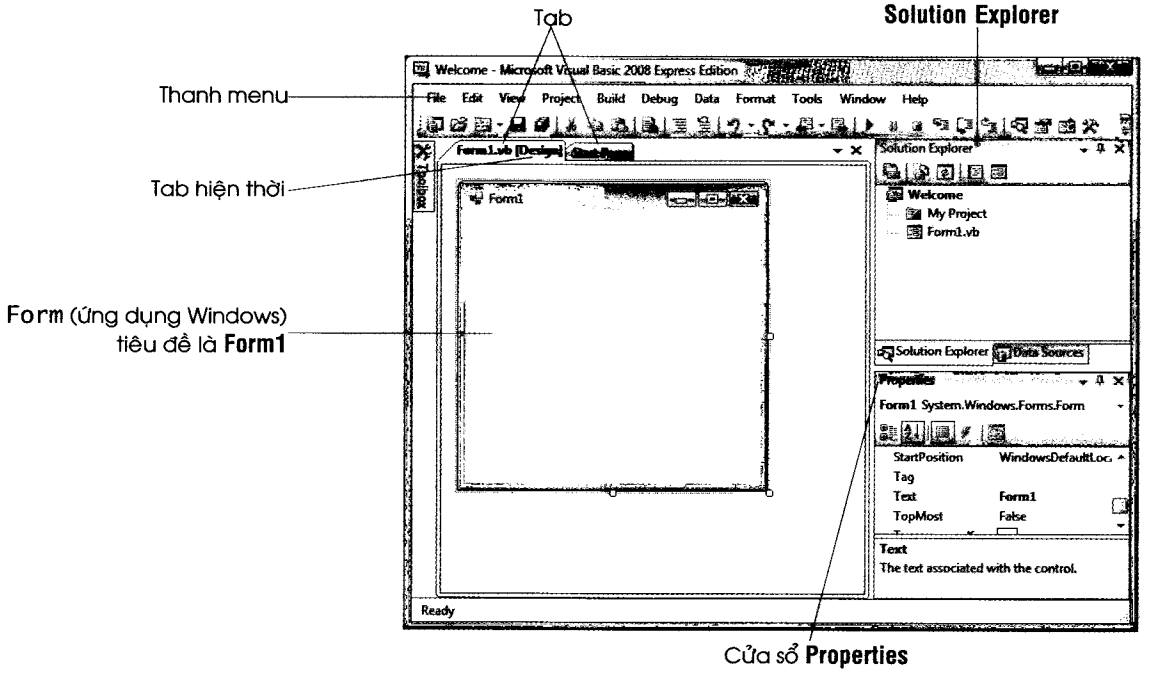
5. ***Thay đổi vị trí lưu project.*** Lưu project vào thư mục `C:\SimplyVB2008`. Để thay đổi vị trí của project, chọn **File > Save All**. Hộp thoại **Save Project** hiện ra (Hình 2.8). Trong hộp thoại này, sử dụng `Button` **Browse...** để tìm thư mục `SimplyVB2008` và nhấn chuột vào **Select Folder** (Hình 2.9). [*Lưu ý*: Nếu bạn sử dụng Windows XP hãy nhấn chuột vào `Button` **Open**]. Sau khi nhập tên và vị trí lưu project trong hộp thoại **Save Project**, nhấn **Save**. IDE lúc này sẽ được hiển thị ở chế độ **Design** (Hình 2.10) với các chức năng cần thiết để tạo ứng dụng Windows. Chú ý rằng màn hình của bạn có thể trông hơi khác một chút - một số cửa sổ như **Solution Explorer** có thể chưa được hiển thị ngay. Chúng tôi sẽ trình bày cách mở những cửa sổ này ở phần sau.



Hình 2.8 Hộp thoại **Save Project**.



Hình 2.9 Hộp thoại **Project Location.**



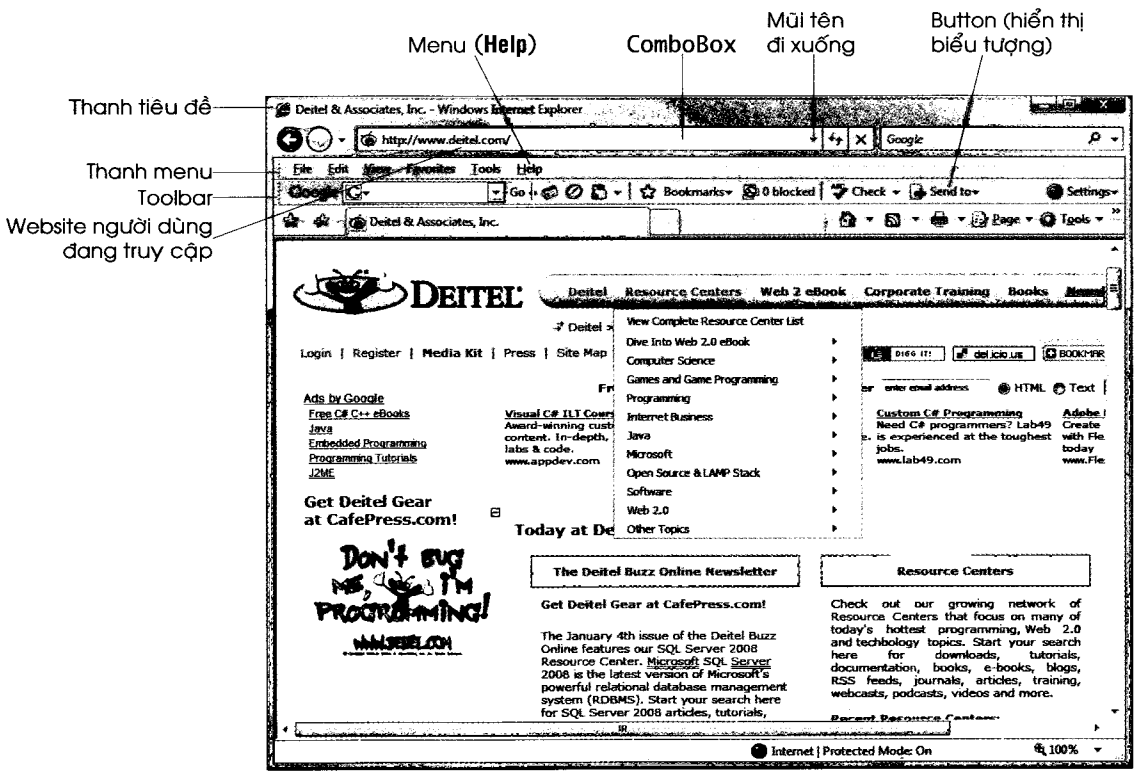
Hình 2.10 Chế độ **Design** của IDE.

Tên của những file đang mở được liệt kê trên **tab (Form1.vb [Design]** và **Start Page** trên Hình 2.10). Để hiển thị nội dung của file, nhấn chuột vào tab chứa file đó. Tab là cách dễ dàng để truy cập nhiều file. **Tab hiện thời (active tab)** được in đậm (**Form1.vb [Design]** trên Hình 2.10).

Nội dung của tab **Form1.vb [Design]** bao gồm một hình chữ nhật màu ghi (gọi là **Form**) là **Windows Form Designer (phần thiết kế giao diện cho Form)**. `Form` (tiêu đề là **Form1**) là cửa sổ chính của ứng dụng Windows Forms mà bạn đang xây dựng. `Form` có thể được thiết kế bằng cách thêm điều khiển (thành phần có thể tái sử dụng) như các `Button`. Nói chung, `Form` cùng với điều khiển tạo ra giao diện đồ họa cho ứng dụng. Người dùng nhập dữ liệu **đầu vào (input)** cho ứng dụng bằng cách nhấn vào bàn phím, nhấn chuột và rất nhiều cách khác. Ứng dụng hiển thị hướng dẫn và kết xuất các thông tin **đầu ra (output)** trên giao diện người dùng

đồ họa. Ví dụ, hộp thoại **New Project** trên Hình 2.6 là một giao diện người dùng đồ họa trong đó người dùng nhấn chuột để chọn các kiểu project và nhập tên project từ bàn phím.

Điều khiển giao diện người dùng đồ họa (ví dụ button) vừa hỗ trợ người dùng nhập liệu, vừa định dạng và hiển thị dữ liệu kết quả. Ví dụ, Internet Explorer (Hình 2.11) hiển thị trang web do người dùng yêu cầu. Giao diện người dùng đồ họa của Internet Explorer có một thanh menu bao gồm sáu menu: **File, Edit, View, Favorites, Tools** và **Help**. Các menu này cho phép người dùng in, lưu file và nhiều tác vụ khác. Bên dưới thanh menu là **toolbar (thanh công cụ)** chứa các button. Mỗi button chứa một hình ảnh - còn gọi là **biểu tượng (icon)** - để phân biệt button đó với những button khác. Khi nhấn chuột, button trên toolbar sẽ thực thi tác vụ (chẳng hạn in, tìm kiếm). Bên trên thanh menu là `ComboBox` (hộp combo) để người dùng nhập địa chỉ của website cần truy cập. Người dùng có thể nhấn chuột vào mũi tên đi xuống của `ComboBox` để chọn website đã truy cập trước đó. Trên cùng của cửa sổ là tiêu đề của trang web mà người dùng đang truy cập. Menu, button và `Label` là một phần giao diện người dùng đồ họa Internet Explorer, cho phép người dùng có thể tương tác với ứng dụng Internet Explorer. Sử dụng Visual Basic, bạn có thể tạo các ứng dụng của riêng bạn với tất cả điều khiển như trên Hình 2.11 và nhiều điều khiển khác nữa.



Hình 2.11 Cửa sổ Internet Explorer với các điều khiển giao diện người dùng đồ họa đã được gắn nhãn (Nội dung website của Deitel & Associates).

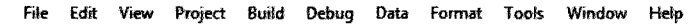
TỰ ÔN TẬP

1. Phần của ứng dụng mà người dùng có thể tương tác được gọi là_____của ứng dụng.
   a) giao diện người dùng đồ họa b) project
   c) solution d) thanh tiêu đề
2. _____chứa một hoặc nhiều project hình thành nên ứng dụng.
   a) Hộp thoại b) `Form`
   c) Solution d) Giao diện người dùng đồ họa

Đáp án: **1)** a. **2)** c.

2.4 Thanh menu và Toolbar

Lập trình viên Visual Basic sử dụng menu từ **thanh menu (menu bar)** trên Hình 2.12. Những menu này chứa lệnh để quản lý IDE và để phát triển và thực thi ứng dụng. Menu hiển thị tùy thuộc vào những gì bạn đang làm trên IDE.



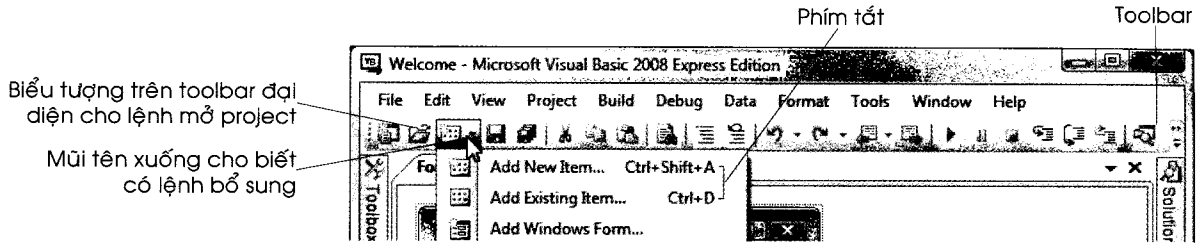
Hình 2.12 Thanh menu của Visual Basic 2008 IDE.

Mỗi menu có một nhóm các **lệnh (command)** liên quan - còn gọi là các **menu item (mục menu)** - mà khi được chọn thì IDE sẽ thực hiện một tác vụ nào đó, chẳng hạn như mở cửa sổ, lưu file, in file và thực thi ứng dụng. Ví dụ, để hiển thị cửa sổ **Toolbox**, bạn chọn **View > Toolbox**. Các menu trên Hình 2.12 được tóm tắt trên Hình 2.13, trong cuốn sách này bạn sẽ cần phải sử dụng nhiều trong số các menu này. [*Lưu ý*: Các menu được hiển thị tùy thuộc vào bạn đang làm gì - ví dụ, một số menu chỉ hiển thị nếu project được mở]. Ở Chương 20, ứng dụng **Typing** (giới thiệu về các sự kiện bàn phím, menu và hộp thoại), bạn sẽ học cách tự tạo và bổ sung các menu và menu item vào trong ứng dụng.

Menu	Mô tả
File	Chứa lệnh để mở, đóng, thêm và lưu project cũng như in dữ liệu của project và thoát khỏi Visual Studio.
Edit	Chứa lệnh chỉnh sửa như **Cut**, **Paste** và **Undo**.
View	Chứa lệnh để hiển thị các cửa sổ của IDE (như cửa sổ **Solution Explorer**, **Toolbox**, **Properties**) và các toolbar.
Project	Chứa lệnh để quản lý project và các file của project.
Build	Chứa lệnh để dịch ứng dụng Visual Basic.
Debug	Chứa lệnh để gỡ lỗi (ví dụ xác định và sửa lỗi trong ứng dụng) và chạy ứng dụng.
Data	Chứa lệnh để tương tác với **cơ sở dữ liệu (database)** - nơi chứa dữ liệu cho ứng dụng xử lý.
Format	Chứa lệnh để căn lề và tùy chỉnh điều khiển trên `Form`. Menu này chỉ hiển thị khi một thành phần của giao diện người dùng đồ họa được chọn trong chế độ **Design**.
Tools	Chứa lệnh để truy cập những công cụ bổ sung của IDE và các tùy chọn hỗ trợ tùy chỉnh IDE.
Window	Chứa lệnh để ẩn, mở, đóng và hiển thị cửa sổ IDE.
Help	Chứa lệnh để truy cập vào các tính năng trợ giúp của IDE.

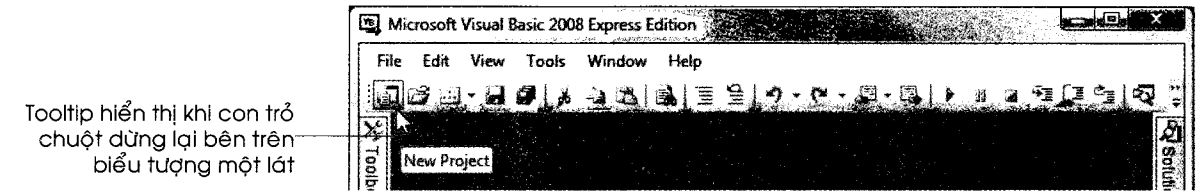
Hình 2.13 Tóm tắt các menu của Visual Basic IDE.

Ngoài việc di chuyển qua các menu trên thanh menu, bạn có thể truy cập rất nhiều lệnh thông thường từ toolbar của IDE (Hình 2.14), toolbar này chứa nhiều biểu tượng đồ họa đại diện cho các lệnh. Để thực hiện lệnh thông qua toolbar, bạn chỉ cần nhấn chuột vào biểu tượng của lệnh. Một số biểu tượng có mũi tên đi xuống bên cạnh, khi nhấn chuột vào mũi tên này sẽ hiển thị các lệnh bổ sung.



Hình 2.14 Toolbar IDE.

Thật khó nhớ mỗi biểu tượng trên toolbar đại diện cho lệnh gì. Hãy trỏ chuột vào biểu tượng và dừng lại một lát để hiển thị mô tả cho biểu tượng gọi là **tooltip** (Hình 2.15). Tooltip giúp bạn làm quen với những tính năng của IDE.



Hình 2.15 Hiển thị tooltip.

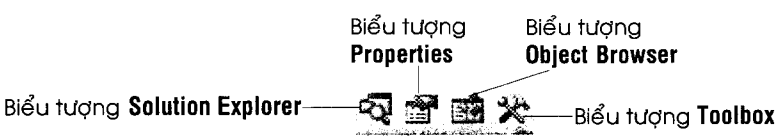
TỰ ÔN TẬP

1. _____chứa nhóm lệnh liên quan.
   a) Menu item b) Menu
   c) Tooltip d) Các lựa chọn trên đều sai
2. Khi trỏ chuột lên biểu tượng trên toolbar của IDE một vài giây,_____được hiển thị.
   a) toolbox b) toolbar
   c) menu d) tooltip

Đáp án: 1) b. 2) d.

2.5 Cửa sổ IDE Visual Basic 2008 Express Edition

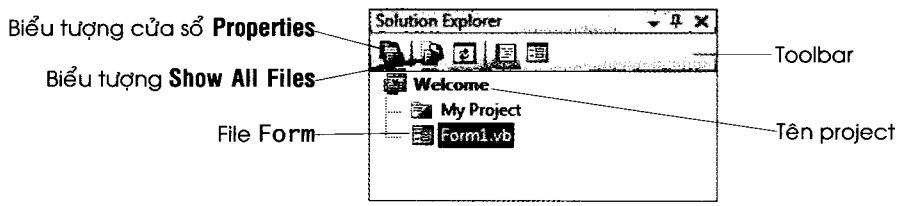
IDE cung cấp cửa sổ để truy cập vào những file liên quan của project để chỉnh sửa form và điều khiển trên form bằng cách thay đổi thuộc tính của chúng (tên, màu sắc...). Những cửa sổ này cung cấp phương tiện trực quan cho các tác vụ lập trình thông thường như quản lý các file trong project. Trong phần này, bạn sẽ làm quen với một số cửa sổ - **Solution Explorer, Properties, Toolbox** - là những cửa sổ thiết yếu để xây dựng ứng dụng Visual Basic. Bạn có thể truy cập những cửa sổ này bằng cách sử dụng các biểu tượng trên toolbar của IDE (Hình 2.16) hoặc bằng cách chọn tên cửa sổ trên menu **View**. [*Lưu ý*: Những biểu tượng này có thể không được hiển thị nếu cửa sổ IDE quá nhỏ. Nếu bạn không nhìn thấy các biểu tượng được hiển thị như trên Hình 2.16, hãy mở rộng cửa sổ IDE].



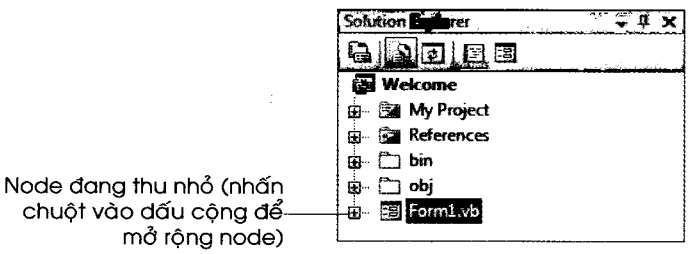
Hình 2.16 Những biểu tượng trên toolbar của bốn cửa sổ Visual Basic IDE.

Solution Explorer

Cửa sổ **Solution Explorer** (mặc định nằm ở bên phải của IDE như trên Hình 2.10) cung cấp khả năng truy cập đến các file của solution. Cửa sổ này cho phép bạn quản lý file một cách trực quan. Cửa sổ **Solution Explorer** hiển thị danh sách file trong một project và danh sách project trong một solution. (Trong cuốn sách này bạn chỉ tạo một project trong mỗi ứng dụng, tuy nhiên hãy ghi nhớ rằng ứng dụng hoàn toàn có thể chứa một hoặc nhiều project). Nếu cửa sổ **Solution Explorer** không được hiển thị trên IDE, bạn có thể hiển thị cửa sổ này bằng cách nhấn chuột vào biểu tượng **Solution Explorer** trên toolbar (Hình 2.16) hoặc bằng cách chọn **View > Solution Explorer**. Khi IDE vừa mới được mở, cửa sổ **Solution Explorer** chưa có file nào được hiển thị. Khi một project được mở, cửa sổ **Solution Explorer** sẽ hiển thị nội dung. Hình 2.17 hiển thị nội dung của ứng dụng **Welcome**. Theo mặc định, IDE hiển thị file mà bạn có thể cần phải chỉnh sửa. Những file khác do IDE tạo ra sẽ bị ẩn đi. Nhấn chuột vào biểu tượng **Show All Files** để hiển thị tất cả các file trong solution, bao gồm cả những file do IDE tạo ra (Hình 2.18).



Hình 2.17 Solution Exporer khi project đang được mở.



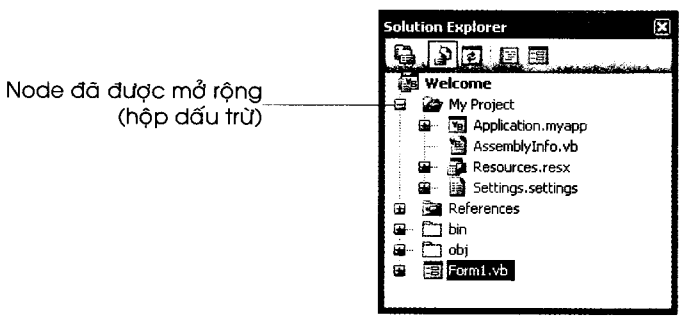
Hình 2.18 Sử dụng biểu tượng **Show All Files** để hiển thị tất cả các file trong solution.

Welcome là project duy nhất trong solution. File chứa Form trên Hình 2.10 có tên là Form1.vb. (Các file Form trong Visual Basic có phần mở rộng là .vb, viết tắt của "Visual Basic").

Hộp dấu cộng (plus box) và **hộp dấu trừ (minus box)** bên trái **My Project**, **References**, **bin**, **obj** và **Form1.vb** được gọi là **node**. Hộp dấu cộng và hộp dấu trừ dùng để mở rộng và thu nhỏ thông tin của node.

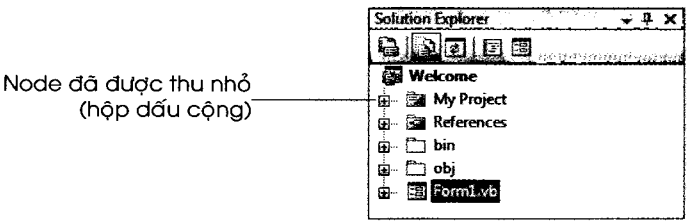
Di chuyển trong project với Solution Explorer

1. *Mở rộng node*. Sau khi nhấn chuột vào biểu tượng **Show All Files** (Hình 2.18), nhấn chuột vào hộp dấu cộng ở bên trái thư mục **My Project** để mở rộng node. Cửa sổ **Solution Explorer** sẽ hiển thị như trên Hình 2.19.



Hình 2.19 Node đã được mở rộng.

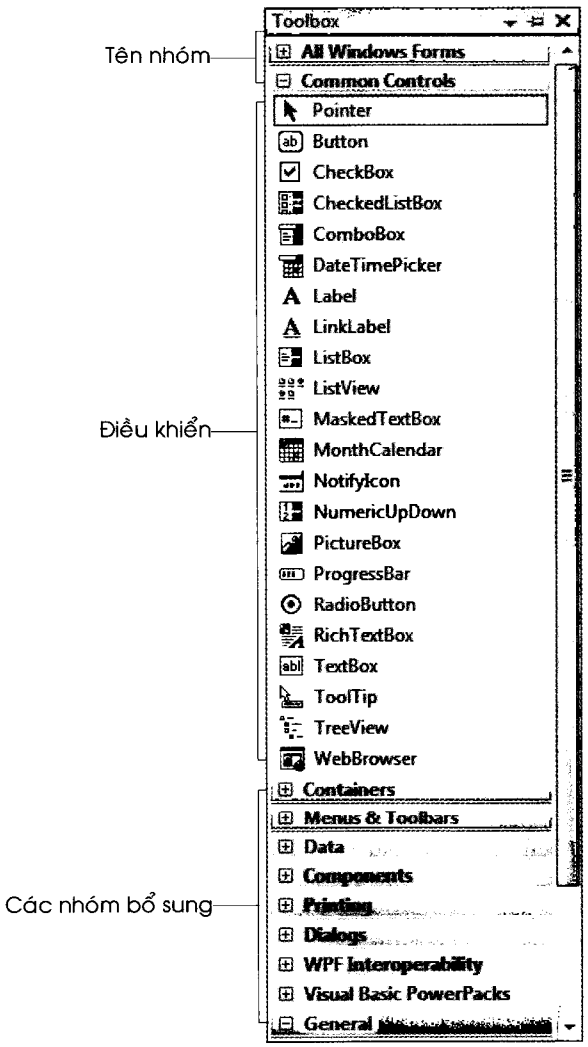
2. ***Thu nhỏ node***. Nhấn chuột vào hộp dấu trừ ở bên trái thư mục **My Project** (Hình 2.19) để thu nhỏ node. Hộp dấu trừ lúc này trở thành dấu cộng như trên Hình 2.20.



Hình 2.20 Node đã được thu nhỏ.

Toolbox

Sử dụng kỹ thuật lập trình trực quan, bạn có thể "kéo thả" điều khiển lên `Form` nhanh chóng và dễ dàng thay vì xây dựng chúng từ đầu - một công việc phức tạp và tốn thời gian. Tương tự như để lái xe, bạn không cần biết cách chế tạo động cơ, do đó bạn cũng không cần biết cách xây dựng điều khiển để có thể tạo ra giao diện người dùng đồ họa. **Toolbox (hộp công cụ)** (Hình 2.21) chứa rất nhiều điều khiển để giúp bạn làm việc đó. Bạn sẽ sử dụng **Toolbox** khi bạn tạo ứng dụng **Welcome** trong Chương 3. Nếu **Toolbox** chưa được hiển thị, hãy chọn **View > Toolbox**.



Hình 2.21 **Toolbox** hiển thị nội dung của tab **Common Controls**.

Toolbox sắp xếp điều khiển vào các nhóm - **All Windows Forms, Common Controls, Containers, Menu & Toolbars, Data, Components, Printing, Dialog, WPF Interoperability, Visual Basic PowerPacks** và **General**. Khi nhấn chuột vào tên nhóm, **Toolbox** sẽ hiển thị tất cả các điều khiển trong nhóm đó. Bạn có thể di chuyển qua các điều khiển bằng **mũi tên thanh cuộn (scroll arrow)** nếu có ở phía bên phải của **Toolbox**. Trong những chương còn lại, bạn sẽ sử dụng rất nhiều điều khiển trong **Toolbox**.

Cửa sổ Properties

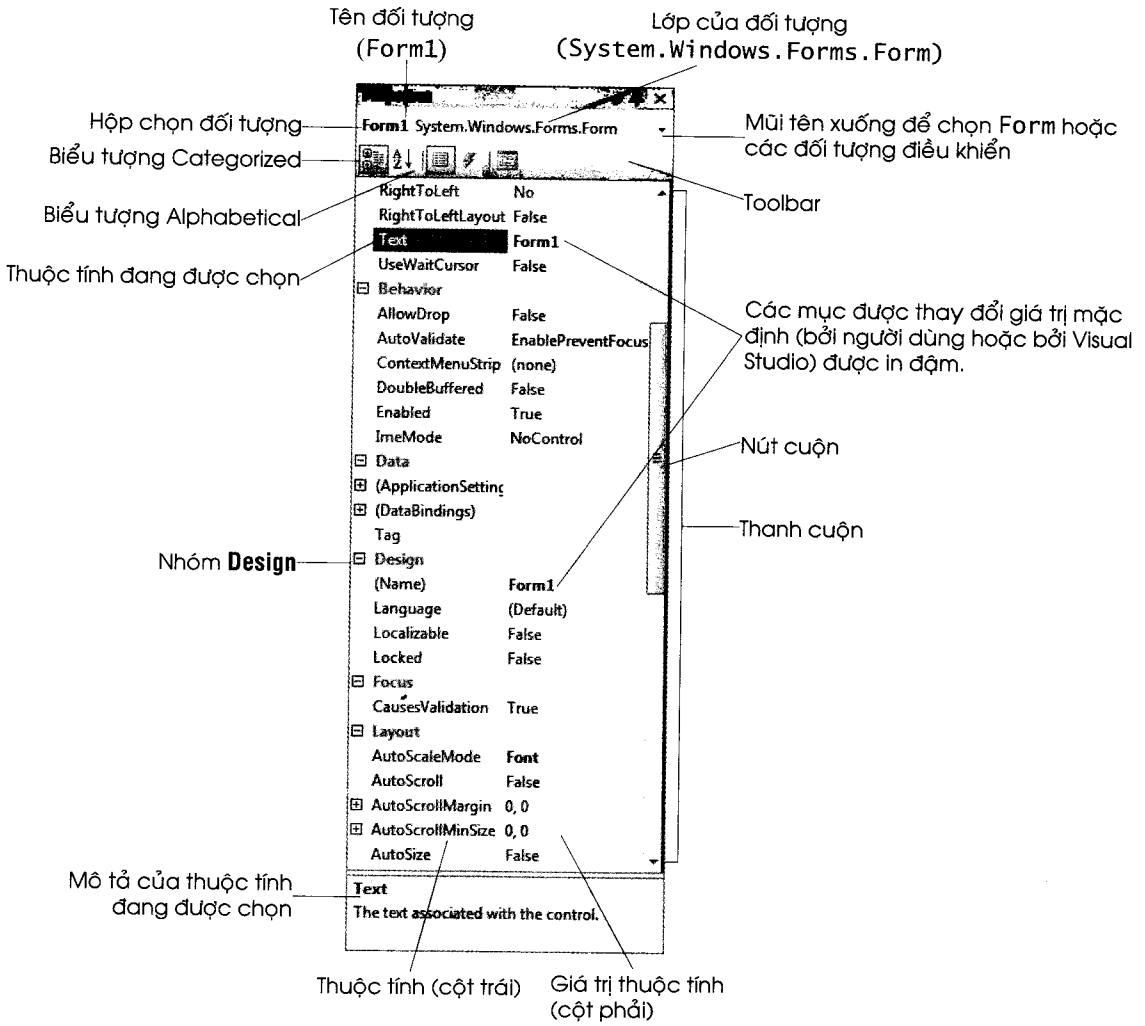
Cửa sổ **Properties** hiển thị **thuộc tính (properties)** của `Form` và đối tượng điều khiển. Thuộc tính xác định các đặc tính của đối tượng như kích thước, màu sắc, vị trí.

Cửa sổ **Properties** cho phép bạn thiết lập thuộc tính đối tượng một cách trực quan mà không phải viết mã. Cửa sổ này cung cấp một số lợi ích sau đây:

- Bạn có thể xem mô tả ngắn gọn về thuộc tính được chọn giúp bạn hiểu ý nghĩa của thuộc tính.
- Bạn có thể thiết lập thuộc tính nhanh chóng.
- Bạn có thể xem thuộc tính nào có thể tùy chỉnh, đôi khi bạn còn được biết giá trị nào được chấp nhận cho thuộc tính nào đó. Giá trị thuộc tính mà bạn chỉnh sửa sẽ được in đậm.
- Bạn sẽ không phải nhớ hay tìm kiếm tài liệu Visual Basic (xem Mục 2.7) để biết được những thiết lập cho thuộc tính.

Những tính năng mang lại các lợi ích trên đã được thiết kế để đảm bảo các thiết lập đều chính xác và nhất quán trong suốt project. Nếu cửa sổ **Properties** không hiển thị, chọn **View > Properties Window** (hoặc nhấn *F4*). Hình 2.22 hiển thị cửa sổ **Properties** của `Form`.

- Mỗi `Form` hoặc đối tượng điều khiển đều có một tập các thuộc tính của riêng nó. Ở trên cùng cửa sổ **Properties** là **hộp chọn đối tượng (component object box)**, cho phép bạn chọn đối tượng cần xem thuộc tính trong cửa sổ **Properties**. Bạn cũng có thể chọn đối tượng đó trong Form Designer.
- Bạn có thể chắc chắn mình đang thao tác với đúng thuộc tính của đối tượng bởi tên và kiểu lớp của đối tượng được hiển thị trong hộp chọn đối tượng. Đối tượng `Form` thuộc lớp `System.Windows.Forms.Form` và được gán tên (chẳng hạn `Form1`). Bạn sẽ tìm hiểu về kiểu lớp của điều khiển trong chương sau. Các biểu tượng trên toolbar được sắp xếp hoặc theo bảng chữ cái (nếu bạn nhấn chuột vào **biểu tượng Alphabetical**) hoặc theo nhóm (nếu bạn chọn **biểu tượng Categorized**). Hình 2.22 hiển thị cửa sổ **Properties** với các thuộc tính của nó được sắp xếp theo nhóm. Mỗi thanh ngang ở bên trái của thanh cuộn là một nhóm các thuộc tính liên quan. Ví dụ, nhóm **Design** tập hợp bốn thuộc tính liên quan. Các nhóm được hiển thị trên Hình 2.22 là **Behavior, Data, Design, Focus** và **Layout**. Chú ý rằng mỗi nhóm là một node.
- Cột bên trái của cửa sổ **Properties** liệt kê tên thuộc tính của các đối tượng; cột bên phải hiển thị giá trị của mỗi thuộc tính. Ở chương tiếp theo, bạn sẽ học cách thiết lập thuộc tính cho các đối tượng.
- Bạn có thể duyệt qua danh sách các thuộc tính bằng cách kéo thanh cuộn lên hoặc xuống.
- Bất cứ khi nào bạn chọn thuộc tính, mô tả của thuộc tính sẽ hiển thị ở phía dưới cửa sổ **Properties**. (Thuộc tính `Text` đang được chọn trên Hình 2.22).



Hình 2.22 Cửa sổ **Properties** hiển thị thuộc tính của Form.

TỰ ÔN TẬP

1. _____cho phép bạn thêm điều khiển vào Form một cách trực quan.
   a) **Solution Explorer** b) Cửa sổ **Properties**
   c) **Toolbox** d) Cửa sổ **Dynamic Help**
2. Cửa sổ_____cho phép bạn xem file trong solution.
   a) **Properties** b) **Solution Explorer**
   c) **Toolbox** d) Các lựa chọn trên đều sai

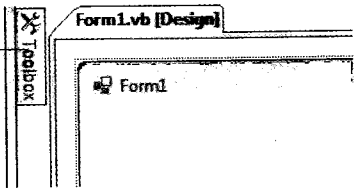
Đáp án: 1) c. 2) b.

2.6 Tự động ẩn

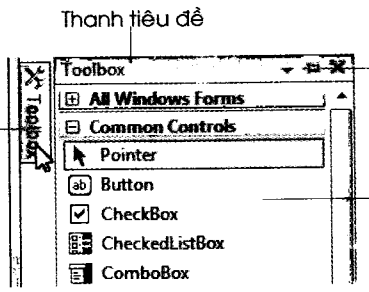
Visual Basic cung cấp tính năng **tự động ẩn (auto-hide)** để tiết kiệm không gian. Tính năng này cho phép bạn ẩn hoặc hiển thị cửa sổ **Toolbox**, **Solution Explorer** và cửa sổ **Properties** trong IDE. Khi tính năng tự động ẩn được kích hoạt cho những cửa sổ này, tab đại diện cho cửa sổ bị ẩn sẽ hiển thị dọc theo cạnh của cửa sổ IDE.

Sử dụng tính năng tự động ẩn

1. ***Kích hoạt tính năng tự động ẩn và hiển thị cửa sổ bị ẩn.*** Để kích hoạt tính năng tự động ẩn, bạn hãy nhấn chuột vào biểu tượng có hình ghim dọc trên cửa sổ để chuyển thành biểu tượng có hình ghim nằm ngang. Toolbar nằm dọc theo một trong các cạnh của IDE chứa một hoặc nhiều tab, mỗi tab là một cửa sổ bị ẩn (Hình 2.23). Trỏ chuột lên tab **Toolbox** để hiển thị **Toolbox** (Hình 2.24).

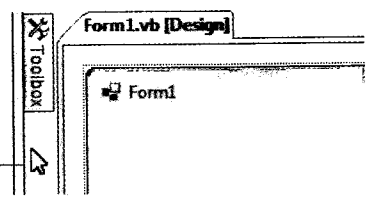


Hình 2.23 Tab cửa sổ bị ẩn.



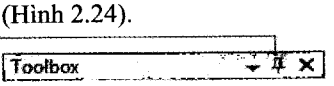
Hình 2.24 Hiển thị một cửa sổ bị ẩn với tính năng tự động ẩn được kích hoạt.

2. ***Ẩn cửa sổ.*** Di chuyển con trỏ chuột ra khỏi khu vực cửa sổ **Toolbox** để ẩn **Toolbox** (Hình 2.25).



Hình 2.25 Ẩn **Toolbox** bằng cách di chuyển con trỏ chuột ra khỏi vị trí của **Toolbox**.

3. ***Tắt tính năng tự động ẩn.*** Giữ cửa sổ **Toolbox** mở và vô hiệu hóa tính năng tự động ẩn bằng cách nhấn chuột vào biểu tượng **đinh ghim** nằm trên thanh tiêu đề trên Hình 2.24. Chú ý rằng khi cửa sổ được ghim, biểu tượng đinh ghim sẽ đứng thẳng (Hình 2.26), còn nếu tính năng tự động ẩn được kích hoạt thì biểu tượng đinh ghim sẽ hiển thị ở dạng nằm ngang (Hình 2.24).



Hình 2.26 Biểu tượng đinh ghim ở vị trí đứng thẳng.

TỰ ÔN TẬP

1. Visual Basic cung cấp tính năng tiết kiệm không gian gọi là_____.
   a) tự động đóng
   b) ẩn
   c) thu nhỏ
   d) tự động ẩn
2. Khi tính năng tự động ẩn được kích hoạt, biểu tượng đinh ghim của nó sẽ_____.
   a) nằm ngang
   b) đứng thẳng
   c) đóng xuống
   d) nằm chéo

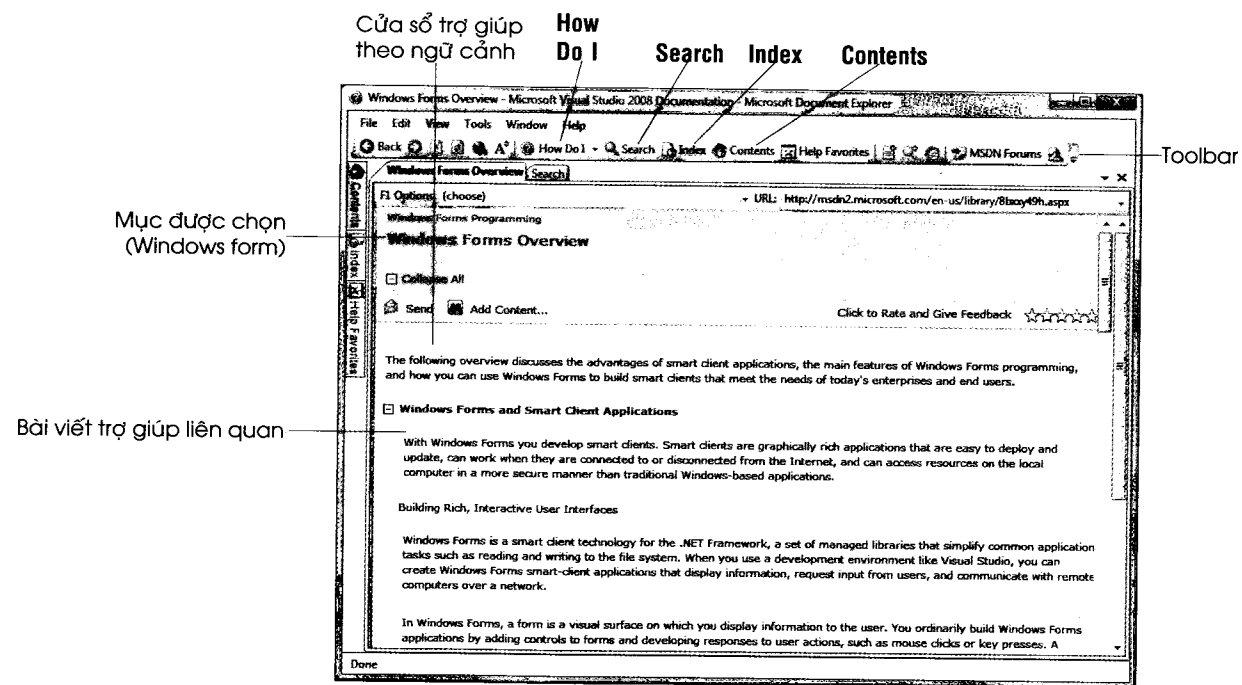
Đáp án: **1)** d. **2)** a.

2.7 Sử dụng Help

Visual Basic cung cấp rất nhiều tính năng trợ giúp. Các lệnh trên menu **Help** được tóm tắt trên Hình 2.27. Sử dụng **Help** là một cách hay để thu thập thông tin nhanh chóng về IDE và các tính năng của IDE. **Help** cung cấp danh sách các bài viết liên quan đến "nội dung hiện tại" (chẳng hạn các đối tượng nằm ở vị trí con trỏ hiện tại). Visual Basic cũng cung cấp các **trợ giúp theo ngữ cảnh (context - sensitive help)** hiển thị các bài viết trợ giúp liên quan hơn là một danh sách các bài viết chung chung (Hình 2.28). Để sử dụng trợ giúp theo ngữ cảnh, nhấn chuột vào một mục (chẳng hạn form), sau đó nhấn phím *F1*. Cửa sổ trợ giúp cung cấp các bài trợ giúp, đoạn mã mẫu và thông tin "Getting Started". Ngoài ra, còn có toolbar cung cấp truy cập đến tính năng trợ giúp **How Do I**, **Search**, **Index** và **Contents**. Để quay lại IDE, có thể đóng cửa sổ trợ giúp hoặc chọn biểu tượng IDE trên thanh task bar của Windows.

Lệnh	Mô tả
How Do I?	Chứa liên kết đến các chủ đề liên quan, bao gồm làm thế nào để nâng cấp ứng dụng và tìm hiểu về các dịch vụ web, kiến trúc và thiết kế, file, dữ liệu,...
Search	Tìm kiếm các bài viết trợ giúp dựa trên từ khóa tìm kiếm.
Index	Hiển thị danh sách các đề mục theo bảng chữ cái.
Contents	Hiển thị bảng nội dung được phân nhóm trong đó các bài viết trợ giúp được tổ chức thành các chủ đề.

Hình 2.27 Các lệnh menu **Help**.



Hình 2.28 Cửa sổ trợ giúp theo ngữ cảnh.

TỰ ÔN TẬP

1. _____hiển thị bài viết trợ giúp liên quan dựa trên đối tượng được lựa chọn.
   a) Trợ giúp tích hợp
   b) Trợ giúp theo ngữ cảnh
   c) Trợ giúp ngoại
   d) Trợ giúp dựa vào ngữ cảnh
2. Lệnh **Help**_____hiển thị một danh sách các chủ đề được sắp xếp theo bảng chữ cái.
   a) **Search**
   b) **Browse**
   c) **Contents**
   d) **Index**

Đáp án: 1) b. 2) d.

2.8 Lưu và đóng project trong Visual Basic

Sau khi chỉnh sửa project hoàn tất, bạn nên lưu file và đóng project lại.

Đóng project của ứng dụng Welcome

1. ***Lưu file trong project.*** Trước khi đóng project của ứng dụng **Welcome**, bạn nên lưu file của project lại để đảm bảo rằng mọi thay đổi của project sẽ không bị mất. Mặc dù ở chương này, bạn chưa tạo ra bất kỳ thay đổi nào cho project tuy nhiên trong hầu hết các chương sau, bạn sẽ tạo ra những thay đổi cho project. Để tiến hành lưu file của project, chọn **File > Save All.**
2. ***Đóng project.*** Chọn **File > Close Project.**

2.9 Tài nguyên web

Hãy dành một chút thời gian để ghé thăm các site sau.

`www.deitel.com/VisualBasic2008/`

Site này liệt kê rất nhiều tài nguyên web chúng tôi đã sử dụng để chuẩn bị viết cuốn sách *Simply Visual Basic 2008: một cách tiếp cận hướng ứng dụng* dưới dạng các bài hướng dẫn. Có rất nhiều tài nguyên ở đây cho phép bạn làm quen với thế giới Visual Basic 2008.

`msdn.microsoft.com/vstudio`

Site này là trang chủ của Microsoft Visual Studio. Site này chứa tin tức, tài liệu, dữ liệu có thể tải về cùng các tài nguyên khác.

`msdn.microsoft.com/vbasic/default.aspx`

Site này chứa thông tin về những bản phát hành Visual Basic mới nhất bao gồm các dữ liệu có thể tải về, thông tin và tài nguyên cộng đồng.

`forums.microsoft.com/MSDN/default.aspx?ForumGroupID=10&SiteID=1`

Site này cung cấp truy cập đến các diễn đàn Microsoft Visual Basic, tại đó bạn có thể nhận được những lời giải đáp cho những câu hỏi về ngôn ngữ Visual Basic và IDE.

`msdn.microsoft.com/msdnmag/`

Đây là site *Microsoft Developer Network Magazine.* Site này cung cấp các bài viết và đoạn mã ví dụ về chủ đề lập trình Visual Basic và .NET. Trang web cũng cung cấp bản lưu trữ của các số tạp chí đã phát hành.

`www.vbi.org`

Site này chứa bài viết về Visual Basic, bình luận về các cuốn sách và phần mềm, tài liệu, dữ liệu có thể tải về, liên kết tới các nguồn thông tin khác...

`www.vbcity.com`

Site này cung cấp các bài viết Visual Basic, các hướng dẫn, câu hỏi thường gặp,... Gửi đoạn mã Visual Basic của bạn đến đây để được đánh giá bởi các lập trình viên khác.

2.10 Tổng kết

Chương này giới thiệu về môi trường phát triển tích hợp (IDE) của Visual Basic 2008 Express Edition. Bạn đã được tìm hiểu về nhiều tính năng như tab, menu, thanh menu, toolbar, biểu tượng, tự động ẩn...

Bạn đã tạo ứng dụng Windows Forms chứa đối tượng `Form` tên là `Form1.vb`. Các điều khiển trên `Form` thể hiện giao diện người dùng đồ họa (GUI) của ứng dụng.

Bạn đã được làm việc với cửa sổ **Solution Explorer**, **Toolbox**, **Properties**. Tất cả những cửa sổ này đều vô cùng quan trọng trong việc phát triển ứng dụng Visual Basic. Cửa sổ **Solution Explorer** cho phép bạn quản lý file của solution một cách trực quan. Cửa sổ **Toolbox** chứa một tập các điều khiển (được tổ chức thành các nhóm) cho phép bạn thiết kế giao diện người dùng đồ họa. Cửa sổ **Properties** cho phép bạn tạo các giao diện. Cửa sổ **Properties** cho phép bạn thiết lập thuộc tính cho `Form` và những điều khiển trên `Form`.

Bạn đã được khám phá các tính năng trợ giúp của Visual Basic bao gồm cửa sổ **Help**, menu **Help** và trợ giúp theo ngữ cảnh. Cửa sổ **Help** hiển thị những liên kết liên quan đến đối tượng đang được chọn bằng con trỏ chuột. Bạn đã được tìm hiểu những website chứa thông tin về Visual Basic.

Ở chương sau, bạn sẽ bắt đầu tạo ứng dụng Visual Basic. Bạn sẽ thực hiện theo những hướng dẫn cụ thể từng bước để hoàn tất ứng dụng **Welcome** bằng cách sử dụng kỹ thuật lập trình trực quan và các tính năng của IDE mà bạn đã tìm hiểu ở chương này.

TỔNG KẾT KỸ NĂNG

Tạo project mới

- Chọn **File > New Project...** hoặc **File > Open Project...**
- Nhấn chuột vào liên kết **Create: Project...** hoặc **Open: Project...** (ở trên **Start Page** trong mục **Recent Projects**).
- Chọn **Windows Forms Application** trong khung **Templates:**.
- Nhập tên của project trong `TextBox` **Name:**.
- Nhấn vào `Button` **OK**.

Lưu project

- Chọn **File > Save All**.
- Nhập tên của project vào `TextBox` **Name:**.
- Chọn thư mục của project trong `TextBox` **Location:**.

Hiển thị tooltip cho biểu tượng Visual Basic

- Trỏ chuột lên biểu tượng và giữ nguyên cho đến khi tooltip hiển thị.

Thu nhỏ node trong Solution Explorer

- Nhấn chuột vào hộp dấu trừ của node.

Mở rộng node trong Solution Explorer

- Nhấn chuột vào hộp dấu cộng của node.

Di chuyển qua danh sách các điều khiển trên Toolbox

- Nhấn chuột vào các mũi tên thanh cuộn.

Hiển thị cửa sổ Properties

- Chọn **View > Properties Window** hoặc nhấn phím *F4*.

Hiển thị Solution Explorer

- Chọn **View > Solution Explorer.**

Hiển thị Toolbox

- Chọn **View > Toolbox.**

Hiển thị một cửa sổ bị ẩn

- Trỏ chuột lên tab của cửa sổ bị ẩn.

Tắt chế độ tự động ẩn

- Nhấn chuột vào biểu tượng đinh ghim nằm ngang để chuyển thành biểu tượng đinh ghim đứng thẳng.

Kích hoạt tự động ẩn

- Nhấn chuột vào biểu tượng đinh ghim đứng thẳng để chuyển thành biểu tượng đinh ghim nằm ngang.

Mở cửa sổ Help

- Chọn **Help > How Do I, Help > Search, Help > Contents** hay **Help > Index.**
- Chọn đối tượng bạn cần trợ giúp và nhấn phím *F1*.

THUẬT NGỮ

biểu tượng (icon) - Đại diện dưới dạng đồ họa cho một lệnh trong Visual Studio 2008 IDE.

biểu tượng Alphabetical - Biểu tượng trong cửa sổ **Properties** mà khi nhấn chuột vào thì các thuộc tính sẽ được sắp xếp theo thứ tự bảng chữ cái.

biểu tượng Categorized - Biểu tượng trong cửa sổ **Properties** mà khi nhấn chuột vào thì các thuộc tính sẽ được sắp xếp thành các nhóm.

biểu tượng đinh ghim - Là biểu tượng để kích hoạt hoặc vô hiệu hóa tính năng tự động ẩn.

chế độ Design - Chế độ hiển thị của IDE chứa trình thiết kế Windows Forms cho phép bạn sắp đặt điều khiển cho ứng dụng Windows Forms.

cơ sở dữ liệu (database) - Lưu trữ thông tin để ứng dụng có thể truy cập được.

cửa sổ Properties - Cửa sổ hiển thị các thuộc tính của `Form` hay đối tượng điều khiển.

đầu ra (output) - Kết quả của ứng dụng.

đầu vào (input) - Dữ liệu mà người dùng nhập vào ứng dụng.

`Form` - Là đối tượng trình bày giao diện người dùng đồ họa của ứng dụng Windows.

giao diện đồ họa người dùng (Graphical user interface - GUI) - Là phần trực quan của ứng dụng mà người dùng có thể tương tác được.

hộp chọn đối tượng (component object box) - `ComboBox` nằm ở phía trên cửa sổ **Properties** cho phép bạn chọn `Form` hoặc đối tượng điều khiển mà bạn muốn thiết lập thuộc tính.

hộp dấu cộng (plus box) - Là biểu tượng khi nhấn vào sẽ mở rộng node.

hộp dấu trừ (minus box) - Là biểu tượng mà khi nhấn vào sẽ thu gọn node lại.

hộp thoại (dialog) - Là cửa sổ hiển thị để thu thập thông tin.

hộp thoại New Project - Là hộp thoại cho phép nhấn chuột vào để chọn kiểu ứng dụng muốn tạo.

lệnh Contents - Lệnh hiển thị nội dung được phân loại trong đó các bài viết được sắp xếp theo chủ đề.

menu - Một nhóm các lệnh liên quan, khi được chọn, IDE có thể thực hiện các hành động cụ thể như mở cửa sổ, lưu file, in file và chạy ứng dụng.

menu Data - Menu của IDE chứa lệnh tương tác với database.

menu Debug - Menu của IDE chứa lệnh gỡ lỗi và chạy chương trình.

menu item (mục menu) - Là lệnh nằm trong menu, khi được chọn sẽ khiến ứng dụng thực hiện một hành động cụ thể.

menu Tools - Là menu của IDE chứa lệnh để truy cập những công cụ và tùy chọn bổ sung của IDE cho phép tùy chỉnh IDE.

Microsoft Developer Network (MSDN) - Là thư viện trực tuyến chứa bài viết, dữ liệu tải về và các hướng dẫn về công nghệ của các lập trình viên Visual Basic.

môi trường phát triển tích hợp (Integrated Development Environment IDE) - Phần mềm được dùng để phát triển, tạo tài liệu, chạy và gỡ lỗi ứng dụng.

mũi tên thanh cuộn (scroll arrow) - Các mũi tên ở cuối của thanh cuộn giúp bạn cuộn qua các đối tượng.

project - Là nhóm các file liên quan tạo nên ứng dụng.

solution - Chứa một hoặc nhiều project.

Solution Explorer - Là cửa sổ cung cấp khả năng truy cập đến tất các project và file liên quan trong solution.

Start Page (trang khởi động) - Trang đầu tiên hiển thị khi Visual Studio 2008 được mở.

tab hiện thời (active tab) - Tab của đối tượng đang được hiển thị trên IDE.

template - Điểm khởi đầu cho các dự án bạn tạo ra trong Visual Basic.

thanh địa chỉ (location bar) - `ComboBox` nằm trong trình duyệt web được tích hợp bên trong Visual Basic cho phép bạn nhập địa chỉ của web site muốn truy cập.

thanh menu (menu bar) - Chứa các menu của cửa sổ.

thanh tiêu đề (title bar) - Phần trên cùng của cửa sổ hiển thị tiêu đề của cửa sổ.

thuộc tính (property) - Chỉ ra các đặc tính (như kích thước, màu sắc, vị trí) của Form và các đối tượng điều khiển.

toolbar (thanh công cụ) - Là thanh chứa các button mà khi nhấn vào sẽ thực thi lệnh.

Toolbox - Là cửa sổ chứa điều khiển được sử dụng để xây dựng và tùy biến `Form`.

tooltip - Là mô tả cho biểu tượng, xuất hiện khi con trỏ chuột trỏ đến biểu tượng vài giây.

trình duyệt web được tích hợp bên trong - Trình duyệt web (Internet Explorer) được tích hợp trong Visual Basic 2008 Express cho phép bạn có thể duyệt web.

trợ giúp theo ngữ cảnh (context-sensitive help) - Là tùy chọn trợ giúp (khi nhấn phím *F1*) cung cấp liên kết đến các bài viết liên quan đến đối tượng đang được lựa chọn (là đối tượng đang được nhấn chuột vào).

tự động ẩn (auto-hide) - Tính năng tiết kiệm khoảng trống được sử dụng cho cửa sổ như **Toolbox**, **Properties** và **Solution Explorer** để ẩn cửa sổ cho đến khi con trỏ chuột trỏ vào tab của cửa sổ bị ẩn.

ứng dụng Windows Forms - Là ứng dụng thực thi trên hệ điều hành Windows (Windows XP hoặc Vista) và có giao diện người dùng đồ họa (GUI) - phần trực quan của ứng dụng mà người dùng có thể tương tác.

Visual Studio - Là môi trường phát triển tích hợp của Microsoft (IDE), cho phép lập trình viên phát triển ứng dụng bằng nhiều ngôn ngữ lập trình thuộc .NET.

Windows Form Designer - Sử dụng để thiết kế giao diện của ứng dụng Windows Form.

CÂU HỎI TRẮC NGHIỆM

2.1 Môi trường phát triển tích hợp_____được sử dụng để tạo ứng dụng viết bằng nhiều ngôn ngữ khác nhau.

a) **Solution Explorer**
b) Gates
c) Visual Studio
d) Microsoft

2.2 Phần mở rộng `.vb` là của_____.

a) file Visual Basic
b) file trợ giúp động
c) file trợ giúp
d) file rất lớn

2.3 Hình ảnh trên các button của toolbar được gọi là _____.

a) prototype
b) biểu tượng
c) tooltip
d) tab

2.4 _____cho phép lập trình viên có thể tùy chỉnh điều khiển một cách trực quan mà không phải viết mã.

a) Cửa sổ **Properties**
b) **Solution Explore**
c) Thanh menu
d) **Toolbox**

2.5 _____sẽ ẩn **Toolbox** khi con trỏ chuột di chuyển ra ngoài khu vực của **Toolbox**.

a) Tính năng chọn thành phần
b) Tính năng tự động ẩn
c) Lệnh ghim
d) Lệnh thu nhỏ

2.6 _____sẽ hiển thị khi trỏ chuột trỏ lên biểu tượng nằm trên toolbar một vài giây.

a) Danh sách thả xuống
b) Menu
c) Tooltip
d) Mũi tên đi xuống

2.7 Visual Basic IDE cung cấp_____.

a) tài liệu trợ giúp
b) toolbar
c) cửa sổ để truy cập file của project
d) Các lựa chọn trên đều đúng

2.8 _____chứa danh sách các liên kết hữu ích như **Getting Started** và **Visual Basic Express Headlines.**

a) Cửa sổ **Solution Explorer**
b) Cửa sổ **Properties**
c) **Start Page**
d) Liên kết **Toolbox**

2.9 Cửa sổ **Properties** chứa_____.

a) hộp chọn đối tượng
b) **Solution Explorer**
c) menu
d) thanh menu

2.10 ____có thể được cải tiến bằng cách thêm các thành phần có thể tái sử dụng như `Button`.

a) Điều khiển
b) `Form`
c) Tab
d) Thuộc tính

2.11 Để duyệt web, Visual Basic tích hợp_____.

a) chế độ Web
b) Excel
c) tab **Web**
d) Internet Explorer

2.12 Giao diện người dùng đồ họa của ứng dụng có thể chứa_____.

a) toolbar
b) biểu tượng
c) menu
d) Các lựa chọn trên đều đúng

2.13 _____không chứa biểu tượng đinh ghim.

a) Cửa sổ **Properties**
b) Cửa sổ **Solution Explorer**
c) Cửa sổ **Toolbox**
d) Tab hiện thời

2.14 Khi nhấn vào_____trong cửa sổ **Solution Explorer** sẽ mở rộng node và khi nhấn vào_____thì sẽ thu nhỏ node.

a) hộp dấu trừ, hộp dấu cộng
b) hộp dấu cộng, hộp dấu trừ
c) mũi tên đi lên, mũi tên đi xuống
d) mũi tên sang trái, mũi tên sang phải

2.15 _____của `Form` chỉ ra các thuộc tính như kích thước và vị trí.

a) Node
b) Dữ liệu được nhập vào
c) Thuộc tính
d) Thanh tiêu đề

BÀI TẬP

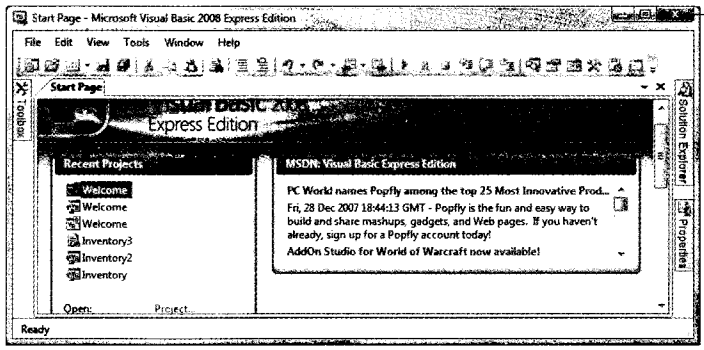
2.16 ***(Đóng và mở Start Page)*** Trong bài tập này, bạn sẽ tìm hiểu cách đóng và mở **Start Page**. Để thực hiện điều này, bạn hãy thao tác theo những bước sau đây:

a) Đóng Visual Basic nếu Visual Basic đang được mở bằng cách chọn **File > Exit** hoặc bằng cách nhấn chuột vào nút đóng.

b) Khởi động Visual Basic 2008 Express Edition.

c) Đóng **Start Page** bằng cách nhấn chuột vào nút đóng (Hình 2.29).

d) Chọn **View > Other Windows > Start Page** để hiển thị **Start Page**.



Hình 2.29 Đóng **Start Page**.

2.17 ***(Kích hoạt tự động ẩn cho cửa sổ Solution Explorer)*** Trong bài tập này, bạn sẽ học cách sử dụng tính năng tự động ẩn của cửa sổ **Solution Explorer** bằng cách thực hiện các bước sau đây:

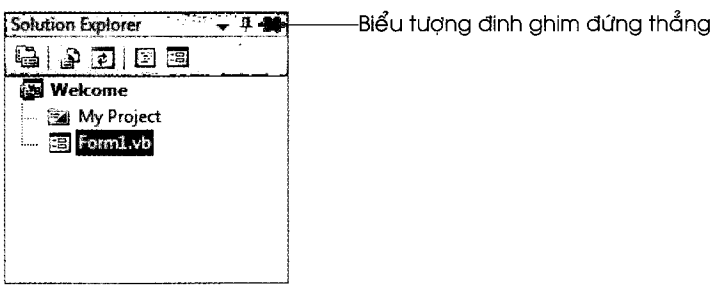
a) Mở **Start Page**.

b) Trên **Start Page**, nhấn chuột vào `Button` **Open Project** để hiển thị hộp thoại **Open Project**. Bạn có thể chuyển sang *Bước* e) nếu ứng dụng **Welcome** đã được mở.

c) Trong hộp thoại **Open Project**, di chuyển đến thư mục `C:\SimplyVB2008\Welcome` và nhấn chuột vào **Open**.

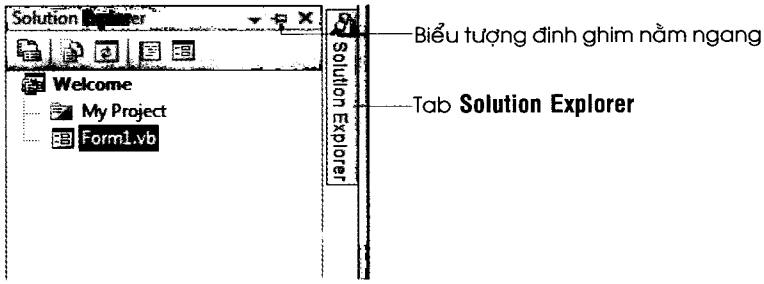
d) Trong hộp thoại **Open Project**, chọn `Welcome` (hoặc `Welcome.sln`) và chọn **Open**.

e) Trỏ chuột lên biểu tượng đinh ghim đứng thẳng trên thanh tiêu đề của cửa sổ **Solution Explorer** (Hình 2.30).



Hình 2.30 Kích hoạt tự động ẩn.

f) Nhấn chuột vào biểu tượng đinh ghim đứng thẳng để tab **Solution Explorer** hiển thị ở bên phải IDE và biểu tượng đinh ghim đứng thẳng chuyển thành biểu tượng đinh ghim nằm ngang (Hình 2.31). Tự động ẩn lúc này đã được kích hoạt cho cửa sổ **Solution Explorer**.



Hình 2.31 Cửa sổ **Solution Explorer** được kích hoạt tính năng tự động ẩn.

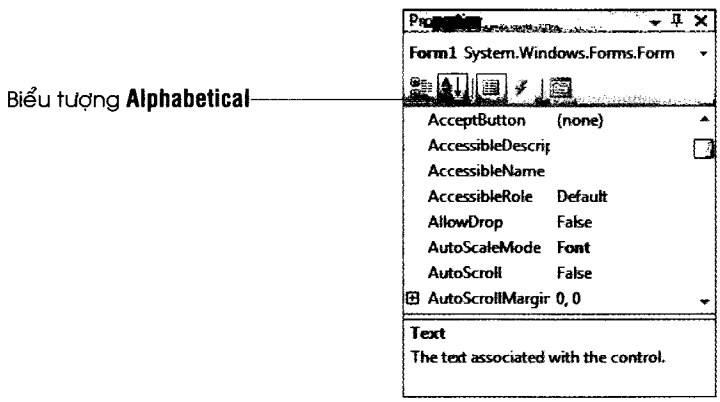
2.18 ***(Sắp xếp thuộc tính trong cửa sổ Properties theo bảng chữ cái)*** Trong bài tập này, bạn sẽ học cách sắp xếp thuộc tính của cửa sổ **Properties** theo bảng chữ cái bằng cách thực hiện những bước sau đây:

a) Mở ứng dụng **Welcome** bằng cách thực hiện các bước từ a) đến d) của bài tập 2.17. Nếu ứng dụng **Welcome** đã được mở thì chuyển luôn sang bước này.

b) Tìm cửa sổ **Properties**. Nếu cửa sổ **Properties** không hiển thị, chọn **View > Properties Window.**

c) Nhấn chuột vào `Form`.

d) Để sắp xếp các thuộc tính theo bảng chữ cái, nhấn chuột vào biểu tượng **Alphabetical** trên cửa sổ **Properties** (Hình 2.32). Các thuộc tính sẽ được hiển thị theo thứ tự bảng chữ cái.



Hình 2.32 Sắp xếp các thuộc tính theo thứ tự bảng chữ cái.

CHƯƠNG 3

Ứng dụng Welcome

Giới thiệu về lập trình trực quan

Mục tiêu

Trong chương này, bạn sẽ tìm hiểu để:

- Thiết lập nội dung trên thanh tiêu đề của **Form**.
- Thay đổi màu nền của **Form**.
- Thêm điều khiển **Label** vào **Form**.
- Hiển thị văn bản trên điều khiển **Label**.
- Thêm điều khiển **PictureBox** vào **Form**.
- Hiển thị hình ảnh trên điều khiển **PictureBox**.
- Chạy ứng dụng.

Nội dung chính

Ngày nay, người dùng thích sử dụng phần mềm có giao diện người dùng đồ họa (GUI), là phần mềm có thể đáp lại hành động của người dùng như nhấn `Button`, nhập liệu... Do đó phần lớn ứng dụng Windows như Microsoft Word và Internet Explorer đều dựa trên GUI. Từ thời điểm này về sau, chúng tôi sẽ gọi giao diện người dùng đồ họa đơn giản là giao diện, vì trong cuốn sách này chúng tôi chỉ đề cập đến ứng dụng phát triển dựa trên giao diện người dùng đồ họa. Với Visual Basic, bạn có thể tạo ứng dụng Windows để nhập, xuất thông tin theo nhiều cách, bạn sẽ được tìm hiểu những cách đó trong cuốn sách này.

Trong chương này, bạn sử dụng kỹ thuật lập trình trực quan để hoàn thiện ứng dụng **Welcome** mà bạn đã tạo trong Chương 2. Bạn sẽ xây dựng giao diện của ứng dụng bằng cách thêm điều khiển `Label` và `PictureBox` vào `Form`, điều khiển `Label` để hiển thị văn bản và điều khiển `PictureBox` để hiển thị hình ảnh. Bạn tùy chỉnh cách hiển thị `Form`, `Label` và `PictureBox` bằng cách thiết lập giá trị thuộc tính trong cửa sổ **Properties**. Bạn sẽ thiết lập các giá trị thuộc tính như màu nền của `Form`, hình ảnh trong `PictureBox` và nội dung văn bản trên `Label`. Bạn cũng sẽ học cách chạy ứng dụng trong Visual Basic 2008 IDE.

3.1 Chạy thử ứng dụng Welcome

Trong chương trước, bạn đã được tìm hiểu về IDE Visual Basic 2008. Trong chương này, bạn sẽ sử dụng Visual Basic để tiếp tục xây dựng ứng dụng **Welcome** mà bạn đã tạo trong Chương 2. Ứng dụng này phải đáp ứng những yêu cầu sau:

Yêu cầu đối với ứng dụng

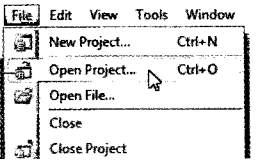
Hãy nhớ lại rằng Công ty phần mềm (Deitel & Associates) đã đề nghị bạn phát triển ứng dụng Visual Basic hiển thị thông điệp "Welcome to Visual Basic 2008!" và hình ảnh biểu tượng của công ty. Hiện tại bạn đã làm quen với IDE Visual Basic, nhiệm vụ của bạn là phát triển ứng dụng này để thỏa mãn yêu cầu của công ty.

Bạn bắt đầu bằng việc chạy thử ứng dụng đã hoàn thiện. Sau đó, bạn tìm hiểu những tính năng bổ sung của Visual Basic cần thiết để xây dựng cho mình một phiên bản của ứng dụng này.

Chạy thử ứng dụng Welcome

1. ***Mở ứng dụng đã hoàn thiện.*** Khởi động Visual Basic và chọn **File > Open Project...** (Hình 3.1) để hiển thị hộp thoại **Open Project** (Hình 3.2). Chọn thư mục `C:\Examples\Tutorial03\CompletedApplication\Welcome`. Chọn file solution **Welcome** (`Welcome.sln`) và nhấn chuột vào `Button` **Open**.

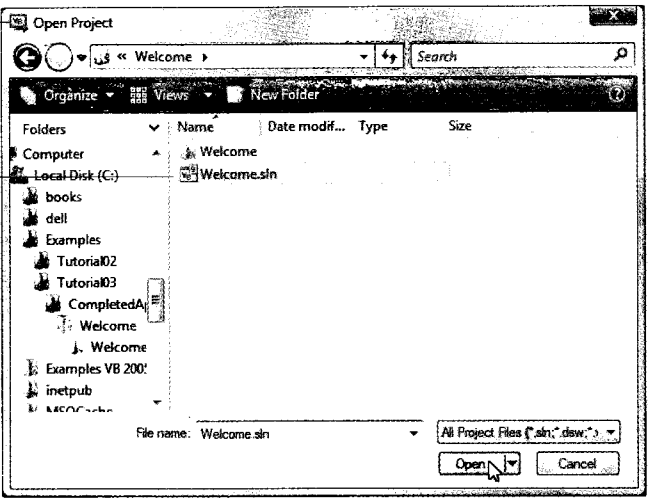
Lệnh **Open Project...** (đang được chọn) mở project có sẵn.



Hình 3.1 Mở project có sẵn bằng lệnh **Open Project...** trên menu **File**.

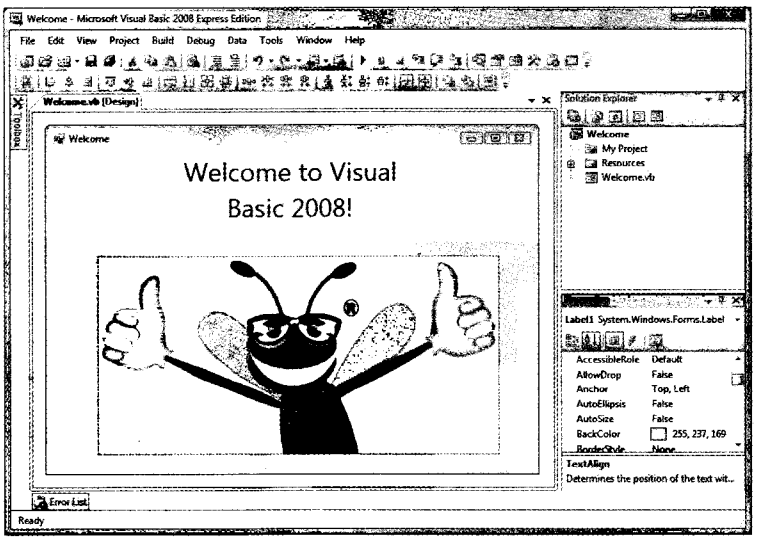
Hộp thoại **Open Project**

File solution **Welcome**



Hình 3.2 Hộp thoại **Open Project** hiển thị nội dung của solution **Welcome**.

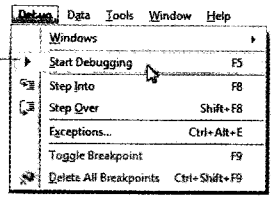
2. ***Mở Form trong chế độ Design.*** Nhấn đúp chuột vào `Welcome.vb` trong **Solution Explorer** để mở `Form` của ứng dụng **Welcome** trong chế độ **Design**. (Hình 3.3).



Hình 3.3 **Form** của ứng dụng **Welcome** trong chế độ **Design**.

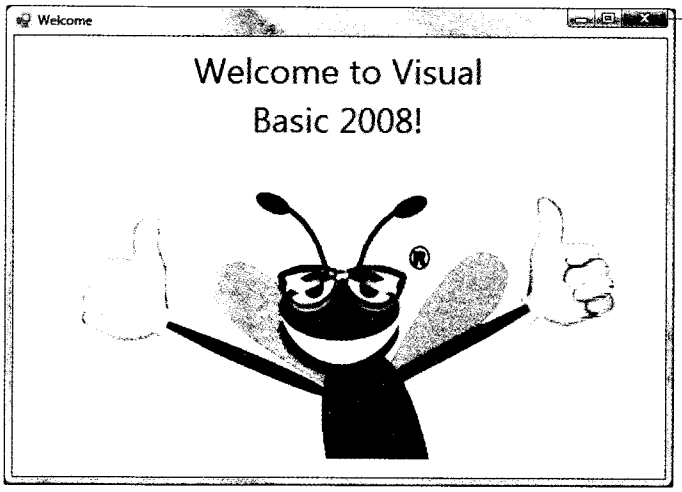
3. ***Chạy ứng dụng Welcome.*** Chọn **Debug > Start Debugging** (Hình 3.4). Lệnh **Start Debugging** sẽ chạy ứng dụng. `Form` **Welcome** sẽ hiển thị như trong Hình 3.5.

Lệnh **Start Debugging** (đang được chọn) sẽ chạy ứng dụng.



Hình 3.4 Chạy ứng dụng **Welcome** bằng lệnh **Start Debugging** trên menu **Debug**.

Nút đóng



Hình 3.5 Ứng dụng **Welcome** đang chạy.

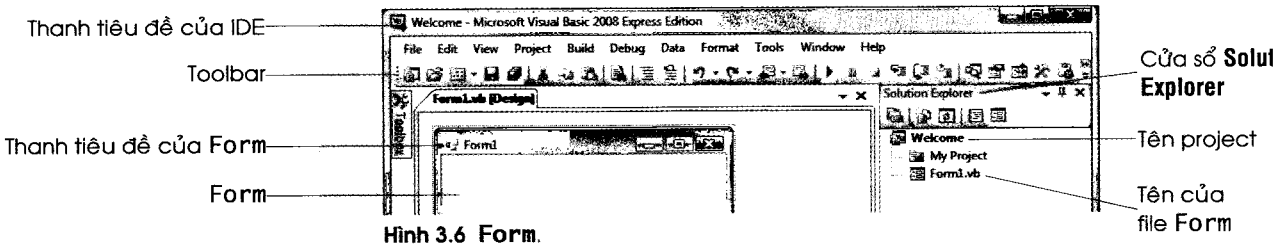
4. ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng bên phải của ứng dụng.
5. ***Đóng project***. Đóng project bằng cách chọn **File > Close Project.**

3.2 Xây dựng ứng dụng Welcome

Trong phần này, bạn sẽ thực hiện những bước cần thiết để phát triển ứng dụng **Welcome**. Ứng dụng này bao gồm `Form` chứa một điều khiển `Label` và một điều khiển `PictureBox`. Người dùng không thể thay đổi nội dung văn bản hiển thị trên điều khiển **`Label`** và hình ảnh hiển thị trên **`PictureBox`**. Bạn không cần phải viết một dòng mã nào để tạo ứng dụng này. Thay vào đó, bạn sẽ sử dụng kỹ thuật **lập trình trực quan (visual programming)** trong đó Visual Basic sẽ viết mã chương trình dựa trên những thao tác của bạn (như nhấn chuột, kéo thả các điều khiển)! Phần ngay sau đây sẽ hướng dẫn bạn cách xây dựng ứng dụng **Welcome** từ solution đã tạo trong Chương 2.

Thay đổi tên file của Form và nội dung hiển thị trên thanh tiêu đề

1. ***Mở project của ứng dụng Welcome***. Nhấn đúp chuột vào file `C:\SimplyVB2008\Welcome\Welcome.sln` mà bạn đã tạo ở Chương 2 để mở ứng dụng. Nhấn đúp chuột vào file `Form1.vb` trong cửa sổ **Solution Explorer** để hiển thị `Form`. Trong Hình 3.6 ứng dụng **Welcome** đang được mở trong IDE.

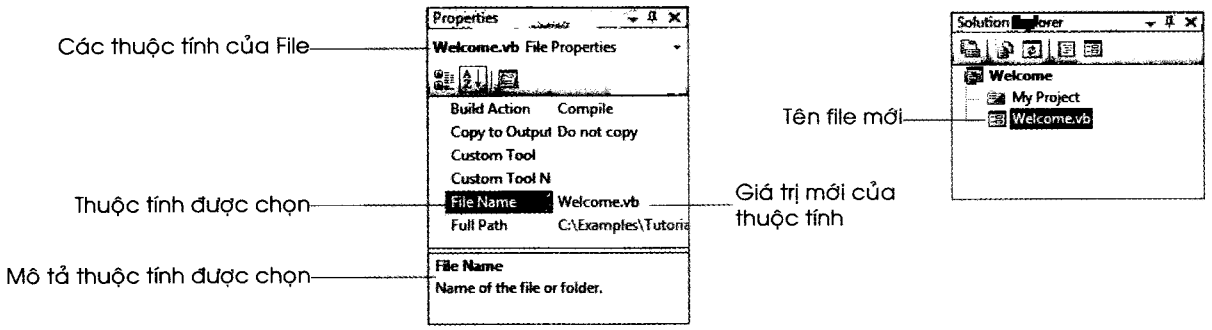


Hình 3.6 Form.

2. ***Thay đổi tên của file Form.*** Khi ứng dụng Windows được tạo, Visual Basic đặt tên Form là Form1.vb. Chọn file Form1.vb từ cửa sổ **Solution Explorer** (Hình 3.6) để hiển thị các thuộc tính của file trong cửa sổ **Properties** (cửa sổ nằm ở bên trái Hình 3.7). Nếu cửa sổ trên chưa được hiển thị, bạn có thể chọn **View > Properties Window** hoặc **View > Solution Explorer** để hiển thị cửa sổ tương ứng. Nhấn đúp chuột vào trường văn bản nằm bên phải thuộc tính **File Name** để chọn tên file hiện tại và nhập Welcome.vb (Hình 3.7). Nhấn phím *Enter* để cập nhật tên file Form mới. Chú ý rằng tên file sẽ thay đổi đồng thời trong cửa sổ **Solution Explorer** (cửa sổ nằm ở bên phải Hình 3.7) và cửa sổ **Properties**.

Thói quen lập trình tốt

Thay đổi tên file của Form (Form1.vb) thành tên mới mô tả cho mục đích của ứng dụng.

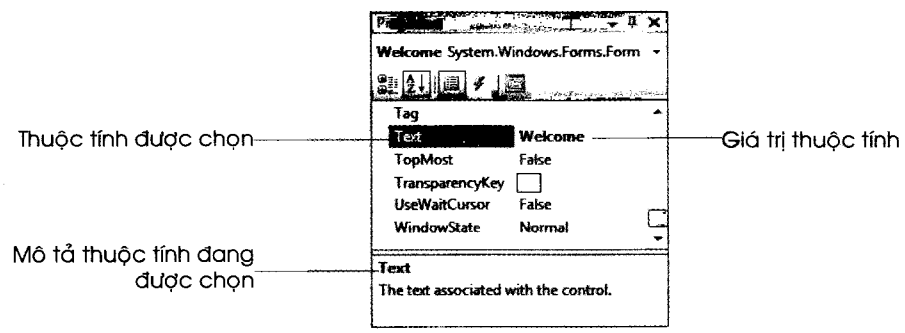


Hình 3.7 Thay đổi tên file của Form.

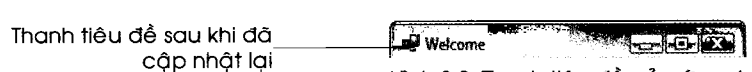
Mẹo thiết kế giao diện

Chọn tiêu đề Form ngắn gọn, có tính gợi tả. Viết hoa những từ không phải là mạo từ, giới từ và liên từ. Không nên sử dụng dấu câu.

3. ***Thay đổi nội dung hiển thị trên thanh tiêu đề của Form.*** Thanh tiêu đề nằm ở phần trên cùng của cửa sổ và chứa tiêu đề của cửa sổ. Sử dụng cửa sổ **Properties** để thay đổi nội dung hiển thị trên thanh tiêu đề của Form từ **Form1** thành **Welcome** (Hình 3.8). Nhấn chuột vào vùng xám trên Form. Giống như trong Hình 3.7, nhấn đúp chuột vào trường văn bản nằm bên phải thuộc tính **Text** trên cửa sổ **Properties** để chọn nội dung hiện tại của trường này và nhập vào chữ **Welcome**. Nhấn phím *Enter* để cập nhật lại thanh tiêu đề của Form (Hình 3.9).



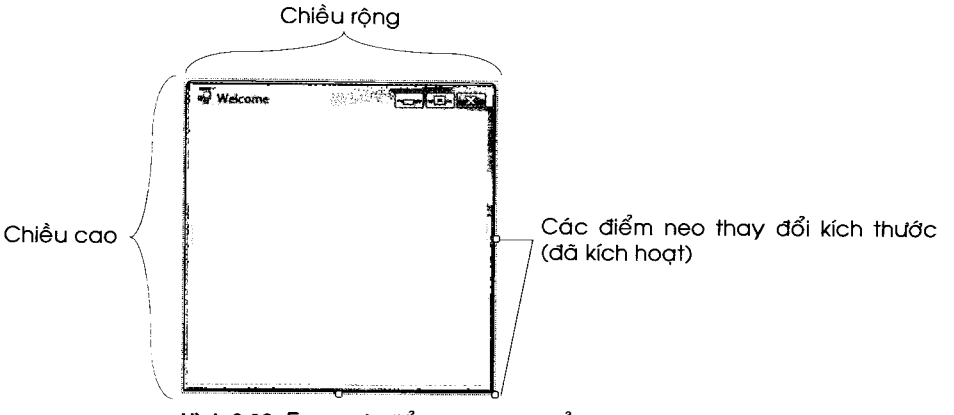
Hình 3.8 Thiết lập thuộc tính Text của Form.



Hình 3.9 Thanh tiêu đề của ứng dụng **Welcome**.

4. ***Lưu project.*** Chọn **File > Save All** để lưu project đã được sửa đổi.

Có một số cách để thay đổi kích thước Form. Nếu thay đổi kích thước không cần thật chính xác, bạn có thể nhấn chuột và kéo một trong các **điểm neo thay đổi kích thước (sizing handle)** của Form (các ô vuông màu trắng nằm xung quanh Form như trong Hình 3.10). Khi trỏ chuột lên các điểm neo này, con trỏ chuột sẽ chuyển thành hình hai mũi tên thể hiện các hướng bạn có thể kéo để thay đổi kích thước của Form.

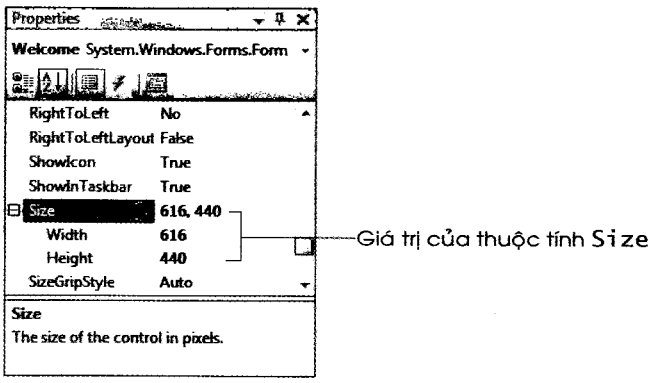


Hình 3.10 Form và điểm neo thay đổi kích thước.

Bạn có thể sử dụng thuộc tính **Size** bao gồm chiều rộng và chiều cao để thay đổi kích thước của Form. Đơn vị được sử dụng để đo kích thước Form là **pixel** (*picture element* - **điểm ảnh** hay còn gọi là **phần tử ảnh**). Pixel là một điểm nhỏ trên màn hình máy tính hiển thị một màu. Thuộc tính Size có hai thành phần là thuộc tính **Width** và **Height**. Thuộc tính Width xác định chiều rộng tính bằng pixel của Form và thuộc tính Height xác định chiều cao tính pixel của Form.

Thiết lập thuộc tính Size cho Form

1. ***Thiết lập chiều rộng và chiều cao của Form.*** Để giao diện ứng dụng **Welcome** của bạn trông giống hệt như Hình 3.5, bạn cần phải thay đổi kích thước của Form và các điều khiển trên Form. Nhấn chuột vào Form để chọn Form. Tìm thuộc tính Size của Form trong cửa sổ **Properties** (Hình 3.11). Nhấn chuột vào nút cộng nằm bên cạnh thuộc tính Size để mở rộng node. Nhập giá trị 616 cho thuộc tính Width và nhấn *Enter*. Nhập giá trị 440 cho thuộc tính Height và nhấn *Enter*. Chú ý rằng giá trị của thuộc tính Size (616,440) sẽ thay đổi khi một trong hai thuộc tính Width và Height thay đổi. Bạn cũng có thể nhập giá trị chiều rộng và chiều cao (phân cách nhau bằng dấu phẩy) trực tiếp vào trường Size.
2. ***Lưu project.*** Chọn **File > Save All** để lưu project đã chỉnh sửa.

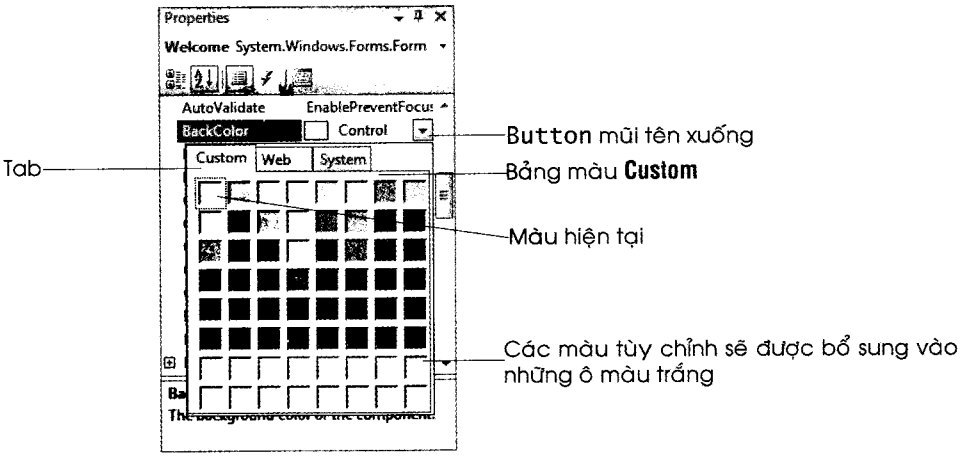


Hình 3.11 Giá trị thuộc tính Size của Form.

Lúc này bạn đã thiết lập xong kích thước của Form, tiếp theo bạn sẽ tiếp tục thay đổi màu nền của Form từ màu xám thành màu vàng.

Thiết lập màu nền cho Form

1. ***Khám phá các màu sắc***. Nhấn chuột vào Form để hiển thị các thuộc tính của Form trong cửa sổ **Properties**. Thuộc tính **BackColor** xác định màu nền của đối tượng. Khi bạn nhấn chuột vào giá trị của thuộc tính **BackColor** trong cửa sổ **Properties**, một Button mũi tên xuống (▼) sẽ hiển thị (Hình 3.12). Khi nhấn chuột vào Button mũi tên xuống, ba tab sẽ được hiển thị: **System** (mặc định), **Web** và **Custom**. Mỗi tab chứa một loạt màu sắc còn gọi là **bảng màu** (**palette**). Tab **System** hiển thị bảng màu chứa các màu sắc được sử dụng trong giao diện Microsoft Windows. Bảng màu này chứa màu sắc cho các điều khiển và desktop Windows. Màu sắc trong tab **System** dựa trên thiết lập màu sắc của Windows. Tab **Web** hiển thị một bảng màu gồm các **màu an toàn cho web** (**web-safe color**)- các màu này sẽ hiển thị giống nhau trên các máy tính khác nhau. Bảng màu trên tab **Custom** cho phép bạn chọn một loạt các màu sắc khác nhau được định nghĩa trước hoặc tạo màu sắc của riêng bạn. Nhấn chuột vào tab **Custom** để hiển thị bảng màu của tab như Hình 3.12.

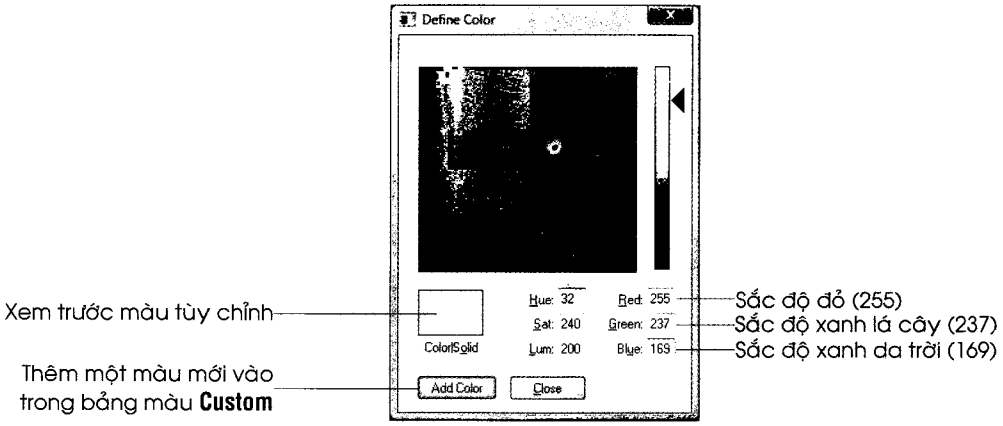


Hình 3.12 Hiển thị bảng màu **Custom** cho thuộc tính BackColor của Form.

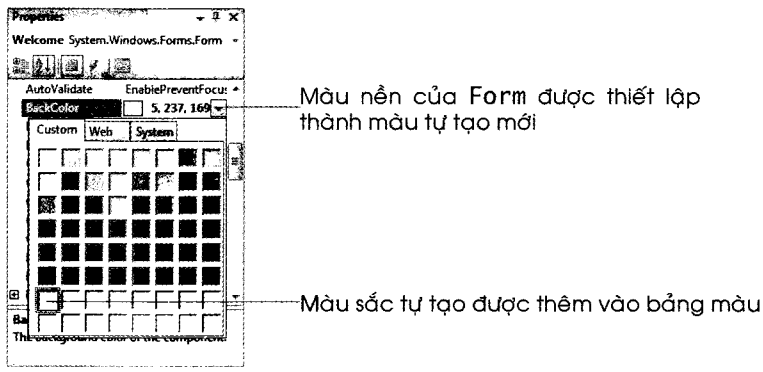
2. ***Thay đổi màu nền cho Form***. Nhấn chuột phải vào bất kì một trong 16 ô trắng nằm bên dưới bảng màu **Custom** để hiển thị hộp thoại **Define Color** (Hình 3.13). Các màu sắc có thể được tạo bằng cách nhập ba giá trị trong các TextBox **Hue:**, **Sat:** và **Lum:**, hay bằng cách nhập các giá trị **Red:**, **Green:** và **Blue:** hoặc bằng cách chọn một màu trong cửa sổ cầu vồng và kéo mũi tên đen lên xuống. Các giá trị trong các TextBox **Red:**, **Green:** và **Blue:** thể hiện lượng màu đỏ, xanh lá cây và xanh da trời cần để pha thành màu mà bạn chọn và thường được gọi là **giá trị RGB**. Sắc độ đỏ, xanh lá cây, xanh da trời đều nằm trong khoảng 0-255 (từ nhạt đến đậm). Để pha được màu trong ví dụ này, sắc độ **Red:** là 255, sắc độ **Green:** là 237 và sắc độ **Blue:** là 169. Nhấn chuột vào Button **Add Color** để đóng hộp thoại. Màu nền của Form lúc này đã được thay đổi và một màu mới đã được bổ sung vào bảng màu **Custom** (Hình 3.14).

Mẹo thiết kế giao diện

Sử dụng các màu sắc trong ứng dụng của bạn nhưng đừng gây phản cảm cho người dùng.



Hình 3.13 Thêm một màu mới vào trong bảng màu **Custom**.



Hình 3.14 Cửa sổ **Properties** sau khi thêm màu tự tạo mới.

3. ***Lưu project.*** Chọn **File > Save All** để lưu project đã chỉnh sửa của bạn.

Lúc này bạn đã hoàn tất việc chỉnh sửa Form, bạn có thể thêm điều khiển vào Form - một Label để hiển thị một lời chào.

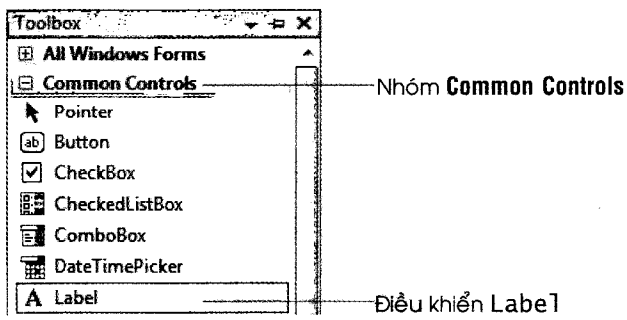
Thêm Label vào Form

Mẹo thiết kế giao diện

Sử dụng Label để hiển thị nội dung văn bản mà bạn không muốn người dùng thay đổi.

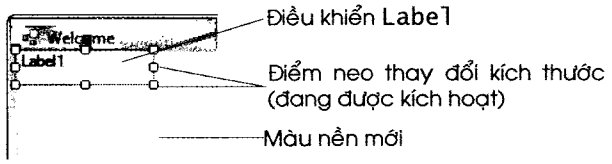
1. ***Thêm điều khiển Label vào Form.*** Nhấn chuột vào nhóm **Common Controls** (các điều khiển cơ bản) trong **Toolbox** (Hình 3.15) nếu nhóm chưa được chọn. Nếu **Toolbox** chưa hiển thị, chọn **View > Toolbox**. Nhấn đúp chuột vào điều khiển Label trong **Toolbox**. Một Label sẽ được hiển thị ở góc trên bên trái của Form (Hình 3.16). Bạn cũng có thể kéo Label từ **Toolbox** và thả lên Form. Bạn sử dụng điều khiển Label này để hiển thị thông điệp chào mừng. Mặc định, Label sẽ hiển thị chữ **Label1**.

   Chú ý rằng màu nền của Label giống như màu nền của Form. Khi một điều khiển Label được thêm vào Form, IDE ban đầu sẽ gán giá trị của thuộc tính BackColor của điều khiển đó bằng giá trị thuộc tính BackColor của Form.

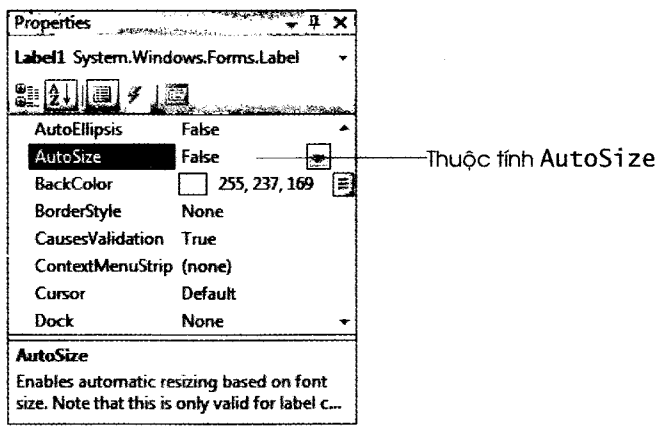


Hình 3.15 Nhấn chuột vào tab **Common Controls** trên **Toolbox**.

2. ***Thiết lập thuộc tính AutoSize của Label***. Nhấn chuột vào `Label` để chọn `Label`. Chú ý rằng thuộc tính của `Label` lúc này đã hiển thị trong cửa sổ **Properties**. Mặc định Visual Basic không cung cấp các điểm neo thay đổi kích thước (Hình 3.16) để thay đổi kích thước của `Label`. Nếu thuộc tính **`AutoSize`** của `Label` là `True`, `Label` sẽ tự mở rộng kích thước hoặc co lại cho vừa với giá trị của thuộc tính `Text` của `Label`. Để có thể thay đổi kích thước bằng tay, bạn cần phải thiết lập thuộc tính `AutoSize` của `Label` thành `False` (Hình 3.17). Bạn có thể thực hiện việc này bằng cách nhấn đúp vào giá trị của thuộc tính `AutoSize` hoặc bằng cách chọn `False` từ drop-down list (danh sách thả xuống).



Hình 3.16 Thêm `Label` vào `Form`.

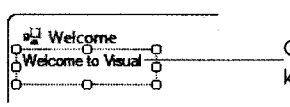


Hình 3.17 Thiết lập thuộc tính `AutoSize` thành `False`.

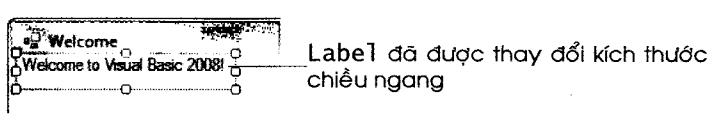
Mẹo thiết kế giao diện

Đảm bảo rằng điều khiển `Label` đủ rộng để hiển thị nội dung của chúng.

3. ***Thay đổi nội dung hiển thị trên Label***. Thuộc tính `Text` của `Label` chỉ ra nội dung hiển thị (**Label1**) của `Label`. Nhập `Welcome to Visual Basic 2008!` vào giá trị thuộc tính `Text` của `Label` và nhấn *Enter*. Chú ý rằng, nội dung này dài hơn so với kích thước `Label` (Hình 3.18). Sử dụng điểm neo thay đổi kích thước để mở rộng `Label` và hiển thị được toàn bộ nội dung trên `Label` (Hình 3.19).

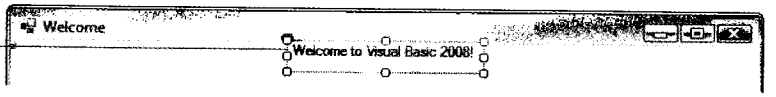


Hình 3.18 `Label` sau khi thay đổi thuộc tính `Text`.



Hình 3.19 `Label` sau khi thay đổi kích thước.

4. ***Căn chỉnh lại Label***. Kéo `Label` lên phía trên của `Form`. Bạn có thể đưa `Label` ra giữa bằng cách chọn **Format > Center In Form > Horizontally** (Hình 3.20).

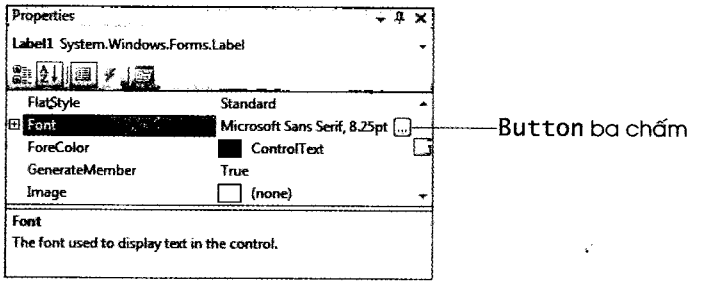


Hình 3.20 Label đã được căn giữa.

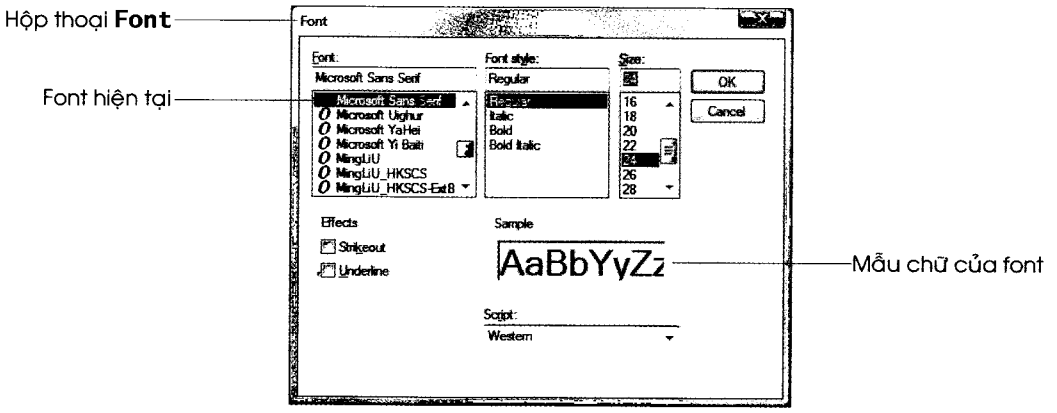
Mẹo thiết kế giao diện

Sử dụng font Segoe UI 9pt để nội dung văn bản hiển thị trên điều khiển dễ đọc hơn.

5. ***Thiết lập font cho Label***. Nhấn chuột vào giá trị của thuộc tính Font để làm hiển thị Button ba chấm (Hình 3.21). Nhấn chuột vào Button ba chấm để hiển thị hộp thoại **Font** (Hình 3.22). Dấu ba chấm là quy ước khi có hộp thoại giúp bạn thiết lập giá trị cho thuộc tính. Trong hộp thoại này, bạn có thể chọn tên font (**Segoe UI**, **Times New Roman**...), kiểu font (**Regular**, **Italic**...) và kích cỡ font (**16**, **18**...) theo đơn vị point (một point bằng 1/72 inch). Nội dung trong Label **Sample** hiển thị font được chọn. Dưới mục **Size:** chọn **24** point. Trong nhóm **Font**, chọn **Segoe UI** và nhấn **OK**. Nếu nội dung của Label không vừa một dòng, nó sẽ tự động nhảy xuống dòng tiếp theo. Sử dụng các điểm neo thay đổi kích thước để mở rộng Label theo chiều dọc để hiển thị nội dung trên hai dòng, sau đó căn chỉnh Label ra giữa như trong *Bước 4*.

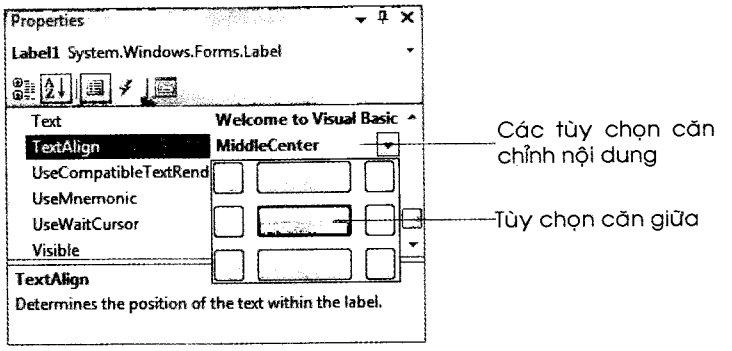


Hình 3.21 Cửa sổ **Properties** hiển thị thuộc tính của Label.



Hình 3.22 Hộp thoại Font để chọn font, kiểu font và kích cỡ.

6. ***Căn chỉnh nội dung của Label***. Để căn chỉnh nội dung trong Label, sử dụng thuộc tính **TextAlign** của Label. Nhấn chuột vào thuộc tính TextAlign để làm hiển thị Button mũi tên xuống. Nhấn chuột vào Button mũi tên xuống để hiển thị lưới 3x3 của Button (Hình 3.23). Vị trí của mỗi Button thể hiện nơi văn bản nằm trong Label. Nhấn chuột vào Button giữa của lưới 3x3 để đưa nội dung vào giữa Label. Giá trị MiddleCenter sẽ được gán cho thuộc tính TextAlign. Bạn cũng có thể thiết lập giá trị của thuộc tính này bằng cách nhấn đúp liên tiếp vào giá trị của thuộc tính ở bên phải tên thuộc tính. Việc này sẽ giúp bạn chuyển qua tất cả các giá trị có thể của thuộc tính TextAlign. Kỹ thuật này sử dụng cho bất kỳ thuộc tính nào cung cấp một tập các tùy chọn thông qua mũi tên xuống ở bên phải giá trị thuộc tính trong cửa sổ **Properties**.



Hình 3.23 Căn giữa nội dung của `Label`.

7. ***Lưu project***. Chọn **File > Save All** để lưu project đã chỉnh sửa.

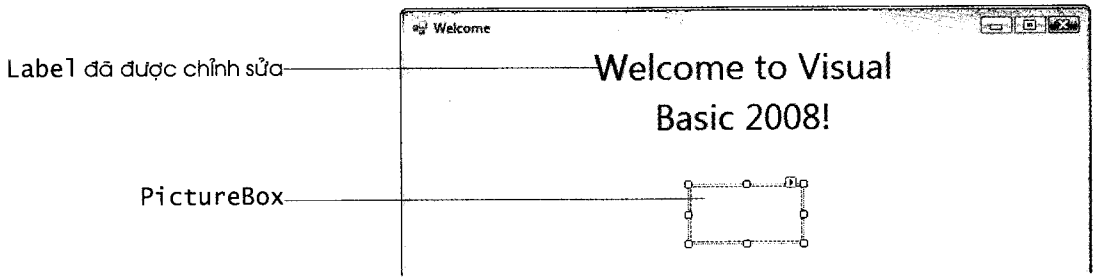
Để hoàn tất ứng dụng Windows Visual Basic đầu tiên, bạn cần phải thêm hình ảnh biểu tượng của công ty và chạy ứng dụng. Chúng tôi sử dụng điều khiển `PictureBox` để thêm hình ảnh vào `Form` trước khi chạy ứng dụng. Phần ngay sau đây sẽ hướng dẫn bạn các bước thêm hình ảnh vào `Form`.

Thêm hình ảnh và chạy ứng dụng Welcome

1. ***Thêm điều khiển `PictureBox` vào `Form`***. `PictureBox` cho phép bạn hiển thị hình ảnh. Để thêm điều khiển `PictureBox` vào `Form`, nhấn đúp chuột vào biểu tượng điều khiển `PictureBox`

   PictureBox

   trên `Toolbox`. Khi `PictureBox` hiện ra, kéo `PictureBox` vào vị trí bên dưới `Label` và kéo ra giữa `Form` (Hình 3.24).



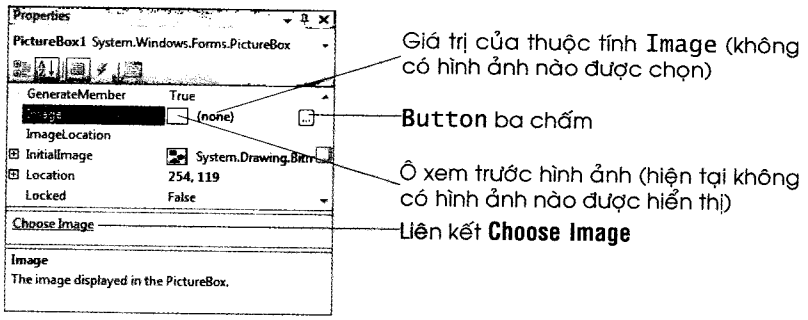
Hình 3.24 Thêm và căn chỉnh `PictureBox`.

2. ***Thiết lập thuộc tính `Image`***. Nhấn chuột vào `PictureBox` để hiển thị thuộc tính của `PictureBox` trong của sổ **Properties**. Tìm thuộc tính **`Image`** để hiển thị hình ảnh có sẵn (nếu có). Nếu chưa có hình ảnh nào được gắn vào thuộc tính **`Image`**, giá trị của thuộc tính sẽ là (none) (Hình 3.25) và một ô trắng rỗng sẽ hiển thị ở bên trái của (none). Bạn có thể sử dụng một số định dạng ảnh phổ biến bao gồm *PNG* (*Portable Network Graphics*), *GIF* (*Graphic Interchange Format*), *JPEG* (*Joint Photographic Experts Group*) và *BMP* (*Windows Bitmap*).

   Trong ứng dụng này, bạn sử dụng hình ảnh có định dạng PNG. Để tạo hình ảnh mới cần phải có phần mềm chỉnh sửa ảnh như Adobe Photoshop (`www.adobe.com`), Corel Paint Shop Pro Photo X2 (`www.corel.com`), Adobe Fireworks (`www.adobe.com`), Microsoft Paint (đi kèm với Windows), Paint.NET (nguồn mở của `www.getpaint.net`) và Picnik (`www.picnik.com`; dịch vụ chỉnh sửa ảnh trực tuyến). Trong cuốn sách này, bạn sẽ không phải tạo hình ảnh; thay vào đó, bạn sẽ được cung cấp các hình ảnh sẽ được sử dụng trong mỗi chương).

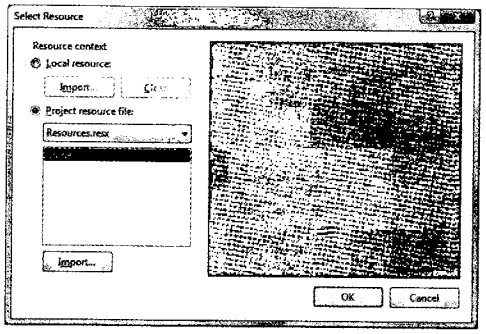
Mẹo thiết kế giao diện

Sử dụng các `PictureBox` để cải thiện giao diện với các hình ảnh không muốn người dùng thay đổi.



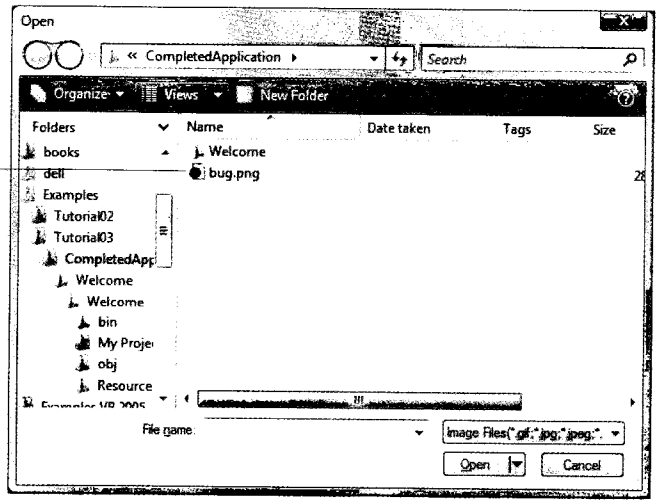
Hình 3.25 Thuộc tính `Image` của `PictureBox`.

3. ***Hiển thị hình ảnh.*** Trong cửa sổ **Properties**, nhấn chuột vào giá trị thuộc tính `Image` của `PictureBox` để làm hiển thị `Button` ba chấm (Hình 3.25). Nhấn chuột vào `Button` ba chấm để hiển thị **hộp thoại Select Resource** (Hình 3.26). Bạn có thể mở hộp thoại bằng cách nhấn vào liên kết **Choose Image** trong phần mô tả thuộc tính (Hình 3.25) hoặc bằng cách nhấn chuột phải vào điều khiển `PictureBox`. Hộp thoại **Select Resource** được sử dụng để đưa (import) file (ví dụ như file hình ảnh) vào ứng dụng. Nhấn chuột vào `Button` **Import...** để tìm hình ảnh cần chèn. Trong ví dụ này, hình ảnh được chèn là `bug.png`. Khi hộp thoại **Open** hiện ra, di chuyển đến thư mục `C:\Examples\Tutorial03\CompletedApplication`. Chọn file `bug.png` và nhấn chuột vào `Button` **Open** (Hình 3.27). Hình ảnh xem trước sẽ được hiển thị trong hộp thoại **Select Resource** (Hình 3.28). Nhấn **OK** để chèn hình ảnh vào ứng dụng. Chú ý rằng `PictureBox` không hiển thị toàn bộ hình ảnh (Hình 3.29). Bạn sẽ giải quyết vấn đề này ở bước sau.

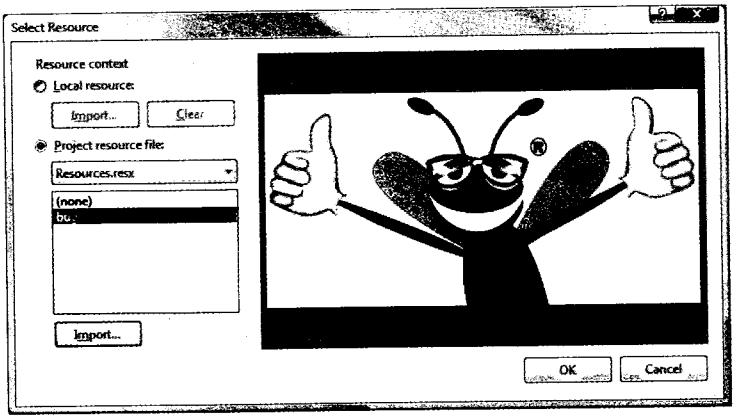


Hình 3.26 Hộp thoại **Select Resource** để chọn hình ảnh cho `PictureBox`.

File `bug.png` (có thể chỉ hiển thị là `bug` tùy thuộc vào chế độ hiển thị phần mở rộng của file trên máy tính của bạn).



Hình 3.27 Hộp thoại **Open** được dùng để chọn hình ảnh cho `PictureBox`.

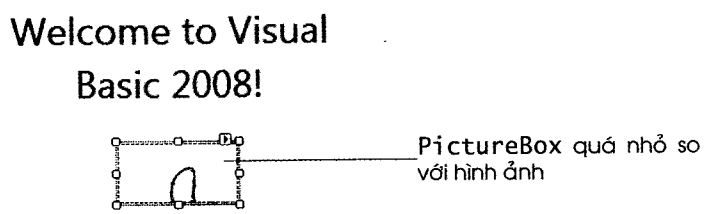


Hình 3.28 Hộp thoại **Select Resource** hiển thị hình ảnh xem trước của hình ảnh được chọn.

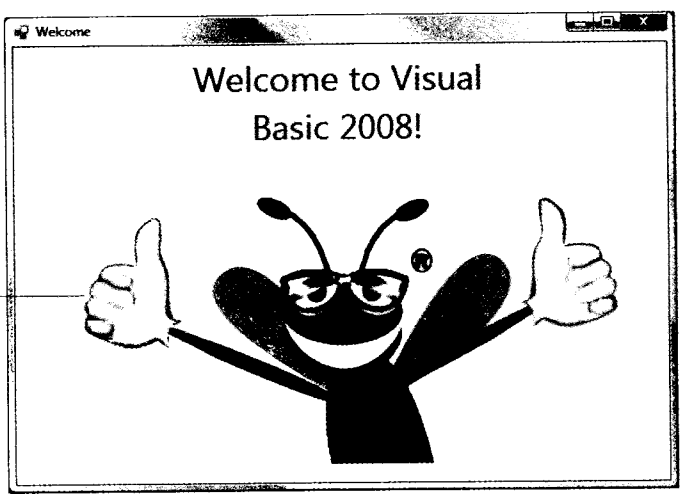
Mẹo thiết kế giao diện

Hình ảnh sẽ vừa với `PictureBox` nếu thay đổi thuộc tính `SizeMode` của `PictureBox` thành `StretchImage`.

4. ***Thay đổi kích thước hình ảnh cho vừa với `PictureBox`.*** Chúng tôi cần hình ảnh phải vừa với `PictureBox`. Thuộc tính **`SizeMode`** của `PictureBox` quy định cách `PictureBox` hiển thị hình ảnh. Để hình ảnh vừa với `PictureBox`, thay đổi thuộc tính của `SizeMode` thành **`StretchImage`**. Lúc này kích thước của hình ảnh (cả chiều rộng và chiều cao) sẽ điều chỉnh bằng với kích thước của `PictureBox`. Để thay đổi kích thước của `PictureBox`, nhấn đúp vào thuộc tính `Size` và nhập `500, 250`. Căn chỉnh hình ảnh ra giữa theo phương ngang bằng cách nhấn vào `PictureBox` và chọn **Format > Center in Form > Horizontally**. `Form` lúc này sẽ trông giống như trong Hình 3.30. [*Lưu ý*: Bạn có thể sẽ cần di chuyển hình ảnh lên hoặc xuống một chút để `Form` của bạn trông giống như Hình 3.30. Để di chuyển hình ảnh lên xuống, bạn chỉ cần nhấn chuột vào `PictureBox` sau đó sử dụng phím mũi tên lên hoặc mũi tên xuống tương ứng.]

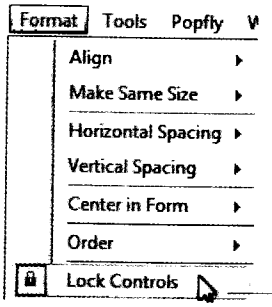


Hình 3.29 Hình ảnh mới được thêm vào.



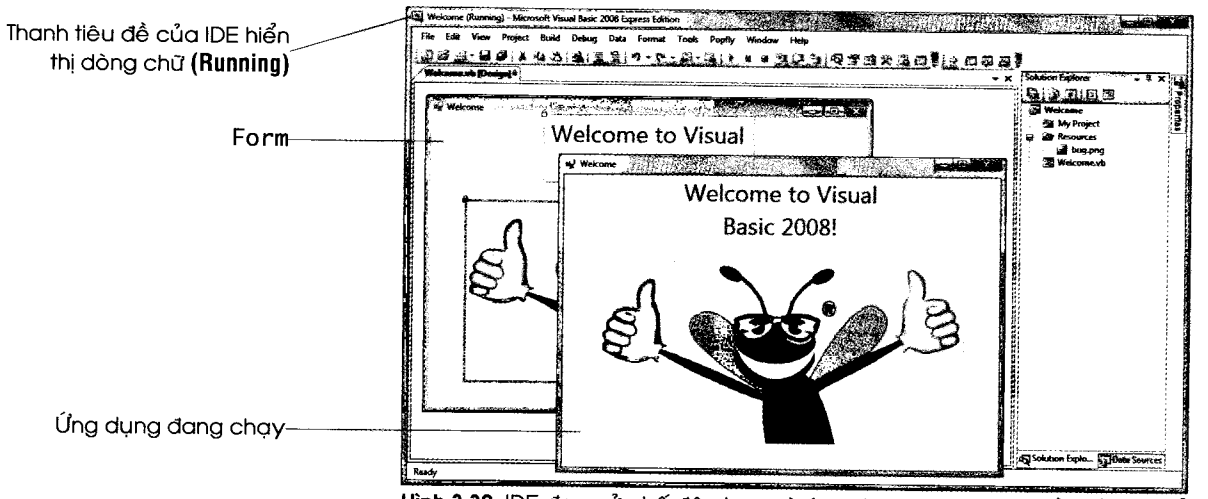
Hình 3.30 `PictureBox` hiển thị hình ảnh.

5. ***Khóa các điều khiển trên Form.*** Lập trình viên thường hay vô tình làm thay đổi kích thước và vị trí của điều khiển trên `Form`. Để đảm bảo rằng điều khiển giữ nguyên vị trí, chọn **Format > Lock Controls** (Hình 3.31). Thao tác này sẽ khóa tất cả các điều khiển trên `Form`. Bạn có thể khóa một điều khiển cụ thể bằng cách thiết lập thuộc tính `Locked` của điều khiển cần khóa thành `True`. Bạn có thể tìm hiểu về các tính năng khác bằng cách nhấn chuột phải vào `Form` trong chế độ **Design**.



Hình 3.31 Khóa các điều khiển bằng menu **Format**.

6. ***Lưu project.*** Chọn **File > Save All** để lưu project bạn đã chỉnh sửa. Bạn nên thường xuyên lưu file vào thư mục `C:\SimplyVB2008`. Tuy nhiên chú ý rằng không nhất thiết phải lưu file của project nếu bạn có ý định chạy ứng dụng. Khi ứng dụng Visual Basic được chạy trong IDE, file project sẽ tự động được lưu.
7. ***Chạy ứng dụng.*** Dòng chữ **Welcome.vb [Design]** ở trên cùng vùng thiết kế thể hiện rằng chúng ta đang làm việc ở **chế độ thiết kế (design mode)**. Ở chế độ thiết kế, ứng dụng không chạy và bạn sẽ có thể truy cập đến tất cả các cửa ổ của IDE (như **Toolbox**, **Properties**), menu và toolbar. Ở **chế độ chạy (run mode)**, ứng dụng sẽ chạy và bạn chỉ có thể tương tác với một vài tính năng của IDE. Các tính năng không sử dụng được sẽ bị vô hiệu hóa (chuyển thành màu xám). Chọn **Debug > Start Debugging** để chạy ứng dụng. Trong Hình 3.22, IDE đang ở chế độ chạy. Chú ý rằng có rất nhiều biểu tượng và menu trên toolbar bị vô hiệu hóa.
8. ***Đóng ứng dụng.*** Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng, ( ). IDE lúc này sẽ được chuyển sang chế độ thiết kế.
9. ***Đóng IDE.*** Đóng IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.



Hình 3.32 IDE đang ở chế độ chạy và ứng dụng đang chạy nằm ở cửa sổ trên cùng.

TỰ ÔN TẬP

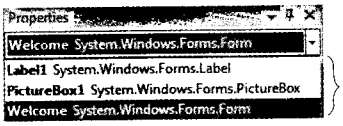
1. Thuộc tính_____của Form quy định nội dung hiển thị trên thanh tiêu đề của Form.
   a) Title
   b) Text
   c) (Name)
   d) Name
2. Thuộc tính_____quy định cách nội dung được căn chỉnh trong Label.
   a) Alignment
   b) AlignText
   c) Align
   d) TextAlign

Đáp án: 1) b. 2) d.

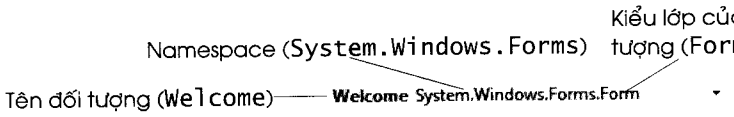
3.3 Các đối tượng được sử dụng trong ứng dụng Welcome

Trong Chương 1 và 2, bạn đã biết rằng điều khiển là thành phần phần mềm có thể tái sử dụng được gọi là đối tượng. Ứng dụng **Welcome** sử dụng đối tượng Form, đối tượng Label và đối tượng PictureBox để tạo giao diện hiển thị đoạn văn bản và hình ảnh. Mỗi đối tượng là một thể hiện của một lớp được định nghĩa trong .NET Framework Class Library. Đối tượng Form được tạo bằng Visual Basic IDE. Các đối tượng Label và PictureBox được tạo khi nhấn đúp chuột vào biểu tượng trên **Toolbox**.

Chúng tôi sử dụng cửa sổ **Properties** để thiết lập thuộc tính cho mỗi đối tượng. Nhớ lại rằng ComboBox nằm ở trên cùng của cửa sổ **Properties** (còn gọi là hộp chọn đối tượng) hiển thị tên và kiểu lớp của Form và đối tượng điều khiển đang được chọn (Hình 3.33). Trong Hình 3.34, hộp chọn đối tượng hiển thị tên (Welcome) và tên lớp (Form) của đối tượng Form. Trong .NET Framework Class Library, các lớp được tổ chức thành một dạng giống như các thư mục gọi là **namespace**. Kiểu lớp được sử dụng trong ứng dụng này có namespace là System.Windows.Forms. Namespace này chứa các lớp điều khiển và lớp Form. Bạn sẽ được tìm hiểu về những namespace khác trong các chương sau.



Hình 3.33 Hộp chọn đối tượng được mở rộng để hiển thị tất cả các đối tượng của ứng dụng **Welcome**.



Hình 3.34 Tên và lớp của một đối tượng được hiển thị trên hộp chọn đối tượng trong cửa sổ **Properties**.

TỰ ÔN TẬP

1. ComboBox nằm ở trên cùng của cửa sổ **Properties** là_____.
   a) hộp chọn đối tượng
   b) hộp điều khiển
   c) hộp đối tượng điều khiển
   d) hộp thành phần
2. .NET Framework Class Library tổ chức các lớp thành các_____.
   a) tập hợp
   b) hộp tên
   c) namespace
   d) không gian lớp

Đáp án: 1) a. 2) c.

3.4 Tổng kết

Chương này giới thiệu về lập trình trực quan trong Visual Basic. Bạn đã biết rằng lập trình trực quan giúp bạn thiết kế và tạo thành phần giao diện người dùng đồ họa cho ứng dụng nhanh chóng và dễ dàng bằng cách kéo và thả điều khiển lên `Form`.

Trong quá trình tạo ứng dụng **Welcome**, bạn đã sử dụng cửa sổ **Properties** để thiết lập dòng tiêu đề, kích thước (chiều rộng và chiều cao), màu nền của `Form` bằng các thuộc tính `Text`, `Size` và `BackColor`. Bạn đã biết rằng `Label` là điều khiển hiển thị văn bản và `PictureBox` là điều khiển hiển thị hình ảnh. Bạn hiển thị văn bản trong `Label` bằng cách thiết lập thuộc tính `Text` và `TextAlign` và hiển thị hình ảnh bằng cách thiết lập các thuộc tính `Image` và `SizeMode` của điều khiển `PictureBox`.

Bạn cũng đã xem xét mối quan hệ giữa điều khiển và lớp. Bạn cũng đã biết rằng lớp trong .NET Framework Class Library được gom thành các nhóm được tổ chức giống như thư mục gọi là namespace và điều khiển là thể hiện (hay đối tượng) của lớp .NET Framework Class Library. Những lớp trong .NET Framework Class Library được sử dụng ở chương này (`Form`, `Label` và `PictureBox`) thuộc về namespace `System.Windows.Forms`. Bạn đã sử dụng hộp chọn đối tượng từ cửa sổ **Properties** để xem tên, namespace và kiểu lớp của đối tượng.

Trong chương sau, bạn sẽ tiếp tục tìm hiểu về lập trình trực quan. Cụ thể hơn, bạn sẽ tạo ứng dụng với điều khiển được thiết kế để lấy dữ liệu do người dùng nhập vào.

TỔNG KẾT KỸ NĂNG

Tạo giao diện nhanh chóng và hiệu quả

- Sử dụng kỹ thuật lập trình trực quan.

Thêm điều khiển vào `Form`

- Nhấn đúp chuột vào điều khiển trên **Toolbox** để đặt điều khiển lên góc trên bên trái của `Form` hoặc kéo điều khiển từ **Toolbox** lên `Form`.

Căn chỉnh điều khiển

- Sử dụng lệnh trên menu **Format.**

Thay đổi kích thước của `Form` hoặc điều khiển bằng điểm neo thay đổi kích thước

- Nhấn chuột và kéo một trong các điểm neo thay đổi kích thước của đối tượng.

Thiết lập kích thước của `Form` hoặc điều khiển bằng thuộc tính `Size`

- Sử dụng cửa sổ **Properties** để nhập chiều rộng và chiều cao của `Form` hoặc điều khiển vào trường `Size`.

Thiết lập chiều rộng và chiều cao của `Form` hoặc điều khiển

- Sử dụng cửa sổ **Properties** để nhập giá trị cho thuộc tính `Width` và `Height` (hoặc sử dụng trường thuộc tính `Size`).

Thiết lập màu nền cho `Form`

- Sử dụng cửa sổ **Properties** để thiết lập giá trị cho thuộc tính `BackColor` của `Form`.

Thêm điều khiển `Label` vào `Form`

- Nhấn đúp chuột vào điều khiển `Label` trên **Toolbox** để đặt điều khiển ở góc trên bên trái của `Form` hoặc kéo `Label` từ **Toolbox** lên trên `Form`.

Thiết lập thuộc tính `Text` của `Label`

- Sử dụng cửa sổ **Properties** để thiết lập thuộc tính `Text` của `Label`.

Thiết lập thuộc tính `Font` của `Label`

- Nhấn chuột vào giá trị của thuộc tính `Font` trong cửa sổ **Properties** để hiển thị `Button` ba chấm bên cạnh. Nhấn chuột vào `Button` ba chấm để hiển thị hộp thoại **Font**. Thay đổi tên, kiểu và kích thước font cho nội dung văn bản trên `Label`.

Căn chỉnh văn bản trong Label

- Sử dụng cửa sổ **Properties** để thiết lập thuộc tính `TextAlign` của `Label`.

Thay đổi kích thước của Label

- Sử dụng cửa sổ **Properties** để thiết lập thuộc tính `AutoSize` thành `False`, sau đó sử dụng điểm neo thay đổi kích thước trong `Form` Designer.

Thêm một hình ảnh vào Form

- Sử dụng điều khiển `PictureBox` để hiển thị hình ảnh. Trong cửa sổ **Properties**, nhấn chuột vào `Button` ba chấm nằm bên cạnh giá trị thuộc tính `Image` của `PictureBox` hoặc nhấn chuột vào liên kết **Choose Image** trong phần mô tả thuộc tính để tìm hình muốn chèn trong hộp thoại **Select Resource**.
- Thay đổi kích thước của hình ảnh cho vừa với kích thước của `PictureBox` bằng cách thiết lập giá trị của thuộc tính `SizeMode` thành `StretchImage`.

Hiển thị thuộc tính của Form hoặc điều khiển trong cửa sổ Properties

- Nhấn chuột vào `Form` hoặc điều khiển trên `Form`.

THUẬT NGỮ

bảng màu (palette) - Tập các màu.

chế độ chạy (run mode) - IDE cho biết ứng dụng đang thực thi.

chế độ thiết kế (design mode) - Là chế độ IDE cho phép bạn tạo ứng dụng sử dụng cửa sổ của Visual Studio 2008, toolbar và thanh menu.

điểm neo thay đổi kích thước (sizing handle) - Có dạng hình vuông, khi được kích hoạt có thể sử dụng để thay đổi kích thước `Form` hay điều khiển của `Form`.

giá trị RGB - Sắc độ màu đỏ, xanh da trời và xanh lá cây cần thiết để pha thành một màu nào đấy.

hộp thoại Select Resource - Sử dụng để đưa file (import file) vào ứng dụng.

`Label` - Điều khiển hiển thị đoạn văn bản mà người dùng không thể thay đổi.

lập trình trực quan (visual programming) - Kỹ thuật trong đó Visual Basic tự động viết mã chương trình dựa trên thao tác của bạn (như nhấn, kéo và thả điều khiển).

màu an toàn cho web (web-safe color) - Màu được hiển thị giống nhau trên những máy khác nhau.

namespace - Các lớp trong thư viện .Net Framework Class Library được tổ chức giống như kiểu tổ chức thư mục.

`PictureBox` - Điều khiển hiển thị hình ảnh.

pixel (điểm ảnh) - Điểm nhỏ xíu trên màn hình máy tính chỉ hiển thị một màu.

`StretchImage` - Giá trị thuộc tính `SizeMode` của `PictureBox` dùng để thay đổi kích thước của hình ảnh sao cho vừa khít với `PictureBox`.

thuộc tính `AutoSize` của `Label` - Xác định `Label` có tự động điều chỉnh kích thước cho vừa với nội dung không.

thuộc tính `BackColor` - Chỉ định màu nền của `Form` hay của điều khiển.

thuộc tính `File Name` - Chỉ định tên của file mã nguồn.

thuộc tính `Font` - Chỉ định tên, kiểu và kích thước font của văn bản hiển thị trên `Form` hay trên điều khiển.

thuộc tính `Height` - Thuộc tính này là thành phần của thuộc tính `Size`, cho biết chiều cao của `Form` hay điều khiển được tính bằng pixel.

thuộc tính `Image` - Cho biết tên file của ảnh được hiển thị trên `PictureBox`.

thuộc tính `Locked` - Ngăn không cho di chuyển và thay đổi kích thước điều khiển.

thuộc tính `Size` - Thuộc tính này chỉ định chiều cao và chiều rộng theo pixel của `Form` hay điều khiển.

thuộc tính SizeMode - Thuộc tính này chỉ định cách hình ảnh được hiển thị trên PictureBox.

thuộc tính Text - Chỉ định nội dung văn bản hiển thị bởi Form hay Label.

thuộc tính TextAlign - Chỉ định cách căn chỉnh văn bản trong Label.

thuộc tính Width - Là thành phần của thuộc tính Size, cho biết chiều rộng của Form hay điều khiển được tính bằng pixel.

HƯỚNG DẪN THIẾT KẾ GIAO DIỆN

Thiết kế tổng thể

- Sử dụng màu sắc trong ứng dụng nhưng đừng gây phản cảm đối với người dùng.

Form

- Chọn tiêu đề ngắn, gợi tả. Viết hoa các từ không phải là mạo từ, giới từ, liên từ. Không nên sử dụng các dấu câu.
- Sử dụng font Segoe UI 9pt để văn bản hiển thị trên điều khiển dễ đọc hơn.

Label

- Sử dụng Label để hiển thị văn bản mà bạn không muốn người dùng thay đổi.
- Đảm bảo rằng điều khiển Label đủ lớn để hiển thị nội dung. Bạn có thể thiết lập để thuộc tính AutoSize thành True để tự động thay đổi kích thước Label cho vừa nội dung cần hiển thị hoặc thiết lập AutoSize thành False và thay đổi kích thước của Label một cách thủ công.

PictureBox

- Sử dụng PictureBox để bổ sung hình ảnh mà bạn không muốn người dùng thay đổi vào giao diện.
- Các hình ảnh sẽ vừa vặn với PictureBox nếu bạn thiết lập giá trị của thuộc tính SizeMode thành StretchImage.

ĐIỀU KHIỂN, SỰ KIỆN, THUỘC TÍNH & PHƯƠNG THỨC

Label A Label

Điều khiển này hiển thị nội dung văn bản trên Form mà bạn không muốn người dùng thay đổi.

- ***Trên giao diện khi ứng dụng chạy***

Welcome to Visual
Basic 2008!

- ***Thuộc tính***

  Text - Chỉ định nội dung hiển thị trên Label.

  Font - Chỉ định tên, kiểu, kích thước font của văn bản hiển thị trên Label.

  TextAlign - Chỉ định văn bản được căn chỉnh ra sao trong Label.

  AutoSize - Cho phép tự động thay đổi kích thước của Label cho vừa với nội dung.

PictureBox PictureBox

Điều khiển này hiển thị hình ảnh trên Form.

- ***Trên giao diện khi ứng dụng chạy***

- ***Thuộc tính***

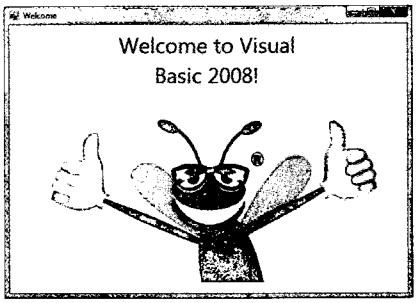
  Image - Chỉ định hình ảnh hiển thị trong PictureBox.

  SizeMode - Chỉ định cách hình ảnh hiển thị trong PictureBox.

  Size - Chỉ định chiều rộng và chiều cao (tính bằng pixel) của PictureBox.

Form Là cửa sổ chính của ứng dụng có giao diện đồ họa.

- ***Trên giao diện khi ứng dụng chạy***



- ***Thuộc tính***

`BackColor` - Chỉ định màu nền của `Form`.

`Font` - Chỉ định tên, kiểu, kích cỡ font của bất kì nội dung văn bản nào hiển thị trên `Form`. Mặc định các điều khiển trên `Form` sẽ sử dụng font chữ này.

`Size` - Chỉ định chiều rộng và chiều cao (tính bằng pixel) của `Form`.

`Text` - Chỉ định nội dung văn bản hiển thị trên thanh tiêu đề của `Form`.

CÂU HỎI TRẮC NGHIỆM

3.1 Thuộc tính_____chỉ định màu nền của `Form`.

a) `BackColor` b) `BackgroundColor`

c) `RGB` d) `Color`

3.2 Để lưu file của project, chọn_____.

a) **Save > Solution > Save Files** b) **File > Save**

c) **File > Save All** d) **File > Save As...**

3.3 Khi `Button` ba chấm nằm bên phải giá trị thuộc tính **Font** được nhấn,_____ được hiển thị.

a) hộp thoại **Font Property** b) hộp thoại **New Font**

c) hộp thoại **Font Settings** d) hộp thoại **Font**

3.4 Thuộc tính_____của `PictureBox` chứa hình ảnh xem trước của hình ảnh được hiển thị bởi `PictureBox`.

a) `Picture` b) `ImageName`

c) `Image` d) `PictureName`

3.5 Khi thiết lập thuộc tính `BackColor`, tab_____cho phép bạn tạo màu sắc của riêng mình.

a) **Custom** b) **Web**

c) **System** d) **User**

3.6 Thuộc tính_____của `PictureBox` chỉ định cách hình ảnh sẽ được hiển thị trên `PictureBox`.

a) `Size` b) `Height`

c) `Width` d) `SizeMode`

3.7 Thuộc tính_____chỉ định nội dung văn bản sẽ hiển thị trong điều khiển `Label`.

a) `Caption` b) `Data`

c) `Text` d) `Name`

3.8 Ở chế độ_____, ứng dụng được thực thi.

a) khởi động (start) b) chạy (run)

c) ngắt (break) d) thiết kế (design)

3.9 Lệnh_____ngăn lập trình viên vô tình thay đổi kích cỡ và vị trí của điều khiển của Form.

a) **Lock Controls**
b) **Anchor Controls**
c) **Lock**
d) **Bind Controls**

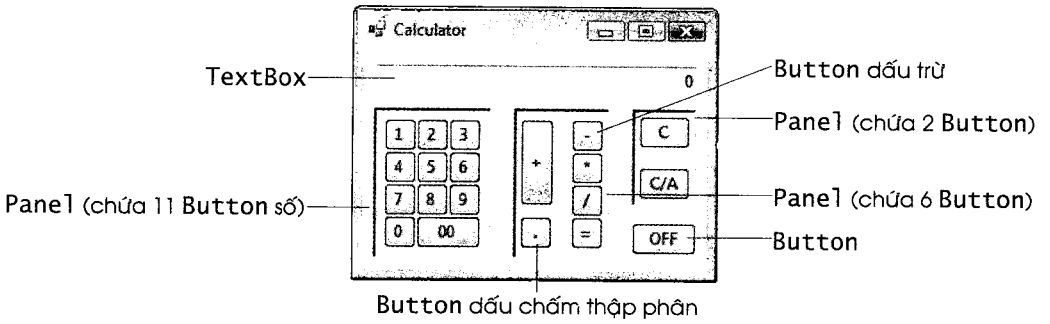
3.10 Pixel là_____.

a) điểm ảnh
b) điều khiển trên **Toolbox**
c) tập các font
d) tập các màu sắc trên tab **Web**

BÀI TẬP

Đối với những bài tập từ 3.11 đến 3.16, với mỗi bài tập bạn phải tạo giao diện. Bạn phải sử dụng kỹ thuật lập trình trực quan được trình bày trong chương này để tạo giao diện. Bạn chỉ mới tạo giao diện nên ứng dụng của bạn vẫn chưa thực sự hoạt động. Ví dụ, giao diện ***Calculator*** *trong bài tập 3.11 sẽ không hoạt động giống như một chiếc máy tính bỏ túi khi nhấn chuột vào các* Button. *Bạn sẽ được tìm hiểu cách làm cho các ứng dụng hoạt động đầy đủ chức năng ở những chương sau. Hãy tạo mỗi ứng dụng trong project riêng. Nếu bạn vô tình nhấn đúp chuột vào một điều khiển trong chế độ* ***Design****, IDE sẽ hiển thị mã nguồn của* Form. *Để quay lại chế độ* ***Design****, chọn* ***View > Designer****.*

3.11 ***(GUI Calculator)*** Tạo giao diện cho máy tính bỏ túi như trong Hình 3.35.



Hình 3.35 Giao diện **Calculator**.

a) ***Tạo project mới.*** Tạo **Windows Forms Application** mới tên là Calculator.

b) ***Đổi tên file Form.*** Đặt tên file Form là Calculator.vb.

c) ***Thay đổi giá trị thuộc tính của Form.*** Thay đổi thuộc tính Text của Form thành Calculator. Thay đổi thuộc tính Font thành Segoe UI 9 pt. Thay đổi thuộc tính Size của Form thành 272, 204. Chú ý rằng Visual Studio sẽ thay đổi kích thước của Form khi bạn đổi kích cỡ font của Form, *nên hãy thay đổi kích cỡ font trước khi thiết lập kích thước Form.*

d) ***Thêm TextBox vào Form.*** Thêm TextBox vào Form bằng cách nhấn đúp vào TextBox trên **Toolbox**. Điều khiển TextBox giúp người dùng có thể nhập dữ liệu vào ứng dụng. Thiết lập thuộc tính Text của TextBox thành 0. Thay đổi thuộc tính Size thành 240, 23. [*Lưu ý*: Bạn không thể thay đổi chiều cao của TextBox một dòng. Bạn sẽ được tìm hiểu cách tạo TextBox nhiều dòng trong Chương 11]. Thiết lập thuộc tính TextAlign thành Right để căn phải nội dung của TextBox. Cuối cùng, thiết lập thuộc tính Location của TextBox thành 8, 16 để đưa điều khiển lên góc trên bên trái của Form.

e) ***Thêm Panel đầu tiên vào Form.*** Điều khiển Panel được dùng để nhóm các điều khiển khác. Nhấn đúp vào biểu tượng Panel (Panel) trong nhóm **Containers** của **Toolbox** để thêm Panel vào Form. Thay đổi thuộc tính BorderStyle của Panel thành Fixed3D để khiến phần bên trong của Panel thụt vào. Thay đổi thuộc tính Size thành 88,112. Cuối

cùng, thiết lập thuộc tính `Location` thành `8,48`. `Panel` này chứa các phím số của máy tính.

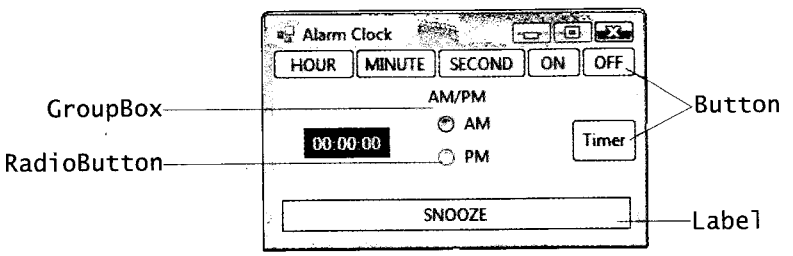
f) ***Thêm Panel thứ hai vào Form.*** Nhấn chuột vào `Form`. Nhấn đúp chuột vào biểu tượng `Panel` trong **Toolbox** để thêm một `Panel` khác vào `Form`. Thay đổi thuộc tính `BorderStyle` của `Panel` thành `Fixed3D`. Thay đổi thuộc tính `Size` thành `72,112`. Cuối cùng, thiết lập thuộc tính `Location` thành `112,48`. `Panel` này chứa các phím phép tính của máy tính.

g) ***Thêm Panel thứ ba (cuối cùng) vào Form.*** Nhấn chuột vào `Form`. Nhấn đúp chuột vào biểu tượng `Panel` trong **Toolbox** để thêm một `Panel` khác vào `Form`. Thay đổi thuộc tính `BorderStyle` của `Panel` thành `Fixed3D`. Thay đổi thuộc tính `Size` thành `48,72`. Cuối cùng thiết lập thuộc tính `Location` thành `200,48`. `Panel` này chứa các phím **C** (Clear - xóa) và **C/A** (Clear All - xóa tất cả).

h) ***Thêm Button vào Form.*** Có 20 `Button` trên máy tính. Để thêm `Button` vào `Panel`, kéo một `Button` (`Button`) từ **Toolbox** và thả nó lên `Panel` tương ứng. Thay đổi thuộc tính `Text` của mỗi `Button` thành phím tương ứng của nó trên bàn phím. Giá trị bạn nhập vào thuộc tính `Text` sẽ hiển thị trên `Button`. Cuối cùng, thay đổi kích thước `Button` bằng thuộc tính `Size`. Mỗi `Button` được gắn nhãn `0`, `9`, `*`, `/`, `-`, `=` và `.` (dấu chấm thập phân) có kích cỡ `24,24`. Các `Button` **00** và **OFF** có kích thước `48,24`. Button `+` có kích thước `24,64`. Các `Button` **C** (clear - xóa) và **C/A** (clear all - xóa hết) có kích thước `38,24`. Để bố trí các `Button` số như trong Hình 3.35, chọn `Button` **1** và thiết lập thuộc tính `Location` của nó thành `6,6` và thuộc tính `Lock` thành `True`. Đặt các `Button` **2** và **3** ở bên phải `Button` **1**. Chọn ba `Button` ở dòng đầu (**1**, **2** và **3**) bằng cách nhấn chuột vào `Button` **1** sau đó giữ phím Shift trong khi chọn các `Button` **2** và **3**. Vị trí `Button` sau sẽ dựa trên `Button` mà bạn đã chọn đầu tiên (trong ví dụ này là `Button` **1**). Sử dụng tùy chọn **Format > Horizontal Spacing > Remove** bỏ đi khoảng trống giữa các `Button` đứng liền kề nhau. Chọn **Format > Align > Middles** để đặt chúng thành một đường thẳng. Tương tự sắp xếp các `Button` **1**, **4**, **7** và **0** thẳng cột bằng cách chọn **Format > Vertical Spacing > Remove** và **Format > Align > Centers**. Bạn có thể kéo thả các `Button` số còn lại vào vị trí - IDE sẽ tự động xếp các `Button` thẳng hàng với các `Button` xung quanh nó. Menu **Format** chứa nhiều tùy chọn hữu ích. [*Lưu ý*: Bạn có thể hiển thị những tùy chọn của menu **Format** trên toolbar Visual Studio bằng cách nhấn chuột phải lên toolbar của IDE và chọn **Layout**].

i) ***Lưu và đóng project.*** Chọn **File > Save All** để lưu sửa đổi. Sau đó chọn **File > Close Project** để đóng project của ứng dụng.

3.12 ***(Giao diện Alarm Clock)*** Tạo giao diện đồng hồ báo thức như Hình 3.36.



Hình 3.36 Giao diện **Alarm Clock.**

a) ***Tạo project mới.*** Tạo **Windows Forms Application** mới và đặt tên là `AlarmClock`.

b) ***Đổi tên file Form.*** Đặt tên file Form là `AlarmClock.vb`.

c) ***Thay đổi giá trị thuộc tính của Form.*** Thay đổi thuộc tính `Font` của `Form` thành `Segoe UI 9pt`. Thay đổi thuộc tính `Text` thành `AlarmClock`. Thay đổi thuộc tính `Size` của `Form` thành `281,176`. Hãy nhớ thay đổi kích thước của thuộc tính `Font` trước khi thay đổi thuộc tính `Size` của `Form`.

d) ***Thêm Button vào Form.*** Thêm sáu `Button` vào `Form`. Thay đổi thuộc tính `Text` của mỗi `Button` thành những giá trị tương ứng. Thay đổi thuộc tính `Size` của các `Button` **Hour**, **Minute** thành `60,23`. Thay đổi `Size` của `Button` **Second** thành `65,23`. Thay đổi `Size` của `Button` **ON**, **OFF** thành `40,23`. `Button` **Timer** có kích thước là `48,32`. Sử dụng tùy chọn **Format > Horizontal Spacing > Remove** để bố trí các `Button` trên vào dòng đầu tiên như trên Hình 3.36.

e) ***Thêm Label vào Form.*** Thêm `Label` vào `Form`. Thay đổi thuộc tính `Text` thành **SNOOZE**. Thiết lập thuộc tính `AutoSize` thành `False` và `Size` là `254,23`. Thiết lập thuộc tính `TextAlign` của `Label` thành `MiddleCenter`. Cuối cùng vẽ một đường biên thể hiện cạnh của `Label` **SNOOZE**, thay đổi thuộc tính `BorderStyle` của `Label` **SNOOZE** thành `FixedSingle`.

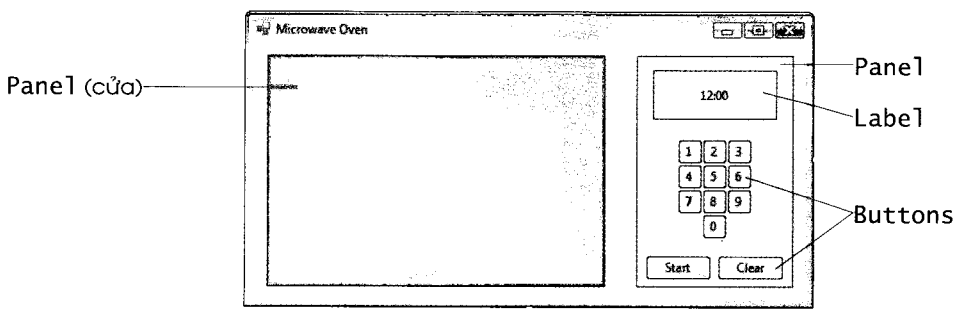
f) ***Thêm GroupBox vào Form.*** `GroupBox` giống như `Panel` ngoại trừ `GroupBox` có thể hiển thị tiêu đề. Để thêm `GroupBox` vào `Form`, nhấn đúp chuột vào điều khiển `GroupBox` (GroupBox) trên tab **Container** của **Toolbox**. Thay đổi thuộc tính `Text` thành `AM/PM` và thiết lập thuộc tính `Size` thành `72,72`. Để đặt `GroupBox` vào đúng vị trí trên `Form`, thiết lập `Location` của `GroupBox` thành `104,29`.

g) ***Thêm RadioButton AM/PM vào GroupBox.*** Thêm hai `RadioButton` vào `Form` bằng cách kéo điều khiển `RadioButton` (RadioButton) từ **Toolbox** vào `Form` hai lần. Thay đổi thuộc tính `Text` của một `RadioButton` thành `AM` và `RadioButton` còn lại thành `PM`. Sau đó bố trí các `RadioButton` như trên Hình 3.36 bằng cách thiết lập `Location` của `RadioButton` **AM** thành `16,16` và của `RadioButton` **PM** thành `16,40`. Thiết lập thuộc tính `AutoSize` thành `False` và thiết lập kích thước của `Size` thành `48,24`.

h) ***Thêm Label hiển thị thời gian vào Form.*** Thêm một `Label` vào `Form` và thay đổi thuộc tính `Text` của `Label` thành `00:00:00`. Thay đổi thuộc tính `BorderStyle` thành `Fixed3D` và `BackColor` thành `Black`. Thiết lập thuộc tính `AutoSize` thành `False` và `Size` thành `64,23`. Sử dụng thuộc tính `Font` để làm cho thời gian hiển thị đậm hơn. Thay đổi `ForeColor` thành `Silver` (nằm trong tab **Web**) để thời gian tương phản với nền đen. Thiết lập `TextAlign` thành `MiddleCenter` để căn nội dung ra giữa `Label`. Sắp xếp `Label` như trong Hình 3.36.

i) ***Lưu và đóng project.*** Chọn **File > Save All** để lưu thay đổi. Sau đó chọn **File > Close Project** để đóng project của ứng dụng.

3.13 ***(Giao diện Microwave Oven)*** Tạo giao diện giả lập lò vi sóng như Hình 3.37.



Hình 3.37 Giao diện **Microwave Oven**.

a) ***Tạo project mới.*** Tạo **Windows Forms Application** mới và đặt tên là `Microwave`.

b) ***Đổi tên file Form.*** Đặt tên file `Form` là `Microwave.vb`.

c) ***Thay đổi giá trị thuộc tính của Form.*** Thay đổi thuộc tính `Font` của `Form` thành `Segoe UI 9pt` và thuộc tính `Text` thành `Microwave Oven`. Thay đổi thuộc tính `Size` thành `552,288`.

d) ***Thêm cửa lò vi sóng.*** Thêm `Panel` vào `Form` bằng cách nhấn đúp chuột vào điều khiển `Panel` (Panel) trong **Toolbox**. Chọn `Panel` và thay đổi thuộc tính `BackColor` thành `Silver` (nằm trong tab **Web**) trong cửa sổ **Properties**. Sau đó thay đổi thuộc tính `Size` thành `328,224`. Tiếp theo thay đổi thuộc tính `BorderStyle` thành `FixedSingle`. Bố trí `Panel` như trong Hình 3.37 bằng cách sử dụng mũi tên bốn chiều (⊕) ở góc trên bên trái của `Panel` được chọn.

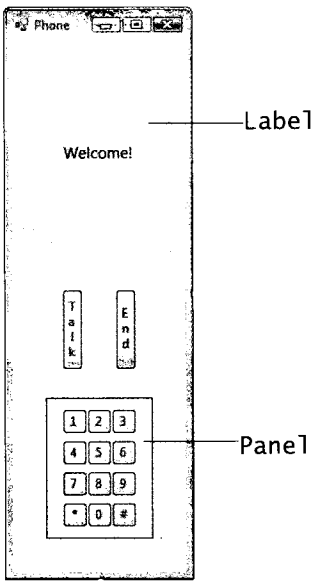
e) ***Thêm một Panel khác.*** Thêm `Panel` khác và thay đổi thuộc tính `Size` của `Panel` thành `152,224` và `BorderStyle` thành `FixedSingle`. Đặt `Panel` lên bên phải `Panel` cửa như trong Hình 3.37.

f) ***Thêm đồng hồ cho lò vi sóng.*** Thêm một `Label` vào `Panel` bên phải bằng cách nhấn chuột một lần vào `Label` trên **Toolbox**, sau đó nhấn chuột vào bên trong `Panel` bên phải. Thay đổi thuộc tính `Text` của `Label` thành `12:00`, `BorderStyle` thành `FixedSingle`, `AutoSize` thành `False` và Size thành `120,48`. Thay đổi `TextAlign` thành `MiddleCenter`. Bố trí vị trí đồng hồ như trong Hình 3.37.

g) ***Thêm bàn phím cho lò vi sóng.*** Đặt `Button` lên `Panel` bên phải bằng cách nhấn chuột vào điều khiển `Button` trên `Toolbox`, sau đó nhấn chuột vào bên trong `Panel` bên phải. Thay đổi thuộc tính `Text` thành `1` và `Size` thành `24,24`. Làm tương tự với chín `Button` còn lại, thay đổi thuộc tính `Text` cho mỗi `Button`. Sau đó thêm các `Button` **Start** và **Clear** với `Size` là `64,24`. Đừng quên thiết lập thuộc tính `Text` cho mỗi `Button` này. Cuối cùng sắp xếp các `Button` như trong Hình 3.37. `Button` **1** nằm ở vị trí 39, 80 và `Button` **Start** nằm ở vị trí `8,192`.

h) ***Lưu và đóng project.*** Chọn **File > Save All** để lưu các thay đổi của bạn. Sau đó chọn **File > Close Project** để đóng project cho ứng dụng này.

3.14 ***(Giao diện Cell Phone)*** Tạo giao diện cho điện thoại di động như trong Hình 3.38.



Hình 3.38 Giao diện **Cell Phone.**

a) ***Tạo project mới.*** Tạo **ứng dụng Windows Forms** mới và đặt tên là Phone.

b) ***Đổi tên file Form.*** Đặt tên file Form là Phone.vb.

c) ***Thay đổi giá trị thuộc tính của Form.*** Thay đổi thuộc tính Font của Form thành Segoe UI 9pt. Thay đổi thuộc tính Text thành Phone và Size thành 184,558.

d) ***Thêm Label hiển thị.*** Thêm Label vào Form. Thay đổi thuộc tính BackColor của Label thành NavajoWhite (trong bảng màu của tab **Web**), Text thành Welcome!, AutoSize thành False và Size thành 156,210. Thay đổi thuộc tính TextAlign thành MiddleCenter. Sau đó bố trí Label như trong Hình 3.38.

e) ***Thêm Panel bàn phím.*** Thêm Panel vào Form. Thay đổi thuộc tính BorderStyle thành FixedSingle và Size thành 104,136.

f) ***Thêm Button để mô phỏng bàn phím.*** Thêm Button bàn phím (tổng cộng 12 Button) vào Form. Mỗi Button trên bàn phím số nên để kích thước là 24,24 và được đặt lên Panel. Thay đổi thuộc tính Text của mỗi Button thành các số từ 0-9, dấu thăng (#) và dấu sao (*). Sau đó thêm hai Button cuối cùng lần lượt có thuộc Text là Talk và End. Thay đổi Size của mỗi Button là 20,80. Chú ý rằng nếu Size của Button quá nhỏ sẽ khiến cho Text được sắp xếp dạng cột dọc.

g) ***Bố trí điều khiển.*** Bố trí toàn bộ các điều khiển để GUI của bạn trông giống như trong Hình 3.38.

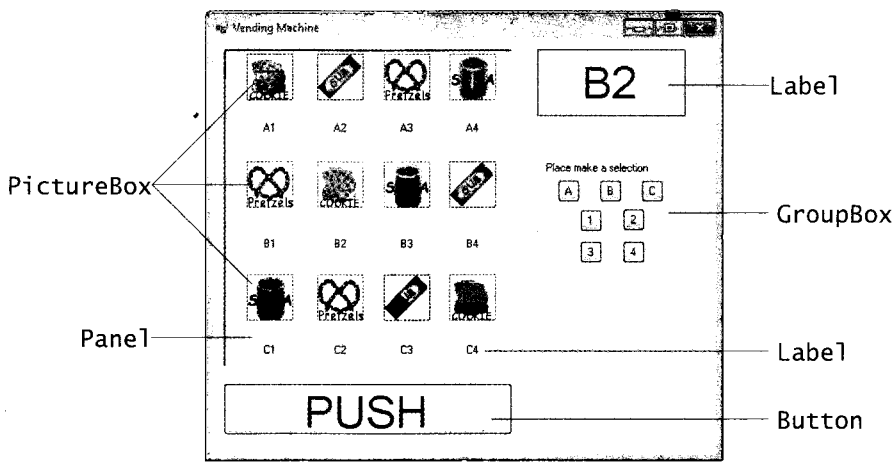
h) ***Lưu và đóng project.*** Chọn **File > Save All** để lưu thay đổi của bạn. Sau đó chọn **File > Close Project** để đóng project của ứng dụng.

3.15 ***(Giao diện Vending Machine)*** Tạo giao diện cho máy bán lẻ tự động như trong Hình 3.39.

a) ***Tạo project mới.*** Tạo **ứng dụng Windows Forms** và đặt tên là VendingMachine.

b) ***Đổi tên file Form.*** Đặt tên file Form là VendingMachine.vb.

c) ***Thay đổi giá trị thuộc tính của Form.*** Thiết lập thuộc tính Font của Form thành Segoe UI 9pt, thuộc tính Text thành Vending Machine và Size thành 560,488.



Hình 3.39 Giao diện **Vending Machine.**

d) ***Thêm Panel hiển thị các loại mặt hàng.*** Thêm Panel vào Form và thay đổi Size của Form thành 312,344 và BorderStyle thành Fixed3D. Thêm PictureBox vào Panel rồi thay đổi kích thước thành 50,50. Sau đó thiết lập thuộc tính Image bằng cách nhấn vào liên kết **Choose Image** và chọn một file từ thư mục C:\Examples\Tutorial03\ExerciseImages\VendingMachine. Tương tự với mười một PictureBox còn lại.

e) ***Thêm Label cho mỗi mặt hàng bán lẻ.*** Thêm `Label` dưới `PictureBox` đầu tiên. Thay đổi thuộc tính `Text` của `Label` thành `A1`, thuộc tính `TextAlign` thành `MiddleCenter`, `AutoSize` thành `False` và `Size` thành `50,16`. Bố trí `Label` giống như trong Hình 3.39. Tương tự đối với những `Label` từ `A2` đến `C4` (11 `Label`).

f) ***Tạo cửa máy bán lẻ (là một Button).*** Thêm một `Button` vào `Form` bằng cách kéo điều khiển `Button` từ **Toolbox** và thả vào bên dưới `Panel`. Thay đổi thuộc tính `Text` của `Button` thành `PUSH`, `Font Size` thành `36`, `Size` thành `312,56`. Sau đó bố trí `Button` lên `Form` như trên Hình 3.39.

g) ***Thêm Label hiển thị lựa chọn.*** Thêm `Label` vào `Form` và thay đổi thuộc tính `Text` thành `B2`, `BorderStyle` thành `FixedSingle`, `Font Size` thành `36`, `TextAlign` thành `MiddleCenter`, `AutoSize` thành `False` và `Size` thành `160,72`.

h) ***Nhóm các Button nhập liệu.*** Thêm `GroupBox` vào bên dưới `Label`, thay đổi thuộc tính `Text` thành `Please make a selection` và `Size` thành `160,136`.

i) ***Thêm Button nhập liệu.*** Cuối cùng thêm `Button` vào `GroupBox`. Thay đổi `Size` của bảy `Button` thành `24,24`. Sau đó thay đổi thuộc tính `Text` của `Button` để mỗi `Button` chứa lần lượt các giá trị `A`, `B`, `C`, `1`, `2`, `3` hoặc `4` như trong Hình 3.39. Sau khi hoàn tất, bố trí các điều khiển trên `Form` cho giống hình.

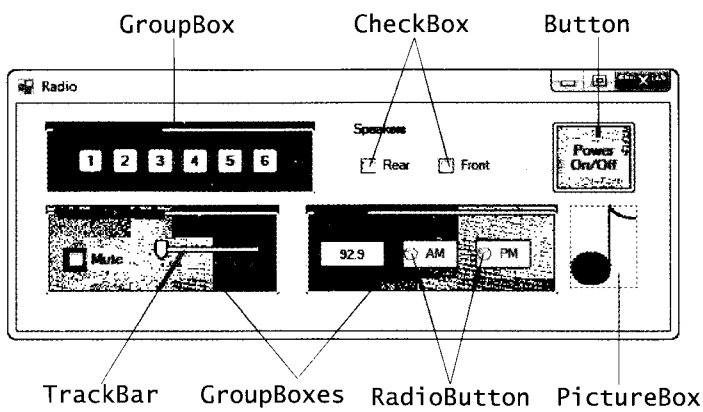
j) ***Lưu và đóng project.*** Chọn **File > Save All** để lưu thay đổi. Sau đó chọn **File > Close Project** để đóng project của ứng dụng.

Bài tập nâng cao ▶ **3.16** ***(Giao diện Radio)*** Tạo giao diện cho radio như Hình 3.40. [*Lưu ý*: Tất cả màu sắc trong bài tập này đều nằm trong bảng màu **Web**]. Trong bài tập này, bạn sẽ tự tạo giao diện theo ý bạn. Hãy thoải mái lựa chọn những thuộc tính khác nhau cho mỗi điều khiển. Hình ảnh hiển thị trong `PictureBox` có thể sử dụng file (`MusicNote.gif`) nằm trong thư mục `C:\Examples\Tutorial03\ExerciseImages\Radio`.

a) ***Tạo project mới.*** Tạo **Windows Forms Application** mới và đặt tên là `Radio`.

b) ***Đổi tên file Form.*** Đặt tên file `Form` là `Radio.vb`.

c) ***Thao tác với thuộc tính của Form.*** Thay đổi thuộc tính `Font` của `Form` thành `Segoe UI 9pt`, thuộc tính `Text` thành `Radio` và `Size` thành `576,240`. Thiết lập `BackColor` thành `PeachPuff`.



Hình 3.40 Giao diện **Radio**.

d) ***Thêm GroupBox Pre-set Stations và các Button.*** Thêm `GroupBox` vào `Form`. Thiết lập `Size` thành `232,64`, `Text` thành `Pre-set Stations`, `ForeColor` thành `Black` và `BackColor` thành `RosyBrown`. Thay đổi `Font` thành đậm. Cuối cùng, thiết lập `Location` của `GroupBox` thành

24,16. Thêm sáu Button vào GroupBox. Thiết lập BackColor của mỗi Button thành PeachPuff và Size là 24, 24. Thay đổi thuộc tính Text của Button lần lượt thành 1, 2, 3, 4, 5, 6.

e) ***Thêm GroupBox Speakers và các CheckBox***. Thêm GroupBox vào Form. Thiết lập Size thành 160,64, Text thành Speakers và ForeColor thành Black. Thiết lập Location thành 280,16. Thêm hai CheckBox vào Form. Thiết lập thuộc tính AutoSize của mỗi CheckBox thành False và Size thành 56,24. Thiết lập thuộc tính Text cho mỗi CheckBox thành Rear và Front.

f) ***Thêm Button Power On/Off***. Thêm Button vào Form. Thiết lập thuộc tính của Button thành Power On/Off, BackColor thành RosyBrown, ForeColor thành Black và Size thành 72,64. Thay đổi kiểu Font thành Bold.

g) ***Thêm GroupBox Volume Control, CheckBox Mute và TrackBar Volume***. Thêm GroupBox vào Form. Thiết lập thuộc tính Text của GroupBox thành Volume Control, BackColor thành RosyBrown, ForeColor thành Black và Size thành 200,80. Thiết lập kiểu Font thành Bold. Thêm CheckBox vào GroupBox. Thiết lập thuộc tính Text của CheckBox thành Mute và Size là 56,19. Thêm TrackBar (TrackBar) nằm trong nhóm **All Windows Forms** lên GroupBox.

h) ***Thêm GroupBox Tuning, Label hiển thị kênh radio và RadioButton AM/FM***. Thêm GroupBox vào Form. Thiết lập thuộc tính Text cho GroupBox thành Tuning, ForeColor thành Black và BackColor thành RosyBrown. Thiết lập kiểu Font thành Bold và Size thành 216,80. Thêm Label vào GroupBox. Thiết lập BackColor thành PeachPuff, BorderStyle thành FixedSingle, TextAlign thành MiddleCenter và Size thành 56,24. Thiết lập Text thành 92.9. Bố trí Label như trong Hình 3.40. Thêm hai RadioButton vào GroupBox. Thay đổi BackColor thành PeachPuff và thay đổi Size thành 45,24. Thiết lập Text của một Button là AM và Label kia là FM.

i) ***Thêm hình ảnh***. Thêm PictureBox vào Form. Thiết lập BackColor thành Transparent, SizeMode thành StretchImage và Size thành 56,72. Thiết lập thuộc tính Image thành C:\Examples\Tutorial03\Exercise03\ExerciseImages\Radio\MusicNote.gif.

j) ***Lưu và đóng project***. Chọn **File > Save All** để lưu các thay đổi. Sau đó chọn **File > Close Project** để đóng project của ứng dụng.

4 CHƯƠNG

Mục tiêu

Trong chương này, bạn sẽ tìm hiểu để:

- Lập trình trực quan, áp dụng hướng dẫn thiết kế giao diện.
- Đổi tên cho **Form**.
- Thêm **TextBox** và **Button** vào **Form**.
- Sử dụng thuộc tính **BorderStyle** cho **Label**.

Nội dung chính

Thiết kế ứng dụng Inventory

Giới thiệu TextBox và Button

Chương này giới thiệu về thiết kế giao diện người dùng đồ họa. Bạn sẽ thiết kế giao diện đồ họa cho ứng dụng kiểm kê đơn giản. Qua mỗi bước, bạn nâng cấp giao diện của ứng dụng bằng cách thêm vào các điều khiển. Bạn thiết kế `Form` để đặt `Label`, `TextBox` và `Button` vào đó. Bạn sẽ tìm hiểu các thuộc tính mới của `Label` và `TextBox`. Ở phần cuối của chương, bạn sẽ tìm thấy danh sách những hướng dẫn thiết kế giao diện giúp bạn tạo ra giao diện người dùng đồ họa đẹp và dễ sử dụng.

4.1 Chạy thử ứng dụng Inventory

Trong chương này, bạn sẽ tạo ứng dụng tính toán số lượng sách giáo khoa mà kho sách của trường đại học đã nhận. Ứng dụng này cần phải đáp ứng những yêu cầu sau đây:

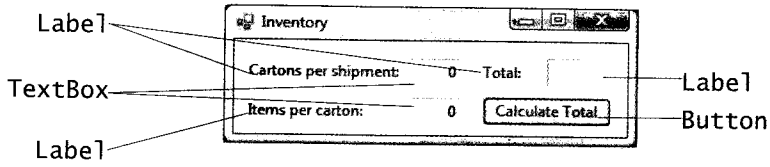
Yêu cầu đối với ứng dụng

Kho sách của trường đại học nhận về các kiện sách giáo khoa. Trong mỗi lô hàng, tất cả các kiện đều chứa cùng một số lượng sách. Quản lý kho muốn sử dụng máy tính để tính toán tổng số sách giáo khoa được chuyển tới kho sách trong mỗi lô hàng. Quản lý nhập số lượng kiện sách đã nhận và số lượng sách giáo khoa trong mỗi kiện sách của mỗi lô hàng, sau đó ứng dụng tính toán tổng số sách giáo khoa trong mỗi lô hàng.

Ứng dụng này thực hiện phép tính đơn giản. Người dùng (quản lý kho) nhập vào `TextBox` số kiện hàng và số sách trong mỗi kiện, sau đó nhấn vào `Button`. Ứng dụng sẽ nhân hai con số với nhau rồi hiển thị kết quả là tổng số sách giáo khoa đã nhận được. Bạn bắt đầu bằng chạy thử một ứng dụng hoàn chỉnh. Sau đó bạn tìm hiểu các tính năng bổ sung của Visual Basic cần thiết để tự xây dựng cho mình một phiên bản của ứng dụng.

Chạy thử ứng dụng Inventory

1. ***Mở ứng dụng hoàn chỉnh***. Mở thư mục `C:\Examples\Tutorial04\CompleteApplication\Inventory` để tìm ứng dụng **Inventory**. Nhấn đúp vào file `Inventory.sln` để mở ứng dụng trong IDE Visual Basic. Tùy thuộc vào cấu hình hệ thống của mình, bạn có thể sẽ không thấy đuôi mở rộng `.sln` của file. Trong trường hợp này, hãy nhấn đúp vào file có tên `Inventory` có chứa biểu tượng file solution .
2. ***Chạy ứng dụng Inventory.*** Chọn **Debug > Start Debugging** để chạy ứng dụng. `Form` **Inventory** xuất hiện giống như Hình 4.1.



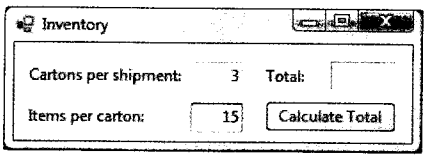
Chú thích hình

- ***Cartons per shipment***: *Số kiện hàng mỗi lô hàng*
- ***Items per carton***: *Số sách trong mỗi kiện hàng*

Hình 4.1 `Form` ứng dụng **Inventory** với dữ liệu mặc định được hiển thị.

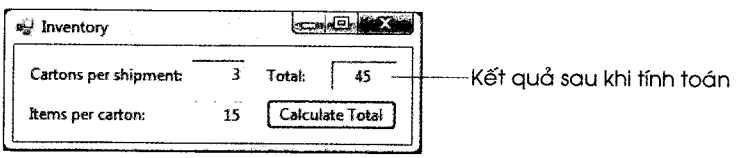
Lưu ý rằng ứng dụng này có hai điều khiển `TextBox` và `Button` mà bạn chưa sử dụng trong ứng dụng **Welcome**. **`TextBox`** là điều khiển cho phép người dùng nhập dữ liệu từ bàn phím và có thể hiển thị dữ liệu cho người dùng. `Button` là điều khiển làm cho ứng dụng thực hiện một hành động khi được nhấn.

3. ***Nhập số lượng vào ứng dụng***. Một vài điều khiển (ví dụ như `TextBox`) không hiển thị văn bản tự mô tả - chúng ta nhắc đến những điều khiển này bằng cách sử dụng `Label` định danh chúng. Chẳng hạn, ta nhắc đến `TextBox` bên phải `Label` **Cartons per shipment:** là `TextBox` **Cartons per shipment:**. Nhập 3 vào `TextBox` **Cartons per shipment:**. Nhập 15 vào `TextBox` **Items per Carton:**. Hình 4.2 thể hiện `Form` sau khi các giá trị này đã được nhập vào.



Hình 4.2 Ứng dụng **Inventory** với số liệu mới được nhập vào.

4. ***Tính toán tổng số sách đã nhận được***. Nhấn `Button` **Calculate Total**. Khi `Button` này được nhấn thì ngay lập tức ứng dụng nhân hai số vừa được nhập vào và hiển thị kết quả (45) trên `Label` bên phải của **Total:** (Hình 4.3).



Hình 4.3 Kết quả của việc nhấn `Button` **Calculate Total** trong ứng dụng **Inventory**.

5. ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.
6. ***Đóng project***. Chọn **File > Close Project**.

4.2 Xây dựng ứng dụng Inventory

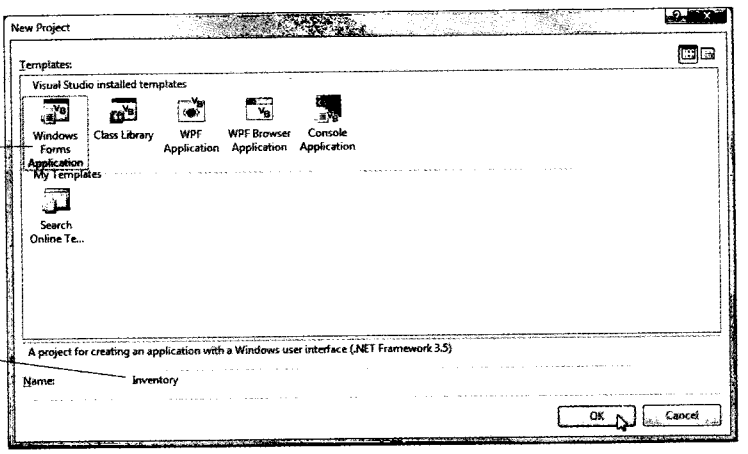
Bây giờ bạn đã chạy thử ứng dụng hoàn chỉnh, bạn sẽ bắt đầu tự xây dựng cho riêng mình một phiên bản của ứng dụng. Bạn sẽ tạo project có Form chứa những điều khiển cần thiết cho ứng dụng **Inventory**. Sau đó bạn lưu solution vào thư mục làm việc của mình, `C:\SimplyVB2008`. [*Lưu ý*: Chúng tôi giả sử bạn đã tạo thư mục này như theo hướng dẫn trong phần *Chuẩn bị*. Nếu quyết định sử dụng thư mục khác, hãy lưu ứng dụng vào thư mục đó.] Cuối cùng, chúng tôi sẽ hướng dẫn bạn cách đặt lại tên Form.

Tạo ứng dụng mới

1. ***Tạo project mới***. Để tạo ứng dụng Windows, chọn **File > New Project...** để hiển thị hộp thoại **New Project** (Hình 4.4). Từ danh sách template, chọn **Windows Forms Application**. Nhập `Inventory` vào TextBox **Name:** và nhấn Button **OK**.

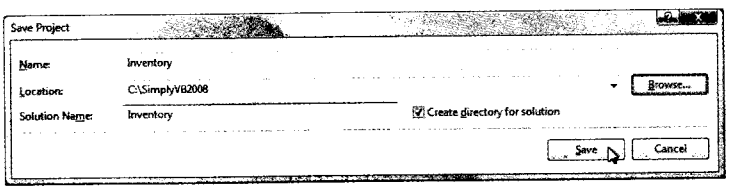
Chọn **Windows Forms Application** từ khung **Templates:**

TextBox **Name:**



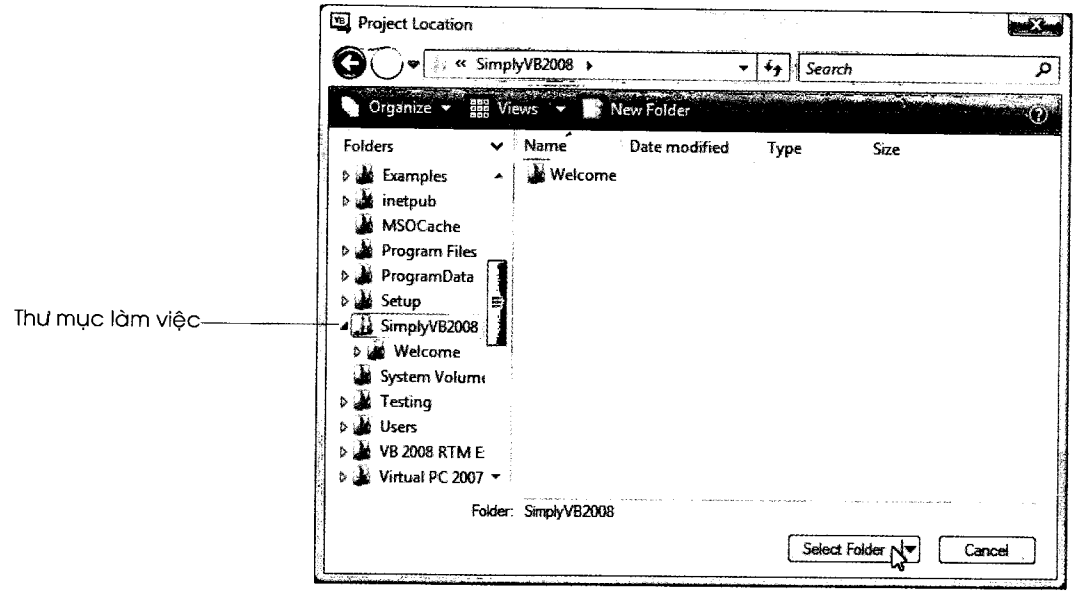
Hình 4.4 Hộp thoại **New Project** để tạo ứng dụng mới.

2. ***Lưu project vào thư mục làm việc của bạn***. Bây giờ khi vùng làm việc trống đã được tải, hãy chọn **File > Save All**, để hiển thị hộp thoại **Save Project** (Hình 4.5). Nhấn Button **Browse...**, hộp thoại **Project Location** sẽ xuất hiện (Hình 4.6). Vì bạn đã tạo thư mục `SimplyVB2008`, di chuyển tới `C:\ SimplyVB2008`. Nhấn **Select Folder** để chọn thư mục và đóng hộp thoại. Thư mục được chọn sau đó xuất hiện ở TextBox **Location:**.

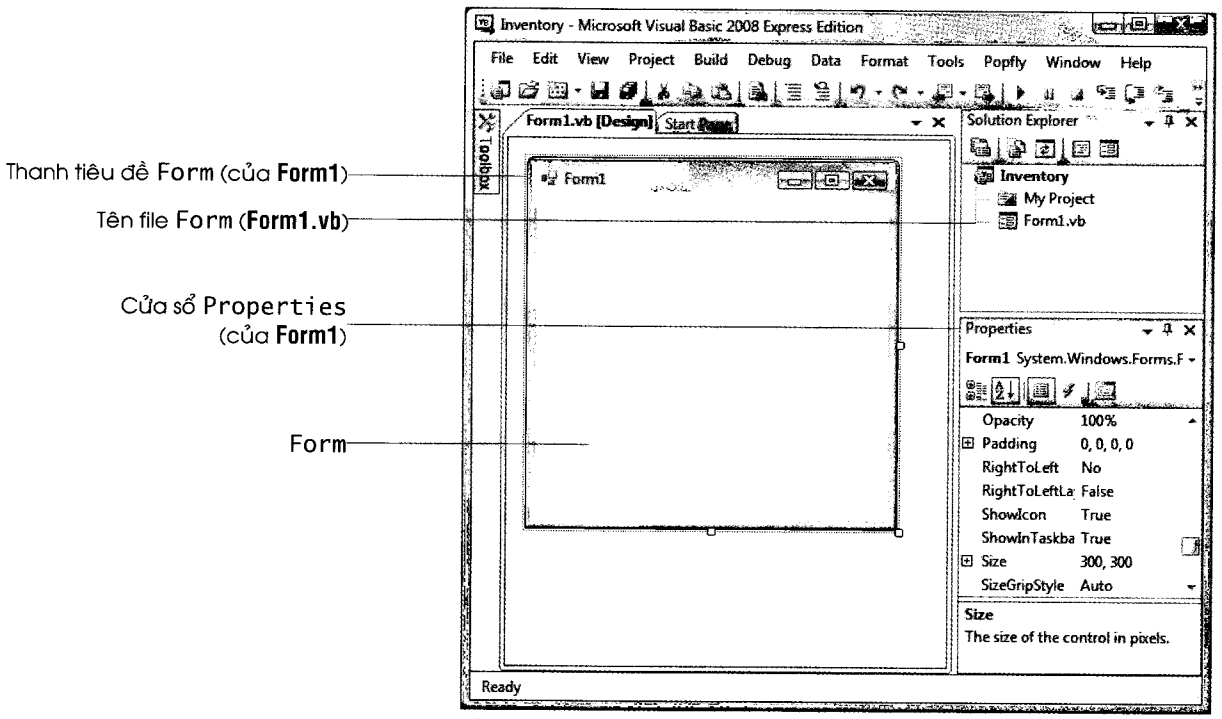


Hình 4.5 Hộp thoại **Save Project** để lưu ứng dụng mới được khởi tạo.

3. ***Xem Form***. Nhấn Button **Save** (Hình 4.5) để đóng hộp thoại **Save Project**. Ứng dụng bao gồm Form có tên là **Form1** (Hình 4.7). Nếu Form này không giống như trên Hình 4.7, hãy chọn **View > Designer**. Sau đó nhấn Form trong IDE để chọn.



Hình 4.6 Hộp thoại **Project Location** được sử dụng để chỉ ra thư mục lưu file cho project.



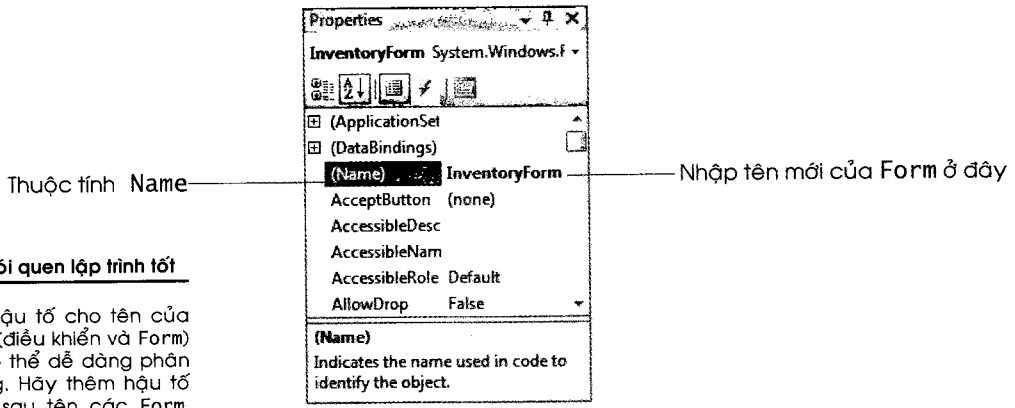
Hình 4.7 Một ứng dụng Windows mới trong Visual Studio.

4. ***Đổi tên file Form.*** Thay đổi tên file của Form thành một tên có ý nghĩa hơn cho ứng dụng là một thói quen tốt. Để làm việc này, nhấn vào tên file của Form (Form1.vb) trong **Solution Explorer**. Sau đó chọn File Name trong cửa sổ **Properties** và nhập Inventory.vb trong ô bên phải. Nhấn *Enter* để cập nhật tên file. Thông thường, bạn cần nhấn *Enter* hoặc chọn một thuộc tính khác để những thay đổi trong cửa sổ **Properties** có hiệu lực.
5. ***Đổi tên đối tượng Form.*** Mỗi đối tượng Form cần có tên riêng và có ý nghĩa để dễ phân biệt. Trong Visual Basic IDE, bạn đặt tên Form bằng cách sử dụng thuộc tính **Name**. Theo mặc định, Visual Basic IDE đặt tên Form là Form1. Khi bạn thay đổi tên file của Form, Visual Basic IDE cập

Thói quen lập trình tốt

Đặt tên duy nhất và có ý nghĩa cho Form để dễ phân biệt.

nhật thuộc tính Name của Form một cách tự động theo tên của file nhưng không có phần mở rộng .vb - trong trường hợp này là Inventory. Nhấn vào Form trong Windows Form Designer. Trong cửa sổ **Properties** (Hình 4.8), tìm và nhấn đúp vào trường bên phải thuộc tính Name, được hiển thị là (Name). Nhập tên InventoryForm, sau đó nhấn *Enter* để cập nhật.



Hình 4.8 Đổi tên Form trong cửa sổ **Properties**.

Thói quen lập trình tốt

Sử dụng hậu tố cho tên của đối tượng (điều khiển và Form) để bạn có thể dễ dàng phân biệt chúng. Hãy thêm hậu tố Form vào sau tên các Form. Viết hoa chữ cái đầu của tên Form vì Form là một lớp, còn đối tượng (ví dụ như điều khiển) nên bắt đầu bằng chữ thường.

6. ***Lưu project***. Chọn **File > Save All** để lưu thay đổi. Lưu lại công việc giúp cho bạn tránh việc bị mất những thay đổi đã thực hiện.

Mẹo thiết kế giao diện

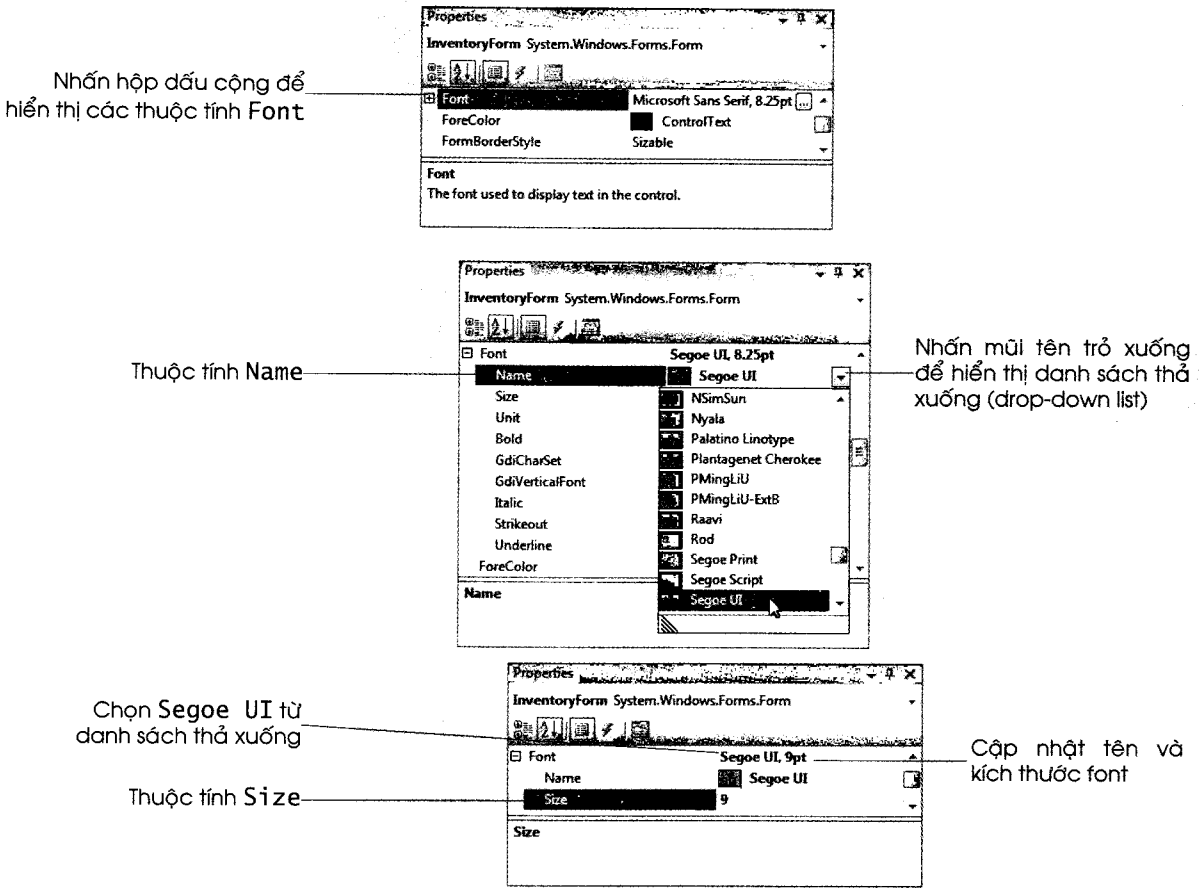
Thay đổi font của Form thành Segoe UI 9pt để tương thích với font được Microsoft khuyến nghị sử dụng cho hệ điều hành Windows.

Tiếp theo, bạn sẽ tìm hiểu cách thay đổi Form bằng cách thiết lập font chữ cho Form. Bạn nên thiết lập font của Form là Segoe UI 9pt, font được Microsoft khuyến nghị cho giao diện của hệ điều hành Windows. Điều này đảm bảo rằng các điều khiển được thêm vào Form đều sử dụng font Segoe UI. Bạn cũng sẽ học cách thay đổi tiêu đề và kích thước Form. Mặc dù đã thay đổi tên file thành Inventory.vb, bạn vẫn cần thay đổi văn bản của thanh tiêu đề để người dùng biết được nội dung của Form. Thay đổi kích thước Form sao cho vừa với nội dung bên trong sẽ làm cho giao diện đẹp hơn.

Tùy chỉnh Form

1. ***Thiết lập font của Form***. Trong chương trước bạn đã dùng hộp thoại **Font** để thay đổi font. Bây giờ bạn sẽ dùng cửa sổ **Properties** để thay đổi font của Form. Chọn Form trong Windows Form Designer. Nếu cửa sổ **Properties** không được mở sẵn, nhấn vào biểu tượng **Properties** trên thanh công cụ IDE hoặc chọn **View > Properties Window**. Để thay đổi font của Form thành Segoe UI 9pt, nhấn hộp dấu cộng (⊞) ở bên trái thuộc tính Font trong cửa sổ **Properties** (Hình 4.9). Việc này làm cho các thuộc tính liên quan đến Font của Form được hiển thị. Trong danh sách vừa xuất hiện, chọn thuộc tính Name của font sau đó nhấn mũi tên trỏ xuống ở bên phải giá trị thuộc tính. Trong danh sách vừa xuất hiện, chọn Segoe UI. [*Lưu ý*: Danh sách có thể chứa các font khác so với trong Hình 4.9 phụ thuộc vào các font được cài đặt trong hệ thống của bạn.] Sau đó, đặt thuộc tính Size của font là 9.

   Chú ý rằng có một vài thuộc tính, ví dụ như Font, có một hộp dấu cộng (⊞) bên cạnh tên thuộc tính để chỉ ra rằng node này có các thuộc tính bổ sung. Chẳng hạn, khi mở rộng node Font, bạn sẽ thấy rằng các thuộc tính bổ sung như Name, Size và Bold, mỗi thuộc tính có một danh sách giá trị của riêng chúng được hiển thị trong cửa sổ **Properties**.



Hình 4.9 Thiết lập font của Form là Segoe UI 9pt.

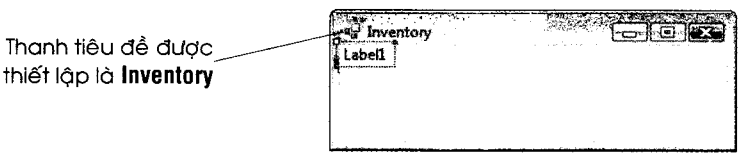
Mẹo thiết kế giao diện

Thay đổi tiêu đề Form để người dùng có thể hiểu được nội dung của Form.

Mẹo thiết kế giao diện

Tiêu đề Form nên sử dụng kiểu viết hoa tiêu đề sách.

2. ***Thiết lập nội dung văn bản trên thanh tiêu đề cho Form.*** Văn bản trên thanh tiêu đề Form được xác định bởi thuộc tính **Text** của Form. Để hiển thị thuộc tính của Form trong cửa sổ **Properties**, nhấn vào Form trong Windows Form Designer. Nhấn đúp vào trường bên phải thuộc tính Text trong cửa sổ **Properties**, nhập Inventory và nhấn *Enter*. Bạn nên sử dụng kiểu viết hoa tiêu đề sách để thiết lập nội dung cho tiêu đề Form. **Kiểu viết hoa tiêu đề sách (book-title capitalization)** là kiểu viết hoa chữ cái đầu của mỗi từ quan trọng trong văn bản và không kết thúc bằng dấu chấm câu (chẳng hạn, *Simply Visual Basic 2008*). Thanh tiêu đề cập nhật được thể hiện trong Hình 4.10.



Hình 4.10 Form đã thay đổi kích thước, hiển thị tiêu đề mới.

3. ***Thay đổi kích thước Form.*** Nhấn đúp vào trường bên phải thuộc tính Size trong cửa sổ **Properties**, sau đó nhập 320,112 và nhấn *Enter* (Hình 4.10). Lưu ý rằng Form bây giờ có cùng kích thước với ứng dụng hoàn chỉnh mà bạn đã chạy thử ở đầu chương.
4. ***Lưu project.*** Chọn **File > Save All** để lưu lại thay đổi.

Tới đây, khi đã tạo và thay đổi Form, bạn sẽ thêm điều khiển vào giao diện. Label mô tả mục đích của điều khiển trên Form và cũng có thể được sử dụng để hiển thị kết quả tính toán. Trong phần tiếp theo, bạn tìm hiểu cách thêm điều khiển Label và thiết lập tên, văn bản hiển thị và vị trí cho mỗi Label trên Form.

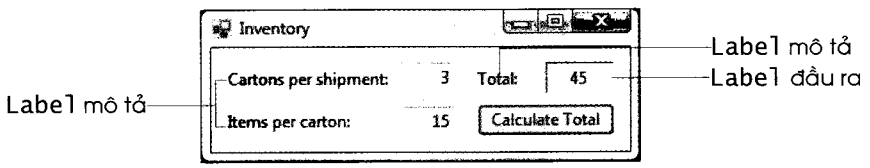
TỰ ÔN TẬP

1. _____là font được Microsoft khuyến cáo cho giao diện hệ điều hành Windows.
   a) Arial b) Microsoft Sans Serif
   c) Segoe UI d) Times New Roman
2. Tiêu đề Form sẽ sử dụng kiểu viết hoa_____.
   a) tiêu đề sách b) toàn bộ
   c) không viết hoa d) đầu câu

Đáp án: 1) c. 2) a.

4.3 Thêm Label vào ứng dụng Inventory

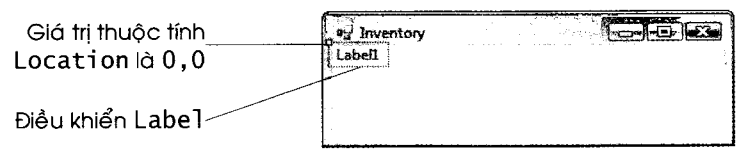
Mặc dù bạn không để ý, nhưng có bốn Label trong ứng dụng. Bạn có thể dễ dàng nhận ra ba trong số Label đó từ ứng dụng bạn thiết kế trong Chương 3. Label thứ tư có đường viền nhưng không chứa văn bản cho đến khi người dùng nhấn Button **Calculate Total** (Hình 4.11). Label thường được dùng để định danh cho những điều khiển khác trên Form. **Label mô tả (Descriptive Label)** giúp người dùng hiểu được mục đích của mỗi điều khiển, còn **Label đầu ra** được sử dụng để hiển thị kết quả đầu ra của chương trình.



Hình 4.11 Label được sử dụng trong ứng dụng **Inventory**.

Thêm Label vào Form

1. ***Thêm điều khiển Label vào Form.*** Nhấn vào nhóm **All Windows Forms** trong **Toolbox**. Sau đó nhấn đúp vào điều khiển Label trong **Toolbox** để thêm Label vào Form (Hình 4.12).



Hình 4.12 Thêm một Label vào Form.

Mẹo thiết kế giao diện

Hãy để khoảng trống giữa biên của Form và các điều khiển của chúng.

2. ***Thiết lập vị trí của Label.*** Nếu cửa sổ **Properties** chưa được mở, hãy chọn **View > Properties Window**. Trong cửa sổ **Properties**, thiết lập thuộc tính Location của Label là 9,15. Thiết lập giá trị thuộc tính Location như vậy để tạo khoảng cách giữa Label và các biên của Form. Như đã tìm hiểu trong chương trước, bạn có thể kéo một điều khiển từ **Toolbox** vào Form. Bạn cũng có thể điều chỉnh vị trí một điều khiển bằng cách chọn điều khiển đó và sử dụng phím mũi tên để di chuyển.

   Thuộc tính **Location** của Label chỉ ra vị trí của góc trên bên trái của điều khiển trên Form. IDE gán giá trị 0, 0 cho góc trên cùng bên trái của Form, không bao gồm thanh tiêu đề (Hình 4.12). Thuộc tính Location của điều khiển được thiết lập theo vị trí tương đối của điều khiển so với điểm trên cùng bên trái của Form. Khi chỉ số đầu tiên (tọa độ x) của thuộc

Mẹo thiết kế giao diện

Thuộc tính `Location` có thể được sử dụng để chỉ ra vị trí chính xác của điều khiển trên `Form`.

tính `Location` tăng, điều khiển chuyển dịch sang phải. Nếu chỉ số thứ hai (tọa độ *y*) của thuộc tính `Location` tăng, điều khiển chuyển dịch xuống phía dưới của `Form`. Trong trường hợp này, giá trị `9,15` chỉ ra rằng `Label` được đặt ở 9 pixel về phía phải của góc trên bên trái và 15 pixel bên dưới góc trên bên trái. Giá trị thuộc tính `Location` 16, 48 chỉ ra rằng `Label` được đặt ở 16 pixel về phía phải góc trên bên trái và 48 pixel bên dưới góc trên bên trái.

Thói quen lập trình tốt

Thêm hậu tố `Label` vào sau tên của điều khiển `Label`.

3. ***Thiết lập thuộc tính Name và Text của Label***. Trong cửa sổ **Properties**, nhấn đúp vào trường bên phải thuộc tính `Text`, sau đó nhập `Cartons per shipment:`. Thiết lập giá trị thuộc tính `Name` là `cartonsLabel`.

   Khi nhập giá trị cho thuộc tính `Text` của `Label`, bạn nên sử dụng kiểu viết hoa đầu câu. **Kiểu viết hoa đầu câu (sentence-style capitalization)** chỉ viết hoa chữ cái đầu tiên của từ đầu tiên trong văn bản đó. Các từ khác đều viết thường ngoại trừ tên riêng (chẳng hạn, *Deitel*).

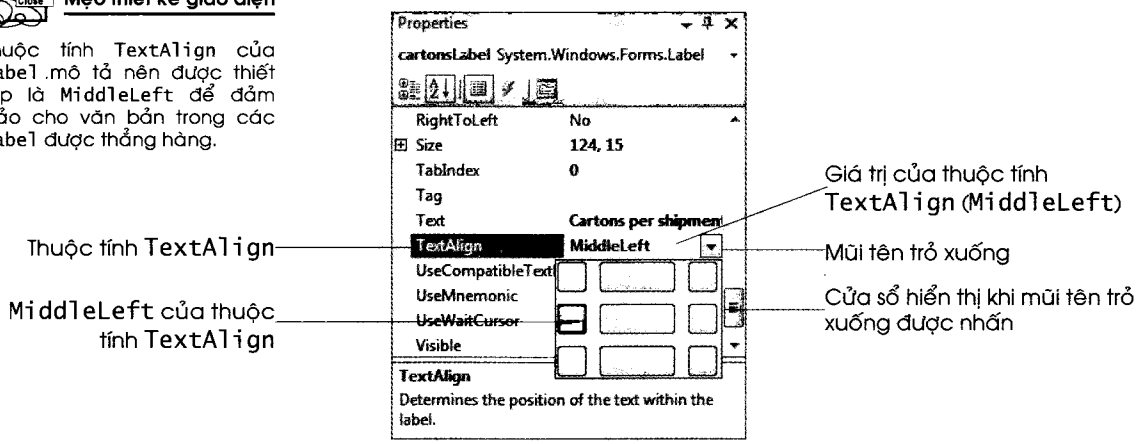
Mẹo thiết kế giao diện

Đối với `Label` sử dụng để miêu tả mục đích của điều khiển nên được viết hoa đầu câu và kết thúc với dấu hai chấm. `Label` như vậy được gọi là `Label` mô tả.

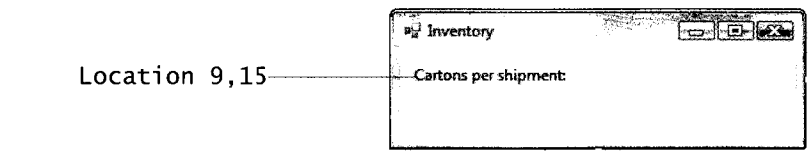
4. ***Căn chỉnh văn bản trong Label***. Chọn thuộc tính `TextAlign` trong cửa sổ **Properties**, sau đó nhấn mũi tên trỏ xuống trong trường bên phải (Hình 4.13). Thuộc tính `TextAlign` thiết lập cách căn chỉnh cho văn bản trong phạm vi điều khiển. Nhấn vào mũi tên trỏ xuống sẽ mở ra cửa sổ để bạn chọn cách căn lề cho văn bản trong `Label` (Hình 4.13). Trong cửa sổ này, chọn hình chữ nhật ở giữa bên trái cho biết văn bản trong `Label` sẽ được đặt ở giữa theo chiều dọc và vào bên trái theo chiều ngang của điều khiển. Giá trị của thuộc tính thay đổi thành `MiddleLeft`. Hình 4.14 hiển thị `Label` sau khi bạn thiết lập thuộc tính cho nó.

Mẹo thiết kế giao diện

Thuộc tính `TextAlign` của `Label` mô tả nên được thiết lập là `MiddleLeft` để đảm bảo cho văn bản trong các `Label` được thẳng hàng.



Hình 4.13 Thay đổi thuộc tính `TextAlign` của `Label`.



Hình 4.14 Giao diện sau khi `Label` đã được tùy chỉnh.

5. ***Lưu project***. Chọn **File > Save All** để lưu thay đổi.

Bây giờ bạn sẽ thêm những `Label` còn lại vào `Form`. Những `Label` này giúp người dùng hiểu nhập thông tin gì cho đầu vào và hiểu được đầu ra của ứng dụng hiển thị thông tin gì. Các `Label` này định danh cho những điều khiển bạn sẽ thêm vào `Form` sau này.

Thêm các Label còn lại vào Form

Mẹo thiết kế giao diện

Canh trái hoặc phải cho Label mô tả nếu các Label được sắp xếp theo chiều dọc.

Mẹo thiết kế giao diện

Sử dụng Label mô tả để định danh cho Label đầu ra.

Mẹo thiết kế giao diện

Đặt đầu ra của ứng dụng bên dưới và/hoặc bên phải điều khiển đầu vào của Form.

Mẹo thiết kế giao diện

Nếu Label đầu ra được sắp xếp theo hàng dọc để hiển thị số được dùng trong các phép tính toán học (như trong hóa đơn), hãy sử dụng giá trị `MiddleRight` cho thuộc tính `TextAlign`.

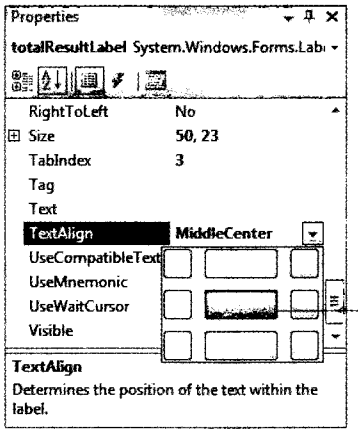
Mẹo thiết kế giao diện

Hãy thiết lập để Label đầu ra khác với Label mô tả bằng cách đặt giá trị thuộc tính `BorderStyle` của Label đầu ra là `Fixed3D`.

Thói quen lập trình tốt

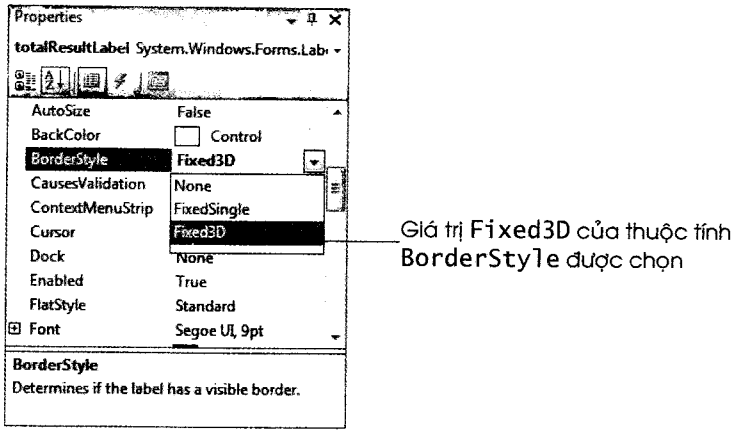
Ban đầu, giá trị đầu ra nên được xóa hoặc được cung cấp giá trị mặc định. Khi ứng dụng thực hiện tính toán cho giá trị đó, thuộc tính `Text` của Label được cập nhật giá trị mới. Bạn sẽ học cách làm việc này trong chương sau.

1. ***Thêm Label mô tả thứ hai.*** Nhấn đúp vào điều khiển `Label` trong **Toolbox** để thêm `Label` thứ hai. Thiết lập thuộc tính `Location` của `Label` là `9,46`. Đặt giá trị thuộc tính `Text` của `Label` là `Items per carton:` và thay đổi thuộc tính `Name` của `Label` này là `itemsLabel`. Sau đó đặt giá trị thuộc tính `TextAlign` của `Label` là `MiddleLeft`.
2. ***Thêm Label mô tả thứ ba.*** Nhấn đúp vào điều khiển `Label` trong **Toolbox** để thêm `Label` thứ ba. Đặt giá trị thuộc tính `Location` của `Label` là `190,15`. Thiết lập thuộc tính `Text` của `Label` là `Total:` và thay đổi thuộc tính `Name` của `Label` này thành `totalLabel`. Sau đó đặt giá trị thuộc tính `TextAlign` của `Label` là `MiddleLeft`.
3. ***Thêm Label đầu ra.*** Để thêm `Label` thứ tư, nhấn đúp vào điều khiển `Label` trong **Toolbox**. Đặt giá trị thuộc tính `AutoSize` của `Label` là `False`. Sau đó thiết lập thuộc tính `Size` của `Label` là `50,23` và thuộc tính `Location` của `Label` là `243,11`. Lưu ý rằng những thiết lập này làm cho văn bản trong `Label` đầu ra thẳng hàng với văn bản trong `label` mô tả tương ứng. Sau đó đặt tên `Label` này là `totalResultLabel`. Đặt giá trị thuộc tính `TextAlign` của `Label` là `MiddleCenter`. Đối với các `Label` trước, bạn đã đặt thuộc tính này là `MiddleLeft`. Để chọn giá trị `MiddleCenter`, hãy thực hiện như ở *Bước 2*, nhưng chọn hình chữ nhật trung tâm như trong Hình 4.15. Bạn sử dụng cách căn lề văn bản `MiddleCenter` để hiển thị kết quả tính toán, vì cách căn lề này phân biệt giá trị trong `Label` đầu ra với những giá trị trong `Label` mô tả (có thuộc tính `TextAlign` được đặt là `MiddleLeft`).



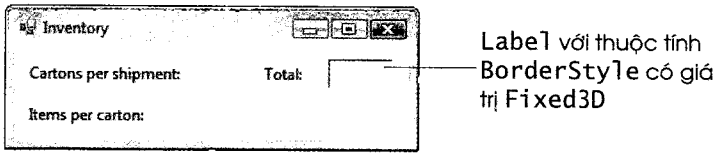
Hình 4.15 Thiết lập thuộc tính `TextAlign` là `MiddleCenter`.

4. ***Thay đổi thuộc tính BorderStyle cho Label.*** `TotalResultLabel` hiển thị kết quả tính toán của ứng dụng. Do đó, bạn phải làm cho `Label` này có giao diện khác so với những `Label` khác. Để làm việc này, bạn phải thay đổi hình thức đường viền của `Label` bằng cách thay đổi giá trị của thuộc tính **BorderStyle**. Gán giá trị `Fixed3D` (Hình 4.16) cho thuộc tính `BorderStyle` của `totalResultLabel`, làm cho `Label` có giao diện 3 chiều (Hình 4.17) [*Lưu ý*: Nếu được chọn, `FixedSingle` hiển thị đường viền đậm.]



Hình 4.16 Thay đổi thuộc tính `BorderStyle` của `Label` thành `Fixed3D`.

5. ***Xóa nội dung thuộc tính `Text` của `Label`.*** Khi `Label` được thêm vào `Form`, thuộc tính `Text` được gán tên mặc định của `Label` đó. Trong trường hợp này bạn nên xóa nội dung văn bản trên `Label` vì bạn không muốn hiển thị nội dung văn bản cho tới khi `totalResultLabel` được gán một giá trị có nghĩa trong ứng dụng. Để làm việc đó, xóa văn bản bên phải thuộc tính `Text` trong cửa sổ **Properties** và nhấn *Enter*. Hình 4.17 hiển thị giao diện với tất cả các `Label` vừa được thêm vào.



Hình 4.17 Giao diện với tất cả các `Label` vừa được thêm vào.

6. ***Lưu project.*** Chọn **File > Save All** để lưu thay đổi.

TỰ ÔN TẬP

1. Giá trị_____của thuộc tính `Location` xác định góc trên cùng bên trái (không bao gồm thanh tiêu đề) của `Form`.
   a) `1, 1` b) `0, 0`
   c) `1, 0` d) `0, 1`
2. `Label` đầu ra nên_____.
   a) khác biệt so với những `Label` khác
   b) có thuộc tính `Text` rỗng hay một giá trị mặc định khi khởi chạy
   c) sử dụng `Fixed3D` cho thuộc tính `BorderStyle`
   d) Các lựa chọn trên đều đúng

Đáp án: 1) b. 2) d.

4.4 Thêm TextBox và Button vào Form

Ứng dụng **Inventory** cần dữ liệu nhập vào từ người dùng để tính toán tổng số sách giáo khoa được chuyển đến kho trong mỗi lô hàng. Cụ thể là, người dùng sẽ nhập số kiện sách và số sách trong mỗi kiện. Vì dữ liệu này được nhập vào từ bàn phím nên bạn sẽ dùng điều khiển `TextBox`. Tiếp theo bạn sẽ học cách thêm `TextBox` vào `Form` và thiết lập các thuộc tính cho chúng. Sau đó bạn sẽ thêm điều khiển `Button` để hoàn chỉnh giao diện của bạn.

Thêm TextBox vào Form

Thói quen lập trình tốt

Thêm hậu tố TextBox vào sau tên của điều khiển TextBox.

Mẹo thiết kế giao diện

Sử dụng TextBox để nhập dữ liệu từ bàn phím.

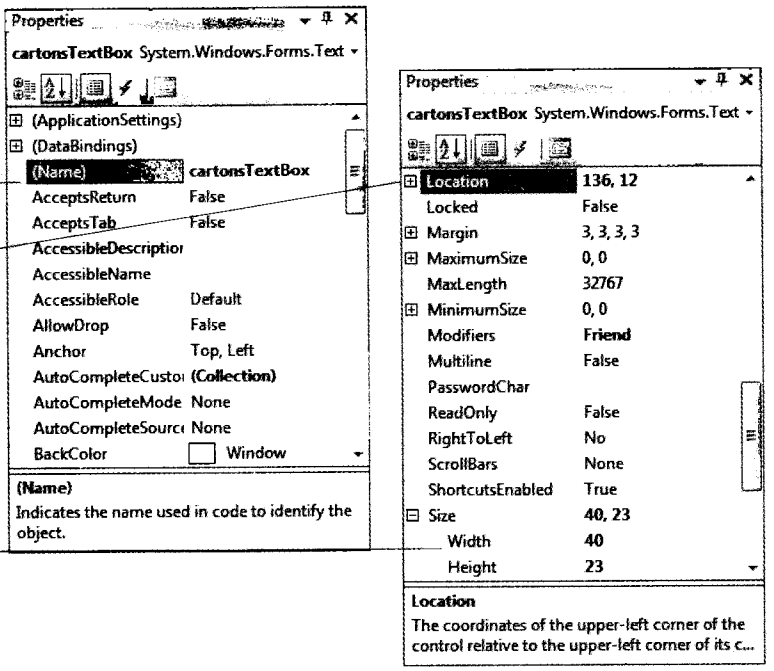
1. ***Thêm TextBox vào Form.*** Nhấn đúp vào điều khiển TextBox

trên **Toolbox** để thêm TextBox vào Form. Thiết lập thuộc tính cho TextBox giống như với Label. Để đặt tên TextBox, chọn thuộc tính Name của TextBox trong cửa sổ **Properties** và nhập cartonsTextBox vào trường bên phải thuộc tính (Hình 4.18). Đặt thuộc tính Width của TextBox là 40 và thuộc tính Location là 136,12. Thuộc tính kích thước và vị trí này sẽ làm cho văn bản trong TextBox thẳng hàng với văn bản trong Label miêu tả nó. Thiết lập thuộc tính Text là 0 (Hình 4.19). Điều này sẽ làm cho TextBox có giá trị khởi tạo bằng 0 khi ứng dụng chạy.

Thuộc tính **Name** được thiết lập là **cartonsTextBox**

Thuộc tính **Location** được thiết lập là 136,12

Thuộc tính **Width** được thiết lập là 40



Hình 4.18 Cửa sổ **Properties** cho TextBox cartonsTextBox.

Mẹo thiết kế giao diện

Điều chỉnh để TextBox đủ rộng cho thông tin đầu vào mà nó sẽ nhận.

2. ***Thay đổi thuộc tính TextAlign của TextBox.*** Thay đổi giá trị thuộc tính TextAlign của cartonsTextBox thành Right. Lưu ý rằng, khi bạn nhấn mũi tên trỏ xuống ở bên phải thuộc tính này, cửa sổ trong Hình 4.13 không xuất hiện. Đó là bởi vì TextBox có ít tùy chọn hơn, vì vậy tùy chọn đó được hiển thị dưới dạng danh sách. Chọn Right trong danh sách (Hình 4.19). Thông thường, khi nhiều TextBox dùng để nhập dữ liệu số được xếp theo chiều dọc, văn bản của chúng nên được căn phải.

Mẹo thiết kế giao diện

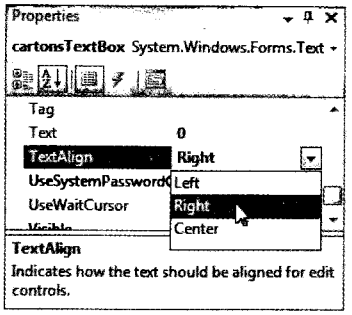
Đặt mỗi Label mô tả ở trên hoặc bên trái của điều khiển (ví dụ như TextBox) mà Label đó định danh.

3. ***Thêm TextBox thứ hai vào Form.*** Nhấn đúp vào điều khiển TextBox trong **ToolBox**. Đặt tên cho TextBox là itemsTextBox. Thiết lập giá trị thuộc tính Width là 40 và Location là 136,43. Thiết lập này đảm bảo cho cạnh phải của các TextBox được thẳng hàng. Thiết lập này cũng làm cho văn bản trong TextBox thẳng hàng với văn bản trong Label mô tả nó. Đặt thuộc tính Text là 0 và thuộc tính TextAlign là Right. Hình 4.20 cho thấy Form sau khi TextBox được thêm vào và các thuộc tính đã được thiết lập.

Lưu ý rằng nhiều thuộc tính của TextBox thứ hai có giá trị giống với các thuộc tính tương ứng của TextBox thứ nhất (ví dụ Width, Text và TextAlign). Khi tạo nhiều điều khiển của cùng một loại với nhiều giá trị giống nhau, sẽ dễ hơn nếu ta copy điều khiển đầu tiên. Để làm việc này, chọn điều khiển bạn muốn copy và nhấn *Ctrl + C*. Sau đó nhấn *Ctrl + V* để paste nó lên Form. Điều chỉnh lại vị trí của điều khiển mới và điều chỉnh các thuộc tính cần thiết.

Mẹo thiết kế giao diện

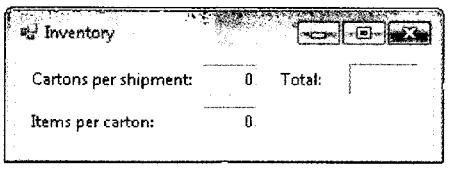
Hãy để TextBox đủ rộng cho đầu vào mà nó sẽ nhận.



Hình 4.19 Thiết lập giá trị `Right` cho thuộc tính `TextAlign` của điều khiển `TextBox`.

Mẹo thiết kế giao diện

`Label` mô tả và điều khiển mà nó định danh phải được canh dòng bên trái nếu chúng được xếp theo hàng dọc.



Hình 4.20 Giao diện sau khi các `TextBox` đã được thêm vào và chỉnh sửa.

4. ***Lưu project.*** Chọn **File > Save All** để lưu thay đổi.

Mẹo thiết kế giao diện

Văn bản trong `Label` mô tả và văn bản trong điều khiển `Label` mà nó định danh phải được căn dưới nếu chúng được xếp ngang hàng.

Lưu ý rằng các điều khiển được căn theo chiều ngang và chiều dọc. Thông thường, bạn nên đặt `Label` mô tả ở trên hoặc bên trái điều khiển mà nó mô tả (ví dụ như `TextBox`). Nếu bạn đặt các điều khiển trên cùng một dòng, văn bản của `Label` mô tả và văn bản của điều khiển mà nó mô tả cần được bố trí thẳng hàng. Tuy nhiên, nếu bạn đặt điều khiển theo chiều dọc, `Label` được đặt bên trên điều khiển mà nó mô tả và các điều khiển được căn trái. Việc tuân theo những chỉ dẫn đơn giản này làm cho ứng dụng của bạn có hình thức đẹp hơn và dễ sử dụng hơn vì các điều khiển trên ứng dụng được tổ chức rõ ràng, ngăn nắp.

Bây giờ người dùng đã có thể nhập dữ liệu vào `TextBox`, bạn cần tìm cách cho phép người dùng ra lệnh cho ứng dụng thực thi việc tính toán và hiển thị kết quả. Cách phổ biến nhất để người dùng làm việc đó là nhấn `Button`. Phần tiếp theo sẽ giải thích cách thêm `Button` vào ứng dụng **Inventory**.

Thêm Button vào Form

Mẹo thiết kế giao diện

Các `Button` thường đặt từ trên xuống dưới, bắt đầu từ góc trên bên phải của `Form` hoặc được sắp xếp trên cùng một dòng bắt đầu từ góc dưới bên phải của `Form`.

Thói quen lập trình tốt

Thêm hậu tố `Button` vào sau tên điều khiển `Button`.

1. ***Thêm Button vào Form.*** Thêm `Button` vào `Form` bằng cách nhấn đúp vào điều khiển `Button`

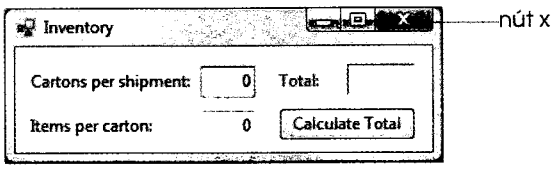
trên **Toolbox**. Thiết lập giá trị thuộc tính của `Button` cũng tương tự như thiết lập thuộc tính cho `Label` hay `TextBox`. Nhập `calculateButton` vào thuộc tính `Name` của `Button`.

Thiết lập giá trị thuộc tính `Size` của `Button` là `100,24` và `Location` là `193,42`. Lưu ý rằng những thiết lập này làm cho cạnh trái và cạnh phải của `Button` được gióng với những `Label` ở phía trên (Hình 4.21). Bạn cũng có thể thực hiện việc này bằng cách căn cạnh trái của `Button` với `Label` ở bên trên nó. Sau đó kéo điểm neo thay đổi kích thước ở bên phải `Button` cho đến khi đường thẳng màu xanh xuất hiện cho biết rằng `Button` đã được căn phải với `Label` bên trên nó. Nhập `Calculate Total` vào thuộc tính `Text` của `Button`. Thuộc tính `Text` của `Button` hiển thị giá trị lên `Button`. Bạn nên sử dụng lối viết hoa tiêu đề sách cho thuộc tính `Text` của `Button`. Khi gán nhãn cho `Button`, hãy giữ cho văn bản càng ngắn càng tốt mà vẫn chỉ ra được chức năng cho `Button`.

Mẹo thiết kế giao diện

Khi bạn kéo một điều khiển, IDE sẽ hiển thị các dòng màu xanh nước biển và màu tím gọi là đường gióng (snapline). Dòng màu xanh giúp bạn xếp vị trí của các điều khiển gióng với nhau. Dòng màu tím giúp bạn xếp vị trí các điều khiển sao cho văn bản trên điều khiển được gióng với nhau.

2. ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng (Hình 4.21). Lưu ý rằng sẽ không có hành động nào xảy ra nếu bạn nhấn `Button` **Calculate Total**. Đó là bởi vì bạn chưa viết mã cho ứng dụng để phản hồi lại hành động nhấn chuột. Trong Chương 5, bạn sẽ viết mã để hiển thị tổng số sách (trong `totalResultLabel`) của lô hàng khi nhấn `Button`.



Hình 4.21 Chạy ứng dụng sau khi hoàn thiện cài đặt.

3. ***Đóng ứng dụng.*** Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.
4. ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

Mẹo thiết kế giao diện

Các `Button` được gán nhãn bằng cách sử dụng thuộc tính `Text` của chúng. Các nhãn này sử dụng kiểu viết hoa tiêu đề sách và càng ngắn gọn càng tốt nhưng vẫn rõ nghĩa để người dùng hiểu được.

TỰ ÔN TẬP

1. Thuộc tính_____của `Button` thiết lập nội dung văn bản hiển thị trên `Button`.
   a) `Name` b) `Text`
   c) `Title` d) `Face`
2. Các `Button` thường_____trên `Form`.
   a) nằm trên cùng một dòng, từ góc dưới bên phải
   b) được căn lề với văn bản trên thanh tiêu đề
   c) được sắp xếp từ trên xuống dưới, bắt đầu từ góc trên bên trái
   d) Hoặc a hoặc c

Đáp án: **1)** b. **2)** d.

4.5 Tổng kết

Trong chương này, bạn bắt đầu xây dựng ứng dụng **Inventory** bằng việc thiết kế giao diện người dùng đồ họa. Bạn đã tìm hiểu cách dùng `Label` để miêu tả điều khiển và thiết lập các thuộc tính `TextAlign` và `BorderStyle` của `Label`. Bạn đã dùng những thuộc tính này để phân biệt giữa `Label` mô tả và `Label` đầu ra.

Sau khi thiết lập tiêu đề cho `Form`, bạn đã thêm `TextBox` để cho phép người dùng nhập dữ liệu từ bàn phím. Cuối cùng bạn thêm `Button` vào ứng dụng **Inventory**, cho phép người dùng yêu cầu ứng dụng thực hiện hành động (trong trường hợp này là thực hiện phép nhân và hiển thị kết quả). Khi thêm điều khiển vào `Form`, bạn cũng đã học được một vài mẹo thiết kế giao diện giúp cho bạn tạo ra những giao diện người dùng đồ họa hấp dẫn và trực quan.

Chương tiếp theo sẽ hướng dẫn bạn viết mã trong Visual Basic, sẽ được thực thi khi người dùng nhấn `Button` **Calculate Total**. Khi `Button` được nhấn, ứng dụng nhận tín hiệu được gọi là sự kiện. Bạn sẽ tìm hiểu cách lập trình ứng dụng để phản hồi lại sự kiện đó bằng việc thực hiện phép tính nhân và hiển thị kết quả.

TỔNG KẾT KỸ NĂNG

Tạo Project mới

- Chọn **File > New Project...** để tạo project.
- Trong hộp thoại **New Project**, chọn **Windows Forms Application** và thiết lập tên cho ứng dụng trong `TextBox` **Name:**.
- Chọn **File > Save All** để lưu project vào thư mục làm việc của bạn (`C:\SimplyVB2008`) bằng cách chọn thư mục từ hộp thoại **Project Location**.

Thiết lập Font của Ứng dụng là Segoe UI

- Chọn `Segoe UI` từ `Combobox` của thuộc tính `Font Name` trong cửa sổ **Properties** của `Form`. Đặt giá trị thuộc tính `Font Size` là 9.

Tạo Label mô tả

- Thêm `Label` vào `Form`, sau đó thay đổi thuộc tính `TextAlign` thành `MiddleLeft`.
- Sử dụng kiểu viết hoa đầu câu cho văn bản trên label và kết thúc văn bản của label bằng dấu hai chấm (:).

Tạo Label đầu ra

- Thêm một `Label` vào `Form` và thay đổi thuộc tính `Border Style` thành `Fixed3D` và thuộc tính `TextAlign` thành `MiddleCenter`.

Cho phép người dùng nhập dữ liệu từ bàn phím

- Thêm một điều khiển `TextBox` vào `Form`.

Phát tín hiệu để ứng dụng thực hiện hành động

- Thêm `Button` vào `Form` và viết mã để thực hiện hành động này. (Bạn sẽ học cách viết mã chương trình trong Chương 5.)

THUẬT NGỮ

điều khiển `Button` - Khi được nhấn sẽ làm cho ứng dụng thực hiện một hành động.

điều khiển `TextBox` - Lấy về dữ liệu do người dùng nhập vào từ bàn phím.

font `Segoe UI` - Font được Microsoft khuyến nghị sử dụng cho ứng dụng trên Windows.

kiểu viết hoa đầu câu (sentence-style capitalization) - Là kiểu chỉ viết hoa chữ cái đầu tiên của từ đầu tiên trong văn bản. Những chữ cái còn lại trong văn bản được viết thường, trừ chữ cái đầu của tên riêng (ví dụ, **Cartons per shipment**).

kiểu viết hoa tiêu đề sách (book-title capitalization) - Là kiểu viết hoa chữ cái đầu tiên của mỗi từ trong văn bản (ví dụ: **Calculate Total**).

`Label` đầu ra (output `Label`) - Là `Label` được sử dụng để hiển thị kết quả.

`Label` mô tả (descriptive `Label`) - Là `Label` được dùng để mô tả cho điều khiển khác trên `Form`, giúp cho người dùng hiểu được mục đích của điều khiển được mô tả.

thuộc tính `BorderStyle` - Chỉ ra kiểu đường viền của `Label`, cho phép phân biệt điều khiển này với điều khiển khác một cách trực quan. Thuộc tính `BorderStyle` có thể có giá trị `None` (không có đường viền), `FixedSingle` (đường viền là một đường đậm) hoặc `Fixed3D` (cho `Label` một giao diện có "chiều sâu").

thuộc tính `Location` - Chỉ ra vị trí (tọa độ *x* và *y*) góc trên bên trái của điều khiển. Thuộc tính này được dùng để thiết lập vị trí chính xác cho điều khiển trên `Form`.

thuộc tính `Name` - Gán một tên riêng, duy nhất và có nghĩa cho điều khiển để dễ phân biệt.

thuộc tính `Text` - Thiết lập nội dung văn bản hiển thị trên điều khiển.

HƯỚNG DẪN THIẾT KẾ GIAO DIỆN

Thiết kế tổng thể

- Tạo khoảng cách giữa cạnh của `Form` với những điều khiển trong đó.
- Mặc dù bạn có thể kéo điều khiển `Label` đến một vị trí trên `Form`, thuộc tính `Location` được sử dụng để chỉ ra vị trí chính xác của `Label` đó.
- Đặt điều khiển hiển thị kết quả đầu ra của ứng dụng bên dưới và/hoặc bên phải điều khiển đầu vào của `Form`.
- Khi bạn kéo điều khiển, IDE hiển thị các đường kẻ màu xanh và màu tím gọi là đường gióng (snapline). Đường kẻ màu xanh giúp sắp xếp vị trí các điều khiển gióng với nhau. Đường kẻ màu tím giúp sắp xếp vị trí của các điều khiển sao cho văn bản trên điều khiển gióng với nhau.

`Button`

- `Button` được gán nhãn bằng thuộc tính `Text`. Các nhãn này sử dụng kiểu viết hoa tiêu đề sách và càng ngắn càng tốt nhưng vẫn rõ nghĩa để người dùng có thể hiểu được.
- `Button` nên được sắp xếp từ trên xuống dưới bắt đầu từ góc trên bên phải của `Form` hoặc được sắp xếp trên cùng một dòng bắt đầu từ góc dưới bên phải của `Form`.

`Form`

- Thay đổi tiêu đề `Form` để người dùng biết được mục đích của `Form`.
- Nên sử dụng kiểu viết hoa tiêu đề sách cho tiêu đề của `Form`.
- Thay đổi font của `Form` thành `Segoe UI 9pt`, là font được Microsoft khuyến cáo cho hệ điều hành Windows.

`Label`

- `Label` được dùng để miêu tả mục đích của điều khiển, sử dụng kiểu viết hoa đầu câu và kết thúc với dấu hai chấm. Kiểu `Label` này được gọi là `Label` mô tả.
- Thuộc tính `TextAlign` của một `Label` mô tả được đặt giá trị `MiddleLeft` để đảm bảo rằng văn bản trong các `Label` được gióng với nhau.
- Đặt `Label` mô tả ở trên hoặc bên trái điều khiển (ví dụ, `TextBox`) mà `Label` đó định danh.
- Căn lề cạnh trái hoặc phải cho các `Label` mô tả nếu các `Label` được xếp theo chiều dọc.
- Sử dụng `Label` mô tả để định danh cho `Label` đầu ra.
- `Label` đầu ra nên được hiển thị khác với `Label` mô tả bằng cách thiết lập thuộc tính `BorderStyle` của `Label` đầu ra là `Fixed3D`.
- Nếu các `Label` đầu ra được xếp theo chiều thẳng đứng để hiển thị số được sử dụng trong tính toán (chẳng hạn như trong hóa đơn), hãy sử dụng giá trị `MiddeRight` cho thuộc tính `TextAlign`.
- `Label` mô tả và điều khiển mà nó mô tả nên được căn lề trái nếu chúng được xếp theo chiều dọc.
- Văn bản trong `Label` mô tả và văn bản trong điều khiển mà nó mô tả nên được gióng hàng nếu chúng được sắp xếp theo chiều ngang.

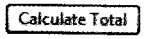
`TextBox`

- Sử dụng `TextBox` để nhập dữ liệu từ bàn phím.
- Mỗi `TextBox` có một `Label` mô tả, chỉ ra cho người dùng biết nội dung cần nhập vào.
- Hãy để `TextBox` đủ rộng cho đầu vào mà nó sẽ nhận.

ĐIỀU KHIỂN, SỰ KIỆN, THUỘC TÍNH & PHƯƠNG THỨC

Button (Button) Khi được nhấn sẽ ra lệnh cho ứng dụng thực hiện một hành động.

- ***Trên giao diện khi ứng dụng chạy***



- ***Thuộc tính***

`Location` - Vị trí của tương đối `Button` trên `Form` so với góc trên cùng bên trái.

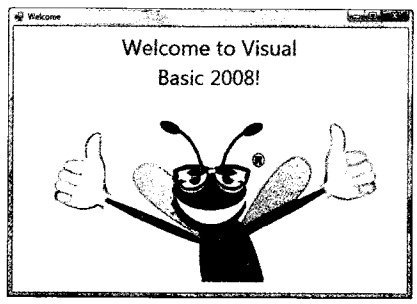
`Name` - Tên được sử dụng để xác định `Button`. Tên nên bao gồm hậu tố `Button`.

`Size` - Chiều rộng và chiều cao (bằng pixel) của `Button`.

`Text` - Văn bản được hiển thị trên `Button`.

Form Là cửa sổ chính của ứng dụng giao diện người dùng đồ họa.

- ***Trên giao diện khi ứng dụng chạy***



- ***Thuộc tính***

`BackColor` - Màu nền của `Form`.

`Font` - Tên font, kiểu và kích thước của văn bản hiển thị trên `Form`. Theo mặc định, các điều khiển của `Form` sử dụng font được thiết lập cho `Form`.

`Name` - Tên được sử dụng để xác định `Form`. Tên nên bao gồm hậu tố `Form`.

`Size` - Chiều rộng và chiều cao của `Form` (bằng pixel).

`Text` - Văn bản được hiển thị trên thanh tiêu đề của `Form`.

Label **A** Label Điều khiển này hiển thị văn bản mà người dùng không thể thay đổi.

- ***Trên giao diện khi ứng dụng chạy***

- ***Thuộc tính***

`AutoSize` - Cho phép tự động thay đổi kích thước của `Label` để vừa với nội dung.

`BoderStyle` - Xác định hình dáng đường viền của `Label`.

`Font` - Tên font, kiểu và kích thước của văn bản được hiển thị trên `Label`.

`Location` - Vị trí tương đối của `Label` trên `Form` so với góc trên bên trái của `Form`.

`Name` - Tên sử dụng để xác định `Label`. Tên nên bao gồm hậu tố `Label`.

`Size` - Chỉ ra chiều rộng và chiều cao của `Label` (bằng pixel).

`Text` - Văn bản được hiển thị trên `Label`.

`TextAlign` - Chỉ ra cách văn bản được căn chỉnh trong phạm vi `Label`.

TextBox TextBox Điều khiển này cho phép người dùng nhập dữ liệu từ bàn phím.

- ***Trên giao diện khi ứng dụng chạy***

- ***Thuộc tính***

`Location` - Vị trí tương đối của `TextBox` trên `Form` so với góc trên bên trái của `Form`.

`Name` - Tên sử dụng để xác định `TextBox`. Tên nên bao gồm hậu tố `TextBox`.

`Size` - Chiều rộng và chiều cao của `TextBox` (bằng pixel).

`Text` - Văn bản được hiển thị trong `TextBox`.

`TextAlign` - Xác định cách văn bản được căn chỉnh trong phạm vi `TextBox`.

`Width` - Xác định chiều rộng (bằng pixel) của `TextBox`.

CÂU HỎI TRẮC NGHIỆM

4.1 Ứng dụng Windows mới được tạo bằng cách chọn_____từ menu **File**.

a) **New Program** b) **New File...**

c) **New Project...** d) **New Application**

4.2 Thuộc tính `BorderStyle` của `Label` có thể được thiết lập giá trị là_____.

a) `Fixed3D` b) `Single`

c) `3D` d) Các lựa chọn trên đều đúng

4.3 Khi tạo `Label`, bạn có thể chỉ ra_____cho `Label`.

a) kiểu căn lề b) kiểu đường viền

c) kích thước d) Các lựa chọn trên đều đúng

4.4 Thay đổi giá trị của thuộc tính_____sẽ làm thay đổi tên file của `Form`.

a) `Name` b) `File`

c) `File Name` d) `Full Path`

4.5 _____là một hậu tố được thêm vào sau tên của `TextBox`.

a) `Text Box` b) `Text`

c) `Box` d) `TextBox`

4.6 _____giúp cho người dùng hiểu mục đích của điều khiển.

a) `Button` b) `Label` mô tả

c) `Label` đầu ra d) Thanh tiêu đề

4.7 _____là điều khiển cho phép người dùng nhập dữ liệu từ bàn phím.

a) `Button` b) `TextBox`

c) `Label` d) `PictureBox`

4.8 `Label` mô tả dùng_____.

a) kiểu viết hoa đầu câu b) kiểu viết hoa tiêu đề sách

c) dấu hai chấm ở cuối văn bản d) Cả a và c

4.9 Bạn nên sử dụng font_____trong ứng dụng trên hệ điều hành Windows.

a) Segoe UI b) MS Sans Serif

c) Times d) Palatino

4.10_____ là hậu tố nên được thêm vào sau tên của `Button`.

a) `Press` b) `Label`

c) `Click` d) `Button`

BÀI TẬP

Ở cuối mỗi chương bạn sẽ tìm thấy bản tóm tắt các mẹo thiết kế giao diện được liệt kê trong phần Hướng dẫn thiết kế giao diện. Danh sách đầy đủ các chỉ dẫn thiết kế giao diện sắp xếp theo từng điều khiển, xuất hiện trong Phụ lục A. Trong những bài tập sau, bạn sẽ thấy `Form` của Visual Basic không tuân thủ những chỉ dẫn thiết kế giao diện được giới thiệu trong chương này. Cho mỗi bài tập, hãy thay đổi các thuộc tính điều khiển sao cho kết quả cuối cùng của bạn tuân thủ theo các chỉ dẫn được giới thiệu trong chương. Lưu ý rằng những ứng dụng này không cung cấp chức năng gì.

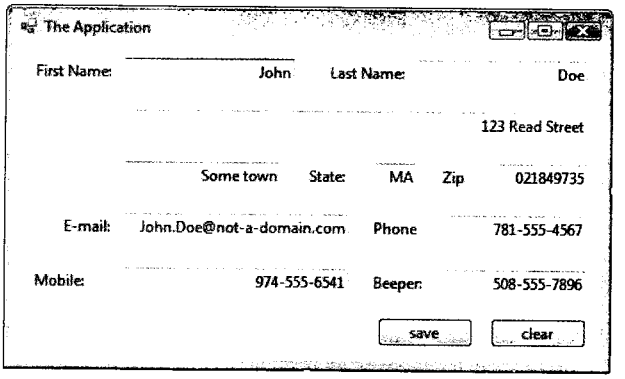
4.11 ***(Giao diện Address Book)*** Trong bài tập này, bạn áp dụng các chỉ dẫn thiết kế giao diện vừa học được để thiết kế giao diện cho cuốn sổ địa chỉ (Hình 4.22).

a) ***Copy template của ứng dụng vào thư mục làm việc.*** Copy thư mục `C:\Example\Tutorial04\Exercises\AddressBook` vào thư mục `C:\SimplyVB2008`.

b) ***Mở file template của ứng dụng.*** Nhấn đúp vào file `AddressBook.sln` trong thư mục `AddressBook` để mở ứng dụng.

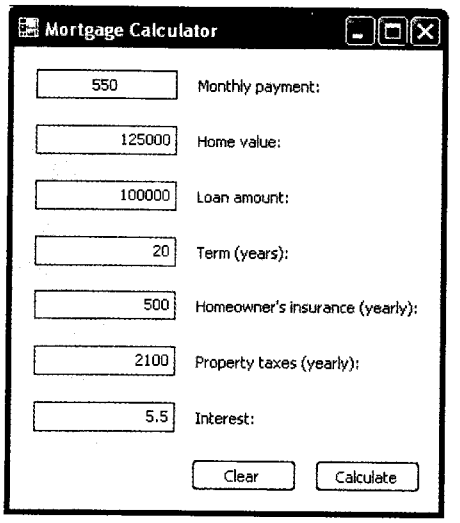
c) ***Áp dụng theo chỉ dẫn thiết kế giao diện.*** Sắp xếp lại các điều khiển và thay đổi các thuộc tính sao cho giao diện tuân thủ những chỉ dẫn thiết kế bạn vừa học được.

d) ***Lưu project.*** Chọn **File > Save All** để lưu lại các thay đổi của bạn.



Hình 4.22 Ứng dụng **Address Book** chưa được áp dụng các hướng dẫn thiết kế giao diện.

4.12 ***(Giao diện Mortgate Calculator)*** Trong bài tập này, bạn áp dụng những hướng dẫn thiết kế giao diện vừa học để thiết kế giao diện cho bảng tính toán vay thế chấp (Hình 4.23).



Hình 4.23 Ứng dụng **Mortgate Calculator** khi chưa áp dụng các hướng dẫn thiết kế giao diện.

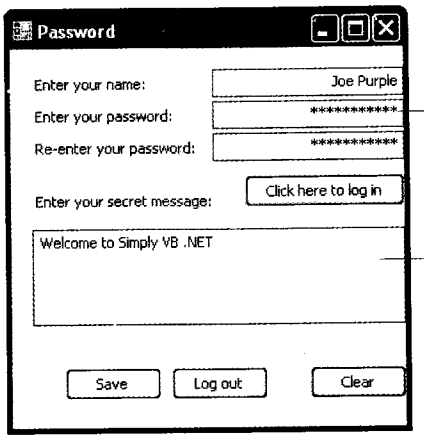
a) ***Copy template của ứng dụng vào thư mục làm việc***. Copy thư mục `C:\Example\Tutorial04\Exercises\MortgateCalculator` vào thư mục `C:\SimplyVB2008`.

b) ***Mở file template của ứng dụng***. Nhấn đúp vào file `MortgateCalculator.sln` trong thư mục `MortgateCalculator` để mở ứng dụng.

c) ***Áp dụng chỉ dẫn thiết kế giao diện***. Sắp xếp lại các điều khiển và thay đổi các thuộc tính sao cho giao diện tuân thủ những hướng dẫn thiết kế bạn vừa học được.

d) ***Lưu project***. Chọn **File > Save All** để lưu lại thay đổi.

4.13 ***(Giao diện Password)*** Trong bài tập này, bạn áp dụng những hướng dẫn thiết kế giao diện vừa học để thiết kế giao diện cho ứng dụng gửi tin được bảo vệ bằng mật khẩu (Hình 4.24).



Hình 4.24 Ứng dụng **Password** chưa được áp dụng các chỉ dẫn thiết kế giao diện.

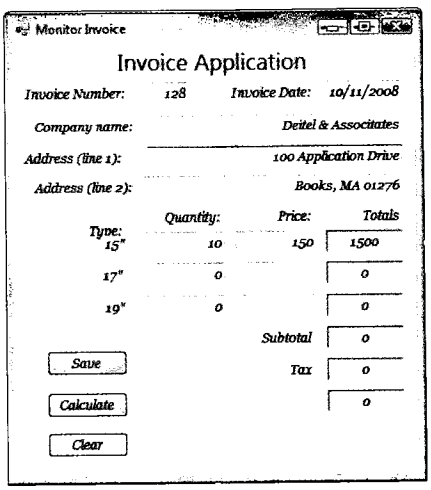
a) ***Copy template của ứng dụng vào thư mục làm việc***. Copy thư mục `C:\Example\Tutorial04\Exercises\Password` vào thư mục `C:\SimplyVB2008` của bạn.

b) ***Mở file template của ứng dụng***. Nhấn đúp vào file `Password.sln` trong thư mục `Password` để mở ứng dụng.

c) ***Áp dụng chỉ dẫn thiết kế giao diện***. Sắp xếp lại các điều khiển và thay đổi các thuộc tính sao cho giao diện tuân thủ các hướng dẫn thiết kế bạn vừa học được.

d) ***Lưu project***. Chọn **File > Save All** để lưu lại thay đổi.

Bài tập nâng cao ▶ **4.14** ***(Giao diện Invoice)*** Trong bài tập này, bạn áp dụng các chỉ dẫn thiết kế giao diện vừa học được để thiết kế giao diện cho ứng dụng tính hóa đơn (Hình 4.25).



Hình 4.25 Ứng dụng **Invoice** chưa được áp dụng các chỉ dẫn thiết kế giao diện.

a) ***Copy template của ứng dụng vào thư mục làm việc***. Copy thư mục `C:\Example\Tutorial04\Exercises\MonitorInvoice` vào thư mục `C:\SimplyVB2008`.

b) ***Mở file template của ứng dụng***. Nhấn đúp vào file `MonitorInvoice.sln` trong thư mục `MonitorInvoice` để mở ứng dụng.

c) ***Áp dụng chỉ dẫn thiết kế giao diện***. Sắp xếp lại các điều khiển và thay đổi các thuộc tính sao cho giao diện tuân thủ các hướng dẫn thiết kế bạn vừa học được.

d) ***Lưu project***. Chọn **File > Save All** để lưu lại thay đổi.

CHƯƠNG 5

Mục tiêu

Trong chương này, bạn sẽ tìm hiểu để:

- Thêm xử lý sự kiện cho điều khiển **Button**.
- Viết mã cho xử lý sự kiện.
- Truy cập giá trị thuộc tính bằng mã Visual Basic.
- Sử dụng toán tử gán và toán tử nhân.
- Sử dụng IDE Visual Basic để hạn chế lỗi biên dịch.

Nội dung chính

Hoàn thiện ứng dụng Inventory

Giới thiệu lập trình

Chương này giới thiệu những kiến thức lập trình cơ bản để tạo ứng dụng có thể tương tác với người dùng. Bạn sẽ tìm hiểu những khái niệm này khi thêm chức năng (bằng mã Visual Basic) cho ứng dụng **Inventory** bạn đã thiết kế trong Chương 4. Thuật ngữ **chức năng** (**functionality**) mô tả hành động ứng dụng có thể thực thi. Trong chương này, bạn sẽ xem xét các **sự kiện** (**event**) trên giao diện đại diện cho hành động của người dùng như nhấn `Button` hoặc thay đổi giá trị trong `TextBox`, và **xử lý sự kiện** (**event handler**) là một đoạn mã sẽ được thực thi khi sự kiện xảy ra (hay còn gọi là sự kiện được kích hoạt). Bạn sẽ hiểu được vì sao sự kiện và xử lý sự kiện là thiết yếu đối với lập trình ứng dụng Windows.

5.1 Chạy thử ứng dụng Inventory

Trong chương này bạn sẽ hoàn thiện ứng dụng **Inventory** đã thiết kế trong Chương 4. Nhớ lại rằng ứng dụng này cần phải đáp ứng những yêu cầu sau:

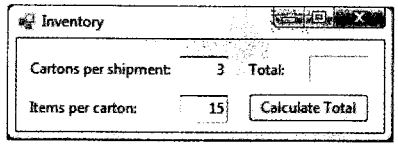
Yêu cầu đối với ứng dụng

Kho sách của trường đại học nhận về các kiện sách giáo khoa. Trong mỗi lô hàng, tất cả các kiện đều chứa cùng một số lượng sách. Quản lý kho muốn sử dụng máy tính để tính toán tổng số sách giáo khoa được chuyển tới kho sách trong mỗi lô hàng. Quản lý nhập số lượng kiện sách đã nhận và số lượng sách giáo khoa trong mỗi kiện sách của mỗi lô hàng, sau đó ứng dụng tính toán tổng số sách giáo khoa trong mỗi lô hàng.

Sau khi xem xét, quản lý kho đã đồng ý với thiết kế của bạn. Bây giờ, bạn phải thêm mã vào ứng dụng để khi người dùng nhấn `Button`, ứng dụng sẽ nhân số kiện với số sách giáo khoa chứa trong mỗi kiện và hiển thị kết quả là tổng số sách giáo khoa đã nhận được. Bạn sẽ bắt đầu bằng chạy thử ứng dụng đã hoàn thiện. Sau đó bạn tìm hiểu các tính năng bổ sung của Visual Basic cần thiết để tự xây dựng cho mình một phiên bản của ứng dụng này.

Chạy thử ứng dụng Inventory

1. ***Mở ứng dụng hoàn chỉnh***. Mở thư mục `C:\Example\Tutorial05\CompletedApplication\Inventory2` để tìm ứng dụng **Inventory**. Nhấn đúp `Invetory2.sln` để mở ứng dụng trong Visual Basic IDE.
2. ***Chạy ứng dụng Inventory***. Chọn **Debug > Start Debugging** để chạy ứng dụng (Hình 5.1). Nhập 3 vào `TextBox` **Cartons per shipment:**. Nhập 15 vào `TextBox` **Items per cartons:**. Hình 5.1 cho thấy `Form` sau khi nhập giá trị.

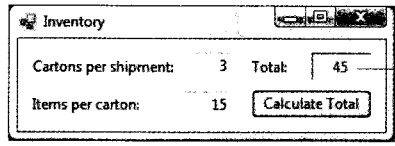


Chú thích hình

- ***Cartons per shipment:*** *Số kiện hàng trong mỗi lô hàng*
- ***Items per carton:*** *Số sách trong mỗi kiện hàng*

Hình 5.1 Ứng dụng **Inventory** sau khi đã nhập các giá trị.

3. ***Tính toán tổng số sách đã nhận được***. Nhấn `Button` **Calculate Total**. Ứng dụng nhân hai giá trị bạn vừa nhập vào và hiển thị kết quả (45) trên `Label` ở bên phải của **Total:** (Hình 5.2).



Hình 5.2 Kết quả sau khi nhấn `Button` **Calculate Total**.

4. ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.
5. ***Đóng IDE***. Chọn **File > Exit**.

5.2 Giới thiệu mã Visual Basic

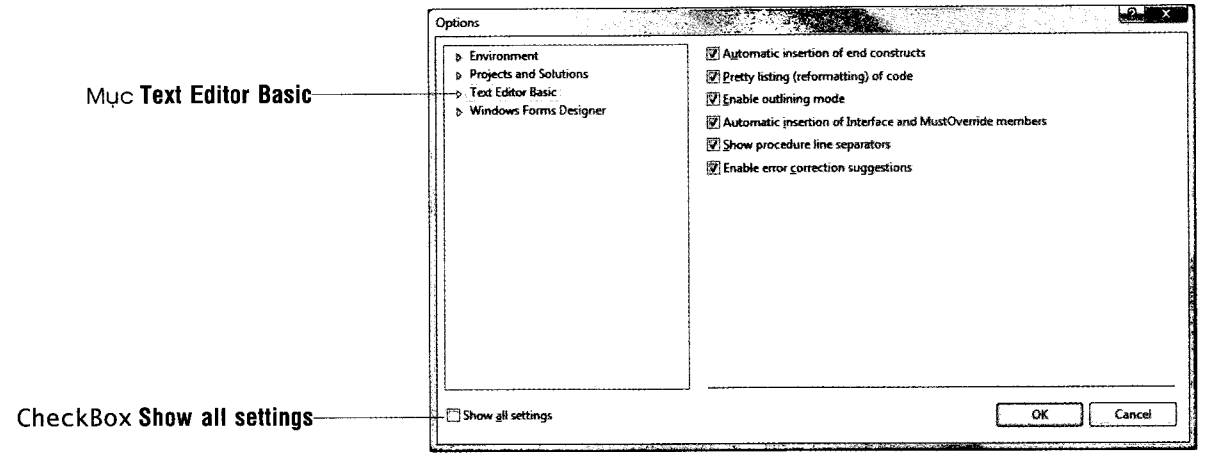
Trong Chương 3 và Chương 4, bạn đã được giới thiệu về khái niệm lập trình trực quan cho phép tạo ra giao diện đồ họa người dùng mà không cần viết một dòng mã nào. Trong phần này, bạn sẽ kết hợp lập trình trực quan với kỹ thuật lập trình thông thường để nâng cấp ứng dụng **Inventory**.

Trước khi bắt đầu xem và viết mã, bạn phải tùy chỉnh cách hiển thị và định dạng mã trên IDE. Trong phần tiếp theo, bạn sẽ mở template của ứng dụng và thay đổi thiết lập hiển thị và định dạng để dễ làm việc với mã và tuân theo hướng dẫn của chúng tôi hơn. Thêm chỉ số dòng, điều chỉnh độ rộng tab và thiết lập font và màu giúp cho bạn xem mã dễ dàng.

Tùy chỉnh IDE

1. ***Copy template của ứng dụng vào thư mục làm việc***. Copy `C:\Examples\Tutorial05\TemplateApplication\Inventory2` vào thư mục `C:\SimplyVB2008`. Thư mục này chứa ứng dụng đã được tạo ra bằng cách tuân theo các bước trong Chương 4.
2. ***Mở file template của ứng dụng Inventory***. Nhấn đúp vào file `Inventory2.sln` trong thư mục `Inventory2` để mở ứng dụng trong Visual Basic IDE. Nếu có lỗi xuất hiện khi bạn copy hoặc thay đổi template, hãy tham khảo với quản trị hệ thống để chắc chắn rằng bạn có quyền sửa đổi ứng dụng này.
3. ***Hiển thị số dòng***.Trong phần thảo luận về lập trình, chúng ta sẽ tham chiếu đến từng dòng mã cụ thể thông qua chỉ số dòng. Để giúp bạn xác định được vị trí thêm mã vào ví dụ, cần thiết lập để IDE hiển thị số dòng trong mã. Chọn **Tools > Options...**, trong hộp thoại **Options** mới xuất hiện (Hình 5.3), mở rộng mục **Text Editor Basic** bằng cách nhấn

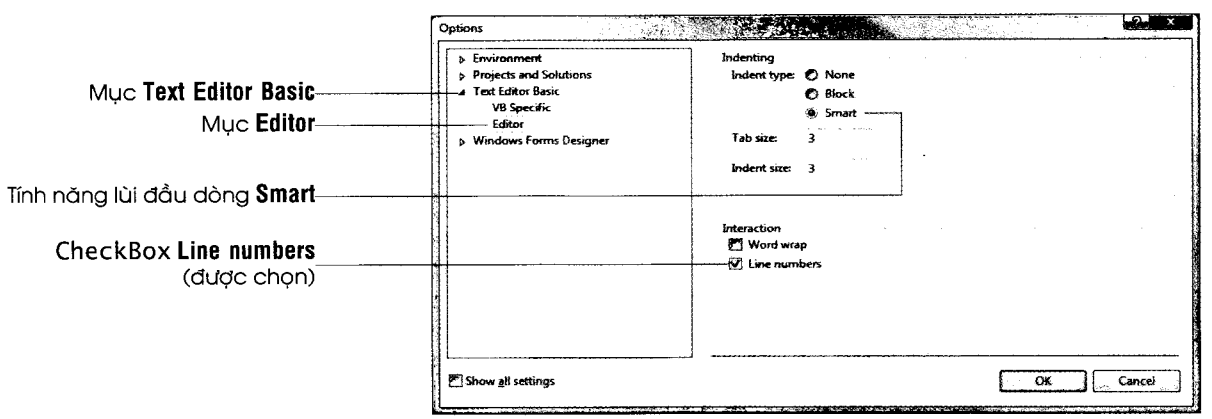
vào hình tam giác bên cạnh nó. [*Lưu ý*: Hãy chắc chắn rằng CheckBox **Show All settings** *không* được chọn, nếu không thì hộp thoại sẽ hiển thị nhiều mục khác nữa.] Chọn mục con **Editor** vừa xuất hiện (Hình 5.4) và tìm nhóm CheckBox **Interaction** trong mục này. Nếu CheckBox bên cạnh **Line numbers** chưa được chọn, nhấn vào bên trong CheckBox này để chọn. Nếu đã được chọn, bạn không cần làm gì nữa, tuy nhiên, đừng đóng hộp thoại.



Hình 5.3 Hộp thoại **Options**.

4. ***Thiết lập kích thước tab***. Giống như việc lùi đầu dòng dòng đầu tiên của mỗi đoạn văn khi bạn viết thư, sử dụng khoảng cách phù hợp khi viết mã rất quan trọng. Lùi đầu dòng mã làm cho chương trình dễ đọc hơn. Bạn có thể kiểm soát việc lùi dòng bằng cách sử dụng tab. Trong hộp thoại **Options** bạn vừa mở ở bước trước (Hình 5.4), nhập 3 cho cả hai trường tab size và indent size.

   Thiết lập kích thước tab chỉ ra số ký tự trắng mà mỗi ký tự tab đại diện. Thiết lập **Indent size:** xác định số ký tự trắng mà mỗi lùi đầu dòng được Visual Basic IDE tự động thêm vào. IDE thêm ba ký tự trắng nếu bạn đang sử dụng tính năng lùi dòng **Smart** (Hình 5.4) - bạn có thể tự mình thêm những ký tự này bằng cách nhấn phím *Tab*.



Hình 5.4 Trang thiết lập **General** cho trình xử lý văn bản của Visual Basic.

5. ***Tìm hiểu về font và màu***. Nhấn vào tam giác bên cạnh mục **Environment**, sau đó nhấn vào mục **Fonts** and **Colors** vừa xuất hiện. Màn hình tiếp theo cho phép bạn tùy chỉnh font và màu được sử dụng để hiển thị mã. Visual Basic IDE sử dụng màu và font giúp bạn bạn đọc và viết mã dễ dàng hơn. Lưu ý rằng, nếu thiết lập của bạn không phải là thiết lập mặc định, kết quả

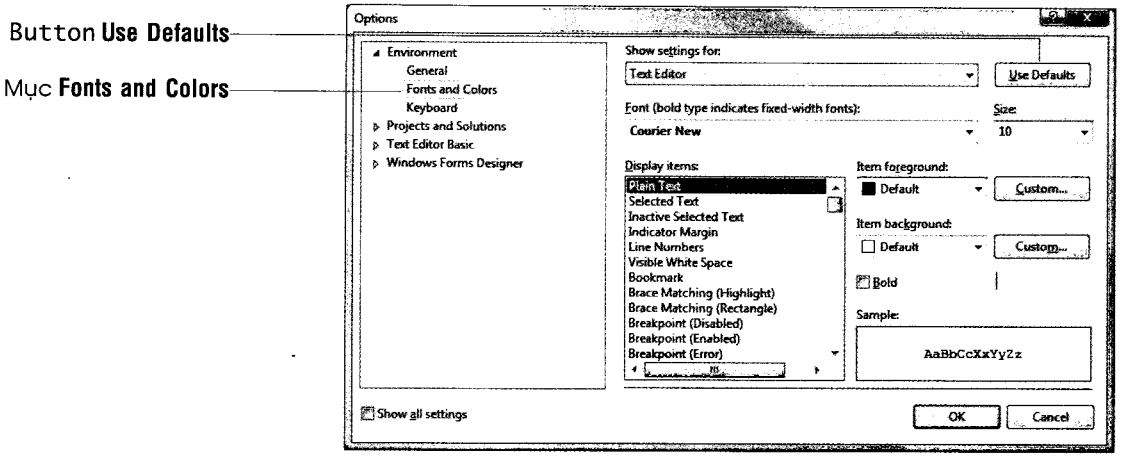
Thói quen lập trình tốt

Bạn có thể thay đổi thiết lập font, màu chữ nếu bạn muốn mã được hiển thị khác đi. Tuy nhiên, để thống nhất với cuốn sách này, chúng tôi đề nghị bạn không thay đổi thiết lập font và màu chữ mặc định.

mà bạn nhìn thấy trên màn hình sẽ không giống với những gì được giới thiệu trong sách này. Nếu bạn cần đặt lại thiết lập mặc định cho font và màu, nhấn `Button` **Use Defaults** (Hình 5.5).

Trong các ví dụ của sách này, bạn sẽ thấy dòng mã **Selected Text** (phần mã được chọn) được đặt nền màu vàng để nhấn mạnh. Thiết lập mặc định cho **Selected Text** là nền xanh nước biển. Bạn nên sử dụng thiết lập mặc định cho máy của mình.

6. ***Áp dụng các thay đổi***. Nhấn `Button` **OK** để áp dụng các thay đổi của bạn và đóng hộp thoại **Options**.



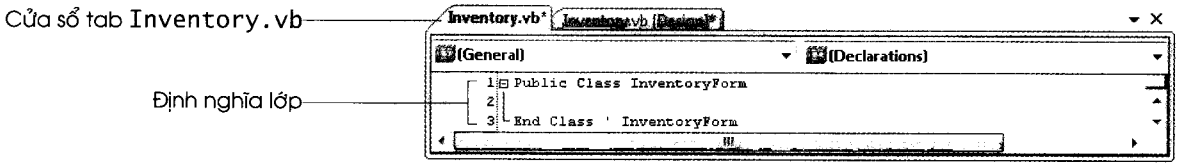
Hình 5.5 Trang **Fonts and Colors**.

Tiếp theo, bạn sẽ được làm quen với mã Visual Basic.

Giới thiệu mã Visual Basic

1. ***Xem mã ứng dụng***. Nếu Windows `Form` Designer chưa mở, nhấn đúp vào file `Inventory.vb` trong cửa sổ **Solution Explorer**. Sau đó chuyển sang **chế độ Code** (nơi mã ứng dụng được hiển thị trong cửa sổ soạn thảo) bằng cách chọn **View > Code** hoặc nhấn *F7*. Tab (`Inventory.vb`) trong Hình 5.6, còn được gọi là **trình soạn thảo mã** (**code editor**), xuất hiện. Lưu ý rằng khi bạn chọn **View > Code**, file `Inventory.vb` phải được chọn trong **Solution Explorer**.

Chú ý rằng IDE mà chúng tôi trình bày mã có thể khác so với IDE của bạn. Để dễ đọc hơn, chúng tôi chỉ trình bày một phần của trình soạn thảo mã và của cửa sổ tab.



Hình 5.6 IDE hiển thị mã của ứng dụng **Inventory**.

Hầu hết các chương trình Visual Basic đều gồm những phần nhỏ được gọi là lớp giúp đơn giản hóa tổ chức của ứng dụng. Nhớ lại từ Chương 1 rằng lớp chứa các nhóm lệnh thực hiện các nhiệm vụ và trả về thông tin khi những nhiệm vụ đó được hoàn thành. Mã trong ứng dụng này định nghĩa lớp ứng dụng **Inventory** của bạn. Tập hợp những dòng mã này được gọi là **định nghĩa lớp** (**class definition**). Hầu hết ứng dụng Visual Basic đều bao gồm tập hợp các dòng mã do lập trình viên viết ra và những lớp có sẵn do Microsoft viết và cung cấp trong .NET Framework Class Library. Chìa

khóa để phát triển ứng dụng Visual Basic thành công là sự kết hợp hợp lý của hai thành phần này. Bạn sẽ học cách sử dụng cả hai kỹ thuật đó trong các chương trình.

2. ***Tìm hiểu định nghĩa lớp.*** Dòng 1 (Hình 5.6) bắt đầu định nghĩa lớp. **Từ khóa `Class`** đưa ra định nghĩa lớp trong Visual Basic và ngay sau nó là **tên lớp** (trong ứng dụng này là `InventoryForm`, giá trị mà bạn nhập vào thuộc tính `Name` của `Form`).

   Tên của lớp là một **định danh (identifier)**, tức là một chuỗi ký tự bao gồm các chữ cái, chữ số và gạch dưới (_). Định danh không được bắt đầu bằng chữ số và không được chứa ký tự trắng. Thí dụ của định danh hợp lệ là `value1`, `label_Value` và `exitButton`. Tên `7welcome` là một định danh không hợp lệ, vì nó bắt đầu bằng chữ số, `input field` cũng là một định danh không hợp lệ vì nó chứa ký tự trắng. Định nghĩa lớp kết thúc ở dòng 3 với từ khóa **`End Class`**. **Từ khóa (keyword)** hay **từ dành riêng (reserved word)** là những từ được Visual Basic đăng ký sử dụng (bạn sẽ học thêm nhiều từ khóa khác nữa trong cuốn sách này). Lưu ý rằng, theo mặc định, các từ khóa có màu xanh nước biển. Bạn có thể tìm thấy danh sách các từ khóa đầy đủ trong Visual Basic ở ở Phụ lục B.

   Từ khóa `Class` đứng sau từ khóa `Public`. Mã cho mỗi `Form` bạn thiết kế trong IDE Visual Basic bắt đầu với từ khóa `Public`. Bạn sẽ học về từ khóa này trong Chương 19.

   Các từ khóa Visual Basic và các định danh không phân biệt **viết hoa** hay **viết thường (not case sensitive)**. Điều này có nghĩa là các chữ cái viết hoa và viết thường được xem như nhau đối với các định danh, vì vậy `InventoryForm` và `inventoryform` được Visual Basic hiểu là cùng một định danh. Mặc dù chữ cái đầu tiên của mỗi từ khóa đều được viết hoa, các từ khóa cũng không phân biệt viết hoa hay viết thường. IDE sẽ áp dụng cách viết đúng cho mỗi chữ cái của từ khóa và định danh, vì vậy khi bạn gõ `classS`, nó sẽ tự động thay đổi thành `Class` khi bạn nhấn phím *Enter*.

Thói quen lập trình tốt

Bạn nên viết hoa chữ cái đầu tiên của định danh lớp, ví dụ như tên `Form`.

TỰ ÔN TẬP

1. Định danh_____.
   a) có thể bắt đầu với bất kỳ chữ cái nào, nhưng không được chứa ký tự trắng
   b) phải bắt đầu với một chữ số, nhưng không được chứa ký tự trắng
   c) không thể bắt đầu bằng một chữ số hay chứa ký tự trắng
   d) không thể bắt đầu bằng một chữ số, nhưng có thể chứa ký tự trắng
2. Từ khóa Visual Basic_____.
   a) phân biệt viết hoa hay viết thường
   b) là các chú thích
   c) không phải là những từ được chỉ định sẵn
   d) không phân biệt viết hoa hay viết thường

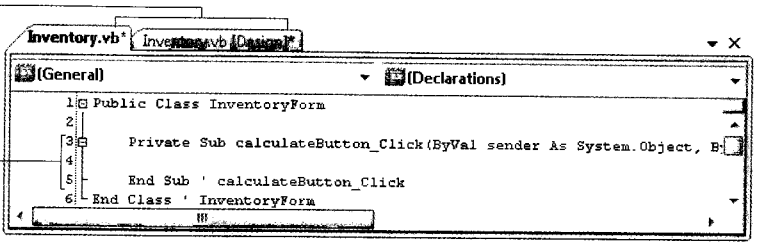
Đáp án: 1) c. 2) d.

5.3 Thêm xử lý sự kiện

Bây giờ khi đã hoàn thiện giao diện, bạn đã sẵn sàng thay đổi ứng dụng để xử lý đầu vào của người dùng. Bạn sẽ làm điều này bằng cách viết mã. Hầu hết ứng dụng Visual Basic trong sách này đều cung cấp chức năng dưới dạng xử lý sự kiện. Nhớ lại rằng xử lý sự kiện thực thi khi sự kiện được kích hoạt, như việc nhấn `Button`. Phần tiếp theo cho bạn thấy cách thêm xử lý sự kiện vào ứng dụng.

Thêm xử lý sự kiện Click cho Button

1. ***Thêm xử lý sự kiện cho Button***. Trong bước này bạn sử dụng Windows Form Designer để tạo xử lý sự kiện và chuyển sang chế độ **Code**. Bắt đầu bằng cách nhấn tab `Inventory.vb [Design]` để xem Windows Form Designer. Sau đó nhấn đúp vào `Button` **Calculate Total** của `Form` để vào chế độ **Code**. Lưu ý rằng mã cho ứng dụng bây giờ có thêm phần xử lý sự kiện mới ở dòng 3-5 trên Hình 5.7.



Hình 5.7 Xử lý sự kiện `calculateButton_Click` trước khi bạn thêm mã lập trình của mình.

Nhấn đúp vào `Button` **Calculate Total** trong chế độ `Design` để Visual Basic IDE tạo xử lý sự kiện **`Click`** của `Button` - mã này sẽ thực thi khi người dùng nhấn `Button` **Calculate Total**. Khi bạn nhấn đúp vào điều khiển, IDE thêm xử lý sự kiện cho điều khiển đó (hoặc hiển thị xử lý sự kiện nếu nó đã có). [*Lưu ý*: Nếu vô tình tạo ra một xử lý sự kiện và không muốn sử dụng nó, bạn có thể đơn giản là xóa đi mã đã được sinh ra]. Mỗi loại điều khiển sẽ có các loại sự kiện khác nhau. Chẳng hạn, nhấn đúp vào điều khiển `Button` sẽ tạo ra xử lý sự kiện `Click` còn nhấn đúp vào điều khiển khác sẽ tạo ra loại xử lý sự kiện khác. Mỗi điều khiển có một kiểu xử lý sự kiện mặc định được sinh ra khi bạn nhấn đúp lên điều khiển đó trong chế độ **Design**.

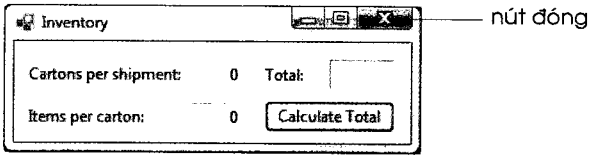
Visual Basic thêm mệnh đề `Handles` ở cuối dòng đầu tiên của mỗi xử lý sự kiện. Cuộn về bên phải trong chế độ **Code** để thấy mệnh đề `Handles` trong xử lý sự kiện `Click` của `Button` **Calculate Total** (dòng 3)

```
Handles calculateButton.Click
```

Mệnh đề `Handles` chỉ ra rằng xử lý sự kiện được gọi ra khi sự kiện `Click` của `calculateButton` được kích hoạt.

Trong Visual Basic, xử lý sự kiện thường tuân thủ quy ước đặt tên *controlName_eventName*. Quy ước này tương tự với mệnh đề `Handles` của xử lý sự kiện; trong đó *ControlName* là tên của điều khiển (giá trị thuộc tính `Name` của điều khiển, trong trường hợp này là `calculateButton`) và *eventName* là tên của sự kiện (trong trường hợp này là `Click`) do điều khiển tạo ra. Khi sự kiện *eventName* được kích hoạt bởi điều khiển *controlName*, xử lý sự kiện *controlName_eventName* sẽ thực thi. Trong ứng dụng này, `calculateButton_Click` xử lý sự kiện `Click` của `Button` **Calculate Total** - nói cách khác, mã trong `calculateButton_Click` sẽ thực thi khi người dùng nhấn `Button` **Calculate Total**.

2. ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng (Hình 5.8). IDE tự động lưu công việc của bạn trước khi chạy ứng dụng. Nhấn `Button` **Calculate Total**.



Hình 5.8 Chạy ứng dụng chưa có chức năng

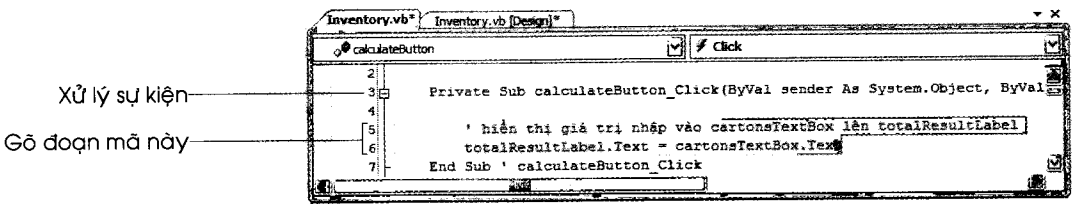
Mặc dù bạn vừa thêm xử lý sự kiện cho sự kiện `Click` của `Button`, không có hành động nào xảy ra khi bạn nhấn `Button` vì bạn chưa lập trình cho xử lý sự kiện. Trong phần tiếp theo, bạn sẽ viết mã cho xử lý sự kiện sao cho khi người dùng nhấn `Button`, văn bản sẽ được hiển thị trên `Label` đầu ra (`totalResultLabel`).

3. ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng - bạn không thể viết mã khi ứng dụng đang chạy.

Lúc này khi đã tạo xử lý sự kiện cho `Button` **Calculate Total**, bạn cần thêm mã để thực hiện hành động. Trong trường hợp này, bạn cần làm cho ứng dụng nhân số kiện sách trong lô hàng với số đầu sách trong mỗi kiện khi người dùng nhấn `Button` **Calculate Total**. Bạn sẽ viết câu lệnh Visual Basic đầu tiên trong phần tiếp theo.

Thêm mã vào xử lý sự kiện rỗng

1. ***Chuyển sang chế độ Code***. Nếu bạn còn chưa ở chế độ **Code**, hãy chọn **View > Code** để xem mã của ứng dụng.
2. ***Thêm mã cho xử lý sự kiện***. Trong xử lý sự kiện, thêm dòng 5-6 trên Hình 5.9 bằng cách gõ phần văn như trong hình. Thêm chú thích ở dòng 7 đứng sau từ khóa `End Sub`.



Hình 5.9 Mã được thêm vào xử lý sự kiện của `Button` **Calculate Total.**

Dòng 5 trên Hình 5.9 bắt đầu **bằng ký tự nháy đơn** (') chỉ ra rằng phần còn lại của dòng mã là **chú thích (comment)**. Bạn thêm chú thích vào chương trình để mã dễ đọc hơn. Những chú thích này giải thích cho mã để những lập trình viên khác khi làm việc với ứng dụng có thể dễ dàng hiểu được. Theo mặc định, chú thích có màu xanh lá cây.

Thói quen lập trình tốt

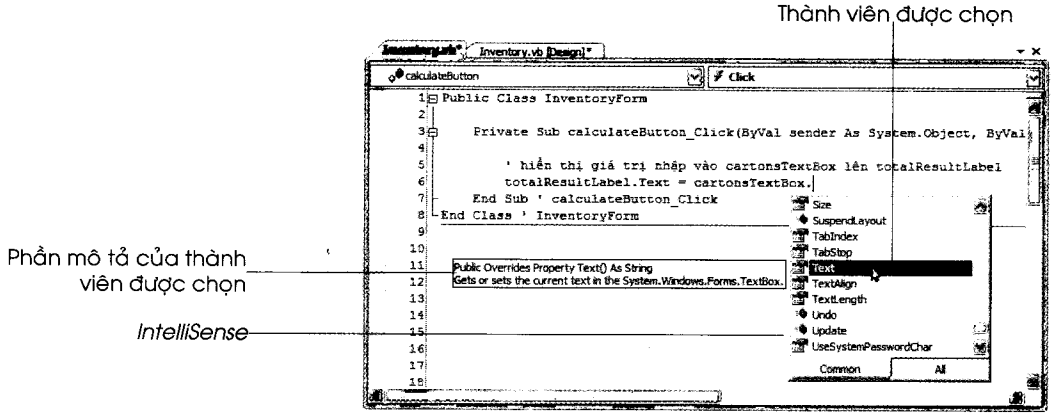
Những chú thích được viết ở cuối dòng lệnh nên có khoảng trống ở phía trước để chương trình dễ đọc hơn.

Chú thích cũng giúp bạn hiểu được mã của mình, nhất là trong trường hợp bạn không xem lại sau một thời gian. Chú thích có thể được đặt trên một dòng riêng (được gọi là "chú thích toàn dòng") hoặc được đặt sau một dòng mã Visual Basic (được gọi là "chú thích cuối dòng").

Trình biên dịch của Visual Basic bỏ qua chú thích - nghĩa là chúng không làm cho máy tính thực hiện bất cứ một hành động nào khi ứng dụng chạy. Chú thích ở dòng 5 đơn giản chỉ ra rằng dòng tiếp theo hiển thị giá trị nhập vào `TextBox` **Cartons per shipment:** trên `Label` **Total:**. Chú thích được hiển thị trong trình soạn thảo mã của Visual Basic IDE có màu xanh lá cây.

Dòng 6 trên Hình 5.9 là **câu lệnh (statement)** thực thi đầu tiên của bạn, câu lệnh này thực hiện một hành động. Theo mặc định, mỗi câu lệnh được viết trên một dòng. Phần sau trong chương này, bạn sẽ biết cách viết một câu lệnh dài hơn một dòng. Lệnh này (dòng 6) truy cập thuộc tính `Text` của `cartonsTextBox` và của `totalResultLabel`. Trong mã Visual Basic, các thuộc tính được truy cập bằng cách đặt một dấu chấm giữa tên điều khiển (chẳng hạn, `totalResultLabel`) và tên thuộc tính

(ví dụ, `Text`). Dấu chấm này được đặt ở bên phải của tên điều khiển, được gọi là **toán tử truy cập thành viên** (**member-access operator**) hay **toán tử chấm** (**dot operator**). Khi bạn gõ tên điều khiển và toán tử truy cập thành viên, sẽ xuất hiện cửa sổ liệt kê các thành viên của đối tượng (Hình 5.10). Tính năng tiện dụng này được gọi là ***IntelliSense***, hiển thị các thành phần có sẵn trong chương trình và có thể được sử dụng trong ngữ cảnh hiện tại, chẳng hạn như tất cả các thành viên của đối tượng. Cửa sổ *IntelliSense* cũng có thể được mở bằng cách nhấn *Ctrl* + *Space*. Bạn cuộn đến thành viên mình quan tâm và chọn thành viên đó. Bạn cũng có thể tiếp tục gõ để giới hạn số lượng lựa chọn. Nhấn vào tên thành viên để hiển thị mô tả của thành viên, nhấn đúp vào nó để thêm thành viên đó vào ứng dụng. Bạn cũng có thể nhấn *Enter* hoặc *Tab* để thêm thành viên đó - *Tab* thêm thành viên còn *Enter* thêm thành viên kèm thêm một dòng mới. *IntelliSense* rất hữu ích trong việc tìm hiểu các thành viên của lớp và ý nghĩa của chúng. Lưu ý rằng cửa sổ *IntelliSense* trên Hình 5.10 hiển thị hai tab - **Common** và **All**. Tab **Common** hiển thị những thành viên thường được sử dụng có thể xuất hiện ở bên phải toán tử chấm. Tab **All** hiển thị mọi thành viên có thể xuất hiện ở bên phải của toán tử dấu chấm. Bạn có thể đóng cửa sổ *IntelliSense* bằng cách nhấn *Esc*.



Hình 5.10 *IntelliSense* được kích hoạt khi viết mã.

Hãy xem xét kỹ hơn dòng 6 trên Hình 5.9. Đọc dòng này từ trái sang phải, ta thấy thuộc tính `Text` của `totalResultLabel` đứng trước dấu bằng (=), dấu bằng này đứng trước giá trị thuộc tính `Text` của `cartonsTextBox`. Ký hiệu "=" như được dùng ở đây, được gọi là **toán tử gán** (**assignment operator**). Các biểu thức ở hai bên của toán tử gán được gọi là **toán hạng** (**operand**). Toán tử gán này gán giá trị ở bên phải của toán tử (**toán hạng phải**) cho biến ở bên trái của toán tử (**toán hạng trái**). Toán tử gán còn được gọi là **toán tử hai ngôi** vì nó có hai toán hạng - trong trường hợp này là `totalResultLabel.Text` và `cartonsTextBox.Text`.

Toàn bộ câu lệnh này được gọi là **lệnh gán** (**assignment statement**) vì nó gán giá trị cho toán hạng bên trái. Trong ví dụ này bạn đã gán giá trị của thuộc tính `Text` của `cartonsTextBox` cho thuộc tính `Text` của `totalResultLabel`. Câu lệnh có nghĩa là, "Thuộc tính `Text` của `totalResultLabel` *nhận* giá trị của thuộc tính `Text` của `cartonsTextBox`". Lưu ý rằng lệnh gán không thay đổi toán hạng phải.

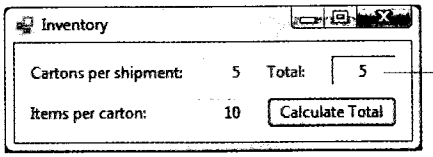
Khi người dùng nhấn `Button` **Calculate Total**, xử lý sự kiện được thực thi, hiển thị giá trị người dùng nhập vào `TextBox` **Cartons per shipment:** trên `Label` đầu ra `totalResultLabel`. Rõ ràng là, đó không phải là kết quả đúng - kết quả đúng là số đơn vị hàng hóa trong một kiện nhân với số

Thói quen lập trình tốt

Thêm chú thích phía sau từ khóa End Sub để cho biết sự kiện nào kết thúc.

kiện có trong lô hàng. Bạn sẽ sửa lỗi này trong phần *Hoàn chỉnh ứng dụng* **Inventory**. Lưu ý rằng chúng ta vừa thêm chú thích ở dòng 7 trên Hình 5.9 để chỉ ra rằng xử lý sự kiện kết thúc ở đây.

3. ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng (Hình 5.11). Nhập 5 vào `TextBox` **Cartons per shipment:** và 10 vào `TextBox` **Items per carton:**, sau đó nhấn `Button` **Calculate Total**. Văn bản của `totalResultLabel` lúc này hiển thị kết quả không đúng. Kết quả hiển thị là 5 (giá trị được nhập vào `TextBox` **Cartons per shipment:**) trong khi kết quả mong muốn là 50. Bạn sẽ chữa lỗi này trong phần tiếp theo.



Hình 5.11 Chạy ứng dụng sau khi viết mã cho phần xử lý sự kiện.

4. ***Đóng ứng dụng.*** Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

TỰ ÔN TẬP

1. Với controlName là tên điều khiển và eventName là tên sự kiện do điều khiển tạo ra. Xử lý sự kiện được Visual Basic IDE sinh ra tuân thủ quy ước_____.
   a) *controlName_eventName* b) *eventName_controlName*
   c) *eventNameControlName* d) *controlNameEventName*
2. Biểu thức ở hai bên của toán tử gán được gọi là _____của nó.
   a) giá trị toán tử b) kết quả
   c) toán hạng d) đối số

Đáp án: 1) a. 2) c.

5.4 Thực hiện phép tính và hiển thị kết quả

Bây giờ, khi đã làm quen với hiển thị kết quả đầu ra trên `Label`, bạn sẽ hoàn thiện ứng dụng **Inventory** bằng cách hiển thị kết quả của phép nhân số kiện trong lô và số sách trong mỗi kiện. Trong phần tiếp theo, bạn sẽ học cách thực hiện thao tác toán học trong Visual Basic.

Hoàn thiện ứng dụng Inventory

Thói quen lập trình tốt

Một câu lệnh dài có thể viết trên nhiều dòng. Nếu một lệnh đơn phải viết thành nhiều dòng, hãy chọn điểm ngắt dòng dễ hiểu, ví dụ như sau toán tử. Nếu lệnh được viết trên hai hay nhiều dòng, hãy thụt đầu dòng những dòng phía sau.

1. ***Thay đổi xử lý sự kiện***. Nếu bạn chưa ở trong chế độ **Code**, hãy chọn **View > Code** hoặc nhấn tab `Inventory.vb`. Thêm ký tự nối dòng (_) ở bên phải các dòng 3-4 với ít nhất một ký tự trắng trước đó, như trên Hình 5.12. Lùi đầu dòng các dòng 4-5. Ký tự gạch dưới là **ký tự nối dòng (line-continuation character)**. Ký tự này chỉ ra rằng dòng tiếp theo là sự tiếp tục của dòng trước đó. Một lệnh có thể chứa nhiều ký tự nối dòng. Tuy nhiên, phải có ít nhất một ký tự trắng đứng trước ký tự nối dòng và chỉ có ký tự khoảng trắng (whitespace character) xuất hiện phía sau ký tự nối dòng. Khoảng trắng là ký tự trắng (space character), ký tự tab hoặc ký tự dòng mới (là ký tự được thêm bằng cách nhấn phím *Enter*). Bạn sử dụng ký tự nối dòng ở dòng 3-4 để tách lệnh đầu tiên của xử lý sự kiện thành ba dòng - việc này làm cho tất cả mã của chương trình nằm gọn trong cửa sổ. Ký tự nối dòng không có hiệu lực khi đặt sau chú thích.

   Thay thế thân xử lý sự kiện `calculateButton_Click` bằng đoạn mã ở dòng 7-9. Chú thích ở dòng 7 chỉ ra rằng bạn sẽ nhân hai giá trị mà người dùng nhập vào và hiển thị kết quả trên `Label`.

Lỗi lập trình thường gặp

Viết một lệnh trên nhiều dòng mà không sử dụng ký tự nối dòng sẽ gây ra lỗi biên dịch.

Mã của ứng dụng **Inventory** đã được thay đổi

```
1  Public Class InventoryForm
2
3      Private Sub calculateButton_Click( _
4          ByVal sender As System.Object, ByVal e As System.EventArgs) _
5          Handles calculateButton.Click
6
7          'nhân giá trị nhập vào và hiển thị kết quả trên Label
8          totalResultLabel.Text = _
9              Val(cartonsTextBox.Text) * Val(itemsTextBox.Text)
10     End Sub ' calculateButton_Click
11 End Class ' InventoryForm
```

Hình 5.12 Sử dụng phép nhân trong ứng dụng **Inventory**.

Lỗi lập trình thường gặp

Đặt ký tự không phải là ký tự khoảng trắng kể cả chú thích sau ký tự nối dòng sẽ gây ra lỗi biên dịch. Lỗi biên dịch sẽ được giới thiệu trong phần Sử dụng IDE để hạn chế lỗi biên dịch trong Mục 5.5.

2. ***Thêm mã thực hiện phép nhân***. Dòng 8-9 thực hiện phép nhân và phép gán. Một lần nữa, bạn sử dụng toán tử gán để gán giá trị cho `totalResultLabel.Text` ở dòng 8. Ký tự nối dòng ở dòng 8 chỉ ra rằng lệnh được tiếp tục sau dòng hiện tại, vì vậy hãy xem toán hạng phải của phép gán ở dòng tiếp theo.

Toán tử gán ở dòng 8 gán kết quả của phép nhân các số do người dùng nhập vào cho `totalResultLabel.Text`. Ở dòng 9, biểu thức `Val(cartonsTextBox.Text)` đứng trước một dấu sao (*) sau đó là biểu thức `Val(itemsTextBox.Text)`. Dấu sao là **toán tử nhân** - toán hạng trái và phải của toán tử được nhân với nhau.

Ứng dụng **Inventory** của bạn không thể ngăn cản người dùng khỏi việc vô tình nhập vào những ký tự không phải số, ví dụ như chữ cái hoặc ký tự đặc biệt như $ hay @. Dòng 9 sử dụng **hàm `Val`** để ngăn không cho những đầu vào đó làm ứng dụng ngừng hoạt động. Hàm là đoạn mã thực hiện một nhiệm vụ khi được gọi (được thực thi) và trả về một giá trị đến nơi mà nó được gọi. Trong trường hợp này, giá trị được `Val` trả về trở thành giá trị được sử dụng trong biểu thức nhân (dòng 9). Bạn gọi một hàm (như ở dòng 9) bằng cách gõ tên của hàm cùng với hai dấu ngoặc đơn. Bất cứ giá trị nào bên trong dấu ngoặc đơn (chẳng hạn, `cartonsTextBox.Text`) đều được gọi là **đối số** (**argument**) của hàm. Đối số là đầu vào của hàm, cung cấp thông tin hàm cần để thực hiện nhiệm vụ của nó. Trong trường hợp này, đối số chỉ ra giá trị mà bạn muốn truyền vào hàm `Val`. Bạn sẽ học cách tự tạo hàm trong Chương 13.

Hàm `Val` lấy về giá trị từ chuỗi ký tự (đầu vào từ bàn phím). Giá trị này được đảm bảo là một giá trị số. Chúng ta sử dụng `Val` vì ứng dụng không thể thực hiện phép tính số học đối với toán tử không phải số. `Val` đọc từng ký tự trong đối số cho tới khi bắt gặp một ký tự không phải là số. Nếu bắt gặp ký tự không phải là số, `Val` trả về số mà nó đã đọc cho đến thời điểm đó. `Val` bỏ qua ký tự trắng (chẳng hạn, "33 5" sẽ được hiểu thành 335). Hình 5.13 là ví dụ về các lệnh gọi tới `Val` và kết quả của những lệnh gọi đó. `Val` nhận biết dấu chấm thập phân là một ký tự số, dấu cộng và dấu trừ khi chúng xuất hiện ở đầu chuỗi (để chỉ ra một số là dương hay là âm). `Val` không nhận biết các ký hiệu như dấu phẩy và $. Nếu hàm `Val` nhận vào một đối số không thể chuyển đổi thành số (chẳng hạn, "b35", bắt đầu bằng một ký tự chứ không phải số), hàm sẽ trả về 0. Kết quả của phép tính được gán cho `totalResultLabel.Text` (dòng 8), để hiển thị kết quả cho người dùng.

Hãy thận trọng khi sử dụng `Val` - mặc dù giá trị trả về là một số, giá trị đó thường không là giá trị mà người dùng mong muốn (xem Hình 5.13). Nếu dữ liệu do người dùng nhập vào không đúng, `Val` sẽ không chỉ ra được lỗi. Hàm này sẽ trả về một giá trị (thường không phải là giá trị mà người dùng mong muốn) và ứng dụng tiếp tục chạy, có thể vẫn sử dụng giá trị không đúng đó cho việc tính toán. Chẳng hạn, một người có

thể nhập vào tổng số tiền là `$10.23`[1] và `Val` coi là 0. Chú ý rằng một lỗi thông thường sẽ làm cho ứng dụng thực thi không chính xác. Visual Basic cung cấp hai cách để quản lý đầu vào không hợp lệ. Một cách là sử dụng khả năng năng xử lý chuỗi của Visual Basic để kiểm tra đầu vào. Bạn sẽ học về những tính năng đó khi đọc cuốn sách này. Một cách khác để quản lý đầu vào không hợp lệ, được gọi là xử lý exception (ngoại lệ), trong đó bạn viết mã để quản lý những lỗi có thể xuất hiện khi ứng dụng thực thi.

3. ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng. Bây giờ người dùng có thể nhập dữ liệu vào cả hai `TextBox`. Khi `Button` **Calculate Total** được nhấn, ứng dụng nhân hai số đã được nhập vào và hiển thị kết quả trên `totalResultLabel`.
4. ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.
5. Lưu project. Chọn **File > Save All** để lưu mã đã thay đổi.

Ví dụ lời gọi hàm Val	Kết quả
`Val("16")`	16
`Val("-3")`	-3
`Val("1.5")`	1.5
`Val("67a4")`	67
`Val("8+5")`	8
`Val("14 Main St.")`	14
`Val("+1 2 3 4 5")`	12345
`Val("hello")`	0

Hình 5.13 Ví dụ lời gọi hàm `Val`.

Hình 5.14 thể hiện mã nguồn ứng dụng **Inventory**. Những dòng mã chứa các khái niệm lập trình mới mà bạn đã học trong chương này được đánh dấu.

```
1  Public Class InventoryForm
2
3     Private Sub calculateButton_Click( _
4        ByVal sender As System.Object, ByVal e As System.EventArgs) _
5        Handles calculateButton.Click
6
7        'nhận giá trị nhập vào và hiển thị kết quả trên Label
8        totalResultLabel.Text = _
9        Val(cartonsTextBox.Text) * Val(itemsTextBox.Text)
10    End Sub ' calculateButton_Click
11 End Class ' InventoryForm
```

Hình 5.14 Mã nguồn ứng dụng **Inventory**.

1: Trong cuốn sách này, chúng tôi sử dụng quy cách viết số tiếng Anh - dùng ký tự "," để phân cách các lớp trong số nguyên và ký tự "." để thể hiện phần nguyên và phần thập phân trong số thập phân - đối với những trường hợp tham chiếu trực tiếp tới mã chương trình hay nội dung trong form của ứng dụng và sử dụng quy cách viết số tiếng Việt trong các trường hợp còn lại.

TỰ ÔN TẬP

1. _____cung cấp dữ liệu mà hàm cần để thực hiện nhiệm vụ của chúng.
   a) Chú thích
   b) Đối số
   c) Đầu ra
   d) Cả a và b
2. Kết quả của `Val ("%5")` là gì?
   a) 5
   b) 0
   c) 500
   d) 0.05

Đáp án: 1) b. 2) b.

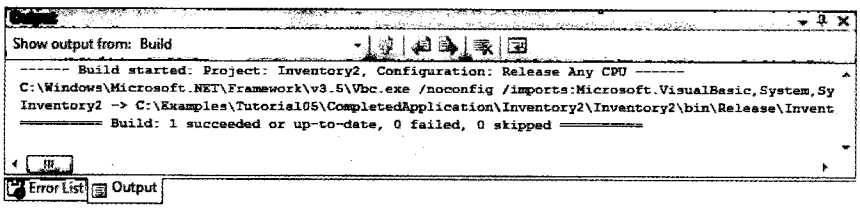
5.5 Sử dụng IDE để hạn chế lỗi biên dịch

Cho tới phần này của cuốn sách, bạn đã thực thi ứng dụng bằng cách chọn **Debug > Start Debugging** để biên dịch và chạy ứng dụng. Nếu bạn không viết mã chính xác, lỗi sẽ xuất hiện trong cửa sổ **Error List**. **Gỡ lỗi** (**debugging**) là quá trình sửa lỗi ứng dụng. Có hai loại lỗi - lỗi biên dịch và lỗi lôgíc.

Lỗi biên dịch (**compilation error**) xuất hiện khi lệnh vi phạm các quy tắc ngữ pháp của ngôn ngữ lập trình hoặc chỉ đơn giản là khi lệnh không đúng trong ngữ cảnh hiện thời. Ví dụ, lỗi biên dịch thường là do viết sai từ khóa hoặc định danh, không dùng ký tự nối dòng khi tách lệnh thành nhiều dòng hoặc sử dụng một định danh sai ngữ cảnh. Một ứng dụng không thể thực thi cho đến khi tất cả các lỗi biên dịch được sửa. Một nhóm nhỏ của lỗi biên dịch được gọi là các **lỗi cú pháp** (**syntax error**). Đây là những lỗi đặc biệt, vi phạm quy tắc ngữ pháp của ngôn ngữ lập trình, ví dụ như không sử dụng ký tự nối dòng khi chia một lệnh thành nhiều dòng.

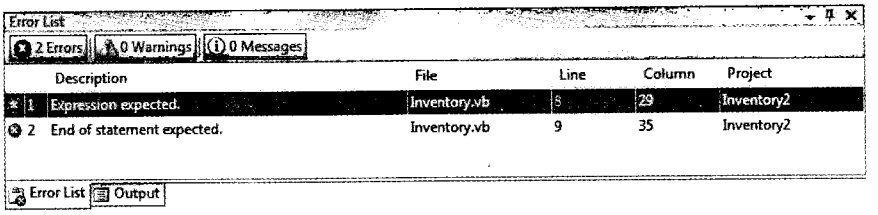
Lỗi lôgíc (**logic error**) không ngăn cản ứng dụng được biên dịch thành công, nhưng làm cho ứng dụng đưa ra kết quả sai. IDE Visual Basic có **trình gỡ lỗi** (**debugger**) cho phép bạn phân tích hoạt động của ứng dụng để xác định xem nó thực thi có đúng hay không.

Bạn có thể biên dịch ứng dụng mà không cần thực thi nó bằng cách chọn **Build > Build** [*Project Name*], trong đó *Project Name* là tên của project hiện thời của bạn. Các lập trình viên thường làm như vậy khi họ muốn xác định trong mã của họ có lỗi biên dịch hay không. Sử dụng **Debug > Start Debugging** hoặc là **Build > Build** [*Project Name*] sẽ hiển thị mọi lỗi biên dịch trong cửa sổ **Error List**. Cửa sổ **Output** hiển thị kết quả biên dịch. Nếu cửa sổ này không xuất hiện, hãy chọn **Debug > Windows > Output** để nó xuất hiện khi gỡ lỗi. Hình 5.15 cho thấy cửa sổ output của một ứng dụng không có lỗi.



Hình 5.15 Kết quả biên dịch thành công trong cửa sổ **Output**.

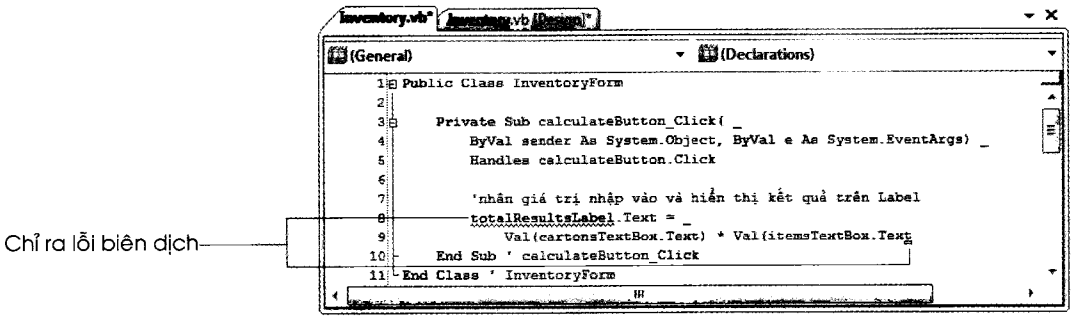
Trong IDE Visual Basic, lỗi biên dịch xuất hiện trong cửa sổ **Error List** cùng với mô tả từng lỗi. Hình 5.16 hiển thị lỗi xuất hiện khi quên thêm ký tự nối dòng cho một lệnh nhiều dòng. Nhấn đúp vào lỗi trong cửa sổ **Error List** để xác định vị trí của lỗi trên mã nguồn. Để có thêm thông tin về lỗi biên dịch, nhấn chuột phải vào lệnh bị lỗi trong cửa sổ **Error List** và chọn **Show Error Help**. Lệnh này hiển thị trang trợ giúp, giải thích thông báo lỗi và hướng dẫn cách khắc phục. Tiếp theo bạn sẽ tạo ra các lỗi biên dịch, xem kết quả và sửa các lỗi đó.



Hình 5.16 Cửa sổ **Error List** liệt kê các lỗi biên dịch.

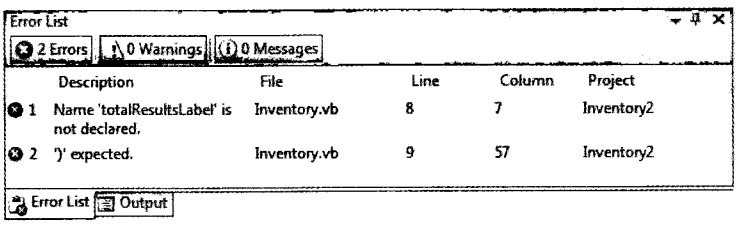
Sử dụng IDE để hạn chế lỗi biên dịch

1. ***Mở ứng dụng hoàn chỉnh***. Nếu ứng dụng **Inventory** đang không được mở, hãy tìm file `Inventory2.sln`, sau đó nhấn đúp để tải ứng dụng vào IDE.
2. ***Tạo lỗi biên dịch***. Bây giờ bạn sẽ tạo lỗi biên dịch cho mục đích tìm lỗi và sửa lỗi sau này. Nếu bạn chưa ở chế độ **Code**, hãy chọn **View > Code**. Mở cửa sổ **Error List** bằng cách chọn **View > Error List**. Thêm một ký tự ("s") vào `Label totalResultLabel` ở dòng 8 và xóa dấu ngoặc đơn bên phải ở cuối lệnh gán ở dòng 9. Hãy chú ý những thay đổi đối với IDE (Hình 5.17).



Hình 5.17 IDE có lỗi biên dịch.

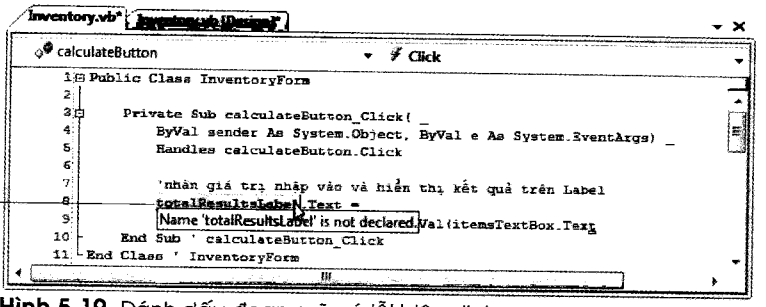
Visual Basic IDE cung cấp khả năng **kiểm tra lỗi thời gian thực** (**real-time error checking**). Khi viết mã trong trình soạn thảo mã, bạn có thể nhận thấy rằng các lỗi biên dịch được thông báo tức thời trong **Error List**. Vị trí chính xác lỗi trong mã của bạn cũng được đánh dấu bằng đường gạch chân hình răng cưa màu xanh. Cửa sổ **Error List** thông báo không nhận diện được định danh `totalResultsLabel` và thiếu dấu ngoặc đơn (Hình 5.18).



Hình 5.18 Error List hiển thị lỗi biên dịch.

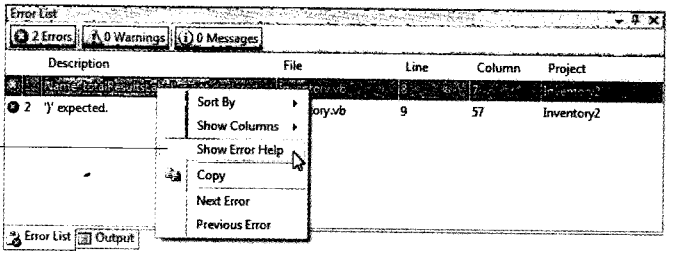
Những tính năng này thông báo cho bạn về những lỗi có thể có và cho bạn cơ hội khắc phục chúng trước khi biên dịch ứng dụng. IDE không chạy ứng dụng có những lỗi mà bạn đã tạo ở phần trên cho đến khi *tất cả* lỗi biên dịch đã được sửa.

3. ***Tìm lỗi biên dịch.*** Nhấn đúp vào một lỗi trong cửa sổ **Error List** để xác định phần mã chứa lỗi này. Nhấn đúp vào lỗi đầu tiên, lỗi ở dòng 8 được đánh dấu (Hình 5.19). Đặt con trỏ lên lỗi biên dịch để hiển thị thông báo lỗi.

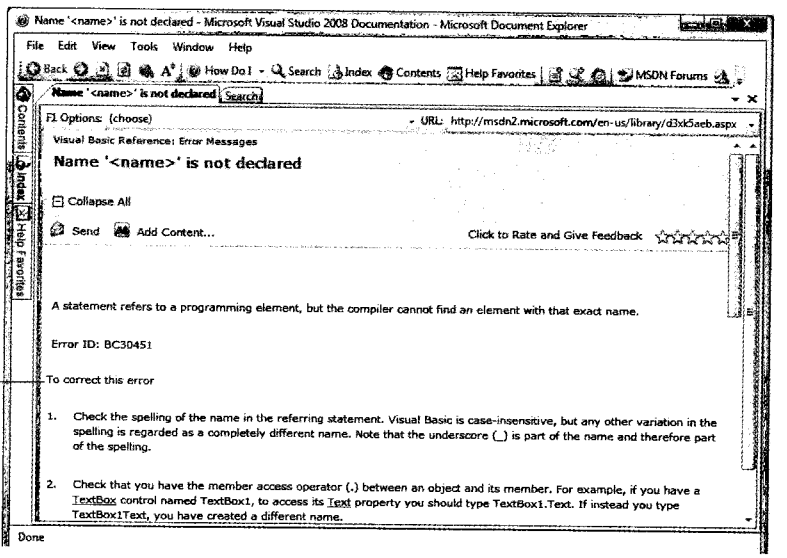


Hình 5.19 Đánh dấu đoạn mã có lỗi biên dịch.

4. ***Sử dụng trợ giúp bổ sung.*** Bạn cũng được cung cấp trợ giúp bổ sung trong menu ngữ cảnh của các lỗi trong **Error List**. Trợ giúp bổ sung được truy cập bằng cách nhấn chuột phải vào lỗi. Nhấn chuột phải vào thông báo lỗi **Name 'totalResultsLabel' is not declared** trong **Error List** rồi chọn **Show Error Help** (Hình 5.20) để hiển thị trang tham khảo có thông tin về dạng thường gặp của lỗi biên dịch, giải pháp có thể thực hiện và đường link tới các tài liệu khác (Hình 5.21). Sau khi đã xem thông tin này, đóng trang trợ giúp bằng cách nhấn vào nút x ở trên cùng, bên phải của trang.



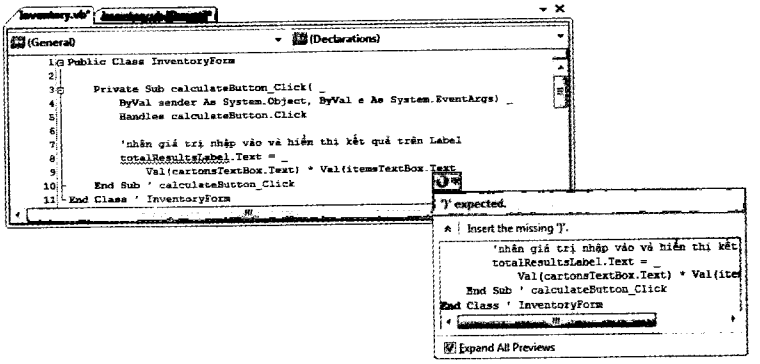
Hình 5.20 Tìm thêm sự trợ giúp.



Hình 5.21 Cửa sổ **Error Help**.

5. ***Sửa lỗi biên dịch.*** Bây giờ, khi đã biết cách tìm và khắc phục lỗi biên dịch, ta hãy trở lại chế độ **Code** và sửa hai lỗi mà bạn đã tạo ra ở *Bước 2*. Xóa bỏ ký tự thừa mà bạn đã thêm vào định danh `totalResultLabel`. Khi bạn sửa lỗi này, hãy chú ý rằng đường răng cưa không biến mất ngay lập tức. Tuy nhiên, khi bạn di chuyển con trỏ tới một dòng khác, IDE kiểm tra lại mã để tìm lỗi, xóa bỏ đường gạch chân răng cưa cho lỗi biên dịch đã được sửa và xóa bỏ lỗi khỏi cửa sổ **Error List**. Để khắc phục lỗi biên dịch thứ hai, di chuột lên hình chữ nhật màu đỏ được đặt ở vị trí mà dấu ngoặc đơn

cần xuất hiện. Biểu tượng **Error Correction Options** ([]) xuất hiện. Nhấn vào biểu tượng này để mở cửa sổ **Error Corection Options** (Hình 5.22). Phần chữ màu xanh ở phía trên cửa sổ cung cấp những gợi ý để sửa lỗi biên dịch. Trong trường hợp này, IDE cho biết rằng thêm một dấu ngoặc đơn sẽ sửa được lỗi biên dịch. Nhấn vào gợi ý - **Insert the missing ')'** - để áp dụng.



Hình 5.22 Sử dụng cửa sổ **Error Correction** để khắc phục lỗi biên dịch.

6. ***Lưu project***. Chọn **File > Save All** để lưu mã đã được sửa đổi. Lúc này ứng dụng đã sẵn sàng để được biên dịch và thực thi.
7. ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

Trong phần này, bạn đã tìm hiểu về các lỗi biên dịch cũng như cách tìm và khắc phục chúng. Trong các chương sau, bạn sẽ tìm hiểu cách tìm và loại bỏ lỗi lôgíc với trình gỡ lỗi thời gian thực của Visual Basic IDE.

TỰ ÔN TẬP

1. Nếu có lỗi biên dịch trong ứng dụng, chúng xuất hiện trong cửa sổ có tên_____.
   a) **Task List** b) **Output**
   c) **Properties** d) **Error List**
2. Lỗi biên dịch xảy ra khi_____.
   a) viết sai từ khóa
   b) thiếu dấu ngoặc đơn
   c) câu lệnh được chia thành nhiều dòng mà không có ký tự nối dòng
   d) Các lựa chọn trên đều đúng

Đáp án: 1) d. 2) d.

5.6 Tổng kết

Trong chương này, bạn được giới thiệu về kỹ thuật lập trình Visual Basic. Bạn đã học cách sử dụng điều khiển `TextBox` để cho phép người dùng nhập dữ liệu và cách sử dụng điều khiển `Button` để phát tín hiệu cho ứng dụng đang chạy thực hiện một hành động cụ thể. Bạn đã biết được rằng một chìa khóa để lập trình tốt là sự kết hợp giữa lập trình trực quan (trong đó IDE viết mã cho bạn) và tự viết mã.

Sau khi tìm hiểu về các toán tử trong Visual Basic, bạn đã viết một vài dòng mã khi thêm xử lý sự kiện vào ứng dụng để thực hiện một phép tính nhân đơn giản. Bạn đã sử dụng hàm `Val` để lấy về giá trị do người dùng nhập vào và sử dụng giá trị đó trong tính toán. Sau đó bạn hiển thị kết quả cho người dùng bằng cách gán giá trị đó vào thuộc tính `Text` của một `Label`. Bạn cũng đã sử dụng chú thích để mã dễ đọc hơn. Bạn đã biết được rằng viết mã trong xử lý sự kiện cho phép một ứng dụng phản hồi lại sự kiện, ví dụ như nhấn `Button`.

Cuối cùng, bạn đã tìm hiểu về các lỗi biên dịch và cách sử dụng IDE Visual Basic để tìm và sửa chữa những lỗi đó. Trong chương tiếp theo, bạn tiếp tục phát triển ứng dụng **Inventory** với việc sử dụng sự kiện `TextChanged`, sự kiện này xảy ra khi người dùng thay đổi giá trị trong `TextBox`. Sau khi áp dụng những hiểu biết về biến, bạn sử dụng trình gỡ lỗi khi ứng dụng chạy để loại bỏ lỗi lôgíc khỏi ứng dụng.

TỔNG KẾT KỸ NĂNG

Truy cập giá trị của thuộc tính bằng mã Visual Basic

- Để truy cập thuộc tính của điều khiển, đặt tên của thuộc tính sau tên điều khiển và toán tử truy cập thành viên (`.`). Ví dụ, để truy cập thuộc tính `Text` của `TextBox` có tên `cartonsTextBox`, sử dụng `CartonsTextbox.Text`.

Thêm chú thích Visual Basic vào mã

- Chú thích bắt đầu bởi ký tự nháy đơn (`'`). Chú thích có thể được đặt trên một dòng (chú thích toàn dòng) hoặc ở cuối một dòng mã (chú thích cuối dòng).

Viết một lệnh trên nhiều dòng

- Thêm ký tự nối dòng (_), đứng sau ít nhất là một ký tự khoảng trắng, để chỉ ra rằng dòng tiếp theo là tiếp tục của dòng trước đó. Chỉ có các ký tự khoảng trắng được đứng sau ký tự nối dòng.

Đặt tên cho xử lý sự kiện

- Sử dụng định dạng *controlName_eventName* để đặt tên cho xử lý sự kiện, trong đó *controlName* là tên của điều khiển và *eventName* là tên của sự kiện.

Thêm xử lý sự kiện cho sự kiện `Click` của `Button`

- Nhấn đúp vào `Button` trong chế độ **Design** để tạo một xử lý sự kiện rỗng, sau đó viết mã thực thi khi sự kiện xảy ra.

Sử dụng lệnh gán

- Sử dụng = (ký hiệu "bằng") để gán giá trị của toán hạng phải cho toán hạng trái.

Sử dụng toán tử nhân

- Sử dụng dấu sao (*) giữa hai biểu thức cần nhân. Toán tử này nhân toán hạng phải với toán hạng trái nếu cả hai toán hạng đều có giá trị số. Lỗi biên dịch xảy ra khi sử dụng toán hạng nhân với các giá trị không phải là kiểu dữ liệu số.

Lấy về giá trị số từ `TextBox`

- Truyền giá trị thuộc tính `Text` của `TextBox` cho hàm `Val`.

Tìm lỗi biên dịch

- Nhấn đúp vào thông báo lỗi trong cửa sổ **Error List**.

Sử dụng trợ giúp ngữ cảnh của lỗi biên dịch

- Nhấn chuột phải vào thông báo lỗi trong cửa sổ **Error List** rồi chọn **Show Error Help** từ menu ngữ cảnh.

THUẬT NGỮ

chú thích (comment) - Văn bản đứng sau dấu nháy đơn (`'`) và được thêm vào để mã dễ đọc hơn.

chức năng (functionality) - Hành động mà ứng dụng có thể thực thi.

cửa sổ Error List - Cửa sổ hiển thị các lỗi biên dịch của mã nguồn.

cửa sổ Output - Cửa sổ hiển thị kết quả của quá trình biên dịch.

định danh (identifier) - Một tập hợp các ký tự bao gồm các chữ cái, chữ số và dấu gạch dưới, dùng để đặt tên các thành phần của chương trình như lớp, điều khiển và biến.

định nghĩa lớp (class definition) - Mã nguồn của lớp, bắt đầu bằng từ khóa `Public Class` và kết thúc với từ khóa `End Class`.

đối số (argument) - Đầu vào của hàm, cung cấp thông tin mà hàm cần để thực hiện tác vụ.

gỡ lỗi (debugging) - Là quá trình khắc phục lỗi của ứng dụng.

hàm `Val` - Lọc một số từ đối số của hàm nếu có thể. Việc này tránh được những lỗi xảy ra khi nhập dữ liệu không phải số, trong khi dữ liệu nhập vào bắt buộc phải là dữ liệu số. Tuy nhiên, kết quả của hàm `Val` không phải khi nào cũng là kết quả mà bạn mong đợi.

IntelliSense - Là một tính năng của IDE Visual Basic trợ giúp bạn trong khi phát triển ứng dụng bằng cách cung cấp các cửa sổ liệt kê những thành phần có trong ngữ cảnh hiện tại.

kiểm tra lỗi thời gian thực (real-time error checking) - Tính năng của IDE Visual Basic đưa ra thông báo ngay lập tức về những lỗi có thể xảy ra trong mã nguồn. Ví dụ, lỗi định danh chưa khai báo được chỉ định bằng đường gạch chân răng cưa màu xanh.

ký tự khoảng trắng (whitespace character) - Dấu cách, tab hoặc ký tự xuống dòng.

ký tự nháy đơn (') - Là ký tự mở đầu cho một chú thích.

ký tự nối dòng (line-continuation character) - Dấu gạch dưới (_) đứng sau ít nhất một ký tự trắng, dùng để tiếp tục một lệnh sang dòng tiếp theo.

lệnh (statement) - Là một đơn vị mã, khi được biên dịch và thực thi, thực hiện một hành động.

lệnh gán (assignment statement) - Lệnh copy giá trị này sang giá trị khác. Lệnh gán có chứa toán tử "bằng" (=) làm cho giá trị của toán hạng phải được copy sang toán hạng trái.

lỗi biên dịch (compilation error) - Lỗi xảy ra khi lệnh của chương trình vi phạm quy tắc ngữ pháp của ngôn ngữ lập trình hoặc đơn giản là khi lệnh không đúng trong ngữ cảnh hiện thời.

lỗi cú pháp (syntax error) - Lỗi xảy ra khi câu lệnh chương trình vi phạm quy tắc ngữ pháp của ngôn ngữ lập trình. Lỗi cú pháp là tập hợp con của lỗi biên dịch.

lỗi lôgíc (logic error) - Là lỗi không ngăn cản ứng dụng được biên dịch thành công, nhưng làm cho ứng dụng đưa ra kết quả sai.

mệnh đề `Handles` - Chỉ ra sự kiện được xử lý bởi một xử lý sự kiện và đối tượng xảy ra sự kiện.

phân biệt viết hoa hay viết thường (case sensitive) - Hai từ được đánh vần giống nhau nhưng khi được viết hoa hay viết thường các chữ cái trong đó thì được xem là khác nhau.

sự kiện (event) - Là hành động của người dùng có thể kích hoạt xử lý sự kiện.

sự kiện `Click` - Sự kiện xảy ra khi người dùng nhấn vào điều khiển.

tên lớp - Định danh để xác định duy nhất tên của lớp trong mã nguồn.

toán hạng (operand) - Biểu thức mà toán tử thực hiện tính toán với nó.

toán hạng phải - Là biểu thức ở vế bên phải của toán tử hai ngôi.

toán hạng trái - Biểu thức ở vế bên trái của một toán tử hai ngôi.

toán tử chấm (dot operator) - Xem toán tử truy cập thành viên.

toán tử gán (assignment operator) - Biểu tượng "=" sử dụng để gán giá trị trong lệnh gán.

toán tử nhân - Dấu sao (*) được dùng để nhân hai toán hạng và đưa ra kết quả.

toán tử hai ngôi (binary operator) - Toán tử có hai toán hạng.

toán tử truy cập thành viên (member-access operator) - Còn được gọi là toán tử chấm (.). Cho phép bạn truy cập tới các thuộc tính của điều khiển bằng mã.

trình gỡ lỗi (debugger) - Công cụ cho phép bạn phân tích hoạt động của ứng

dụng để xác định xem ứng dụng có thực thi đúng hay không.

trình soạn thảo mã (code editor) - Cửa sổ trong đó người dùng có thể tạo, hiển thị hoặc viết mã ứng dụng.

từ dành riêng (reserved word) - Cách gọi khác của từ khóa.

từ khóa (keyword) - Từ trong mã đã được đăng ký với trình biên dịch cho một mục đích cụ thể nào đó. Theo mặc định, trong IDE những từ khóa có màu xanh và không thể sử dụng làm định danh.

từ khóa `Class` - Từ khóa bắt đầu một định nghĩa lớp.

từ khóa `End Class` - Đánh dấu sự kết thúc của một định nghĩa lớp.

xử lý sự kiện (event handler) - Phần mã được thực thi khi một sự kiện xảy ra.

ĐIỀU KHIỂN, SỰ KIỆN, THUỘC TÍNH & PHƯƠNG THỨC

`Button` (ab Button) Điều khiển này cho phép người dùng kích hoạt một hành động hoặc sự kiện.

- ***Trên giao diện khi ứng dụng chạy***

  Calculate Total

- ***Sự kiện***

  `Click` - Được kích hoạt khi người dùng nhấn `Button`.

- ***Thuộc tính***

  `Location` - Chỉ ra vị trí của `Button` trên `Form`, là khoảng cách so với góc trên bên trái của `Form`.

  `Name` - Chỉ ra tên được sử dụng để định danh `Button`. Tên nên thêm hậu tố `Button`.

  `Size` - Chiều rộng và chiều cao (bằng pixel) của `Button`.

  `Text` - Nội dung văn bản hiển thị trên `Button`.

CÂU HỎI TRẮC NGHIỆM

5.1 _____đại diện cho một hành động của người dùng, ví dụ như nhấn `Button`.

a) Lệnh
b) Sự kiện
c) Ứng dụng
d) Hàm

5.2 Để chuyển sang chế độ **Code**, chọn_____.

a) **Code > View**
b) **Design > Code**
c) **View > Code**
d) **View > File Code**

5.3 Mã nguồn thực thi chức năng của ứng dụng_____.

a) thường do lập trình viên viết ra
b) không bao giờ dưới dạng một xử lý sự kiện
c) luôn tạo ra giao diện người dùng đồ họa
d) luôn do IDE sinh ra

5.4 Chú thích_____.

a) giúp cho chương trình dễ đọc hơn
b) đứng sau ký tự nháy đơn
c) được trình biên dịch bỏ qua
d) Các lựa chọn trên đều đúng

5.5 _____cho phép một lệnh được viết trên nhiều hơn một dòng (phải có ít nhất một ký tự trắng đứng trước ký tự này).

a) Ký tự nháy đơn (')
b) Ký tự gạch ngang (-)
c) Ký tự nối dòng (_)
d) Ký tự cộng (+)

5.6 _____làm cho ứng dụng đưa ra kết quả sai.

a) Lỗi lôgíc
b) Sự kiện
c) Lệnh gán
d) Lỗi biên dịch

5.7 _____là một phần của mã, thực thi một nhiệm vụ cụ thể và trả về một giá trị.

a) Biến
b) Hàm
c) Toán hạng
d) Định danh

5.8 Từ khóa Visual Basic là_____.

a) định danh
b) từ được đăng ký
c) phân biệt viết hoa hay viết thường
d) thuộc tính

5.9 IDE Visual Basic không chạy chương trình cho tới khi tất cả lỗi_____được sửa.

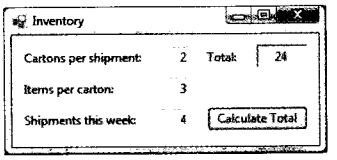
a) lôgíc
b) nghiêm trọng
c) biên dịch
d) thực thi

5.10 Ví dụ về ký tự khoảng trắng là ký tự_____.

a) trắng
b) tab
c) dòng mới
d) Các lựa chọn trên đều đúng

BÀI TẬP

5.11 (***Nâng cấp ứng dụng Inventory***) Mở rộng ứng dụng **Inventory**, thêm `TextBox` để người dùng có thể nhập số lô hàng nhận được trong một tuần. Giả sử rằng mỗi lô hàng có cùng số kiện hàng và mỗi kiện có cùng số đầu sách. Sau đó sửa đổi mã để ứng dụng **Inventory** sử dụng giá trị đó trong khi tính toán.



Chú thích hình

- ***Cartons per shipment:*** *Số kiện hàng trong mỗi lô hàng*
- ***Items per carton:*** *Số sách trong mỗi kiện hàng*
- ***Shipments this week:*** *Số lô hàng trong tuần*

Hình 5.23 Giao diện ứng dụng **Inventory** sau khi được nâng cấp.

a) ***Copy template của ứng dụng vào thư mục làm việc.*** Copy thư mục `C:\Examples\Tutorial05\Exercises\InventoryEnhancement` vào thư mục `C:\SimplyVB2008`.

b) ***Mở file template của ứng dụng.*** Nhấp đúp chuột vào file `InventoryEnhancement.sln` trong thư mục `InventoryEnhancement` để mở ứng dụng.

c) ***Thay đổi kích thước Form.*** Sửa lại kích thước `Form` trong chương này bằng cách thiết lập giá trị thuộc tính `Size` là `320,147`. Di chuyển `Button` xuống phía dưới của `Form` như trên Hình 5.23. Vị trí mới của `Button` là `193,72`.

d) ***Thêm Label.*** Thêm `Label` vào `Form` và thay đổi thuộc tính `Text` thành `Shipments this week:`. Thiết lập giá trị thuộc tính `Location` là `9,77`. Đặt giá trị thuộc tính `Name` là `shipmentsLabel`.

e) ***Thêm TextBox.*** Thêm `TextBox` vào bên phải `Label`. Đặt giá trị thuộc tính `Text` của `TextBox` là `0` và giá trị thuộc tính `Location` là `136,74`. Thiết lập giá trị thuộc tính `TextAlign` và `Size` có cùng giá trị với các `TextBox` khác trong ví dụ của chương này. Thiết lập giá trị thuộc tính `Name` của `TextBox` là `shipmentsTextBox`.

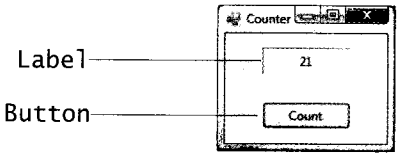
f) ***Thay đổi mã.*** Thay đổi xử lý sự kiện `Click` của `Button` **Calculate Total** để hàm này nhân số lô hàng mỗi tuần với tích của số kiện sách trong mỗi chuyến và số đầu sách trong mỗi kiện.

g) ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng. Nhập các giá trị cho số kiện hàng trong mỗi chuyến, số đầu sách trong mỗi kiện và số lô hàng trong tuần. Nhấn `Button` **Calculate Total** và kiểm tra kết quả được hiển thị có đúng với kết quả nhân ba số hay không. Nhập dữ liệu một vài lần và kiểm tra kết quả mỗi lần.

h) ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

i) ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

5.12 (***Ứng dụng Counter***) Tạo ứng dụng counter gồm một `Label` và `Button` trên `Form`. `Label` ban đầu hiển thị `0`, nhưng mỗi lần người dùng nhấn `Button`, giá trị trong `Label` được tăng thêm `1`. Khi tăng giá trị trên `Label`, bạn cần viết một lệnh ví dụ như `totalLabel.Text = Val(totalLabel.Text)+1`.



Hình 5.24 Giao diện **Counter**.

a) ***Tạo ứng dụng***. Tạo project mới có tên là `Counter`.

b) ***Thay đổi tên của file Form***. Đổi tên của `Form1.vb` thành `Counter.vb`.

c) ***Thay đổi Form***. Thay đổi thuộc tính `Font` của `Form` thành `Segoe UI 9pt` và thuộc tính `Size` thành `176,144`. Thay đổi để `Form` có tiêu đề **Counter**. Đổi tên `Form` thành `CounterForm`.

d) ***Thêm Label***. Thêm `Label` vào `Form` và đặt ở vị trí như trên Hình 5.24. Đảm bảo rằng thuộc tính `Text` của `Label` là `0` và thuộc tính `TextAlign` được thiết lập để văn bản sẽ xuất hiện ở giữa (theo cả chiều ngang và chiều dọc) của `Label`. Việc này có thể được thực hiện bằng cách sử dụng giá trị `MiddleCenter` của thuộc tính `TextAlign`. Thiết lập giá trị `Fixed3D` cho thuộc tính `BorderStyle`. Đặt giá trị thuộc tính `Name` của `Label` là `countTotalLabel`.

e) ***Thêm Button***. Thêm một `Button` vào `Form` giống như trên Hình 5.24. Thiết lập thuộc tính `Text` của `Button` là **Count**. Đặt giá trị thuộc tính `Name` của `Button` là `countButton`.

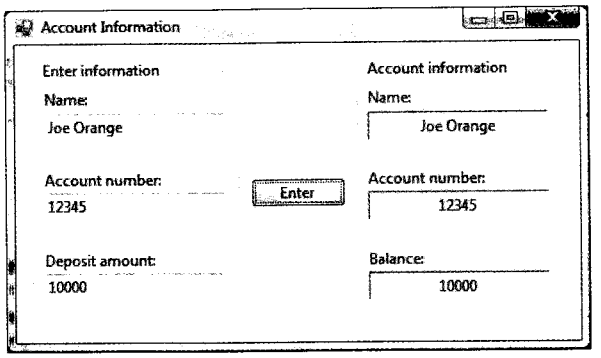
f) ***Tạo xử lý sự kiện***. Thêm một xử lý sự kiện cho `Button` **Count** để giá trị `Label` tăng thêm `1` sau mỗi lần người dùng nhấn `Button` **Count**.

g) ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng. Nhấn `Button` **Count** vài lần và kiểm tra xem kết quả đầu ra có được tăng lên sau mỗi lần hay không.

h) ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

i) ***Đóng IDE***. Đóng cửa sổ IDE Visual Basic bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

5.13 (***Ứng dụng Account Information***) Tạo ứng dụng cho phép một người dùng nhập tên, số tài khoản và số tiền gửi. Sau đó người dùng nhấn `Button` **Enter**, tên và số tài khoản sẽ được copy và hiển thị trên hai `Label` đầu ra. Số tiền gửi nhập vào được thêm vào tổng số tiền trong tài khoản hiển thị trên một `Label` đầu ra khác. Khi cập nhật thông tin hiển thị trên `Label`, bạn cần viết một lệnh ví dụ như `depositsLabel.Text = Val(depositsLabel.Text) + Val(depositAmountTextBox.Text)`.



Chú thích hình

- ***Account number:*** *Số tài khoản*
- ***Deposit amount:*** *Số tiền gửi*
- ***Balance:*** *Tổng số tiền có trong tài khoản*

Hình 5.25 Giao diện **Account Information**.

a) ***Copy file template của ứng dụng vào thư mục làm việc***. Copy thư mục `C:\Examples\Tutorial05\Exercises\AccountInformation` vào thư mục `C:\SimplyVB2008`.

b) ***Mở file template của ứng dụng***. Nhấn đúp chuột vào file `AccountInformation.sln` trong thư mục `AccountInformation` để mở ứng dụng.

c) ***Tạo xử lý sự kiện***. Thêm xử lý sự kiện cho sự kiện `Click` của `Button` **Enter**.

d) ***Lập trình xử lý sự kiện***. Lập trình xử lý sự kiện để copy thông tin từ `TextBox` **Name:** và **Account number:** vào `Label` đầu ra tương ứng của chúng. Sau đó cộng giá trị trong `TextBox` **Deposit amount:** vào giá trị trong `Label` đầu ra **Balance:** rồi hiển thị kết quả lên `Label` đầu ra **Balance:**.

e) ***Chạy ứng dụng.*** Chọn **Debug > Start Debugging** để chạy ứng dụng. Nhập các giá trị như trên Hình 5.25 và nhấn `Button` **Enter**. Kiểm tra xem thông tin tài khoản hiển thị trong các `Label` ở bên phải đã đúng chưa. Nhập các khoản tiền gửi khác nhau và nhấn `Button` **Enter**. Kiểm tra xem tổng số tiền trong tài khoản ở bên phải được cộng giá trị mới chưa.

f) ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

g) ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

Đoạn mã này làm gì? ▶ **5.14** Sau khi nhập `10` vào `priceTextBox` và `1.05` vào `taxTextBox`, người dùng nhấn `Button` có tên `enterButton`. Cho đoạn mã xử lý sự kiện như sau, hãy cho biết kết quả sau khi nhấn chuột là gì?

```
1  Private Sub enterButton_Click(ByVal sender As  _
2     System.Object, ByVal e As System.EventArgs) _
3     Handles enterButton.Click
4
5     outputLabel.Text = Val(priceTextBox.Text) * Val(taxTextBox.Text)
6  End Sub 'enterButton_Click
```

Đoạn mã này có gì sai? ▶ **5.15** Xử lý sự kiện sau sẽ thực thi khi người dùng nhấn `Button` **Calculate**. Xác định lỗi trong đoạn mã.

```
1  Private Sub calculateButton_Click(ByVal sender As _
2     System.Object, ByVal e As System.EventArgs 'dòng thứ hai
3     Handles calculateButton.Click
4
5     resultLabel.Text = priceTextBox.Text * taxTextBox.Text
6  End Sub 'calculateButton_Click
```

Sử dụng trình gỡ lỗi ▶ **5.16** (***Bài tập gỡ lỗi Account Information***) Copy thư mục `C:\Examples\Tutorial05\Exercises\DebuggingExercise` vào thư mục `C:\SimplyVB2008` sau đó chạy ứng dụng **Account Information**. Loại bỏ tất cả các lỗi biên dịch để ứng dụng chạy đúng.

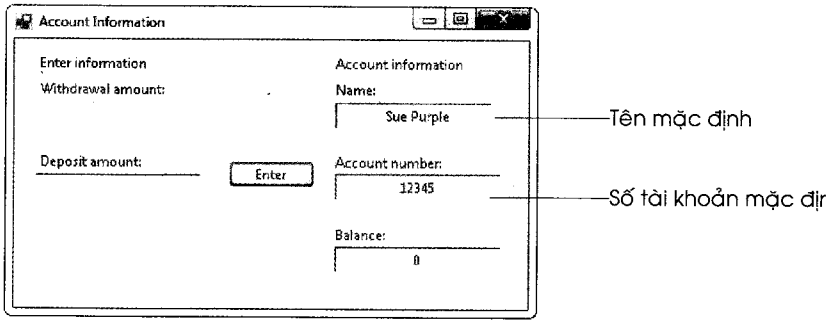
Bài tập nâng cao ▶ **5.17** (***Nâng cấp ứng dụng Account Information***) Thay đổi Bài tập 5.13 để ứng dụng không còn yêu cầu tên và số tài khoản của người dùng mà yêu cầu số tiền rút hoặc gửi. Người dùng có thể nhập cả số tiền rút và tiền gửi cùng một lúc. Khi `Button` **Enter** được nhấn, số tiền trong tài khoản được cập nhật tương ứng.

a) ***Copy file template của ứng dụng vào thư mục làm việc.*** Copy thư mục `C:\Examples\Tutorial05\Exercises\AccountInformation` vào thư mục `C:\SimplyVB2008`.

b) ***Mở file template của ứng dụng.*** Nhấn đúp chuột vào file `AccountInformation.sln` trong thư mục `AccountInformation` để mở ứng dụng.

c) ***Thay đổi giao diện.*** Thay đổi giao diện giống như trên Hình 5.26.

d) ***Thiết lập các giá trị mặc định.*** Thiết lập tên và số tài khoản mặc định với các giá trị như trên Hình 5.26 sử dụng cửa sổ **Properties**.



Chú thích hình

- ***Withdrawal:*** *Số tiền rút*
- ***Deposit amount:*** *Số tiền gửi*
- ***Account information:*** *Thông tin tài khoản*
- ***Account number:*** *Số tài khoản*
- ***Balance:*** *Tổng số tiền có trong tài khoản*

Hình 5.26 Giao diện **Account Information** sau khi đã được nâng cấp.

e) ***Viết mã để thêm chức năng.*** Cập nhật số tiền có trong tài khoản sau mỗi lần rút tiền (giảm tổng số tiền có trong tài khoản) và gửi tiền (tăng tổng số tiền có trong tài khoản). Khi khoản tiền được cập nhật, thiết lập lại giá trị hiển thị trên `TextBox` là 0.

f) ***Chạy ứng dụng.*** Chọn **Debug > Start Debugging** để chạy ứng dụng. Nhập các giá trị rút và gửi tiền khác nhau, nhấn `Button` **Enter** sau mỗi lần nhập. Xác nhận khoản tiền ở bên phải ứng dụng được cập nhật tương ứng với mỗi lần nhấn `Button` **Enter**.

g) ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

h) ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

CHƯƠNG 6

Nâng cấp ứng dụng Inventory

Giới thiệu về biến, khái niệm về bộ nhớ và phép toán số học

Mục tiêu

Trong chương này, bạn sẽ tìm hiểu để:

- Tạo biến.
- Xử lý sự kiện **TextChanged**.
- Áp dụng khái niệm cơ bản về bộ nhớ để sử dụng biến.
- Tìm hiểu về thứ tự ưu tiên của các toán tử số học.
- Thiết lập các breakpoint để gỡ lỗi cho ứng dụng.

Nội dung chính

Trong chương trước, bạn đã phát triển ứng dụng **Inventory** sử dụng phép nhân để tính toán số lượng đầu sách được nhập vào kho. Bạn đã tìm hiểu cách tạo `TextBox` để lấy dữ liệu do người dùng nhập vào từ bàn phím. Bạn cũng đã thêm `Button` vào `Form` và lập trình để `Button` đó phản hồi lại khi người dùng nhấn vào. Trong chương này, bạn sẽ nâng cấp ứng dụng **Inventory** bằng cách sử dụng các khái niệm lập trình mới, bao gồm biến, sự kiện và toán tử.

6.1 Chạy thử ứng dụng Inventory đã được nâng cấp

Trong chương này, bạn sẽ nâng cấp ứng dụng **Inventory** của chương trước bằng cách viết thêm mã. Bạn sẽ sử dụng biến để thực hiện các phép toán trong Visual Basic, và sẽ tìm hiểu khái niệm về bộ nhớ để có thể hiểu cách mà ứng dụng chạy trên máy tính. Hãy nhớ lại rằng, ứng dụng **Inventory** trong Chương 5 của bạn tính số lượng đầu sách từ thông tin do người dùng nhập vào - số kiện hàng và số đầu sách trong mỗi kiện. Ứng dụng được nâng cấp phải đáp ứng những yêu cầu sau:

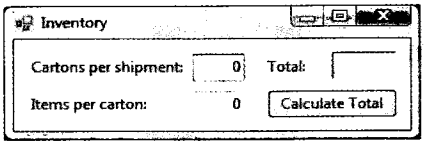
Yêu cầu đối với ứng dụng

*Người quản lý kho thông báo một thiếu sót trong ứng dụng **Inventory** của bạn. Mặc dù ứng dụng tính toán ra kết quả đúng, nhưng kết quả đó vẫn tiếp tục được hiển thị ngay cả sau khi dữ liệu mới đã được nhập vào. Kết quả hiển thị chỉ thay đổi khi người quản lý kho nhấn tiếp vào `Button` **Calculate Total**. Bạn cần thay đổi ứng dụng **Inventory** để xóa kết quả ngay khi người dùng nhập thông tin mới vào bất kỳ `TextBox` nào để tránh sự nhầm lẫn về tính chính xác của kết quả được tính toán.*

Bạn bắt đầu bằng việc chạy thử ứng dụng đã hoàn thiện. Sau đó, bạn tìm hiểu những tính năng bổ sung của Visual Basic cần thiết để xây dựng cho mình một phiên bản của ứng dụng này. Nhìn lướt qua, có vẻ như ứng dụng hoạt động không có gì khác so với ứng dụng trong chương trước. Tuy nhiên, bạn nên chú ý rằng `Label` **Total:** được xóa khi bạn nhập dữ liệu mới vào bất kỳ `TextBox` nào.

Chạy thử ứng dụng Inventory đã được nâng cấp

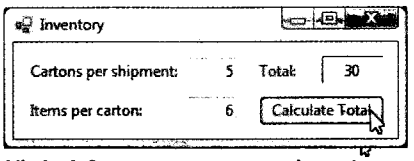
1. ***Mở ứng dụng đã hoàn thiện.*** Mở thư mục `C:\Examples\Tutorial06\CompletedApplication\Inventory3` để tìm ứng dụng đã được nâng cấp **Inventory**. Nhấn đúp vào `Inventory3.sln` để mở ứng dụng trong Visual Basic IDE.
2. ***Chạy ứng dụng Inventory.*** Chọn **Debug > Start** để chạy ứng dụng



Chú thích hình
- ***Cartons per shipment:*** *Số kiện hàng trong mỗi lô hàng*
- ***Items per carton:*** *Số sách trong mỗi kiện hàng*

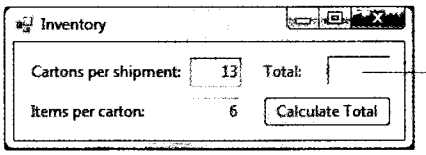
Hình 6.1 Giao diện ứng dụng **Inventory** được hiển thị khi ứng dụng chạy.

3. ***Tính toán số lượng sách trong lô hàng.*** Nhập 5 vào `TextBox` **Cartons per shipment:** và 6 vào `TextBox` **Items per carton:**. Nhấn vào `Button` **Calculate Total**. Kết quả (30) được hiển thị trên `Label` đầu ra **Total:** (Hình 6.2).



Hình 6.2 Chạy ứng dụng **Inventory**.

4. ***Nhập thông số mới.*** Sau khi bạn chỉnh sửa ứng dụng, kết quả được hiển thị trên `Label` **Total:** sẽ được xóa khi người dùng nhập một số lượng mới vào bất kỳ `TextBox` nào. Nhập 13 làm số lượng kiện hàng mới - kết quả tính toán cuối cùng sẽ bị xóa (Hình 6.3). Điều này sẽ được giải thích trong phần sau của chương.



Hình 6.3 Ứng dụng **Inventory** đã được nâng cấp xóa nội dung trên `Label` hiển thị kết quả sau mỗi lần nhập mới.

5. ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.
6. ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

6.2 Biến

Biến (variable) lưu giữ dữ liệu trong ứng dụng của bạn, tương tự như thuộc tính `Text` của `Label` chứa văn bản được hiển thị cho người dùng. Tuy nhiên, không giống như thuộc tính `Text` của `Label`, theo mặc định các **giá trị của biến (value of a variable)** không được hiển thị cho người dùng. Sử dụng biến trong ứng dụng cho phép bạn lưu và xử lý dữ liệu mà không cần phải hiển thị dữ liệu ra cho người dùng, đồng thời lưu dữ liệu mà không cần phải thêm hoặc sử dụng điều khiển. Các biến lưu dữ liệu như các con số, ngày, giờ... Tuy nhiên, mỗi biến được sử dụng trong Visual Basic tương ứng với duy nhất một kiểu dữ liệu. Ví dụ, một biến kiểu số không thể dùng để lưu văn bản.

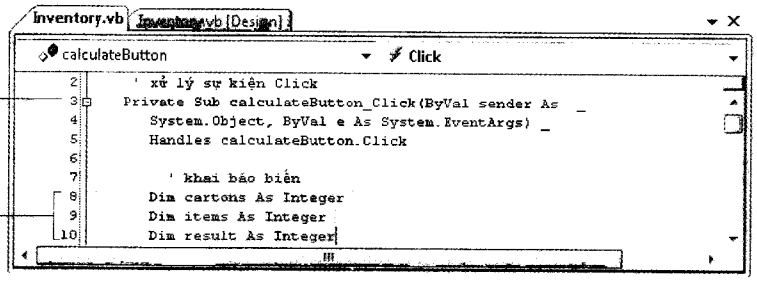
Trong Visual Basic, tất cả các biến phải **được khai báo** hoặc được thông báo, với trình biên dịch bằng cách sử dụng mã chương trình. Tất cả những **khai báo**

(**declaration**) mà bạn sẽ thực hiện trong xử lý sự kiện đều bắt đầu bởi từ khóa **Dim**. Hãy nhớ lại rằng các từ khóa được đăng ký sử dụng bởi Visual Basic (Danh sách đầy đủ các từ khóa của Visual Basic có trong Phụ lục B.)

Phần dưới đây giới thiệu về lập trình với biến. **Tên biến** có thể là một định danh hợp lệ bất kỳ, với điều kiện là, như bạn đã tìm hiểu trong Chương 5, trình biên dịch có thể nhận biết được (và không phải là một từ khóa). Bạn cũng đã tìm hiểu trong chương trước rằng có nhiều ký tự hợp lệ dùng cho việc định danh.

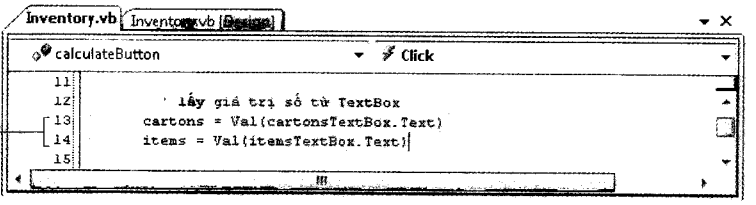
Sử dụng biến trong ứng dụng Inventory

1. ***Copy template của ứng dụng vào thư mục làm việc của bạn.*** Copy thư mục `C:\Examples\Tutorial06\TemplateApplication\Inventory3` vào thư mục `C:\SimplyVB2008` của bạn.
2. ***Mở file template của ứng dụng Inventory.*** Nhấn đúp vào `Inventory3.sln` trong thư mục `Inventory3` để mở ứng dụng trong IDE Visual Basic.
3. ***Thêm khai báo biến (variable declaration) vào xử lý sự kiện `calculateButton_Click`.*** Nếu bạn đang ở chế độ **Design**, hãy chuyển sang chế độ **Code** bằng cách chọn **View > Code**. Thêm các dòng 7-10 trên Hình 6.4 vào phần xử lý sự kiện `calculateButton_Click`. Các dòng 8-10 là các khai báo, bắt đầu bởi từ khóa `Dim`. Chú ý rằng, khi bạn gõ từ khóa `Dim`, cũng như tất cả các từ khóa khác, theo mặc định IDE sẽ đổi nó thành màu xanh. Các từ `cartons`, `items` và `result` là tên biến. Các dòng 8-10 khai báo các biến `cartons`, `items` và `result` lưu dữ liệu kiểu **`Integer`**, sử dụng từ khóa **`As`**. Từ khóa `As` chỉ ra rằng từ theo sau nó (trong trường hợp này là `Integer`) là kiểu biến. Biến kiểu `Integer` lưu giá trị **số nguyên (integer)** - bao gồm toàn bộ các số, ví dụ như 919, 0 và -11. Chú ý rằng ban đầu IDE gạch chân các biến để chỉ ra rằng chúng chưa được sử dụng trong ứng dụng. Điều này giúp lập trình viên tránh đưa vào ứng dụng những biến không cần thiết.



Hình 6.4 Khai báo biến trong phần xử lý sự kiện `calculateButton_Click`.

4. ***Lấy dữ liệu nhập vào từ `TextBox`.*** Cách một dòng trắng sau khai báo biến và thêm các dòng 12-14 trên Hình 6.5 vào phần xử lý sự kiện `calculateButton_Click`. Khi người dùng nhập vào các số và nhấn vào **Calculate Total**, các giá trị của thuộc tính `Text` trong mỗi điều khiển `TextBox` được chuyển đổi thành các giá trị số bởi hàm `Val`. Sau đó các giá trị số này được gán cho các biến `cartons` (dòng 13) và `items` (dòng 14) bởi toán tử gán =. Dòng 13 được hiểu là "`cartons` *nhận* giá trị trả về từ kết quả hàm `Val` với đối số truyền vào là `cartonsTextBox.Text`".
5. ***Lưu project.*** Chọn **File > Save All** để lưu mã đã được sửa đổi.



Hình 6.5 Lấy giá trị số nhập vào từ các `TextBox`.

Thói quen lập trình tốt

dụng các chữ cái và các chữ số trong tên biến.

Thói quen lập trình tốt

nh danh tên biến thường bắt đầu bởi một chữ cái viết thường. Tất cả các từ trong tên biến sau từ đầu tiên nên bắt đầu bởi một chữ cái viết hoa - ví dụ, `firstNumber`. Cách đặt tên này thường được gọi là cách đặt tên theo kiểu lạc đà (được dùng từ hình tượng các bướu trên lưng lạc đà)..

Hàm `Val` chuyển đổi giá trị lấy từ thuộc tính `Text` của `TextBox` và trả về giá trị số có kiểu dữ liệu **`Double`**. Kiểu dữ liệu `Double` được dùng để lưu cả số nguyên và các số thập phân. Thông thường, `Double` lưu các số dấu chấm động, là các số có dấu chấm thập phân, ví dụ như 2.3456 và -845.4680. Biến kiểu `Double` có thể chứa giá trị lớn hơn (hoặc nhỏ hơn) nhiều so với biến kiểu `Integer`.

Sau khi `Val` chuyển hai giá trị do người dùng nhập vào thành giá trị `Double`, các dòng 13-14 ngầm định chuyển đổi các giá trị `Double` đó thành các giá trị `Integer`. Giá trị số nguyên được lấy bằng cách chuyển kiểu giá trị `Double` ở dòng 13 được gán cho biến `cartons` kiểu dữ liệu `Integer`. Tương tự, giá trị số nguyên được lấy bằng cách chuyển kiểu giá trị `Double` ở dòng 14 được gán cho biến `items`. Do `Double` và `Integer` là các kiểu khác nhau, Visual Basic phải thực hiện việc chuyển đổi giữa hai kiểu. Quá trình này được gọi là **ép kiểu ngầm định (implicit conversion)**, vì việc chuyển đổi được thực hiện bởi Visual Basic mà không cần viết thêm mã. Ép kiểu ngầm định từ `Double` thành `Integer` thường không được khuyến khích sử dụng vì có nguy cơ làm mất mát thông tin. Trong Chương 15, bạn sẽ tìm hiểu cách thực hiện ép kiểu tường minh. Bây giờ khi bạn đã gán giá trị cho các biến mới, hãy sử dụng các biến này để tính toán số lượng sách giáo khoa nhận được.

Visual Basic định nghĩa 15 **kiểu dữ liệu cơ sở (primitive data type)**, ví dụ như `Integer` và được liệt kê trên Hình 6.6. Tên của các kiểu dữ liệu cơ sở cũng là các từ khóa. Bên cạnh các kiểu dữ liệu này, Visual Basic cũng định nghĩa kiểu `Object`. Các kiểu dữ liệu cơ sở cùng với kiểu Object được gọi là các **kiểu dữ liệu có sẵn (built-in data type)**. Trong cuốn sách này, bạn sẽ sử dụng một số kiểu dữ liệu có sẵn. Phụ lục C liệt kê các kiểu dữ liệu cơ bản và khoảng giá trị của chúng.

Các kiểu dữ liệu có sẵn (kiểu dữ liệu cơ sở)

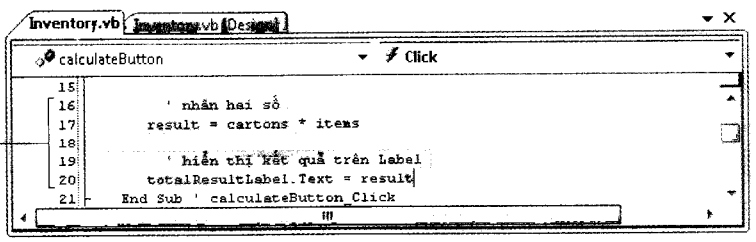
`Boolean`	`Date`	`Integer`	`Long`	`Short`
`Byte`	`Decimal`	`Single`	`Char`	`Double`
`SByte`	`String`	`UInteger`	`Ulong`	`Ushort`

Hình 6.6 Các kiểu dữ liệu có sẵn trong Visual Basic.

Sử dụng biến trong tính toán

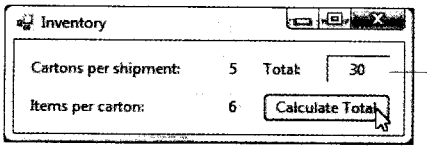
1. ***Thực hiện phép toán nhân***. Bỏ cách một dòng trắng tính từ lệnh cuối cùng mà bạn đã chèn vào và thêm các dòng 16-17 vào phần xử lý sự kiện `calculateButton_Click` (Hình 6.7). Lệnh ở dòng 17 nhân biến `cartons` kiểu `Integer` với biến `items` và gán kết quả cho biến `result`, sử dụng toán tử gán `=`. Câu lệnh được hiểu là "`result` *nhận* giá trị của phép nhân `cartons` với `items`." (Hầu hết các tính toán được thực hiện trong các câu lệnh gán).

Tính toán và hiển thị kết quả



Hình 6.7 Phép toán nhân sử dụng các biến trong `calculateButton_Click`.

2. ***Hiển thị kết quả***. Thêm các dòng 19-20 trên Hình 6.7 vào phần xử lý sự kiện `calculateButton_Click`. Khi việc tính toán đã hoàn tất, dòng 20 hiển thị kết quả của phép toán nhân. Kết quả được gán cho thuộc tính `Text` của `totalResultLabel`. Khi thuộc tính này được cập nhật, `Label` sẽ hiển thị kết quả của phép toán nhân (Hình 6.8).



Hình 6.8 Hiển thị kết quả của phép toán nhân sử dụng biến.

3. ***Chạy ứng dụng.*** Chọn **Debug > Start Debugging** để chạy ứng dụng. Nhập 5 vào `TextBox` **Cartons per shipment:** và 6 vào `TextBox` **Items per carton:**. Sau đó, nhấn vào `Button` **Calculate Total** để kiểm tra kết quả của ứng dụng.
4. ***Đóng ứng dụng.*** Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

TỰ ÔN TẬP

1. Khi Visual Basic chuyển đổi giá trị `Double` thành giá trị `Integer` mà không cần viết mã được gọi là_____.
   a) ép kiểu tường minh
   b) ép kiểu ngầm định
   c) thay đổi kiểu dữ liệu
   d) phép biến đổi
2. Các kiểu dữ liệu đã được định nghĩa trong Visual Basic, ví dụ như `Integer`, được gọi là các kiểu dữ liệu _____.
   a) được cung cấp
   b) đang tồn tại
   c) đã được định nghĩa
   d) có sẵn

Đáp án: **1)** b. **2)** d.

6.3 Xử lý sự kiện TextChanged

Có thể bạn đã để ý thấy rằng thiếu sót, hay lỗi (**bug**), được đề cập trong phần các yêu cầu đối với ứng dụng ở đầu chương vẫn còn trong ứng dụng của bạn. Mặc dù `totalResultLabel` hiển thị kết quả đúng, nhưng khi bạn nhập một số mới vào một `TextBox`, kết quả đó không còn hợp lệ nữa - kết quả được hiển thị không thay đổi cho đến khi bạn nhấn vào `Button` **Calculate Total**, dẫn đến việc người dùng ứng dụng có thể hiểu nhầm. Trong phần tiếp theo, bạn sẽ thêm xử lý sự kiện để xóa kết quả được hiển thị mỗi khi dữ liệu mới được nhập vào.

Xử lý sự kiện TextChanged

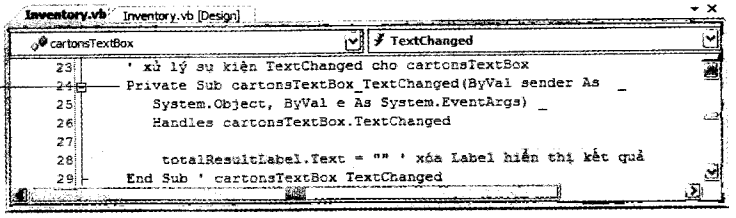
Thói quen lập trình tốt

Nếu một lệnh dài quá độ dài của cửa sổ soạn thảo mã, sử dụng ký tự nối dòng để viết tiếp lệnh ở dòng tiếp theo.

1. ***Thêm xử lý sự kiện cho sự kiện `TextChanged` cho `cartonsTextBox`.*** Chuyển sang chế độ **Design** của IDE và nhấn đúp vào `TextBox` **Cartons per shipment:** để tạo xử lý sự kiện cho sự kiện **`TextChanged`**, xử lý sự kiện sẽ được thực thi khi đoạn văn bản trong `TextBox` thay đổi. Đây là sự kiện mặc định của các `TextBox`. IDE tạo một xử lý sự kiện với phần thân rỗng (không có mã) và đặt con trỏ vào phần thân đó. Chèn dòng 28 trên Hình 6.9 vào mã của bạn. Chú ý rằng, trong đoạn mã này các ký tự nối dòng (_) đã được thêm vào cuối các dòng 24-25 và ghi chú cũng đã được thêm vào ở dòng 23 - trước phần xử lý sự kiện. Hãy nhớ lại rằng, Chương 5 đã cho thấy sử dụng ký tự nối dòng sẽ làm cho mã dễ đọc hơn vì nó tránh được việc viết dòng mã dài quá màn hình.

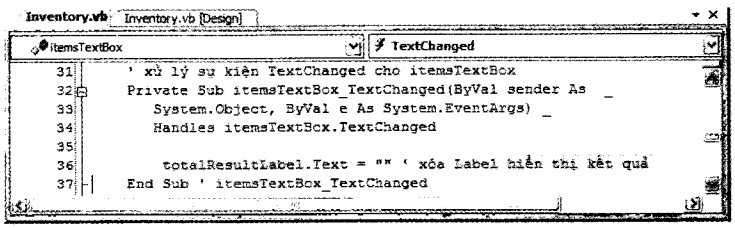
   Theo các yêu cầu đối với ứng dụng trong chương này, ứng dụng cần xóa giá trị trong `totalResultLabel` mỗi khi người dùng thay đổi đoạn văn bản trong bất kỳ `TextBox` nào. Dòng 28 xóa giá trị trong `totalResultLabel`. Ký hiệu `""` (hai dấu nháy kép cạnh nhau) ở dòng 28 được gọi là **chuỗi rỗng (empty string)**, là một giá trị không chứa bất kỳ ký tự nào. Chuỗi rỗng này sẽ thay thế bất kỳ giá trị nào đang được lưu trong `totalResultLabel.Text`. Chú ý rằng bạn cũng có thể biểu diễn một chuỗi rỗng dưới dạng `String.Empty`.

Xử lý sự kiện TextChanged



Hình 6.9 Xử lý sự kiện TextChanged của TextBox **Cartons per shipment:**.

2. ***Thêm xử lý sự kiện cho sự kiện TextChanged cho itemsTextBox.*** Mục tiêu của bạn là kết quả sẽ được xóa mà không cần biết TextBox nào thay đổi giá trị trước. Chuyển sang chế độ **Design** bằng cách nhấn vào tab **Inventory.vb [Design]**. Sau đó, nhấn đúp vào TextBox **Items per carton:** và thêm dòng 36 trên Hình 6.10 vào phần xử lý sự kiện mới. Chú ý rằng dòng này thực hiện công việc giống với dòng 28 - vì bạn muốn hành động tương tự, cụ thể là xóa nội dung hiển thị trên Label.



Hình 6.10 Xử lý sự kiện TextChanged của TextBox **Items per carton:**.

3. ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng. Để kiểm tra ứng dụng, nhập 8 vào TextBox **Cartons per shipment:** và 7 vào TextBox **Items per carton:**. Khi bạn nhấn vào Button **Calculate Total**, số 56 sẽ xuất hiện trên Label hiển thị kết quả. Sau đó, nhập 9 vào TextBox **Items per carton:** để chắc chắn rằng xử lý sự kiện TextChanged xóa nội dung trên Label hiển thị kết quả.
4. ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

Hình 6.11 hiển thị mã nguồn của ứng dụng **Inventory** đã được nâng cấp. Những dòng mã chứa khái niệm lập trình mới mà bạn đã tìm hiểu trong chương này được đánh dấu.

Sử dụng từ khóa Dim để khai báo các biến trong phần xử lý sự kiện

Gán giá trị của thuộc tính cho biến

```
1  Public Class InventoryForm
2      ' xử lý sự kiện Click
3      Private Sub calculateButton_Click(ByVal sender As  _
4          System.Object, ByVal e As System.EventArgs) _
5          Handles calculateButton.Click
6
7          ' khai báo biến
8          Dim cartons As Integer
9          Dim items As Integer
10         Dim result As Integer
11
12         ' lấy giá trị số từ TextBox
13         cartons = Val(cartonsTextBox.Text)
14         items = Val(itemsTextBox.Text)
15
16         ' nhân hai số
17         result = cartons * items
18
19         ' hiển thị kết quả trên Label
```

Gán giá trị của biến cho thuộc tính

Thiết lập thuộc tính `Text` của `Label` thành chuỗi rỗng

```
20          totalResultLabel.Text = result
21      End Sub ' calculateButton_Click
22
23      ' xử lý sự kiện TextChanged cho cartonsTextBox
24      Private Sub cartonsTextBox_TextChanged(ByVal sender As _
25          System.Object, ByVal e As System.EventArgs) _
26          Handles cartonsTextBox.TextChanged
27
28          totalResultLabel.Text = "" ' xóa Label hiển thị kết quả
29      End Sub ' cartonsTextBox_TextChanged
30
31      ' xử lý sự kiện TextChanged cho itemsTextBox
32      Private Sub itemsTextBox_TextChanged(ByVal sender As _
33          System.Object, ByVal e As System.EventArgs) _
34          Handles itemsTextBox.TextChanged
35
36          totalResultLabel.Text = "" ' xóa Label hiển thị kết quả
37      End Sub ' itemsTextBox_TextChanged
38  End Class ' InventoryForm
```

Hình 6.11 Mã nguồn của ứng dụng **Inventory**.

TỰ ÔN TẬP

1. _____được thể hiện bằng "" trong Visual Basic.
   a) Ký tự rỗng
   b) Chuỗi rỗng
   c) Giá trị rỗng
   d) Các lựa chọn trên đều sai
2. Sử dụng thuộc tính _____ để xóa đoạn văn bản hiển thị trên một `Label`.
   a) `ClearText`
   b) `Remove`
   c) `Display`
   d) `Text`

Đáp án: **1)** b. **2)** d.

6.4 Khái niệm về bộ nhớ

Tên biến - ví dụ như `cartons`, `items` và `result` - tương ứng với các địa chỉ thực tế trong bộ nhớ máy tính. Tất cả các biến đều có **tên**, **kiểu**, **kích thước** và **giá trị**. Trong mã của ứng dụng **Inventory** trên Hình 6.11, khi lệnh (dòng 13)

```
cartons = Val(cartonsTextBox.Text)
```

được thực thi, dữ liệu do người dùng nhập vào được chuyển đổi ngầm định thành `Integer`. Giả thiết rằng người dùng nhập số 12 vào `TextBox` **Cartons per shipment:**. Giá trị đầu vào này được lưu trong `CartonsTextBox.Text`. Khi người dùng nhấn vào **Calculate Total**, trên dòng 13 hàm `Val` chuyển giá trị này thành `Double`, sau đó giá trị `Double` này được chuyển đổi ngầm định thành `Integer`. Tiếp theo, phép gán sẽ đặt giá trị 12 kiểu `Integer` vào địa chỉ của biến `cartons`, như trên Hình 6.12.

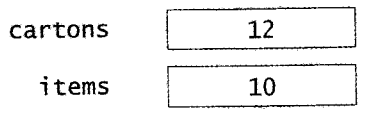
Hình 6.12 Địa chỉ trong bộ nhớ biểu diễn tên và giá trị của biến `cartons`.

Mỗi khi giá trị được đặt vào một địa chỉ trong bộ nhớ, nó sẽ thay thế giá trị đã được lưu từ trước tại địa chỉ đó. Giá trị cũ sẽ bị ghi đè lên (sẽ bị mất).

Giả thiết rằng, sau đó người dùng nhập số 10 vào `TextBox` **Items per carton:** và nhấn vào **Calculate Total**. Trên dòng 14 của Hình 6.11

```
items = Val(itemsTextBox.Text)
```

hàm `Val` chuyển kiểu `itemsTextBox.Text` thành `Double`, sau đó giá trị `Double` này được chuyển đổi ngầm định thành `Integer`. Phép gán sẽ đặt giá trị 10 kiểu `Integer` vào địa chỉ của biến `items` và bộ nhớ sẽ biểu diễn như trên Hình 6.13.

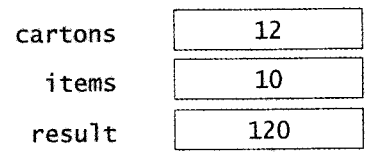


Hình 6.13 Các địa chỉ trong bộ nhớ sau khi giá trị của biến `cartons` và `items` được gán.

Khi Button **Calculate Total** được nhấn, dòng 17 nhân các giá trị này và gán kết quả cho biến `result`. Lệnh

```
result = cartons * items
```

thực hiện phép nhân và thay thế (nghĩa là ghi đè lên) giá trị trước đó của biến `result`. Sau khi `result` đã được tính toán, bộ nhớ sẽ biểu diễn như trên Hình 6.14. Chú ý rằng giá trị của `cartons` và `items` xuất hiện đúng như giá trị mà chúng có trước khi được dùng trong phép nhân. Mặc dù các giá trị này được sử dụng khi máy tính thực hiện tính toán, chúng sẽ không bị mất đi. Điều này cho thấy rằng khi giá trị được đọc từ một địa chỉ trong bộ nhớ sẽ **không bị làm mất giá trị (nondestructive)** - nghĩa là giá trị sẽ không bị ghi đè lên.



Hình 6.14 Các vị trí trong bộ nhớ sau khi thực hiện phép nhân.

TỰ ÔN TẬP

1. Khi giá trị được đặt vào một địa chỉ trong bộ nhớ, nó sẽ_____giá trị đã được lưu từ trước tại địa chỉ đó.
   a) copy b) thay thế
   c) tự thêm vào d) xóa
2. Khi giá trị được đọc từ bộ nhớ, nó sẽ_____.
   a) bị ghi đè lên
   b) được thay thế bởi giá trị mới
   c) được chuyển đến địa chỉ mới trong bộ nhớ
   d) không bị ghi đè lên

Đáp án: **1)** b. **2)** d.

6.5 Phép toán số học

Hầu hết các chương trình đều thực hiện các phép tính toán số học. Trong chương trước, bạn đã thực hiện phép nhân số học bằng cách sử dụng toán tử nhân (*). Các **toán tử số học (arithmetic operator)** được tổng hợp trên Hình 6.15. Chú ý rằng các toán tử này sử dụng một số ký tự đặc biệt không được dùng trong đại số. Ví dụ, **dấu sao** (*) ký hiệu cho phép nhân, từ khóa **Mod** biểu diễn **toán tử mô-đun (modulus operator)**, **dấu sổ chéo ngược** (\) biểu diễn phép chia số nguyên và **dấu mũ** (^) biểu diễn phép toán lũy thừa. Hầu hết các toán tử số học trên Hình 6.15 là các **toán tử hai ngôi (binary operator)**, mỗi toán tử này đều có hai toán hạng.

Các phép toán trong Visual Basic .NET	Toán tử số học	Biểu thức đại số	Biểu thức trên Visual Basic 2008
Cộng	+	$f+7$	`f + 7`
Trừ	-	$p-c$	`p - c`
Nhân	*	bm	`b * m`
Chia (số thập phân)	/	x/y hoặc $\frac{x}{y}$ hoặc $x \div y$	`x / y`
Chia (số nguyên)	\	không có	`v \ u`
Mô-đun (chia lấy phần dư)	Mod	$r \bmod s$	`r Mod s`
Lũy thừa	^	q^p	`q ^ p`
Toán tử một ngôi âm	-	$-e$	`-e`
Toán tử một ngôi dương	+	$+g$	`+g`

Hình 6.15 Toán tử số học.

Ví dụ, biểu thức `sum + value` chứa toán tử hai ngôi + và hai toán hạng `sum` và `value`. Visual Basic cũng cung cấp **toán tử một ngôi**, là toán tử mà chỉ có một toán hạng. Ví dụ, toán tử một ngôi dương (+) và âm (-) cho phép người lập trình có thể viết các biểu thức dưới dạng +9 (một số dương) và -19 (một số âm).

Visual Basic phân biệt toán tử dùng trong **phép chia số nguyên** (dấu sổ chéo ngược, \) và **phép chia số có dấu chấm động** (dấu sổ chéo xuôi, /). Phép chia dấu chấm động chia hai số (số nguyên hoặc số thập phân) và trả về một số dấu chấm động (số thập phân). Phép chia số nguyên xem các toán hạng là số nguyên và trả về kết quả là một số nguyên. Khi số dấu chấm động (số thập phân) được sử dụng trong phép chia số nguyên, trước tiên chúng được làm tròn như sau:

- Các số kết thúc bởi `.5` được làm tròn thành số *chẵn* gần nhất - ví dụ `6.5`, được làm tròn *xuống* 6 và `7.5` được làm tròn *lên* 8.
- Tất cả các số dấu chấm động còn lại được làm tròn thành số nguyên gần nhất - ví dụ, `7.1` được làm tròn *xuống* 7 và `7.7` được làm tròn *lên* 8.

Lỗi lập trình thường gặp

Cố gắng chia cho 0 sẽ gây ra **lỗi thực thi** (là lỗi xảy ra khi ứng dụng đang chạy). Lỗi chia cho 0 làm cho ứng dụng kết thúc.

Sau đó phép chia sẽ được thực hiện. Điều này có nghĩa là, mặc dù `4.5 \ 2` cho kết quả là 2 như mong muốn, `5.5 \ 2` lại cho kết quả là 3, vì `5.5` được làm tròn thành 6 *trước khi* phép chia được thực hiện. Tương tự như vậy, mặc dù `7.1 \ 4` cho kết quả là 1 như mong muốn, `7.7 \ 4` lại cho kết quả là 2, vì `7.7` được làm tròn thành 8 *trước khi* phép chia được thực hiện. Chú ý rằng bất kỳ phần thập phân nào trong kết quả của phép chia số nguyên sẽ bị loại bỏ (hay còn gọi là bị chặt bớt) mà không làm tròn. Không toán tử chia nào cho phép chia cho 0. Nếu bạn viết mã thực hiện phép chia cho 0, **lỗi thực thi** (**runtime error**) hay còn gọi "exception" ("ngoại lệ") sẽ xảy ra. Theo mặc định, lỗi này sẽ làm cho ứng dụng kết thúc.

Toán tử mô-đun, `Mod`, cho kết quả là phần dư của phép chia. Biểu thức `x Mod y` cho kết quả là phần dư sau khi chia `x` cho `y`. Như vậy, `7 Mod 4` cho kết quả là 3, còn `17 Mod 5` cho kết quả là 2. Toán tử này thường được dùng với toán hạng kiểu `Integer`, nhưng cũng có thể dùng với các kiểu dữ liệu khác. Toán tử mô-đun có thể được áp dụng để giải quyết một số bài toán thú vị, ví dụ như xác định xem một số có phải là bội số của một số khác hay không. Nếu `a` và `b` là các số, `a Mod b` sẽ cho kết quả là 0 nếu `a` là một bội số của `b`. `8 Mod 3` cho kết quả là 2, do đó 8 không phải là bội số của 3. Ngược lại cả `8 Mod 2` và `8 Mod 4` đều cho kết quả là 0, vì 8 là bội số của cả 2 và 4.

Các biểu thức số học trong Visual Basic phải được viết **trên cùng một dòng** (**straight-line form**) để bạn có thể gõ vào trình soạn thảo. Ví dụ, phép chia `7.1` cho `4.3` không thể được viết là

$$\frac{7.1}{4.3}$$

nhưng được viết trên một dòng dưới dạng `7.1 / 4.3`. Hay 3 lũy thừa 2 không thể được viết dưới dạng 3^2 nhưng được viết trên một dòng dưới dạng `3 ^ 2`.

Dấu ngoặc đơn được sử dụng trong biểu thức của Visual Basic để nhóm các phép toán giống như trong biểu thức đại số. Để nhân a với tổng b + c, bạn phải viết

```
a * (b + c)
```

Visual Basic thực hiện các phép toán trong các biểu thức số học theo trình tự được xác định bởi **thứ tự ưu tiên của các toán tử (rules of operator precedence)**, thứ tự nói chung là giống với quy luật trong đại số. Các quy luật này giúp cho Visual Basic thực hiện các phép toán theo thứ tự chính xác.

Thứ tự ưu tiên của các toán tử

1. ***Các toán tử nằm bên trong dấu ngoặc đơn sẽ được ưu tiên trước.*** Do đó, *dấu ngoặc đơn được sử dụng để sắp xếp thứ tự ưu tiên theo cách mà bạn mong muốn.* Dấu ngoặc đơn được ưu tiên ở mức cao nhất. Với các **cặp ngoặc đơn lồng nhau** (một cặp dấu ngoặc đơn được chứa trong một cặp dấu ngoặc đơn khác), các toán tử nằm trong cặp dấu ngoặc đơn trong cùng được ưu tiên trước.
2. ***Toán tử lũy thừa được ưu tiên tiếp theo.*** Nếu biểu thức chứa nhiều toán tử lũy thừa, các toán tử đó sẽ được ưu tiên theo thứ tự từ trái sang phải.
3. ***Toán tử một ngôi dương và toán tử một ngôi âm (+ và -) được ưu tiên tiếp theo.*** Nếu biểu thức chứa nhiều toán tử này, các toán tử đó sẽ được ưu tiên theo thứ tự từ trái sang phải.
4. ***Toán tử nhân và toán tử chia số có dấu chấm động được ưu tiên tiếp theo.*** Nếu biểu thức chứa nhiều toán tử nhân và toán tử chia số có dấu chấm động, các toán tử đó sẽ được ưu tiên theo thứ tự từ trái sang phải.
5. ***Toán tử chia số nguyên được ưu tiên tiếp theo.*** Nếu biểu thức chứa nhiều toán tử chia số nguyên, các toán tử đó sẽ được ưu tiên theo thứ tự từ trái sang phải.
6. ***Toán tử mô-đun được ưu tiên tiếp theo.*** Nếu biểu thức chứa nhiều toán tử chia lấy phần dư, các toán tử đó sẽ được ưu tiên theo thứ tự từ trái sang phải.
7. ***Toán tử cộng và toán tử trừ được ưu tiên cuối cùng.*** Nếu biểu thức chứa nhiều toán tử cộng và toán tử trừ, các toán tử đó sẽ được ưu tiên theo thứ tự từ trái sang phải.

Chú ý rằng chúng ta đã đề cập đến các cặp dấu ngoặc đơn lồng nhau. Không phải tất cả các biểu thức chứa nhiều cặp dấu ngoặc đơn đều có các cặp ngoặc đơn lồng nhau. Ví dụ, mặc dù biểu thức

```
a * (b + c) + c * (d + e)
```

chứa nhiều cặp ngoặc đơn, không có cặp ngoặc đơn nào trong số đó là lồng nhau. Các cặp ngoặc đơn này được coi là "có cấp độ như nhau" và được ưu tiên theo thứ tự từ trái sang phải.

Hãy xem xét một số biểu thức dựa trên thứ tự ưu tiên của các toán tử. Mỗi ví dụ đưa ra một biểu thức đại số và một biểu thức tương đương của nó trong Visual Basic.

Biểu thức sau tính toán trung bình cộng của ba số.

Đại số: $m = \frac{(a+b+c)}{3}$

Visual Basic:

```
m = (a + b + c) / 3
```

Các dấu ngoặc đơn là cần thiết vì phép chia số có dấu chấm động có thứ tự ưu tiên cao hơn phép cộng. Tổng của `(a + b + c)` sẽ được chia cho 3. Nếu bỏ đi các dấu ngoặc, chúng ta sẽ có biểu thức sai là `a + b + c / 3`, tương đương với biểu thức đại số

$$a+b+\frac{c}{3}$$

Biểu thức sau là một phương trình đường thẳng

Đại số: $y = mx + b$

Visual Basic: `y = m * x + b`

Không cần thiết phải có dấu ngoặc đơn. Phép nhân sẽ được ưu tiên trước, do nó có thứ tự ưu tiên cao hơn phép cộng. Phép gán xảy ra sau cùng vì nó có thứ tự ưu tiên thấp hơn phép nhân và phép cộng.

Để hiểu kỹ hơn về thứ tự ưu tiên của các toán tử, hãy xem xét xem biểu thức $y = ax^2 + bx + c$ được ưu tiên như nào:

```
y = a * x ^ 2 + b * x + c
    6   2   1   4   3   5
```

Các số được khoanh tròn bên dưới biểu thức chỉ ra thứ tự mà Visual Basic thực hiện tính toán. Hãy nhớ lại rằng trong Visual Basic thì x^2 được biểu diễn dưới dạng `x ^ 2`. Đồng thời chú ý rằng toán tử gán được thực hiện cuối cùng vì nó có thứ tự ưu tiên thấp hơn bất kỳ toán tử số học nào.

Thói quen lập trình tốt

Hãy sử dụng cặp ngoặc đơn phụ trong các biểu thức số học phức tạp để cho biểu thức dễ đọc hơn.

Giống như trong đại số, bạn được phép thêm các **cặp ngoặc đơn phụ** (**redundant parentheses**) vào biểu thức để biểu thức dễ đọc hơn, mặc dù nếu không thêm thì biểu thức vẫn đúng. Ví dụ, lệnh gán trên có thể sử dụng các cặp ngoặc đơn phụ để biểu thức rõ ràng hơn:

```
y = (a * (x ^ 2)) + (b * x) + c
```

TỰ ÔN TẬP

1. Các biểu thức số học trong Visual Basic phải được viết_____để thuận tiện trong việc nhập biểu thức vào máy tính.
   a) sử dụng dấu ngoặc đơn
   b) trên nhiều dòng
   c) trên cùng một dòng
   d) Các lựa chọn trên đều sai
2. Biểu thức ở bên phải của toán tử gán (=) luôn được thực hiện_____khi phép gán được thực hiện.
   a) trước
   b) sau
   c) cùng một lúc
   d) Các lựa chọn trên đều sai

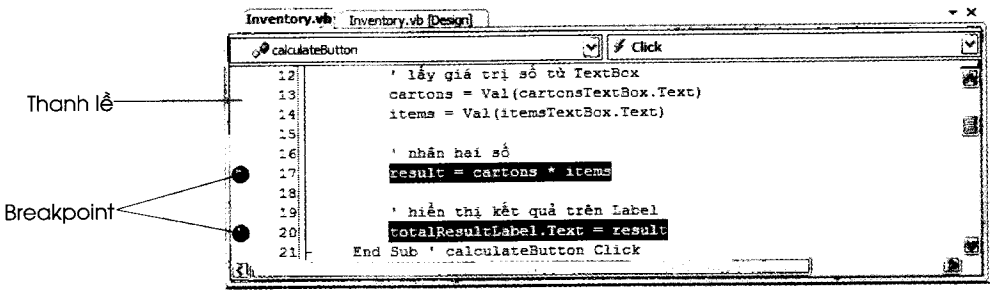
Đáp án: **1)** c. **2)** a.

6.6 Sử dụng trình gỡ lỗi: Breakpoint

Trình gỡ lỗi là một trong những công cụ quan trọng nhất của bạn trong việc phát triển các ứng dụng, khi mà bạn đã trở nên quen thuộc với các tính năng của nó. Bạn đã được giới thiệu về trình gỡ lỗi trong Chương 5, khi bạn sử dụng nó để tìm và khắc phục các lỗi cú pháp. Trong chương này, bạn sẽ tiếp tục tìm hiểu về trình gỡ lỗi và sử dụng **breakpoint (điểm dừng)**, để xác định xem ứng dụng đang làm gì khi nó chạy. Breakpoint là một đánh dấu được thiết lập tại bất kỳ dòng mã nào có thể thực thi. Khi chạy đến một breakpoint, ứng dụng sẽ tạm dừng và bạn có thể xem bên trong ứng dụng để đảm bảo không mắc lỗi lôgíc, ví dụ như lỗi tính toán không chính xác. Trong phần tiếp theo, *Sử dụng trình gỡ lỗi: Breakpoint*, bạn sẽ tìm hiểu cách sử dụng các breakpoint trong trình gỡ lỗi của Visual Basic IDE.

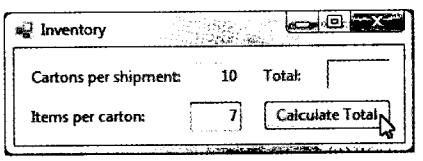
Sử dụng trình gỡ lỗi: Breakpoint

1. ***Thêm breakpoint vào IDE Visual Basic.*** Hãy chắc chắn rằng project `Inventory3` đang được mở ở chế độ **Code** của IDE. Để chèn breakpoint vào IDE, có thể nhấn vào **thanh lề (margin indicator bar)** - phần lề mầu xám ở bên trái cửa sổ Code, Hình 6.16 - bên cạnh dòng mã mà ở đó bạn muốn ứng dụng tạm dừng hoặc cũng có thể nhấn chuột phải vào dòng mã đó và chọn **Breakpoint > Insert Breakpoint**. Bạn có thể thiết lập số lượng breakpoint tùy theo ý muốn. Thiết lập các breakpoint ở các dòng 17 và 20 trong mã của bạn. Một hình tròn đặc xuất hiện tại vị trí bạn đã nhấn vào, chỉ ra rằng breakpoint đã được thiết lập (Hình 6.16). Khi ứng dụng chạy, nó sẽ tạm ngừng việc thực thi tại những dòng có breakpoint. Ứng dụng được gọi là đang ở **chế độ ngắt (break mode)** khi trình gỡ lỗi tạm dừng thực thi ứng dụng. Breakpoint có thể được thiết lập trong chế độ thiết kế (design mode), chế độ ngắt và chế độ chạy (run mode).

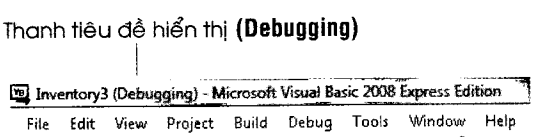


Hình 6.16 Thiết lập hai breakpoint.

2. ***Bắt đầu quá trình gỡ lỗi.*** Sau khi thiết lập breakpoint trong trình soạn thảo mã, chọn **Debug > Start Debugging** để bắt đầu quá trình gỡ lỗi. Khi ứng dụng Windows đang được gỡ lỗi, cửa sổ ứng dụng xuất hiện (Hình 6.17) cho phép tương tác với ứng dụng (nhập dữ liệu vào và hiển thị kết quả). Nhập 10 và 7 vào các `TextBox` và nhấn vào **Calculate Total** để tiếp tục. Thanh tiêu đề của IDE hiển thị **(Debugging)** (Hình 6.18) chỉ ra rằng IDE đang ở trong chế độ ngắt.

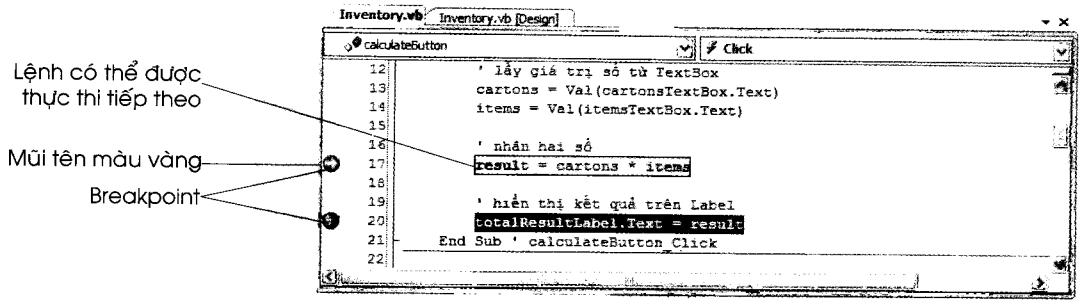


Hình 6.17 Ứng dụng **Inventory** đang chạy.



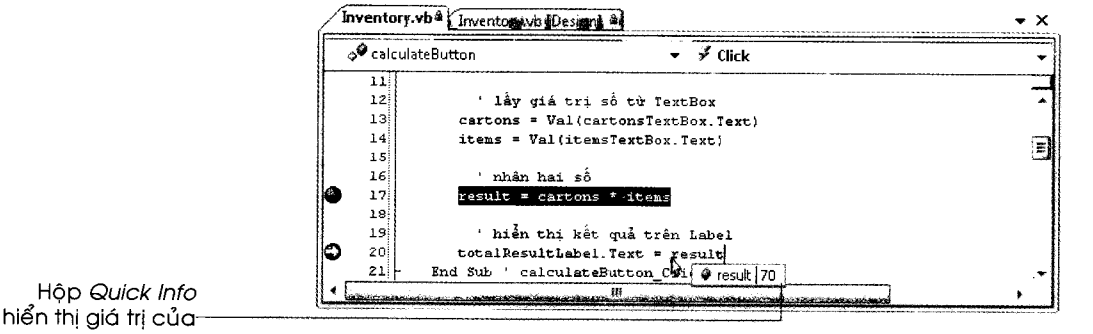
Hình 6.18 Thanh tiêu đề của IDE đang hiển thị **(Debugging)**.

3. ***Kiểm tra quá trình thực thi ứng dụng.*** Ứng dụng đang chạy sẽ tạm ngừng ở breakpoint đầu tiên, IDE sẽ trở thành **cửa sổ hiện thời (active window)** (Hình 6.19). Mũi tên màu vàng ở bên trái dòng 17 cho biết dòng này chứa lệnh được thực thi tiếp theo.



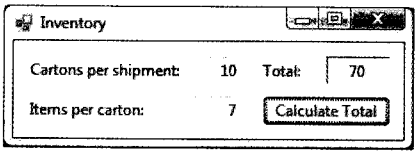
Hình 6.19 Ứng dụng đang chạy tạm dừng ở breakpoint đầu tiên.

4. ***Sử dụng lệnh Continue để tiếp tục thực thi ứng dụng.*** Để tiếp tục thực thi ứng dụng, chọn **Debug > Continue** (hoặc nhấn *F5*). Ứng dụng sẽ chạy tiếp cho đến khi gặp breakpoint tiếp theo ở dòng 20. Chú ý rằng khi bạn đặt con trỏ chuột bên trên tên biến `result`, giá trị mà biến đang chứa sẽ được hiển thị trong hộp ***Quick Info*** (Hình 6.20). Bạn sẽ có cảm giác như đang nhìn vào bên trong máy tính để xem giá trị của biến. Việc này sẽ giúp bạn phát hiện lỗi lôgíc trong ứng dụng.



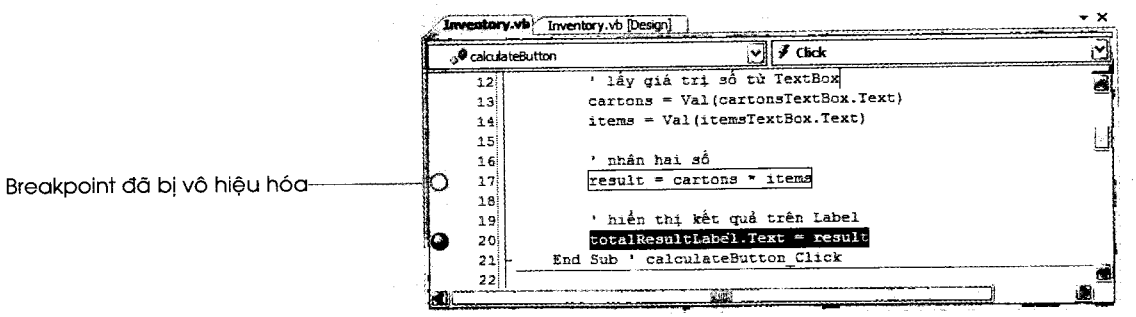
Hình 6.20 Hiển thị giá trị của biến bằng cách đặt con trỏ chuột bên trên tên biến.

5. ***Kết thúc thực thi ứng dụng.*** Sử dụng lệnh **Debug > Continue** để hoàn thành thực thi ứng dụng. Khi không còn breakpoint nào để tạm ngừng việc thực thi, ứng dụng sẽ chạy cho đến khi hoàn tất và kết quả đầu ra sẽ được hiển thị trên `Label` **Total:** (Hình 6.21).



Hình 6.21 Hiển thị kết quả của ứng dụng.

6. ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.
7. ***Vô hiệu hóa breakpoint***. Để vô hiệu hóa breakpoint, nhấn chuột phải vào dòng lệnh mà tại đó nó đã được thiết lập và chọn **Breakpoint > Disable Breakpoint**. Breakpoint đã bị vô hiệu hóa được chỉ ra bởi một hình tròn rỗng màu nâu sẫm (Hình 6.22). Chỉ tạm thời vô hiệu hóa thay vì loại bỏ breakpoint bởi vô hiệu hóa cho phép bạn tái kích hoạt (bằng cách nhấn vào bên trong vòng tròn rỗng) trên thanh lề. Bạn cũng có thể thực hiện việc này bằng cách nhấn chuột phải vào dòng lệnh đã được đánh dấu bởi vòng tròn rỗng màu nâu sẫm và chọn **Breakpoint > Enable Breakpoint**.



Hình 6.22 Breakpoint đã bị vô hiệu hóa.

8. ***Loại bỏ breakpoint.*** Để loại một breakpoint không còn cần thiết, nhấn chuột phải vào dòng mã nơi nó được thiết lập và chọn **Breakpoint > Delete Breakpoint**. Bạn cũng có thể loại bỏ breakpoint bằng cách nhấn vào hình tròn màu nâu sẫm trong thanh lề. Để loại bỏ *tất cả* các breakpoint, chọn **Debug > Delete All Breakpoints** từ thanh menu.
9. ***Lưu project.*** Chọn **File > Save All** để lưu mã đã được sửa đổi của bạn.
10. ***Đóng IDE.*** Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

Trong phần này, bạn đã tìm hiểu cách sử dụng trình gỡ lỗi để thiết lập breakpoint, từ đó có thể kiểm tra kết quả của mã khi ứng dụng đang chạy. Bạn cũng đã tìm hiểu cách tiếp tục chạy ứng dụng sau khi nó tạm dừng tại breakpoint, cùng với cách vô hiệu hóa và loại bỏ breakpoint.

TỰ ÔN TẬP

1. Breakpoint không thể được thiết lập tại _____.
   a) chú thích
   b) dòng mã có thể thực thi được
   c) lệnh gán
   d) lệnh tính toán số học
2. Khi việc thực thi ứng dụng tạm dừng tại breakpoint, lệnh tiếp theo được thực thi là lệnh_____breakpoint.
   a) trước
   b) sau
   c) tại vị trí của
   d) Các lựa chọn trên đều sai

Đáp án: 1) a. 2) c.

6.7 Tổng kết

Trong chương này, bạn đã thêm biến vào ứng dụng **Inventory**. Bạn bắt đầu bằng việc sử dụng biến để có kết quả giống như trong ứng dụng **Inventory** trước. Sau đó, bạn đã nâng cấp ứng dụng **Inventory**, sử dụng sự kiện `TextChanged` cho phép thực thi mã để xóa giá trị trên `Label` hiển thị kết quả khi người dùng thay đổi giá trị của bất kỳ `TextBox` nào.

Bạn đã tìm hiểu khái niệm về bộ nhớ, cách biến được đọc ra và ghi vào bộ nhớ. Bạn sẽ áp dụng khái niệm này trong các ứng dụng được xây dựng ở những chương tiếp theo, đó là những ứng dụng dựa nhiều vào biến. Bạn cũng đã tìm hiểu cách thực hiện các phép toán số học trong Visual Basic, cùng với thứ tự ưu tiên của các toán tử để thực hiện các biểu thức toán học chính xác. Cuối cùng, bạn đã tìm hiểu cách thêm breakpoint trong trình gỡ lỗi. Breakpoint cho phép bạn tạm ngừng việc thực thi ứng dụng và kiểm tra giá trị của biến. Tính năng này sẽ cho thấy tác dụng trong việc tìm và sửa lỗi lôgíc.

Trong chương tiếp theo, bạn sẽ thiết kế giao diện và viết mã để tạo ứng dụng tính tiền lương. Bạn sẽ sử dụng mã giả, là một ngôn ngữ không chính thống hỗ trợ thiết kế ứng dụng. Bạn cũng sẽ tìm hiểu cách sử dụng cửa sổ **Watch** trong trình gỡ lỗi, một công cụ hữu ích giúp loại bỏ lỗi lôgíc.

TỔNG KẾT KỸ NĂNG

Khai báo biến

- Sử dụng từ khóa `Dim`.
- Sử dụng định danh tên biến hợp lệ.
- Sử dụng từ khóa `As` để cho biết rằng từ theo sau nó là kiểu dữ liệu của biến.
- Chỉ ra kiểu dữ liệu ví dụ như `Integer` hoặc `Double`.

Xử lý sự kiện TextChanged của TextBox

- Nhấn đúp vào `TextBox` trên `Form` để tạo xử lý sự kiện `TextChanged` rỗng.
- Thêm mã vào trong phần xử lý sự kiện để thực thi khi nội dung văn bản trong `TextBox` thay đổi.

Đọc giá trị từ một địa chỉ trong bộ nhớ

- Sử dụng tên biến (như đã được khai báo trong lệnh `Dim` của biến) tại vị trí trong mã cần đến giá trị của biến.

Thay thế giá trị tại một địa chỉ trong bộ nhớ

- Sử dụng tên biến, theo sau bởi toán tử gán (=), cùng với một biểu thức đưa ra giá trị mới.

Biểu diễn các số âm và dương

- Sử dụng các toán tử một ngôi dương (+) và âm (-).

Thực hiện phép toán số học

- Viết biểu thức số học trên cùng một dòng.
- Sử dụng thứ tự ưu tiên của các toán tử để xác định thứ tự thực hiện các phép tính.
- Sử dụng toán tử + để thực hiện phép cộng.
- Sử dụng toán tử - để thực hiện phép trừ.
- Sử dụng toán tử * để thực hiện phép nhân.
- Sử dụng toán tử / để thực hiện phép chia số có dấu chấm động.
- Sử dụng toán tử \ (dấu sổ chéo ngược) để thực hiện phép chia số nguyên, phép chia coi các toán hạng như các `Integer` và trả về một kết quả kiểu `Integer`.
- Sử dụng toán tử ^ để thực hiện phép toán lũy thừa.
- Sử dụng toán tử mô-đun, `Mod`, để lấy phần dư của phép chia.
- Sử dụng các dấu ngoặc đơn để quản lý thứ tự của các toán tử và làm cho biểu thức trở nên rõ ràng.

Thiết lập Breakpoint

- Nhấn vào thanh lề (phần lề mầu xám ở bên trái cửa sổ Code) bên cạnh dòng lệnh mà bạn muốn tạm dừng hoặc nhấn chuột phải vào dòng mã đó và chọn **Breakpoint > Insert Breakpoint.**

Tiếp tục thực thi ứng dụng sau khi chuyển sang chế độ ngắt trong quá trình gỡ lỗi.

- Chọn **Debug > Continue.**

Vô hiệu hóa breakpoint

- Nhấn chuột phải vào dòng mã chứa breakpoint và chọn **Breakpoint > Disable Breakpoint.**

Kích hoạt breakpoint

- Kích hoạt breakpoint đã bị vô hiệu hóa bằng cách nhấn vào bên trong vòng tròn rỗng trên thanh lề.
- Bạn cũng có thể kích hoạt breakpoint đã bị vô hiệu hóa bằng cách nhấn chuột phải vào dòng lệnh đã được đánh dấu bởi vòng tròn rỗng màu nâu sẫm và chọn **Breakpoint > Enable Breakpoint.**

Loại bỏ breakpoint

- Nhấn chuột phải vào dòng mã chứa breakpoint và chọn **Breakpoint > Delete Breakpoint.**
- Bạn cũng có thể loại bỏ breakpoint bằng cách nhấn vào hình tròn màu nâu sẫm trên thanh lề.

THUẬT NGỮ

biến (variable) - Địa chỉ trong bộ nhớ máy tính lưu trữ giá trị.

breakpoint (điểm dừng)- Vị trí tại đó ứng dụng tạm dừng thực thi, được chỉ ra bởi một hình tròn đặc màu nâu sẫm.

cặp ngoặc đơn lồng nhau (nested parentheses) - Xảy ra khi một biểu thức nằm trong một cặp dấu ngoặc đơn ở bên trong một biểu thức khác cũng được bao bởi một cặp dấu ngoặc đơn. Trong trường hợp này, các toán tử nằm trong cặp dấu ngoặc đơn trong cùng sẽ được thực hiện trước.

cặp ngoặc đơn phụ (redundant parentheses) - Các cặp ngoặc đơn được thêm vào biểu thức để biểu thức dễ đọc hơn.

chế độ ngắt (break mode) - Chế độ của IDE khi tạm dừng thực thi ứng dụng. Chế độ này được kích hoạt thông qua trình gỡ lỗi.

chuỗi rỗng (empty string) - Chuỗi không chứa bất kỳ một ký tự nào.

cửa sổ hiện thời (active window) - Cửa sổ trên cùng trên màn hình.

dấu mũ (^) - Toán tử lũy thừa. Toán tử này nâng lũy thừa toán hạng bên trái với số mũ được xác định bởi toán hạng bên phải của nó.

dấu sao (*) - Toán tử nhân. Các toán hạng bên phải và bên trái của toán tử được nhân với nhau.

dấu sổ chéo ngược (\) - Toán tử chia số nguyên. Toán tử này chia toán hạng bên trái cho toán hạng bên phải của nó và trả về kết quả kiểu `Integer`.

dấu sổ chéo xuôi (/) - Toán tử chia số có dấu chấm động. Toán tử này chia toán hạng bên trái cho toán hạng bên phải của nó và trả về một số dấu chấm động (số thập phân).

ép kiểu ngầm định (implicit conversion) - Phép chuyển đổi từ một kiểu dữ liệu thành một kiểu dữ liệu khác được thực hiện bởi Visual Basic.

giá trị của biến (value of a variable) - Một mẫu dữ liệu được lưu tại địa chỉ của biến trong bộ nhớ.

hộp *Quick Info* - Hiển thị giá trị của một biến trong quá trình gỡ lỗi.

khai báo (declaration) - Thông báo biến mới cho trình biên dịch. Sau đó biến có thể được dùng trong mã Visual Basic.

khai báo biến (variable declaration) - Thông báo cho trình biên dịch tên và kiểu của biến.

kích thước của biến (size of a variable) - Số lượng byte cần thiết để lưu giá trị thuộc kiểu dữ liệu của biến.

kiểu của biến (type of a variable) - Chỉ ra loại dữ liệu có thể được lưu trong biến và phạm vi của các giá trị được lưu.

kiểu dữ liệu có sẵn (built-in data type) - Kiểu dữ liệu đã được định nghĩa trong Visual Basic, ví dụ như `Integer`.

kiểu dữ liệu cơ sở (primitive data type) - Kiểu dữ liệu đã được định nghĩa trong Visual Basic, ví dụ như `Integer`.

kiểu dữ liệu `Double` - Lưu cả số nguyên và số thập phân. Thông thường, `Double` chứa số có dấu chấm động.

kiểu dữ liệu `Integer` - Chứa giá trị nguyên.

lỗi (bug) - Sai sót trong chương trình làm cho ứng dụng thực hiện không chính xác.

lỗi thực thi (runtime error) - Lỗi gây ảnh hưởng trong thời gian chạy.

Mod (toán tử mô-đun) - Toán tử mô-đun trả về phần dư của phép chia.

phép chia số có dấu chấm động (floating-point division) - Chia hai số (số nguyên hoặc số thập phân) và trả về một số có dấu chấm động.

phép chia số nguyên (integer division) - Phép chia số nguyên thực hiện trên hai toán hạng kiểu `Integer` và trả về kết quả kiểu `Integer`. Phần thập phân của kết quả được loại bỏ.

số nguyên (integer) - Số nguyên, ví dụ như `919`, `-11`, `0` và `138624`.

sự kiện TextChanged - Xảy ra khi nội dung văn bản trong `TextBox` thay đổi.

tên biến (name of a variable) - Định danh được sử dụng trong ứng dụng để truy cập hoặc thay đổi giá trị của biến.

thanh lề (margin indicator bar) - Phần lề của IDE nơi breakpoint được hiển thị.

thao tác không làm mất giá trị trong bộ nhớ (nondestructive memory operation) - Quá trình xử lý trong đó giá trị trong bộ nhớ không bị ghi đè lên.

thứ tự ưu tiên của các toán tử (rules of operator precedence) - Các quy luật xác định thứ tự thực hiện các phép toán trong biểu thức.

toán tử hai ngôi (binary operator) - Toán tử có hai toán hạng.

toán tử lũy thừa (`^`) - Toán tử này nâng lũy thừa toán hạng bên trái với số mũ được chỉ ra bởi toán hạng bên phải của nó.

toán tử một ngôi (unary operator) - Toán tử có duy nhất một toán hạng.

toán tử số học (arithmetic operator) - Các toán tử `+`, `-`, `*`, `/`, `\`, `^` và `Mod`.

trên cùng một dòng (straight-line form) - Cách ghi biểu thức số học để có thể trình bày được trên mã Visual Basic.

từ khóa As - Dùng trong khai báo biến. Chỉ ra rằng từ theo sau nó (ví dụ như `Integer`) là kiểu dữ liệu của biến.

từ khóa Dim - Cho biết đó là lời khai báo biến.

ĐIỀU KHIỂN, SỰ KIỆN, THUỘC TÍNH & PHƯƠNG THỨC

TextBox (TextBox) Điều khiển này cho phép người dùng nhập dữ liệu vào từ bàn phím.

- ***Trên giao diện khi ứng dụng chạy***

  0

- ***Sự kiện***

  `TextChanged` - Được kích hoạt khi nội dung văn bản trong `TextBox` bị thay đổi.

- ***Thuộc tính***

  `Location` - Chỉ ra vị trí tương đối của `TextBox` trên `Form` so với góc trên bên trái của `Form`.

  `Name` - Chỉ ra tên được dùng để định danh `TextBox`. Nên thêm hậu tố `TextBox` vào sau tên.

  `Size` - Chỉ ra chiều rộng và chiều cao (bằng pixel) của `TextBox`.

  `Text` - Chỉ ra nội dung văn bản được hiển thị ban đầu trên `TextBox`.

  `TextAlign` - Chỉ ra cách căn chỉnh văn bản trong `TextBox`.

CÂU HỎI TRẮC NGHIỆM

6.1 Cặp dấu ngoặc đơn được thêm vào biểu thức chỉ để làm cho nó dễ đọc hơn được gọi là cặp dấu ngoặc đơn _____.

a) cần thiết
b) phụ
c) được chứa trong cặp ngoặc đơn khác
d) lồng nhau

6.2 Toán tử_____thực hiện phép chia số nguyên.

a) \
b) +
c) Mod
d) ^

6.3 Tất cả các biến đều có_____.

a) tên
b) giá trị
c) kiểu dữ liệu
d) Các lựa chọn trên đều đúng

6.4 Trong Visual Basic, biểu thức số học phải được viết_____.

a) trên cùng một dòng
b) từ trên xuống dưới
c) từ trái sang phải
d) từ phải sang trái

6.5 Biểu thức số học được thực hiện _____.

a) từ phải sang trái
b) từ trái sang phải
c) tuân theo thứ tự ưu tiên của các toán tử
d) cả b và c

6.6 Khai báo biến trong các xử lý sự kiện bắt đầu bởi từ khóa _____ .

a) Declare
b) Dim
c) Sub
d) Integer

6.7 Nhập giá trị vào một TextBox sẽ kích hoạt sự kiện _____ .

a) TextAltered
b) ValueChanged
c) ValueEntered
d) TextChanged

6.8 Hàm_____chuyển đổi giá trị người dùng nhập vào trong TextBox thành giá trị kiểu Double.

a) Convert
b) MakeDouble
c) Val
d) WriteDouble

6.9 Biến chứa giá trị nguyên phải được khai báo kiểu_____.

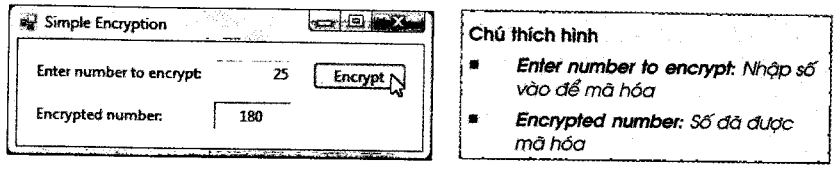
a) Integer
b) Int
c) IntVariable
d) Các lựa chọn trên đều sai

6.10 Kiểu dữ liệu trong khai báo biến được viết sau từ khóa_____.

a) IsA
b) Type
c) Dim
d) As

BÀI TẬP

6.11 (***Ứng dụng Simple Encryption***) Ứng dụng này sử dụng kỹ thuật đơn giản để mã hóa một số. Mã hóa là quá trình biến đổi dữ liệu sao cho chỉ những người hợp lệ nhận được dữ liệu mới có thể khôi phục lại thay đổi và xem dữ liệu ban đầu trước khi mã hóa. Người dùng nhập dữ liệu cần mã hóa vào `TextBox`. Sau đó ứng dụng sẽ nhân số đã nhập với 7 rồi cộng thêm 5. Ứng dụng hiển thị số đã được mã hóa trong `Label` như trên Hình 6.23.



Hình 6.23 Kết quả của ứng dụng **Simple Encryption** đã hoàn thiện.

a) ***Copy template của ứng dụng vào thư mục làm việc***. Copy thư mục `C:\Examples\Tutorial06\Exercises\SimpleEncryption` vào thư mục `C:\SimplyVB2008` của bạn.

b) ***Mở file template của ứng dụng***. Nhấn đúp vào `SimpleEncryption.sln` trong thư mục `SimpleEncryption` để mở ứng dụng.

c) ***Viết mã cho xử lý sự kiện `Click`***. Mã hóa số trong xử lý sự kiện `Click` của `Button` **Encrypt** bằng cách sử dụng kỹ thuật đã nêu ở trên. Dữ liệu do người dùng nhập vào cần được lưu trong một biến có tên là `number` kiểu `Integer` trước khi nó được mã hóa. Sau đó xử lý sự kiện sẽ hiển thị số đã được mã hóa.

d) ***Xóa kết quả***. Thêm xử lý sự kiện cho sự kiện `TextChanged` của `TextBox` **Enter number to encrypt:**. Xử lý sự kiện này sẽ xóa hết nội dung trên `Label` hiển thị kết quả mỗi khi người dùng nhập vào một giá trị mới.

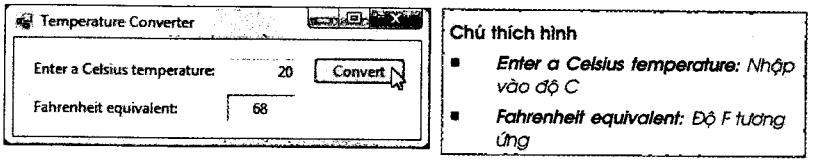
e) ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng. Nhập `25` vào `TextBox` **Enter number to encrypt:** và nhấn vào `Button` **Encrypt**. Kiểm tra xem giá trị `180` có được hiển thị trên `Label` hiển thị kết quả **Encrypted number:** không. Nhập những giá trị khác và nhấn vào `Button` **Encrypt** sau mỗi lần nhập. Kiểm tra xem các giá trị mã hóa tương ứng có được hiển thị không.

f) ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

g) ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

6.12 (***Ứng dụng Temperature Converter***) Viết một ứng dụng chuyển đổi độ C thành độ F tương ứng. Hình 6.24 hiển thị ứng dụng đã hoàn thiện. Bạn hãy sử dụng công thức sau:

$$F = \frac{9}{5} C + 32$$



Hình 6.24 Ứng dụng **Temperature Converter** đã hoàn thiện.

a) ***Copy template của ứng dụng vào thư mục làm việc***. Copy thư mục `C:\Examples\Tutorial06\Exercises\TemperatureConversion` vào thư mục `C:\SimplyVB2008` của bạn.

b) ***Mở file template của ứng dụng***. Nhấn đúp vào `TemperatureConversion.sln` trong thư mục `TemperatureConversion` để mở ứng dụng.

c) ***Viết mã cho xử lý sự kiện Click***. Thực hiện chuyển đổi trong xử lý sự kiện `Click` của `Button` **Convert**. Định nghĩa biến `Integer` để lưu độ C do người dùng nhập vào và kết quả sau khi chuyển đổi. Hiển thị độ F tương ứng với độ C.

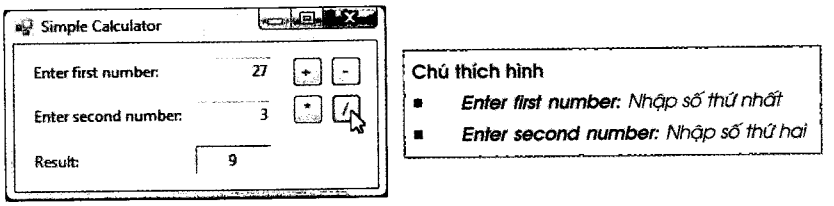
d) ***Xóa kết quả hiển thị***. Xóa hết kết quả trong sự kiện `TextChanged` của `TextBox` **Enter a Celsius temperature:**.

e) ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng của bạn. Nhập 20 vào `TextBox` **Enter a Celsius temperature:** và nhấn vào `Button` **Convert**. Kiểm tra xem giá trị 68 có được hiển thị trên `Label` hiển thị kết quả không. Nhập những giá trị độ C khác và nhấn vào `Button` **Convert** sau mỗi lần nhập. Sử dụng công thức trên để kiểm tra xem độ F tương ứng có được hiển thị sau mỗi lần nhập không.

f) ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

g) ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

6.13 (***Ứng dụng Calculator đơn giản***) Trong bài tập này, bạn sẽ thêm tính năng cho ứng dụng tính toán đơn giản. Ứng dụng cho phép người dùng nhập vào hai số trên `TextBox`. Có bốn `Button` được ghi nhãn lần lượt là +, -, / và *. Khi người dùng nhấn vào các `Button` đó, ứng dụng thực hiện phép toán tương ứng trên các số trong các `TextBox` và hiển thị kết quả. Ứng dụng cũng xóa hết kết quả tính toán khi người dùng nhập vào giá trị mới. Hình 6.25 hiển thị ứng dụng đã hoàn thiện.



Hình 6.25 Kết quả của ứng dụng **Calculator**.

a) ***Copy template của ứng dụng vào thư mục làm việc***. Copy thư mục `C:\Examples\Tutorial06\Exercises\SimpleCalculator` vào thư mục `C:\SimplyVB2008` của bạn.

b) ***Mở file template của ứng dụng***. Nhấn đúp vào `SimpleCalculator.sln` trong thư mục `SimpleCalculator` để mở ứng dụng.

c) ***Viết mã cho xử lý sự kiện Click của phép cộng***. Xử lý sự kiện này cộng hai số và hiển thị kết quả.

d) ***Viết mã cho xử lý sự kiện Click của phép trừ***. Xử lý sự kiện này trừ số thứ nhất cho số thứ hai và hiển thị kết quả.

e) ***Viết mã cho xử lý sự kiện Click của phép nhân***. Xử lý sự kiện này nhân hai số và hiển thị kết quả.

f) ***Viết mã cho xử lý sự kiện Click của phép chia***. Xử lý sự kiện này chia số thứ nhất cho số thứ hai và hiển thị kết quả.

g) ***Xóa kết quả.*** Viết xử lý sự kiện cho sự kiện `TextChanged` của các `TextBox`. Viết mã để xóa `resultLabel` sau khi người dùng nhập giá trị mới vào một `TextBox` nào đó.

h) ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng của bạn. Nhập số thứ nhất và số thứ hai, sau đó kiểm tra bằng cách nhấn vào từng `Button` và xem kết quả. Lặp lại quá trình này với hai giá trị mới và kiểm tra xem kết quả ứng với `Button` được nhấn có được hiển thị đúng hay không.

i) ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

j) ***Đóng IDE.*** Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

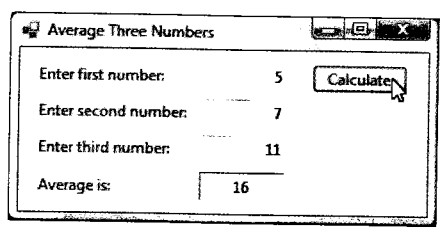
Đoạn mã này làm gì? ▶ **6.14** Đoạn mã này thay đổi các biến `number1`, `number2` và `result`. Giá trị cuối cùng của các biến này là gì?

```
1  Dim number1 As Integer
2  Dim number2 As Integer
3  Dim result As Integer
4
5  number1 = 5 * (4 + 6)
6  number2 = 2 ^ 2
7  result = number1 \ number 2
```

Đoạn mã này có gì sai? ▶ **6.15** Tìm lỗi trong đoạn mã sử dụng biến để thực hiện tính toán dưới đây.

```
1  Dim number1 As Integer
2  Dim number2 As Integer
3  Dim result As Integer
4
5  number1 = (4 * 6 ^ 4) / (10 Mod 4 - 2)
6  number2 = (16 \ 3) ^ 2 * 6 + 1
7  result = number1 - number 2
```

Sử dụng trình gỡ lỗi ▶ **6.16** (***Average Three Numbers***) Bạn vừa viết ứng dụng lấy vào ba số từ các `TextBox`, lưu ba số đó trong các biến, sau đó tìm trung bình cộng của các số đó (chú ý rằng trung bình cộng được làm tròn đến số nguyên gần nhất). Kết quả được hiển thị trên `Label` (Hình 6.26, ở đó kết quả sai được hiển thị). Tuy nhiên, bạn sẽ sớm thấy rằng số được hiển thị trong `Label` không phải là trung bình cộng mà là một số không đúng với đầu vào. Sử dụng trình gỡ lỗi để tìm và loại bỏ lỗi này.



Hình 6.26 Ứng dụng **Average Three Numbers.**

a) ***Copy template vào thư mục làm việc của bạn.*** Copy thư mục `C:\Examples\Tutorial06\Exercises\AverageDebugging` vào thư mục `C:\SimplyVB2008` của bạn.

b) ***Mở file template của ứng dụng.*** Nhấn đúp vào `AverageDebugging.sln` trong thư mục `AverageDebugging` để mở ứng dụng.

c) ***Chạy ứng dụng.*** Chọn **Debug > Start Debugging** để chạy ứng dụng. Xem kết quả được hiển thị để thấy rằng kết quả không chính xác.

d) ***Đóng ứng dụng.*** Đóng ứng dụng và xem file `Average.vb` ở chế độ **Code**.

e) ***Thiết lập các breakpoint.*** Thiết lập breakpoint trong phần xử lý sự kiện `calculateButton_Click`. Chạy lại ứng dụng và sử dụng trình gỡ lỗi để tìm lỗi.

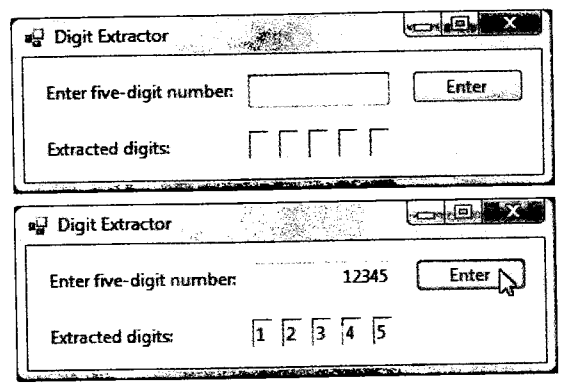
f) ***Tìm và sửa lỗi.*** Sau khi bạn đã tìm ra các lỗi, hãy thay đổi ứng dụng sao cho nó tính toán đúng trung bình cộng của ba số.

g) ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng của bạn. Nhập ba số như trên Hình 6.26 vào các `TextBox` và nhấn vào `Button` **Calculate**. Kiểm tra xem ứng dụng có hiển thị đúng đầu ra là 8 không.

h) ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

i) ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

Bài tập nâng cao ▶ **6.17** (***Digit Extraction***) Viết ứng dụng cho phép người dùng nhập số có năm chữ số vào `TextBox`. Sau đó ứng dụng sẽ phân chia số đó thành các chữ số đơn lẻ và hiển thị mỗi chữ số trên một `Label`. Ứng dụng hiển thị kết quả như trên Hình 6.27. [*Gợi ý*: Bạn có thể sử dụng toán tử `Mod` để tách chữ số hàng đơn vị từ một số. Ví dụ, `12345 Mod 10` cho kết quả là `5`. Bạn có thể sử dụng phép chia số nguyên (`\`) để "bóc tách" các chữ số từ một số. Ví dụ, `12345\100` cho kết quả là `123`. Bây giờ bạn có thể tách số 3 ra bằng cách sử dụng toán tử `Mod`. Áp dụng cách này cho các chữ số còn lại].



Hình 6.27 Giao diện ứng dụng **Digit Extractor**.

a) ***Tạo ứng dụng.*** Tạo một project mới có tên là `DigitExtractor`. Đổi tên file `Form1.vb` thành `DigitExtractor.vb`. Đổi tên `Form` thành `DigitExtractorForm`. Thêm các `Label`, một `TextBox` và một `Button` vào `Form` của ứng dụng. Đặt tên `TextBox` là `inputTextBox` và tên `Button` là `enterButton`. Đặt tên cho các điều khiển còn lại một cách lôgíc dựa trên cách đặt tên đã được hướng dẫn trong các chương trước.

b) ***Thêm xử lý sự kiện cho sự kiện `Click` của `enterButton`***. Trong chế độ **Design**, nhấn đúp vào `enterButton` để tạo xử lý sự kiện `enterButton_Click`. Trong phần xử lý sự kiện này, tạo năm biến kiểu `Integer`. Sử dụng phép chia số nguyên và toán tử `Mod` để trích lấy chữ số. Lưu các chữ số đó trong năm biến đã được tạo.

c) ***Thêm xử lý sự kiện cho sự kiện `TextChanged` của `inputTextBox`***. Trong chế độ **Design**, nhấn đúp vào `inputTextBox` để tạo xử lý sự kiện `inputTextBox_TextChanged`. Trong phần xử lý sự kiện này, xóa nội dung trên năm `Label` được dùng để hiển thị các chữ số. Xử lý sự kiện này xóa hết kết quả đầu ra mỗi khi một giá trị mới được nhập vào.

d) ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng. Nhập vào một số có năm chữ số và nhấn vào `Button` **Enter**. Nhập vào một số có năm chữ số mới và kiểm tra xem kết quả hiển thị trước đó có bị xóa không.

e) ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

f) ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

CHƯƠNG 7

Ứng dụng Wage Calculator

Giới thiệu về giải thuật, mã giả và điều khiển chương trình

Mục tiêu

Trong chương này, bạn sẽ tìm hiểu để:

- Hiểu cơ bản về kỹ thuật giải quyết vấn đề.
- Hiểu về cấu trúc điều khiển.
- Hiểu về mã giả và biết cách viết mã giả.
- Sử dụng câu lệnh lựa chọn **If...Then** và **If... Then...Else** để chọn lựa giữa các hành động khác nhau.
- Sử dụng toán tử gán.
- Sử dụng cửa sổ **Watch** của trình gỡ lỗi.

Nội dung chính

Trước khi viết ứng dụng, việc hiểu biết kỹ lưỡng về vấn đề cần giải quyết là vô cùng quan trọng. Điều này sẽ giúp bạn tạo ra kế hoạch tiếp cận giải quyết vấn đề hợp lý. Khi viết một ứng dụng, nhận diện được loại khối lệnh sẵn có và sử dụng các nguyên lý xây dựng ứng dụng đã được chứng minh cũng rất quan trọng. Trong chương này, bạn sẽ học về lý thuyết và nguyên lý của **lập trình cấu trúc (structured programming)**. Lập trình cấu trúc là kỹ thuật tổ chức điều khiển chương trình giúp bạn phát triển ứng dụng sáng sủa và dễ dàng hơn trong việc gỡ lỗi và chỉnh sửa. Kỹ thuật này thích hợp với hầu hết các ngôn ngữ lập trình bậc cao, bao gồm cả ngôn ngữ lập trình Visual Basic.

7.1 Chạy thử ứng dụng Wage Calculator

Trong phần này, chúng ta sẽ xem trước ứng dụng **Wage Calculator**. Ứng dụng này phải đáp ứng những yêu cầu sau:

Yêu cầu đối với ứng dụng

Một công ty trả lương tính tổng thu nhập trước thuế mỗi tuần cho nhân viên. Tổng thu nhập hàng tuần của nhân viên được tính dựa trên số giờ làm việc và tiền thù lao mỗi giờ. Cần tạo ra ứng dụng với đầu vào là các thông tin này và tính tổng thu nhập (chưa chịu thuế) hàng tuần cho mỗi nhân viên. Trong ứng dụng này, giả sử rằng số giờ làm việc tiêu chuẩn trong một tuần là 40. Lương tính cho 40 giờ làm hoặc ít hơn được tính bằng cách nhân thù lao mỗi giờ với số giờ làm việc trong mỗi tuần. Khi số giờ làm việc trong tuần vượt quá 40 giờ thì số giờ vượt quá này được xem là thời gian làm thêm giờ và sẽ có thù lao gấp rưỡi so với giờ tiêu chuẩn. Thù lao cho giờ làm thêm đó được tính bằng cách nhân tiền thù lao mỗi giờ với 1,5 và nhân kết quả này với số giờ làm thêm. Tổng số tiền làm thêm giờ cộng với tổng số tiền làm trong 40 giờ tiêu chuẩn là tổng thu nhập trong tuần của nhân viên.

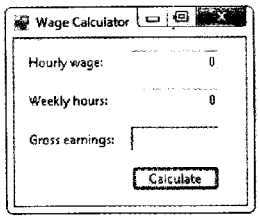
Ứng dụng này tính tổng thu nhập dựa trên tiền thù lao mỗi giờ và số giờ làm việc trong tuần. Thông thường, một nhân viên làm việc 40 giờ một tuần hoặc ít hơn sẽ được trả lương tiêu chuẩn. Nếu nhân viên làm việc nhiều hơn 40 giờ tiêu chuẩn thì thù lao sẽ được tính theo công thức khác. Trong chương này, chúng ta sẽ làm quen với một cấu trúc lập trình hay còn được gọi là cấu trúc điều khiển cho phép bạn thực hiện những tính toán khác nhau dựa trên các thông số khác nhau do người dùng nhập vào. Bạn sẽ bắt đầu bằng việc chạy thử ứng dụng đã hoàn chỉnh. Sau đó, bạn tìm hiểu những tính năng bổ sung của Visual Basic cần thiết để xây dựng cho mình một phiên bản của ứng dụng này.

Chạy thử ứng dụng Wage Calculatev

Mẹo thiết kế giao diện

Khi sử dụng nhiều TextBox theo chiều dọc, hãy căn thẳng hàng cạnh bên phải các TextBox, và nên tạo ra các TextBox có cùng kích thước. Căn lề trái cho các Label mô tả cho TextBox.

1. ***Mở ứng dụng đã hoàn chỉnh.*** Mở thư mục `C:\Examples\Tutorial07\CompletedApplication\WageCalculator` để tìm ứng dụng **Wage Calculator**. Nhấn đúp vào file `WageCalculator.sln` để mở ứng dụng trong IDE của Visual Basic.
2. ***Chạy ứng dụng.*** Chọn **Debug> Start Debugging** để chạy ứng dụng (xem Hình 7.1). Để bố trí giao diện, chúng ta sẽ căn lề bên phải các TextBox và làm cho các TextBox cùng kích thước. Chúng ta cũng căn lề trái các Label mô tả cho các TextBox.

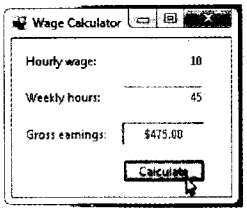


Chú thích hình

- ***Hourly wage:*** *Tiền thù lao mỗi giờ*
- ***Weekly hours:*** *Số giờ làm việc mỗi tuần*
- ***Gross earnings:*** *Tổng thu nhập*

Hình 7.1 Ứng dụng **Wage Calculator**.

3. ***Nhập tiền thù lao mỗi giờ của nhân viên.*** Nhập 10 vào TextBox **Hourly Wage:**.
4. ***Nhập vào tổng số giờ làm việc của nhân viên.*** Nhập 45 vào TextBox **Weekly hours:**.
5. ***Tính tổng thu nhập trước thuế của nhân viên.*** Nhấn vào Button **Calculate**. Kết quả hiển thị trên TextBox **Gross earnings:** là **$475.00** (Hình 7.2). Chú ý rằng tổng thu nhập của nhân viên bằng tổng tiền lương tính theo số giờ làm việc tiêu chuẩn (40*10) và tiền thù lao làm thêm ngoài giờ (5*10*1.5).



Hình 7.2 Tính thu nhập bằng cách nhấn vào Button **Calculate**.

6. ***Đóng ứng dụng.*** Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.
7. ***Đóng IDE.*** Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

7.2 Giải thuật

Các vấn đề tính toán có thể được giải quyết bằng cách thực hiện một loạt các hành động theo một thứ tự cụ thể. Thủ tục để giải quyết một vấn đề bao gồm:

1. các hành động được thực hiện và
2. thứ tự thực hiện các hành động

được gọi là một **giải thuật (algorithm)**. Ví dụ dưới đây minh họa tầm quan trọng của thứ tự thực hiện các hành động. Xét giải thuật "rise-and-shine" (là một thành ngữ chỉ việc thức dậy buổi sáng và chuẩn bị đi làm) được thực hiện bởi một giám đốc điều hành gồm các hành động : (1) ra khỏi giường, (2) cởi bỏ pyjama, (3) tắm, (4) mặc quần áo, (5) ăn sáng, (6) đi làm. Đây là những công việc chuẩn bị thường ngày trước khi đến cơ quan.

Tuy nhiên, giả sử rằng, các bước trên được thực hiện theo thứ tự hơi khác một chút: (1) ra khỏi giường, (2) cởi pyjama, (3) mặc quần áo, (4) tắm, (5) ăn sáng, (6) đi làm. Trong trường hợp này, anh ta sẽ đi làm trong bộ dạng ướt sũng.

Ví dụ này chứng minh rằng trình tự thực thi các hành động thích hợp là vấn đề cốt yếu của một chương trình máy tính. **Điều khiển chương trình (program control)** thực hiện nhiệm vụ sắp xếp đúng thứ tự các lệnh của ứng dụng. Trong chương này, bạn sẽ bắt đầu tìm hiểu về chức năng của điều khiển chương trình trong ngôn ngữ Visual Basic.

TỰ ÔN TẬP

1. _____thực hiện nhiệm vụ sắp xếp đúng thứ tự các lệnh của ứng dụng.
   a) Hành động
   b) Điều khiển chương trình
   c) Cấu trúc điều khiển
   d) Lập trình trực quan
2. Một_____là một kế hoạch nhằm giải quyết vấn đề bao gồm các hành động được thực thi và thứ tự thực thi của các hành động này.
   a) sơ đồ
   b) cấu trúc điều khiển
   c) giải thuật
   d) danh sách có thứ tự

Đáp án: 1) b. 2) c.

7.3 Mã giả

Mã giả là ngôn ngữ không chính quy giúp bạn trình bày các giải thuật. Mã giả rất hữu ích khi phát triển những giải thuật mà sẽ được chuyển thành các đoạn mã có cấu trúc trong ứng dụng Visual Basic. Mã giả giống ngôn ngữ hàng ngày - nó thuận tiện và gần gũi với người dùng, nhưng không phải là một ngôn ngữ lập trình máy tính thật sự.

Mẹo thiết kế giao diện

Mã giả giúp bạn khái niệm hóa ứng dụng trong quá trình phát triển ứng dụng. Sau này, lệnh mã giả có thể được chuyển thành mã Visual Basic.

Lệnh mã giả không được máy tính thực thi, nhưng mã giả giúp bạn xem xét ứng dụng cẩn thận trước khi cố gắng viết bằng ngôn ngữ lập trình, như Visual Basic. Trong chương này, bạn sẽ tìm hiểu một vài ví dụ về mã giả.

Mã giả chỉ chứa các ký tự, vì thế bạn có thể tạo và chỉnh sửa mã giả bằng chương trình soạn thảo văn bản như là trình soạn thảo mã Visual Basic hay Notepad. Chương trình mã giả được chuẩn bị cẩn thận có thể dễ dàng chuyển thành ứng dụng Visual Basic tương ứng. Nhiều chuyển đổi chỉ đơn giản bằng cách thay thế các lệnh mã giả bằng các lệnh Visual Basic tương đương. Hãy xem ví dụ về một lệnh mã giả:

Gán 0 cho biến đếm counter

Lệnh mã giả này diễn tả công việc rất dễ hiểu. Bạn có thể sử dụng một vài lệnh cùng nhau tạo nên một giải thuật nhằm đáp ứng yêu cầu của ứng dụng. Khi bạn hoàn thành mã giả cho giải thuật, bạn có thể chuyển các lệnh mã giả này thành các lệnh Visual Basic tương đương. Lệnh mã giả ở trên có thể chuyển thành lệnh Visual Basic sau:

```
counter = 0
```

Thông thường mã giả chỉ mô tả những **lệnh thực thi** - những hành động sẽ được thực hiện khi ứng dụng Visual Basic tương ứng chạy. Ví dụ về lệnh lập trình mà không được thực thi là lời khai báo.

```
Dim number As Integer
```

Khai báo này cung cấp thông tin về kiểu dữ liệu của biến `number` cho trình biên dịch và hướng dẫn trình biên dịch cấp không gian bộ nhớ cho biến này. Khai báo này không tạo ra bất cứ hành động nào khi thực thi ứng dụng như thao tác nhập, xuất hay tính toán. Bởi vậy chúng ta sẽ không mô tả thông tin này trong mã giả.

TỰ ÔN TẬP

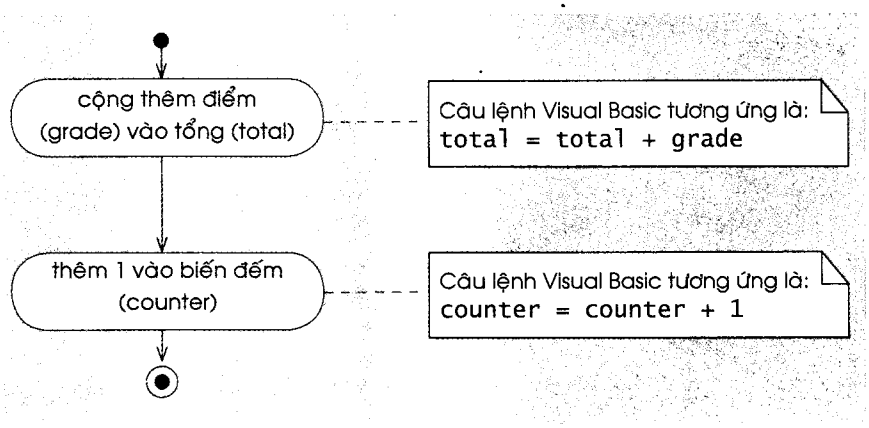
1. _____là một ngôn ngữ không chính quy giúp bạn phát triển giải thuật.
   a) Mã giả
   b) VB-Speak
   c) Chú thích
   d) Các lựa chọn trên đều sai
2. Mã giả_____.
   a) thường chỉ mô tả các khai báo
   b) được thực thi trên máy tính
   c) thường chỉ mô tả các dòng mã thực thi
   d) thường mô tả các khai báo và các dòng mã thực thi

Đáp án: **1)** a. **2)** c.

7.4 Cấu trúc điều khiển

Thông thường, các lệnh trong ứng dụng được thực thi theo thứ tự được viết. Đây gọi là **thực thi tuần tự (sequential execution)**. Ngôn ngữ Visual Basic cho phép bạn thay đổi thứ tự thực thi các câu lệnh. **Chuyển điều khiển (transfer of control)** xảy ra khi một câu lệnh không được thực thi ngay sau khi câu lệnh trước nó (theo thứ tự được viết ra trong ứng dụng) được thực thi. Điều này xảy ra khá phổ biến trong các chương trình máy tính.

Tất cả các chương trình máy tính có thể được viết chỉ sử dụng ba cấu trúc điều khiển: cấu trúc tuần tự, cấu trúc lựa chọn, cấu trúc lặp. **Cấu trúc tuần tự (sequence structure)** là mặc định trong ngôn ngữ Visual Basic, máy tính sẽ thực thi tuần tự các câu lệnh Visual Basic - nghĩa là một lệnh sẽ được thực hiện ngay sau một lệnh khác theo thứ tự xuất hiện trong ứng dụng. **Biểu đồ hoạt động (activity diagram)** trong Hình 7.3 minh họa cấu trúc tuần tự điển hình, trong đó hai phép tính được thực hiện theo thứ tự. Chúng ta sẽ thảo luận chi tiết về biểu đồ hoạt động theo Hình 7.3.



Hình 7.3 Biểu đồ hoạt động minh họa cấu trúc tuần tự.

Biểu đồ hoạt động là một thành phần của **UML (Unified Modeling Language - Ngôn ngữ mô hình hợp nhất)** - một tiêu chuẩn công nghiệp dùng để mô hình hóa hệ thống phần mềm. Biểu đồ hoạt động mô hình hóa các hành động, hay còn gọi là **luồng công việc (workflow)** của một phần trong hệ thống phần mềm. Các hoạt động này có thể chỉ là một phần tạo nên giải thuật, như cấu trúc tuần tự được minh họa trong Hình 7.3. Biểu đồ hoạt động được tạo thành từ các ký hiệu có mục đích đặc biệt, ví dụ như **ký hiệu trạng thái hành động (action-state symbol)** - một hình chữ nhật với cạnh bên trái và bên phải được thay bằng đường vòng cung, **ký hiệu hình thoi (diamond symbol)** và **ký hiệu vòng tròn nhỏ (small circle symbol)**. Các ký hiệu này được nối với nhau bởi **mũi tên chuyển tiếp (transition arrow)** thể hiện luồng hoạt động. Hình 7.3 không có ký hiệu hình thoi nào, ký hiệu này được sử dụng trong biểu đồ hoạt động sau.

Tương tự mã giả, biểu đồ hoạt động giúp bạn phát triển và trình bày các giải thuật, mặc dù nhiều lập trình viên thích sử dụng mã giả hơn. Biểu đồ hoạt động trình bày rõ ràng cách cấu trúc điều khiển thực hiện công việc như thế nào.

Xem xét biểu đồ hoạt động minh họa cấu trúc tuần tự trong Hình 7.3. Biểu đồ hoạt động này chứa hai ký hiệu **trạng thái hành động** (**action state**) biểu diễn các hành động được thực hiện. Mỗi một trạng thái hành động có một **biểu thức hành động** (**action expression**) chỉ rõ hành động được thực hiện - ví dụ "cộng thêm điểm vào tổng" hay "cộng thêm 1 vào biến đếm". Các hành động khác có thể là các thao tác tính toán hay nhập/xuất dữ liệu. Mũi tên trong biểu đồ hoạt động được gọi là mũi tên chuyển tiếp. Các mũi tên này thể hiện **sự chuyển tiếp** (**transition**), chỉ ra thứ tự xảy ra các hành động được biểu diễn bằng các ký hiệu trạng thái hành động. Ứng dụng thực thi các hành động được minh họa bởi biểu đồ hoạt động trong Hình 7.3 đầu tiên cộng thêm giá trị `grade` vào `total`, sau đó cộng thêm `1` vào `counter`.

Hình tròn đặc (**solid circle**) ở trên đỉnh biểu đồ hoạt động biểu diễn **trạng thái khởi đầu** (**initial state**) của hoạt động - điểm bắt đầu của luồng công việc trước khi ứng dụng thực hiện các hoạt động được mô hình hóa. Một hình tròn đặc được bao quanh bởi một vòng tròn rỗng xuất hiện ở cuối biểu đồ hoạt động biểu diễn **trạng thái kết thúc** (**final state**) - điểm kết thúc của luồng công việc sau khi ứng dụng thực hiện xong các hoạt động.

Chú ý, trên Hình 7.3, hình chữ nhật có nếp gấp ở góc trên bên phải, giống như trang giấy, được gọi là **ghi chú** (**note**) trong UML. Các ghi chú này tương tự dòng chú thích trong ứng dụng Visual Basic. Chúng là các lời giải thích, mô tả mục đích của các ký hiệu trên sơ đồ. Trong Hình 7.3, các chú thích UML được sử dụng để chỉ ra đoạn mã Visual Basic tương ứng với trạng thái hành động trong biểu đồ hoạt động. Một **đường nét đứt** (**dotted line**) liên kết giữa ghi chú với đối tượng mà nó mô tả. Thông thường, trong biểu đồ hoạt động không chỉ ra đoạn mã Visual Basic tương ứng, nhưng trong sơ đồ trên chúng tôi đã sử dụng ghi chú nhằm giúp bạn hiểu mối liên kết giữa sơ đồ và mã Visual Basic.

Cấu trúc lựa chọn

Ngôn ngữ Visual Basic cung cấp ba loại **cấu trúc lựa chọn** (**selection structure**). Chúng ta sẽ thảo luận về ba loại cấu trúc này trong chương này và Chương 12. Cấu trúc lựa chọn **`If...Then`** thực hiện một hành động (hay một chuỗi hành động) dựa trên một điều kiện nào đó. **Điều kiện** (**condition**) là một biểu thức có giá trị **đúng** (**true**) hoặc **sai** (**false**) được dùng để đưa ra quyết định. Các điều kiện sẽ **được đánh giá** (**evaluated**) để xác định xem giá trị của chúng là đúng hay sai. Các giá trị này thuộc kiểu dữ liệu **`Boolean`** và trong mã Visual Basic được biểu diễn bằng các từ khóa **`True`** hay **`False`**. Đôi khi, chúng ta gọi một điều kiện là một biểu thức `Boolean`.

Nếu điều kiện có giá trị là `True`, các hành động nằm trong cấu trúc `If... Then` được thực thi. Nếu giá trị này là `False`, các hành động này sẽ bị bỏ qua. Cấu trúc lựa chọn **`If...Then...Else`** thực hiện một hành động (hay một chuỗi các hành động) nếu điều kiện là đúng và sẽ thực hiện một hành động (hay một chuỗi các hành động) khác nếu điều kiện sai. Cấu trúc `Select Case` sẽ được thảo luận trong Chương 12 để thực hiện một trong nhiều hành động tùy thuộc vào giá trị của biểu thức điều kiện.

Cấu trúc `If...Then` được gọi là **cấu trúc lựa chọn đơn** (**single-selection structure**) bởi vì nó lựa chọn hay bỏ qua một hành động (hay một chuỗi hành động). Cấu trúc `If...Then...Else` được gọi là **cấu trúc lựa chọn kép** (**double-selection structure**) bởi vì nó lựa chọn giữa hai hành động (hay hai chuỗi hành động) khác nhau. Cấu trúc `Select Case` được gọi là **cấu trúc đa lựa chọn** (**multiple-selection structure**) bởi vì nó lựa chọn giữa nhiều hành động hay nhiều chuỗi hành động khác nhau.

Cấu trúc lặp

Visual Basic cung cấp bảy loại **cấu trúc lặp** (**repetition structure**) để thực hiện lặp lại một lệnh hay một nhóm các lệnh. Dưới đây liệt kê các cấu trúc lặp cùng với các chương tương ứng giới thiệu về chúng:

- `While...End While`[1]
- `Do While...Loop` (Chương 9)
- `Do Until...Loop` (Chương 9)
- `Do...Loop While` (Chương 10)
- `Do...Loop Until` (Chương 10)
- `For...Next` (Chương 11)
- `For Each...Next`

Từ khóa

Các từ `If`, `Then`, `Else`, `End`, `Select`, `Case`, `While`, `Do`, `Until`, `Loop`, `For`, `Next` và `Each` là các từ khóa của Visual Basic - phần Phụ lục B chứa danh sách toàn bộ các từ khóa của Visual Basic. Chúng ta sẽ thảo luận về các từ khóa của Visual Basic và mục đích sử dụng của các từ khóa xuyên suốt cuốn sách này. Tập hợp các từ khóa trong Visual Basic nhiều hơn hầu hết các ngôn ngữ lập trình thông dụng khác.

Những điểm cần lưu ý về cấu trúc điều khiển

Ngôn ngữ Visual Basic có 11 cấu trúc điều khiển gồm cấu trúc tuần tự, ba loại cấu trúc lựa chọn và bảy loại cấu trúc lặp. Tất cả các ứng dụng Visual Basic được tạo thành bởi sự kết hợp của các loại cấu trúc điều khiển này. Giống như cấu trúc tuần tự trong Hình 7.3, mỗi cấu trúc điều khiển có hai ký hiệu hình tròn nhỏ - một hình tròn đặc màu đen biểu diễn đầu vào của cấu trúc điều khiển và một hình tròn đặc màu đen được bao quanh bởi một hình tròn rỗng để biểu diễn đầu ra.

Tất cả cấu trúc điều khiển trong ngôn ngữ Visual Basic là **cấu trúc điều khiển một đầu vào/ một đầu ra** (**single-entry/single-exit control structure**) - mỗi cấu trúc điều khiển chỉ có một đầu vào và một đầu ra. Các cấu trúc điều khiển này giúp cho việc xây dựng ứng dụng trở nên dễ dàng. Có thể liên kết cấu trúc điều khiển này với cấu trúc điều khiển khác bằng cách kết nối đầu ra của cấu trúc điều khiển này với đầu vào của cấu trúc điều khiển còn lại. Điều này tương tự như việc xếp chồng các khối lên nhau trong xây dựng, do đó được gọi là **xếp chồng cấu trúc điều khiển** (**control-structure stacking**). Một cách khác để kết nối các cấu trúc điều khiển là **lồng cấu trúc điều khiển** (**control-structure nesting**) - một cấu trúc điều khiển được đặt bên trong một cấu trúc khác. Các giải thuật trong ứng dụng Visual Basic đều được tạo thành từ 11 loại cấu trúc điều khiển khác nhau bằng hai cách trên - đây là một mô hình đơn giản. Các cấu trúc điều khiển trong Visual Basic được thực thi như là các lệnh, bởi vậy từ đây trở về sau (sau các bài tập dưới đây), chúng tôi sử dụng thuật ngữ "lệnh" thay cho thuật ngữ "cấu trúc".

TỰ ÔN TẬP

1. Tất cả các ứng dụng Visual Basic có thể được viết từ_____loại cấu trúc điều khiển.
   a) một b) hai
   c) ba d) bốn
2. Quá trình thực thi từng lệnh một theo thứ tự được viết ra trong ứng dụng được gọi là_____.
   a) chuyển điều khiển b) thực thi tuần tự
   c) luồng công việc d) Các lựa chọn trên đều sai

Đáp án: 1) c. 2) b.

1: Chúng tôi không đề cập đến vòng lặp `While...End While` trong cuốn sách này. Cấu trúc của vòng lặp này tương tự như vòng lặp `Do While...Loop`; chúng tôi chỉ cung cấp cho những lập trình viên đã quen với các phiên bản trước của Visual Basic.

7.5 Lệnh lựa chọn If...Then

Một lệnh lựa chọn thực hiện phép chọn giữa hai hành động trong ứng dụng. Ví dụ, giả sử rằng sinh viên có số điểm là 60 hoặc lớn hơn được xem là đỗ (điểm tối đa của bài thi là 100). Lệnh mã giả dưới đây

```
Nếu điểm của sinh viên lớn hơn hoặc bằng 60 thì
    Hiển thị "Passed"
```

xác định xem điều kiện "điểm của sinh viên lớn hơn hoặc bằng 60" là đúng hay sai. Nếu điều kiện là đúng thì dòng chữ "Passed" được hiển thị và câu lệnh mã giả tiếp theo theo thứ tự được "thực thi" (nhớ rằng mã giả không phải là một ngôn ngữ lập trình thực sự.) Nếu điều kiện là sai, lệnh hiển thị sẽ bị bỏ qua và lệnh mã giả tiếp theo sẽ được thực thi.

Lỗi lập trình thường gặp

Thiếu từ khóa Then trong câu lệnh If...Then là một lỗi cú pháp. Môi trường phát triển tích hợp sẽ giúp ngăn chặn lỗi này bằng cách tự động chèn từ khóa Then sau khi bạn viết điều kiện.

Lệnh mã giả *Nếu* trên có thể được viết bằng ngôn ngữ Visual Basic như sau

```
If studentGrade >= 60 Then
    DisplayLabel.Text = "Passed"
End If
```

Mã trong Visual Basic gần như tương ứng với mã giả, điều này càng chứng minh sự hữu ích của mã giả như là công cụ phát triển ứng dụng. Phần thân hay còn được gọi là **khối lệnh (block)** của câu lệnh `If...Then` hiển thị chuỗi `"Passed"` trên `Label`.Từ khóa `End If` kết thúc câu lệnh `If...Then`.

Thói quen lập trình tốt

Viết các lệnh Visual Basic bên trong câu lệnh If...Then lùi vào một chút giúp cho việc đọc mã dễ dàng hơn.

Chú ý đoạn mã trong lệnh `If...Then` được viết lùi vào. Việc viết lùi vào như vậy sẽ làm cho mã dễ đọc hơn. Trình biên dịch Visual Basic bỏ qua những ký tự khoảng trắng, như là dấu cách, tab và dòng mới được sử dụng để viết mã lùi vào, trừ những ký tự khoảng trắng trong chuỗi.

Điều kiện nằm giữa các từ khóa `If` và `Then` xác định xem lệnh (hay chuỗi lệnh) nằm trong lệnh `If...Then` có được thực thi hay không. Nếu điều kiện là đúng, phần thân câu lệnh `If...Then` sẽ được thực thi. Nếu điều kiện là sai, phần thân câu lệnh sẽ không được thực thi. Điều kiện trong lệnh `If...Then` có thể được tạo thành bằng cách sử dụng **toán tử so sánh bằng (equality operator)** và **toán tử quan hệ (relational operator)** - hay còn được gọi là **toán tử so sánh (comparison operator)**. Hình 7.4 liệt kê các toán tử này. Toán tử quan hệ và toán tử so sánh bằng có cùng mức độ ưu tiên.

Lỗi lập trình thường gặp

Thêm các khoảng trống vào giữa các ký hiệu toán tử (ví dụ các toán tử <>, >=, <=, thì được viết thành < >, > =, < =) là một lỗi cú pháp. Lỗi này sẽ được Visual Basic tự động sửa.

Lỗi lập trình thường gặp

Đảo ngược các toán tử (ví dụ các toán tử <> , >= và <= thì được viết thành ><, =>, =<) là một lỗi cú pháp. Visual Basic sẽ tự động sửa lỗi này.

Toán tử quan hệ và toán tử so sánh bằng trong đại số	Toán tử quan hệ và toán tử so sánh bằng trong Visual Basic	Ví dụ về điều kiện trong Visual Basic	Ý nghĩa điều kiện trong Visual Basic
Các toán tử quan hệ			
>	>	x > y	x lớn hơn y
<	<	x < y	x nhỏ hơn y
≥	>=	x >= y	x lớn hơn hoặc bằng y
≤	<=	x <=y	x nhỏ hơn hoặc bằng y
Toán tử so sánh bằng			
=	=	x = y	x bằng y
≠	<>	x <> y	x khác y

Hình 7.4 Toán tử so sánh bằng và toán tử quan hệ.

Hình 7.5 minh họa cú pháp của lệnh `If...Then`. **Cú pháp (syntax)** của lệnh chỉ rõ cách viết lệnh để không xảy ra lỗi cú pháp khi biên dịch. Hãy nhìn kỹ hơn cú pháp lệnh `If...Then`. Dòng đầu tiên trên Hình 7.5 chỉ ra rằng lệnh này phải được bắt đầu với từ khóa `If` , theo sau nó là điều kiện và từ khóa `Then`. Chú ý rằng từ *điều kiện* được in nghiêng. Điều này chỉ ra rằng khi viết câu lệnh `If...Then`, bạn

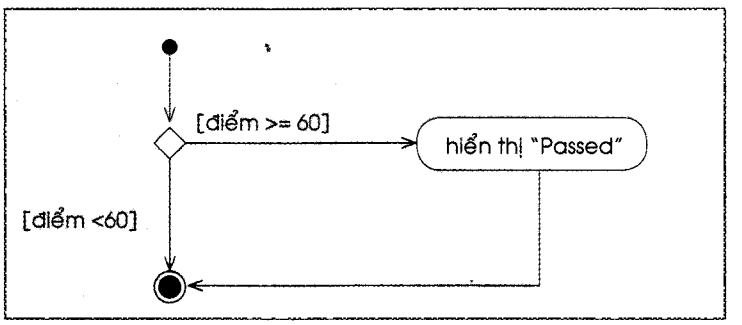
phải thay thế từ *điều kiện* bằng một điều kiện thực sự, đó chính là điều kiện mà bạn muốn đánh giá. Dòng thứ hai chỉ ra rằng dòng chữ *các câu lệnh* phải được thay bằng các lệnh thực sự nằm trong thân câu lệnh `If...Then`. Chú ý rằng chữ *các câu lệnh* được đặt trong dấu ngoặc vuông. Dấu ngoặc vuông này sẽ không xuất hiện trong câu lệnh `If...Then` thực sự mà chỉ ra rằng phần nào của lệnh là tùy chọn (có thể có hoặc không). Trong ví dụ này, dấu ngoặc vuông chỉ ra rằng tất cả các lệnh trong thân lệnh `If...Then` là tùy chọn. Tuy nhiên, nếu không có câu lệnh nào trong thân lệnh `If...Then` thì không hành động nào xảy ra, bất kể điều kiện có giá trị như thế nào. Dòng cuối cùng cho biết lệnh này kết thúc với từ khóa `End If`.

Cú pháp

```
If điều kiện Then
    [Các câu lệnh]
End If
```

Hình 7.5 Cú pháp lệnh `If...Then`.

Hình 7.6 minh họa lệnh lựa chọn đơn `If...Then`. Biểu đồ hoạt động này bao gồm ký hiệu quan trọng nhất trong biểu đồ hoạt động - ký hiệu hình thoi hay **ký hiệu ra quyết định (decision symbol)** chỉ ra hành động nào sẽ được thực hiện. Chú ý rằng, các cặp ngoặc vuông ở phía trên và bên cạnh mũi tên xuất phát từ ký hiệu ra quyết định được gọi là **điều kiện canh giữ (guard condition)**. Ký hiệu ra quyết định cho biết luồng công việc sẽ tiếp tục theo hướng nào dựa trên các điều kiện canh giữ liên kết với ký hiệu đó có giá trị là đúng hay sai. Mỗi mũi tên chuyển tiếp xuất phát từ một ký hiệu ra quyết định có một điều kiện canh giữ (được viết trong một ngoặc vuông ở phía trên hoặc bên cạnh mũi tên chuyển tiếp). Nếu điều kiện canh giữ có giá trị đúng thì luồng công việc sẽ đi đến trạng thái hành động mà mũi tên chuyển tiếp trỏ tới. Ví dụ như trên Hình 7.6, nếu điểm có giá trị lớn hơn hoặc bằng 60 thì ứng dụng hiển thị "Passed", sau đó sẽ chuyển sang trạng thái kết thúc của hoạt động này. Nếu điểm nhỏ hơn 60 thì ngay lập tức ứng dụng sẽ chuyển sang trạng kết thúc mà không hiển thị một thông điệp nào. Tại một thời điểm chỉ có duy nhất một điều kiện canh giữ liên kết với một ký hiệu ra quyết định có giá trị là đúng.



Hình 7.6 Biểu đồ hoạt động minh họa lệnh lựa chọn đơn `If...Then`.

Chú ý rằng lệnh `If...Then` là lệnh một đầu vào một đầu ra. Các biểu đồ hoạt động cho các cấu trúc điều khiển còn lại (ngoài ký hiệu hình tròn nhỏ và mũi tên chuyển tiếp) chỉ chứa các ký hiệu trạng thái-hành động (minh họa các hành động được thực thi) và ký hiệu hình thoi (minh họa quyết định được thực hiện). Việc biểu diễn các cấu trúc điều khiển theo cách này nhằm nhấn mạnh **mô hình lập trình hành động/quyết định**. Để hiểu thêm về quy trình lập trình cấu trúc, chúng ta tưởng tượng có 11 cái thùng, mỗi cái chứa một trong 11 loại cấu trúc điều khiển. Các cấu trúc điều khiển trong mỗi thùng là trống, nghĩa là không có gì được ghi trong ký hiệu trạng thái-hành động và không có điều kiện canh giữ nào được ghi bên cạnh ký hiệu ra quyết định. Nhiệm vụ của bạn là xây dựng ứng dụng, sử dụng các cấu trúc điều khiển tùy theo yêu cầu giải thuật, kết hợp các cấu trúc điều khiển này chỉ bằng hai cách (xếp chồng hoặc lồng cấu trúc) và điền vào các hành động

và các điều kiện canh giữ thích hợp. Mỗi cấu trúc điều khiển này được cài đặt sử dụng lệnh Visual Basic.

TỰ ÔN TẬP

1. Lệnh `If...Then` nào hiển thị đúng kết quả sau: Sinh viên sẽ nhận điểm A trong kỳ thi nếu điểm thi lớn hơn hoặc bằng 90.

   a)
   ```
   If studentGrade <> 90 Then
      displayLabel.Text = "Student received an A"
   End If
   ```

   b)
   ```
   If studentGrade > 90 Then
      displayLabel.Text = "Student received an A"
   End If
   ```

   c)
   ```
   If studentGrade = 90 Then
      displayLabel.Text = "Student received an A"
   End If
   ```

   d)
   ```
   If studentGrade >= 90 Then
      displayLabel.Text = "Student received an A"
   End If
   ```

2. Ký hiệu_____không phải là một toán tử Visual Basic.

   a) * b) ^

   c) % d) <>

Đáp án: **1)** d. **2)** c.

7.6 Lệnh lựa chọn `If...Then...Else` và các biểu thức điều kiện `If`

Như bạn đã tìm hiểu, lệnh lựa chọn `If...Then` sẽ thực hiện một hành động (hay một chuỗi hành động) được chỉ định chỉ khi điều kiện có giá trị là `True`; ngược lại hành động (hay chuỗi hành động) này sẽ bị bỏ qua. Câu lệnh lựa chọn `If...Then... Else` cho phép bạn chỉ ra một hành động (hay một chuỗi hành động) khác sẽ được thực hiện khi điều kiện có giá trị là `False`. Ví dụ như lệnh mã giả dưới đây

```
Nếu điểm của sinh viên lớn hơn hoặc bằng 60 thì
    Hiển thị "Passed"
Trái lại
    Hiển thị "Failed"
```

hiển thị "Passed" nếu điểm có giá trị lớn hơn hoặc bằng 60 hoặc hiển thị "Failed" nếu điểm nhỏ hơn 60. Trong cả hai trường hợp thì sau khi hiển thị thông tin, lệnh mã giả kế tiếp theo thứ tự sẽ được "thực thi". Lệnh mã giả trên được biểu diễn bằng đoạn mã Visual Basic sau

```
If studentGrade >=60 Then
   displayLabel.Text = "Passed"
Else
   displayLabel.Text = "Failed"
End If
```

Thói quen lập trình tốt

Lùi đầu dòng các lệnh nằm trong thân lệnh của lệnh `If... Then...Else` để làm cho mã dễ đọc hơn. (Lưu ý: IDE Visual Basic tự động lùi đầu dòng).

Chú ý rằng thân mệnh đề **`Else`** được viết lùi vào thẳng hàng với thân của mệnh đề `If`. Quy ước lùi đầu dòng nên được áp dụng thống nhất xuyên suốt trong ứng dụng của bạn. Thật là khó để đọc một chương trình mà không sử dụng quy ước đồng nhất về khoảng cách lùi đầu dòng. IDE sẽ giúp bạn duy trì tính thống nhất của việc lùi đầu dòng với tính năng "lùi đầu dòng thông minh". Chức năng này được mặc định trong ngôn ngữ Visual Basic. Lệnh lựa chọn `If...Then...Else` tuân theo cú pháp giống như lệnh lựa chọn `If...Then`. Lệnh này được kết thúc bằng từ khóa `End If`, như Hình 7.7.

Thói quen lập trình tốt

Áp dụng quy ước lùi đầu dòng thống nhất trong toàn ứng dụng để mã nguồn dễ đọc hơn. Tính năng "lùi đầu dòng thông minh" sẽ giúp bạn làm việc này.

Cú pháp

```
If điều kiện Then
    [Các câu lệnh]
Else
    [Các câu lệnh]
End If
```

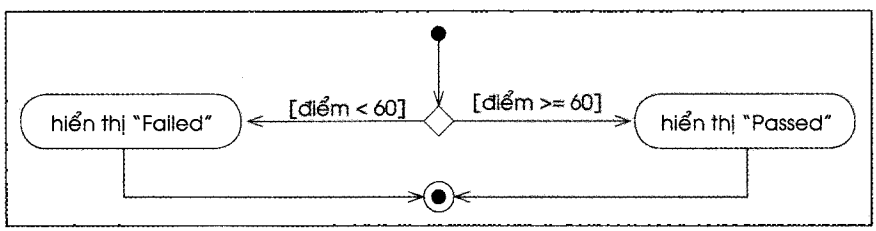
Hình 7.7 Cú pháp lệnh `If...Then...Else`.

Lệnh `If...Then...Else` trên có thể được viết bằng **biểu thức điều kiện `If`** (**conditional `If` expression**) như sau:

```
displayLabel.Text = If (studentGrade >= 60, "Passed", "Failed")
```

Một biểu thức điều kiện `If` bắt đầu bằng từ khóa `If` và theo sau bởi ba biểu thức đặt trong dấu ngoặc đơn - đó là điều kiện, giá trị của biểu thức điều kiện nếu điều kiện là đúng, giá trị của biểu thức điều kiện nếu điều kiện là sai.

Hình 7.8 minh họa luồng điều khiển trong cấu trúc lựa chọn kép `If...Then...Else`. Lại một lần nữa, trừ trạng thái ban đầu, mũi tên chuyển tiếp và trạng thái kết thúc, trong biểu đồ hoạt động chỉ chứa ký hiệu biểu diễn các trạng thái hành động và các quyết định. Trong ví dụ này, điểm có các giá trị nhỏ hơn 60 hoặc lớn hơn 60 hoặc bằng 60. Nếu điểm nhỏ hơn 60 ứng dụng sẽ hiển thị `"Failed"`. Nếu điểm lớn hơn hoặc bằng 60 ứng dụng sẽ hiển thị `"Passed"`. Chúng ta tiếp tục nhấn mạnh mô hình tính toán hành động/quyết định. Hãy tưởng tượng rằng có một chiếc thùng sâu chứa tất cả các lệnh lựa chọn kép rỗng cần để xây dựng ứng dụng Visual Basic. Là một lập trình viên, công việc của bạn là ghép các lệnh lựa chọn này với các lệnh điều khiển khác tùy theo yêu cầu giải thuật (bằng cách xếp chồng hay lồng cấu trúc). Bạn sẽ điền vào các trạng thái hành động và các ký hiệu quyết định với các biểu thức hành động và điều kiện canh giữ phù hợp với giải thuật.



Hình 7.8 Biểu đồ hoạt động minh họa lệnh lựa chọn kép `If...Then...Else`.

Lệnh `If...Then...Else` lồng (Nested `If...Then...Else` statement) kiểm tra đa điều kiện bằng cách đặt lệnh `If...Then...Else` này bên trong lệnh `If...Then...Else` khác. Ví dụ như đoạn mã giả sau đây hiển thị là "A" khi điểm thi lớn hơn hoặc bằng 90, "B" nếu điểm nằm trong khoảng từ 80-89, "C" nếu điểm nằm trong khoảng từ 70-79, "D" nếu điểm từ 60-69 và "F" với các trường hợp còn lại.

```
Nếu điểm của sinh viên lớn hơn hoặc bằng 90 thì
    Hiển thị "A"
Trái lại
    Nếu điểm của sinh viên lớn hơn hoặc bằng 80 thì
        Hiển thị "B"
    Trái lại
        Nếu điểm của sinh viên lớn hơn hoặc bằng 70 thì
            Hiển thị "C"
        Trái lại
            Nếu điểm của sinh viên lớn hơn hoặc bằng 60 thì
                Hiển thị "D"
            Trái lại
                Hiển thị "F"
```

Đoạn mã giả trên có thể được chuyển thành đoạn mã Visual Basic dưới đây:

Thói quen lập trình tốt

Nếu có nhiều mức lùi đầu dòng, mỗi mức nên lùi vào xa hơn về bên phải cùng một khoảng cách.

```
1  If studentGrade >= 90 Then
2     displayLabel.Text = "A"
3  Else
4     If studentGrade >= 80 Then
5        displayLabel.Text = "B"
6     Else
7        If studentGrade >= 70 Then
8           displayLabel.Text = "C"
9        Else
10          If studentGrade >= 60 Then
11             displayLabel.Text = "D"
12          Else
13             displayLabel.Text = "F"
14          End If
15       End If
16    End If
17 End If
```

Hình 7.9 Đoạn mã Visual Basic được chuyển từ đoạn mã giả.

Nếu `studentGrade` lớn hơn hoặc bằng 90, điều kiện đầu tiên được đánh giá có giá trị "`True`" và lệnh `display.Text = "A"` được thực thi. Với giá trị `studentGrade` lớn hơn hoặc bằng 90, ba điều kiện còn lại đạt giá trị "`True`". Tuy nhiên, những điều kiện này không bao giờ được đánh giá vì các điều kiện này được đặt trong mệnh đề `Else` của lệnh `If...Then...Else` ở ngoài cùng. Điều kiện đầu tiên đạt giá trị "`True`" do đó tất cả các lệnh trong mệnh đề `Else` bị bỏ qua. Bây giờ giả sử `studentGrade` có giá trị 75. Điều kiện đầu tiên đạt giá trị "`False`", do đó ứng dụng sẽ thực thi các lệnh trong mệnh đề `Else`. Mệnh đề `Else` này lại chứa một lệnh `If...Then...Else` khác có biểu thức điều kiện là `studentGrade >= 80`. Điều kiện này được đánh giá có giá trị `False`, do đó các lệnh nằm trong mệnh đề `Else` của lệnh `If...Then...Else` này được thực thi. Mệnh đề `Else` này lại chứa một lệnh `If...Then...Else` khác nữa, có biểu thức điều kiện là `studentGrade >= 70`. Điều kiện này có giá trị `True`, do đó lệnh `displayLabel.Text = "C"` được thực thi. Mệnh đề `Else` của lệnh `If...Then...Else` này bị bỏ qua.

Hầu hết các lập trình viên ngôn ngữ Visual Basic thích sử dụng **từ khóa `ElseIf`** để viết lệnh `If...Then...Else` trên, như Hình 7.10 dưới đây.

```
1  If studentGrade >= 90 Then
2     displayLabel.Text = "A"
3  ElseIf studentGrade >= 80 Then
4     displayLabel.Text = "B"
5  ElseIf studentGrade >= 70 Then
6     displayLabel.Text = "C"
7  ElseIf studentGrade >= 60 Then
8     displayLabel.Text = "D"
9  Else
10    displayLabel.Text = "F"
11 End If
```

Hình 7.10 Lệnh `If...Then...Else` sử dụng từ khóa `ElseIf`.

Lỗi lập trình thường gặp

Mệnh đề `Else` phải luôn là mệnh đề cuối cùng của lệnh `If...Then...Else` - nếu phía sau mệnh đề `Else` có một mệnh đề `Else` khác hoặc mệnh đề `ElseIf` là lỗi cú pháp.

Hai lệnh trên là tương đương, nhưng bạn nên sử dụng lệnh sau, vì nó giúp cho bạn tránh việc phải lùi đầu dòng vào sâu trong đoạn mã. Việc lùi đầu dòng vào sâu thường để lại nhiều khoảng trống nhỏ trên một dòng, làm cho đoạn mã trở nên khó đọc. Chú ý phần cuối của lệnh `If...Then...Else` sử dụng từ khóa `Else` để xử lý tất cả các trường hợp còn lại. Mệnh đề `Else` phải luôn là mệnh đề cuối cùng của lệnh `If...Then...Else` - nếu phía sau mệnh đề `Else` có một mệnh đề `Else` khác hoặc mệnh đề `ElseIf` là một lỗi cú pháp. Chú ý rằng lệnh sau cùng chỉ yêu cầu một từ khóa `End If`.

TỰ ÔN TẬP

1. Lệnh `If...Then...Else` là lệnh lựa chọn _____.
   a) đơn b) kép
   c) ba d) lồng
2. Đặt một lệnh `If...Then...Else` bên trong một lệnh `If...Then...Else` khác là một ví dụ về_____.
   a) lệnh `If...Then...Else` lồng nhau b) lệnh `If...Then...Else` xếp chồng
   c) lệnh `If...Then...Else` liên tiếp d) Các lựa chọn trên đều sai

Đáp án: 1) b. 2) a.

7.7 Xây dựng ứng dụng Wage Calculator

Trong phần này, chúng ta sẽ xây dựng ứng dụng **Wage Calculator** bằng cách sử dụng lệnh `If...Then...Else`. Lệnh `If...Then... Else` cho phép bạn lựa chọn giữa việc tính lương thông thường và thù lao làm thêm giờ dựa trên số giờ làm việc. Đoạn mã giả dưới đây mô tả các thao tác cơ bản của ứng dụng **Wage Calculator**.

```
Khi người dùng nhấn vào Button Calculate
    Lấy tổng số giờ làm việc và tiền thù lao mỗi giờ trên các TextBox
    Nếu tổng số giờ làm việc nhỏ hơn hoặc bằng 40 thì
        Tổng thu nhập (chưa chịu thuế) bằng số giờ làm việc nhân với
            tiền thù lao mỗi giờ
    Trái lại
        Tổng thu nhập (chưa chịu thuế) bằng 40 nhân với tiền thù lao mỗi
            giờ cộng với số giờ làm thêm nhân với gấp rưỡi tiền
            thù lao mỗi giờ
    Hiển thị giá trị tổng thu nhập (chưa chịu thuế).
```

Visual Studio cung cấp nhiều công cụ lập trình giúp bạn tạo ra những ứng dụng mạnh mẽ và hiệu quả. Với nhiều công cụ có sẵn như vậy thì việc tạo ra một bảng để tổ chức và lựa chọn các thành phần giao diện đồ họa (GUI) tốt nhất là rất cần thiết. Giống như mã giả, bảng này đơn giản hóa nhiệm vụ xây dựng ứng dụng bằng cách tóm tắt các hành động của ứng dụng. Ngoài việc liệt kê các hành động của ứng dụng, bảng này gán các điều khiển và sự kiện vào các hành động được mô tả trong mã giả.

Bây giờ chúng ta sẽ chạy thử ứng dụng **Wage Calculator** và tìm hiểu về biểu diễn mã giả của ứng dụng này. Bạn sẽ sử dụng bảng Hành động/Điều khiển/Sự kiện (Action/Control/Event - ACE) để chuyển đoạn mã giả này thành ngôn ngữ Visual Basic. Hình 7.1 liệt kê các hành động, điều khiển và sự kiện giúp hoàn thành phiên bản ứng dụng của bạn.

Các `Label` ở dòng đầu tiên hiển thị thông tin về ứng dụng tới người dùng. Các `Label` này hướng dẫn người dùng xuyên suốt ứng dụng. Điều khiển `Button` `calculateButton` được sử dụng để tính tổng thu nhập cho nhân viên. Chú ý rằng, cột thứ ba cho biết chúng ta sẽ sử dụng sự kiện `Click` để thực hiện thao tác tính toán. Các `TextBox` chứa dữ liệu do người dùng nhập vào. Điều khiển cuối cùng `earningResultLabel` là một `Label` hiển thị kết quả của ứng dụng.

Bảng ACE cho ứng dụng Wage Calculator

Hành động	Điều khiển	Sự kiện
Hiển thị Label cho các điều khiển của ứng dụng	`wageLabel`, `hoursLabel`, `earningsLabel`	Ứng dụng chạy
	`calculateButton`	`Click`
Lấy tổng số giờ làm việc và tiền thù lao mỗi giờ trên các TextBox	`wageTextBox`, `hoursTextBox`	
Nếu tổng số giờ làm việc nhỏ hơn hoặc bằng 40 thì Tổng thu nhập (chưa chịu thuế) bằng số giờ làm việc nhân với tiền thù lao mỗi giờ		
Trái lại Tổng thu nhập (chưa chịu thuế) bằng 40 nhân với tiền thù lao cộng với số giờ làm thêm nhân với gấp rưỡi tiền thù lao mỗi giờ		
Hiển thị giá trị tổng thu nhập (chưa chịu thuế) .	`earningsResultLabel`	

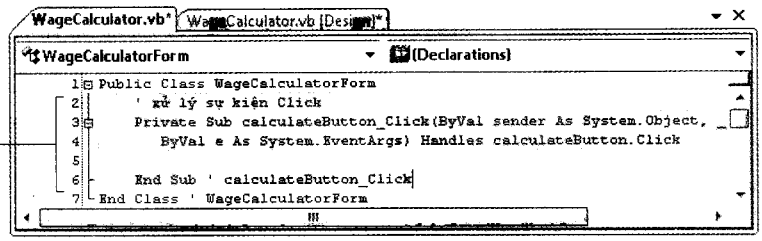
Hình 7.11 Bảng ACE cho ứng dụng **Wage Calculator**.

Bây giờ chúng ta sẽ sử dụng mã giả và bảng ACE để hoàn thành ứng dụng **Wage Calculator**. Phần dưới đây sẽ hướng dẫn bạn các bước thêm sự kiện `Click` cho `Button` **Calculate** và khai báo các biến cần cho việc tính tổng thu nhập của nhân viên. Nếu bạn quên không thêm mã vào sự kiện `Click`, ứng dụng sẽ không hồi đáp lại sự kiện khi người dùng nhấn vào `Button` **Calculate**.

Khai báo biến có trong phần xử lý sự kiện `Click` của `Button Calculate`

1. ***Copy template vào thư mục làm việc của bạn.*** Copy thư mục `C:\Examples\Tutorial07\TemplateApplication\WageCalculate` vào thư mục `C:\SimplyVB2008`.
2. ***Mở file template của ứng dụng Wage Calculator.*** Nhấn đúp vào file `WageCalculator.sln` trong thư mục `WageCalculation` để mở ứng dụng trong IDE Visual Basic. Nếu ứng dụng không mở ở chế độ **Design**, nhấn đúp vào file **WageCalculator.vb** trong **Solution Explorer**. Nếu **Solution Explorer** chưa được mở, hãy chọn **View > Solution Explorer**.
3. ***Thêm xử lý sự kiện `Click` cho `Button Calculate`.*** Trong ví dụ này, khi sự kiện `Click` của `Button` **Calculate** xảy ra, xử lý sự kiện sẽ tính thu nhập trước thuế. Nhấn đúp vào `Button` **Calculate**. Một xử lý sự kiện được tạo ra và IDE sẽ chuyển sang chế độ **Code**. Dòng 3-6 trên Hình 7.12 minh họa xử lý sự kiện vừa được tạo ra này. Hãy chắc chắn rằng bạn đã thêm các ghi chú và các ký tự nối dòng như trên Hình 7.12 để số dòng trong đoạn mã của bạn khớp với số dòng được hiển thị trong chương này.

Xử lý sự kiện rỗng

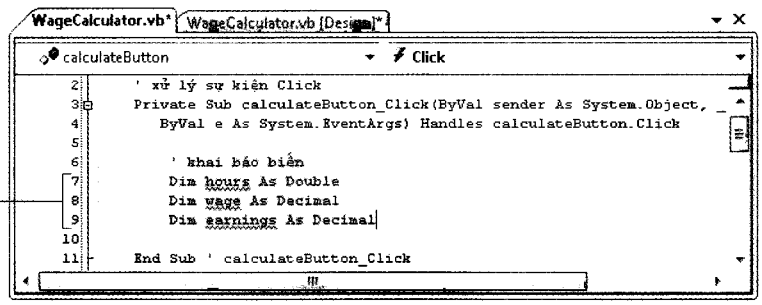


Hình 7.12 Xử lý sự kiện của `Button` **Calculate**.

Từ khóa `End Sub` (dòng 6) cho biết điểm kết thúc của xử lý sự kiện `CalculateButton_Click` . Từ khóa `End Class` (dòng 7) cho biết điểm kết thúc của lớp `WageCalculatorForm`. Chúng ta thường thêm các chú thích để người đọc có thể dễ dàng xác định xử lý sự kiện nào hoặc lớp nào kết thúc mà không cần phải tìm điểm bắt đầu của xử lý sự kiện hay của lớp đó.

4. ***Khai báo biến.*** Ứng dụng này sử dụng kiểu dữ liệu cơ sở `Double` và `Decimal`. Một biến `Double` chứa các chữ số và dấu chấm thập phân. Bởi vì số giờ làm việc và tiền lương thường là các số thập phân, do đó kiểu dữ liệu `Integer` không thích hợp cho ứng dụng này. Thêm các dòng 6-9 trên Hình 7.13 vào thân của xử lý sự kiện `CalculateButton_Click`. Dòng 7 chứa khai báo biến `hours` kiểu `Double`. Biến này chứa số giờ làm việc do người dùng nhập vào.

Khai báo biến



Hình 7.13 Khai báo các biến kiểu `Double` và `Decimal`.

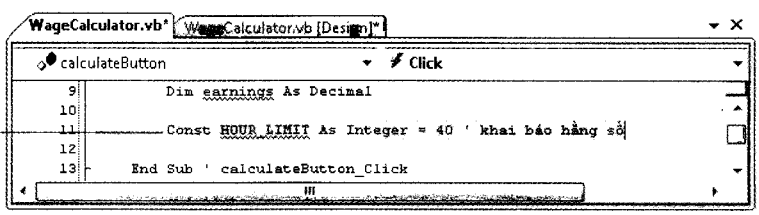
Kiểu **`Decimal`** được dùng để lưu số tiền vì kiểu dữ liệu này tối ưu hóa lỗi làm tròn của các phép tính toán học liên quan đến số tiền. Dòng 8-9 khai báo biến `wage` để lưu tiền thù lao mỗi giờ do người dùng nhập vào và biến `earnings` chứa tổng tiền lương trong tuần.

Thói quen lập trình tốt

Hằng làm cho chương trình dễ đọc hơn bởi hằng cung cấp tên gọi cho các giá trị không thay đổi.

5. ***Khai báo một hằng.*** Thêm dòng 11 trên Hình 7.14 vào cuối xử lý sự kiện `CalculateButton_Click`. Dòng 11 chứa một **hằng (constant)**, một định danh chứa giá trị không bị thay đổi sau khi khai báo biến được khởi tạo. Hằng được khai báo với từ khóa **`Const`**. Chúng ta sẽ gán số giờ làm việc tối đa mà chưa phải tính đến tiền làm thêm giờ cho hằng `HOUR_LIMIT` (giá trị này là 40). Chú ý rằng tên hằng được viết hoa để nhấn mạnh rằng đây là một hằng.

Khai báo hằng



Thói quen lập trình tốt

Viết hoa tất cả các chữ cái của tên hằng, tách các từ của tên hằng bằng dấu gạch dưới (_)

Hình 7.14 Tạo ra một hằng.

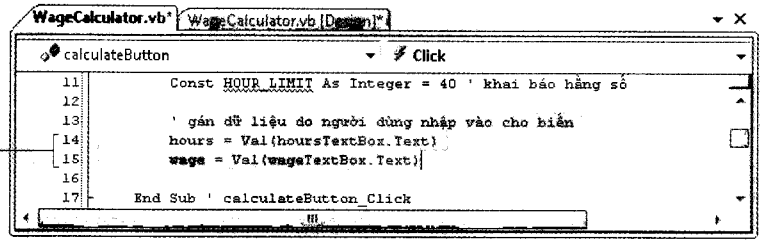
6. ***Lưu project.*** Chọn **File > Save All** để lưu mã đã được sửa đổi.

Bây giờ, bạn đã khai báo xong biến, bạn có thể sử dụng các biến này để lấy dữ liệu do người dùng nhập vào, sau đó sử dụng dữ liệu này để tính toán và hiển thị thu nhập của người dùng. Phần sau sẽ hướng dẫn bạn sử dụng lệnh `If...Then... Else` để xác định thu nhập của người dùng.

Tính tổng thu nhập cho người dùng

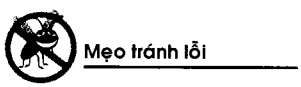
1. ***Lấy dữ liệu đầu vào từ các TextBox.*** Thêm các dòng 13-15 trên Hình 7.15 vào cuối xử lý sự kiện `CalculateButton_Click`. Dòng 14 -15 gán giá trị do người dùng nhập vào `TextBox` cho các biến `hours` và `wage`. Hàm `Val` trả lại giá trị do người dùng nhập vào kiểu `Double` (dòng 14-15). Visual Basic ngầm chuyển giá trị `Double` (kết quả hàm `Val`) sang `Decimal` rồi gán kết quả này cho biến `wage` (dòng 15).

Phép gán giá trị cho biến



Hình 7.15 Gán dữ liệu cho biến.

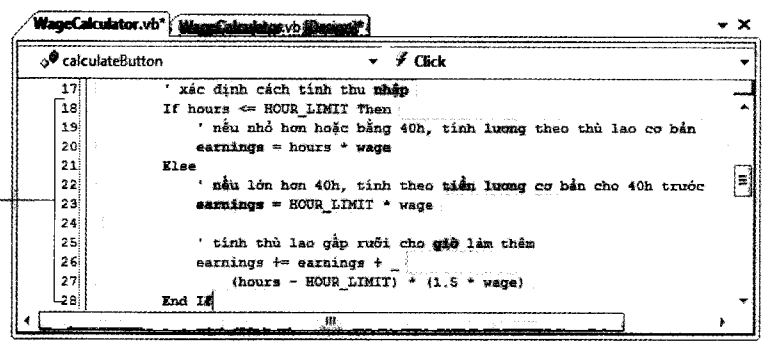
2. ***Tính tổng thu nhập dựa trên số giờ làm việc.*** Bắt đầu thêm lệnh `If... Then...Else` từ dòng 17-28 trên Hình 7.16 vào cuối xử lý sự kiện `CalculateButton_Click`. Sau khi nhập xong dòng 17-18, bấm phím *Enter*. Từ khóa `End If` sẽ được IDE tự động thêm vào. Tiếp theo, thêm lệnh `If...Then...Else` từ dòng 19-27. Bạn cần phải lùi đầu dòng trong khi viết câu lệnh. Lệnh `If...Then...Else` này sẽ xác định xem nhân viên có nhận được số tiền làm thêm giờ ngoài số tiền lương thông thường hay không. Dòng 18 kiểm tra xem giá trị chứa trong biến `hours` có nhỏ hơn hoặc bằng giá trị `HOUR_LIMIT` không. Nếu giá trị này nhỏ hơn hoặc bằng `HOUR_LIMIT`, dòng 20 gán kết quả phép nhân giữa giá trị biến `hours` và `wage` cho biến `earnings`. Khi bạn nhân một biến kiểu `Double` với một biến kiểu `Decimal`, Visual Basic sẽ ngầm chuyển biến `Decimal` thành `Double`. Kết quả kiểu `Double` này được chuyển ngầm thành kiểu `Decimal` khi gán kết quả này cho biến `earnings` kiểu `Decimal`.



Mẹo tránh lỗi

Để giảm lỗi, đôi khi IDE tự động thêm các từ khóa cho bạn. Ví dụ, thêm các từ khóa `End If` khi một lệnh `If...Then` hoặc một lệnh `If...Then...Else` được tạo ra. Điều này sẽ loại trừ khả năng các từ khóa này bị thiếu hoặc bị viết sai.

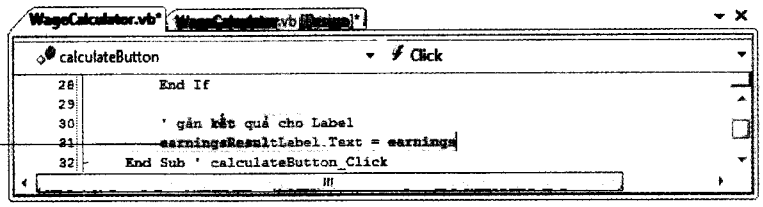
Lệnh `If...Then...Else`



Hình 7.16 Lệnh `If...Then...Else` thực hiện tính tổng thu nhập.

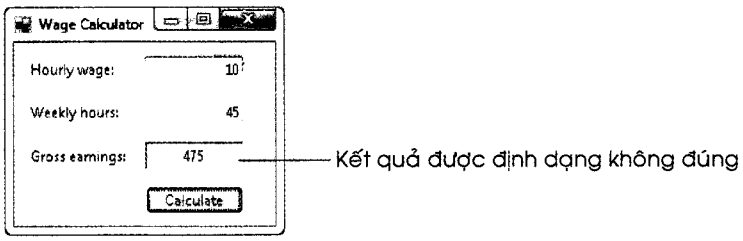
Trái lại, nếu giá trị biến `hours` không nhỏ hơn hoặc bằng `HOUR_LIMIT`, thì ứng dụng sẽ chuyển đến từ khóa `Else` ở dòng 21. Dòng 23 tính lương cho số giờ làm việc theo tiêu chuẩn (được thiết lập bởi hằng `HOUR_LIMIT`) và gán kết quả cho biến `earnings`. Dòng 26-27 xác định tổng số giờ vượt quá `HOUR_LIMIT` (bằng cách sử dụng biểu thức `hours - HOUR_LIMIT`), sau đó nhân với `1.5` lần tiền thù lao mỗi giờ. Kết quả phép tính này chính là thù lao làm thêm giờ của người dùng và kết quả này được cộng thêm với giá trị biến `earnings` rồi gán trở lại biến này.

3. ***Hiển thị kết quả***. Thêm dòng 30-31 trên Hình 7.17 vào cuối xử lý sự kiện `CalcualteButton_Click` . Dòng 31 gán giá trị biến `earnings` vào thuộc tính `Text` của `Label earningsResultLabel` và ngầm chuyển giá trị biến `earnings` này từ kiểu `Decimal` thành kiểu `String`.



Hình 7.17 Gán kết quả vào `Label earningsResultLabel`.

4. ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng. Chú ý rằng, kết quả không được định dạng đúng như kết quả của ứng dụng đã hoàn thiện (Hình 7.18). Bạn sẽ được học cách thêm chức năng này trong Phần 7.9.



Hình 7.18 Ứng dụng tính lương với định dạng kết quả không đúng.

5. ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

TỰ ÔN TẬP

1. Kiểu dữ liệu `Decimal` được dùng để chứa_____.
   a) chữ cái và chữ số
   b) số nguyên
   c) chuỗi
   d) số tiền
2. Hằng được khai báo với từ khóa_____.
   a) `Fixed`
   b) `Constant`
   c) `Final`
   d) `Const`

Đáp án: 1) d. 2) d.

7.8 Toán tử gán

Visual Basic cung cấp một số toán tử gán để rút gọn lệnh gán. Ví dụ, lệnh cộng thêm 3 vào biến `value`:

```
value = value + 3
```

có thể được rút gọn bằng toán tử gán cộng += như sau:

```
value += 3
```

Toán tử += cộng giá trị toán hạng bên phải vào giá trị toán hạng bên trái và gán kết quả cho toán hạng bên trái. Visual Basic cung cấp các toán tử gán tương ứng với các toán tử nhị phân gồm: +, -, *, ^, / và \. Khi toán tử gán được thực hiện, biểu thức phía bên phải toán tử luôn được tính toán đầu tiên, sau đó giá trị này được gán cho biến bên trái. Hình 7.19 chứa các toán tử gán, các biểu thức, ví dụ mẫu và chú thích.

Toán tử gán	Biểu thức ví dụ	Giải thích	Gán
Giả sử rằng c = 4			
+=	c+=7	c=c+7	Gán 11 cho c
-=	c-=3	c=c-3	Gán 1 cho c
*=	c*=4	c=c*4	Gán 16 cho c
/=	c/=2	c=c/2	Gán 2 cho c
\=	c\=3	c=c\3	Gán 1 cho c
^=	c^=2	c=c^2	Gán 16 cho c

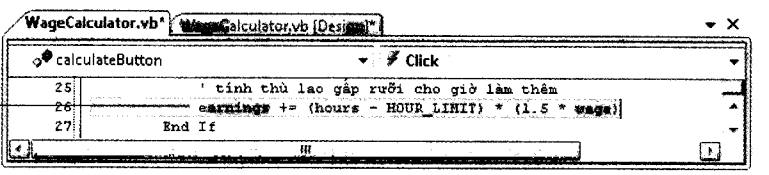
Hình 7.19 Các toán tử gán.

Phần dưới đây minh họa rút gọn công thức tính tiền lương làm thêm giờ bằng cách sử dụng toán tử +=. Chú ý rằng, khi chạy ứng dụng một lần nữa, kết quả sẽ giống với lần chạy trước. Điểm khác nhau duy nhất lệnh dài đã được làm cho ngắn hơn.

Sử dụng toán tử gán cộng

1. ***Thêm toán tử gán cộng***. Thay dòng 26-27 trong Hình 7.16 bởi dòng 26 trong Hình 7.20.

Sử dụng toán tử gán cộng để làm ngắn lệnh



Hình 7.20 Sử dụng toán tử gán cộng trong tính toán.

Trong bước này, chúng ta sử dụng toán tử gán cộng để tạo câu lệnh ngắn hơn. Chú ý rằng lệnh này vẫn thực hiện cùng một hành động - tính số tiền làm thêm giờ và cộng thêm vào tiền lương thông thường.

2. ***Chạy ứng dụng.*** Chọn **Debug > Start Debugging** để chạy ứng dụng. Chú ý, định dạng kết quả vẫn chưa chính xác. Chức năng của ứng dụng này vẫn giống hệt như phần trước. Chỉ khác là chúng ta đã đơn giản hóa bằng cách sử dụng toán tử += để rút gọn một câu lệnh.
3. ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

TỰ ÔN TẬP

1. Toán tử *=__ __.
   a) bình phương giá trị toán hạng bên phải và gán kết quả này cho toán hạng bên trái
   b) cộng giá trị toán hạng bên phải với giá trị toán hạng bên trái và gán kết quả này cho toán hạng bên trái
   c) tạo ra một biến mới và gán giá trị toán hạng bên phải cho biến đó
   d) nhân giá trị toán hạng bên trái với giá trị toán hạng bên phải và gán kết quả này cho toán hạng bên trái
2. Nếu biến `number` được khởi tạo giá trị là 5, giá trị của biến này là bao nhiêu sau khi biểu thức `number -= 3` được thực thi?
   a) 3 b) 5
   c) 7 d) 2

Đáp án: 1) d. 2) d.

7.9 Định dạng văn bản

Trong Visual Basic, có một vài cách để định dạng kết quả đầu ra. Trong phần này, chúng ta làm quen với phương thức `String.Format` để điều chỉnh định dạng văn bản được hiển thị. Việc chỉnh sửa cách văn bản hiển thị được gọi là **định dạng văn bản (text formatting)**. Phương thức này nhận **tham số chuỗi điều khiển định dạng (format control string)**, theo sau đó là các tham số chỉ ra giá trị cần được định dạng. Tham số chuỗi điều khiển định dạng cho biết các tham số còn lại sẽ được định dạng như thế nào.

Hãy nhớ lại rằng, ứng dụng **Wage Calculator** của bạn không hiển thị kết quả tính toán với đúng định dạng số thập phân và ký hiệu $ như bạn đã được nhìn thấy khi chạy thử ứng dụng này. Phần tiếp theo, bạn sẽ tìm hiểu cách định dạng tiền tệ cho giá trị trong `TextBox` **Gross earnings:**.

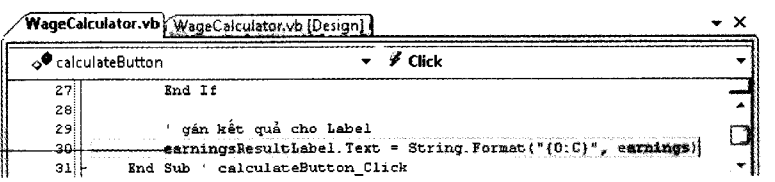
Định dạng cho giá trị tổng thu nhập trước thuế

Mẹo thiết kế giao diện

Định dạng số tiền bằng cách sử dụng ký tự chỉ thị định dạng C (currency).

1. ***Sửa đổi sự kiện `Click` của `Button` Calculate.*** Nếu IDE không hiển thị ở chế độ **Code**, chọn **View > Code**. Thay dòng 31 trên hình 7.17 bởi dòng 30 trên hình 7.21. Dòng 30 truyền đối số chuỗi điều khiển định dạng là "`{0:C}`" và giá trị cần định dạng là `earnings` vào phương thức `String.Format`. Số 0 cho biết rằng đối số 0 (`earnings` - là đối số đầu tiên sau chuỗi điều khiển định dạng) sẽ có định dạng được xác định bởi chữ cái đặt sau dấu hai chấm. Chữ cái này được gọi là ký tự **chỉ thị định dạng (format specifier)**. Trong ví dụ này, chúng ta sử dụng định dạng được chỉ thị bởi chữ cái viết hoa C là ký tự biểu diễn cho **định dạng tiền tệ (currency format)**, được sử dụng để hiển thị giá trị dưới dạng số tiền. Ảnh hưởng của ký tự chỉ thị định dạng C lên cùng một giá trị cần định dạng là khác nhau phụ thuộc vào thiết lập máy tính của bạn. Trong trường hợp này, kết quả hiển thị với dấu $ ở phía trước, sử dụng dấu phẩy để phân tách hàng nghìn và hiển thị với hai số lẻ thập phân (biểu diễn cho cent) bởi vì máy tính của bạn đang được thiết lập vị trí là United States. Bạn có thể thiết lập vị trí cho phù hợp, nhưng điều này nằm ngoài phạm vi của chương này.

Định dạng kết quả hiển thị dưới dạng tiền tệ



Hình 7.21 Sử dụng phương thức **Format** để hiển thị kết quả dưới định dạng tiền tệ.

2. ***Chạy ứng dụng.*** Chọn **Debug > Start Debugging** để chạy ứng dụng. Lúc này kết quả tổng thu nhập trước thuế được hiển thị dưới định dạng tiền tệ.
3. ***Đóng ứng dụng.*** Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.
4. ***Lưu project.*** Chọn **File > Save All** để lưu đoạn mã vừa chỉnh sửa.

Hình 7.22 đưa ra một số ký tự chỉ thị định dạng. Các ký tự chỉ thị định dạng không phân biệt chữ hoa chữ thường, bởi vậy các chữ cái viết hoa và các chữ cái thường có thể sử dụng tương đương. Chú ý rằng mã định dạng D chỉ được dùng với kiểu số nguyên (như kiểu `Integer`, `Byte`, `Short` và `Long`)

Ký tự chỉ thị định dạng	Mô tả
C	Định dạng tiền tệ (Currency). Định dạng tiền tệ dựa vào thiết lập vị trí địa phương của máy tính. Ví dụ với tiền U.S thì ký tự $ đứng trước ký tự số, với mỗi ba chữ số được phân tách bởi dấu phẩy và có hai số lẻ thập phân.
E	Định dạng số mũ (Exponent). Hiển thị một chữ số ở bên trái dấu chấm thập phân và sáu chữ số ở bên phải dấu chấm thập phân, theo sau là ký tự E và số nguyên có ba chữ số biểu diễn số mũ cơ số 10. Ví dụ, 965.2 có định dạng là 9.562000E+002.
F	Định dạng dấu chấm tĩnh (Fixed point). Thiết lập 2 số lẻ thập phân.
G	General. Visual Basic sẽ tự động chọn định dạng E hoặc F cho bạn, tùy thuộc vào định dạng nào có chuỗi ngắn hơn.
D	Định dạng thập phân (Decimal integer). Hiển thị một số nguyên được biểu diễn theo cơ số 10.
N	Định dạng số (Number). Với mỗi ba chữ số được phân tách bởi dấu phẩy và có hai số lẻ thập phân (tùy theo thiết lập vị trí địa phương của máy tính, người Mỹ sử dụng dấu chấm để phân tách phần nguyên và phần thập phân, người Việt lại dùng dấu phẩy để phân tách phần nguyên và phần thập phân).

Hình 7.22 Các ký tự chỉ thị định dạng cho chuỗi.

Hình 7.23 là mã nguồn của ứng dụng **Wage Calculator** trong đó những dòng mã chứa các khái niệm lập trình mới mà bạn vừa được học trong chương này được đánh dấu.

```
1  Public Class WageCalculatorForm
2      ' xử lý sự kiện Click
3      Private Sub calculateButton_Click(ByVal sender As System.Object, _
4          ByVal e As System.EventArgs) Handles calculateButton.Click
5
6          ' khai báo biến
7          Dim hours As Double
8          Dim wage As Decimal
9          Dim earnings As Decimal
10
11         Const HOUR_LIMIT As Integer = 40 ' khai báo hằng số
12
13         ' gán dữ liệu do người dùng nhập vào cho biến
14         hours = Val(hoursTextBox.Text)
15         wage = Val(wageTextBox.Text)
16
17         ' xác định cách tính thu nhập
18         If hours <= HOUR_LIMIT Then
19             ' nếu nhỏ hơn hoặc bằng 40h, tính lương theo thù lao cơ bản
20             earnings = hours * wage
21         Else
22             ' nếu lớn hơn 40h, tính theo tiền lương cơ bản cho 40h trước
23             earnings = HOUR_LIMIT * wage
24
25             ' tính thù lao gấp rưỡi cho giờ làm thêm
26             earnings += (Hours - HOUR_LIMIT) * (1.5 * wage)
27         End If
28
29         ' gán kết quả cho Label
30         earningsResultLabel.Text = String.Format("{0:C}", earnings)
31     End Sub ' calculateButton_Click
32 End Class ' WageCalculatorForm
```

Từ khóa `Const` cho biết đây là một hằng

Điều kiện nằm giữa từ khóa `If` và `Then`

Thân mệnh đề `Else` thực thi khi điều kiện có giá trị là `False`

Toán tử gán gán tổng của toán hạng bên trái với toán hạng bên phải cho toán hạng bên trái

Định dạng kết quả dưới dạng tiền tệ

Hình 7.23 Mã nguồn của ứng dụng **Wage Calculator**.

TỰ ÔN TẬP

1. Phương thức `String.Format` được sử dụng để_____.
   a) tạo các hằng số
   b) điều chỉnh định dạng văn bản
   c) định dạng các lệnh Visual Basic
   d) Các lựa chọn trên đều đúng
2. Định dạng_____hiển thị các giá trị dưới dạng số tiền.
   a) monetary
   b) cash
   c) currency
   d) dollar

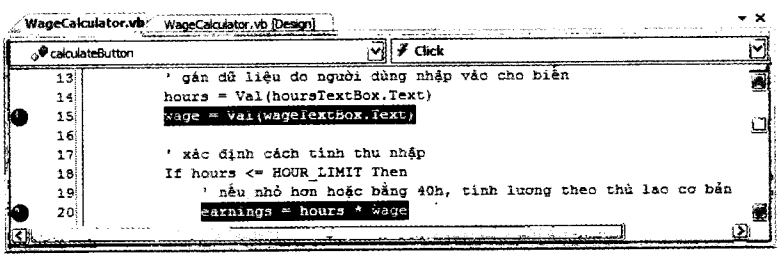
Đáp án: 1) b. 2) c.

7.10 Sử dụng trình gỡ lỗi: Cửa sổ Watch

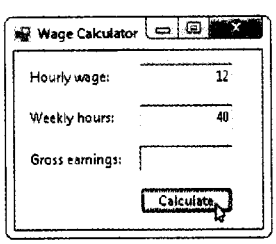
Visual Studio hỗ trợ một số cửa sổ gỡ lỗi. Bạn có thể truy cập cửa sổ này từ menu con **Debug > Windows**. **Cửa sổ Watch** chỉ hiển thị ở chế độ ngắt (break mode) cho phép bạn kiểm tra giá trị của biến hoặc biểu thức. Bạn có thể sử dụng cửa sổ **Watch** để theo dõi sự thay đổi giá trị biến khi ứng dụng thực thi hoặc để thay đổi giá trị của biến bằng cách nhập giá trị mới này trực tiếp vào cửa sổ **Watch**. Mỗi biểu thức hay mỗi biến được thêm vào cửa sổ **Watch** được gọi là một watch. Phần dưới đây sẽ hướng dẫn cách thêm, loại bỏ và thao tác với các watch bằng cách sử dụng cửa sổ **Watch**.

Sử dụng trình gỡ lỗi: Cửa sổ Watch

1. ***Bắt đầu gỡ lỗi.*** Nếu IDE không hiển thị chế độ **Code**, hãy chuyển sang chế độ **Code**. Đặt các điểm dừng (breakpoint) ở dòng 15 và dòng 20 (Hình 7.24). Chọn **Debug > Start Debugging** để chạy ứng dụng. Khi `Form` **Wage Calculator** xuất hiện, nhập giá trị 12 vào `TextBox` **Hourly wage:** và giá trị 40 vào `TextBox` **Weekly hours:** (Hình 7.25). Nhấn vào `Button` **Calculate.**



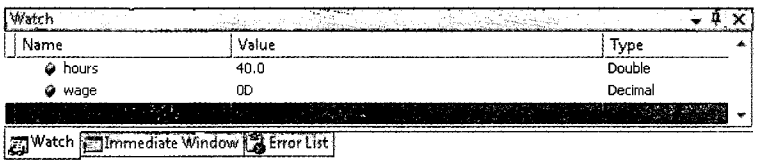
Hình 7.24 Thêm các điểm dừng vào ứng dụng **Wage Calculator**.



Hình 7.25 Ứng dụng **Wage Calculator**.

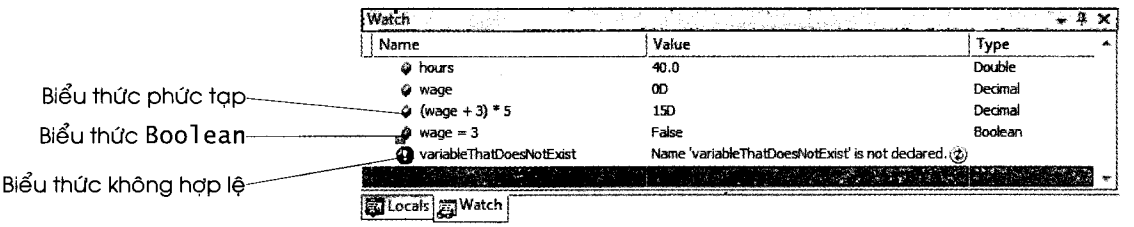
2. ***Tạm dừng thực thi ứng dụng.*** Khi bạn nhấn vào `Button` **Calculate** thì xử lý sự kiện `CalculateButton_Click` thực thi cho đến khi gặp điểm dừng. Khi gặp điểm dừng, ứng dụng sẽ tạm dừng thực thi và IDE sẽ chuyển sang chế độ ngắt. Chú ý rằng cửa sổ hiện thời sẽ thay đổi từ ứng dụng đang chạy sang IDE. **Cửa sổ hiện thời (active window)** là cửa sổ hiện đang được sử dụng và đôi khi còn được gọi là cửa sổ đang được **focus**. Ứng dụng **Wage Calculator** vẫn đang chạy, nhưng có thể bị ẩn đằng sau IDE.

3. ***Kiểm tra dữ liệu***. Một khi ứng dụng ở chế độ ngắt, bạn sẽ thoải mái kiểm tra giá trị của các biến bằng cách sử dụng cửa sổ **Watch** của trình gỡ lỗi. Để hiển thị cửa sổ **Watch** bạn chọn **Debug > Windows > Watch**. Ban đầu thì cửa sổ **Watch** trống. Để thêm các watch, bạn có thể nhập biểu thức vào cột **Name**. Nhấn vào trường đầu tiên của cột **Name**. Gõ `hours` và nhấn phím *Enter*. Giá trị và kiểu dữ liệu của watch được IDE tự động thêm vào (Hình 7.26). Chú ý rằng, giá trị này là `40.0` - giá trị được gán vào biến `hours` ở dòng 14. Nhập `wage` vào dòng tiếp theo và nhấn phím *Enter*. Giá trị được hiển thị cho biến `wage` là `0D`. `D` cho biết rằng biến `wage` chứa giá trị số kiểu `Decimal`. Bạn cũng có thể bôi đen tên biến trong đoạn mã và giữ chuột rồi kéo biến này vào trong cửa sổ **Watch** hoặc nhấn chuột phải vào biến trong đoạn mã và chọn **Add Watch**.



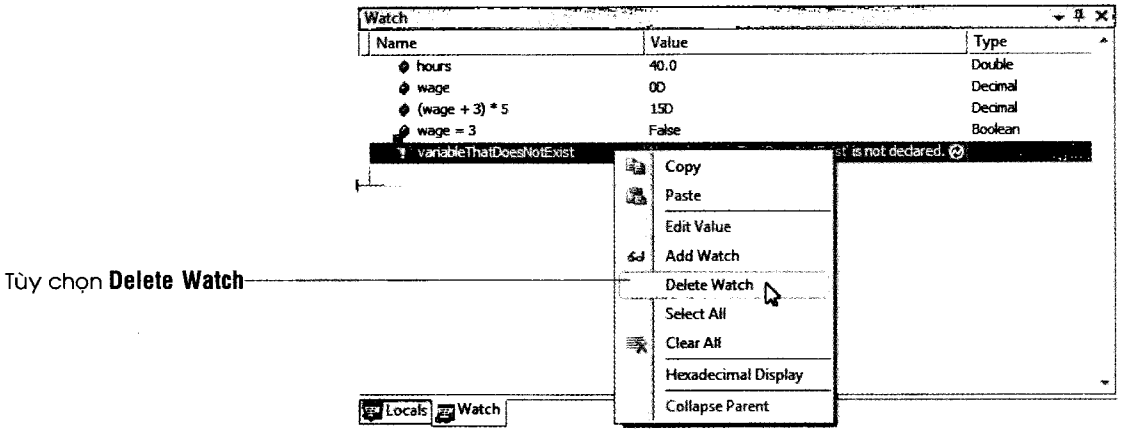
Hình 7.26 Cửa sổ **Watch**.

4. ***Kiểm tra các biểu thức khác nhau***. Thêm biểu thức `(wage + 3) * 5` vào cửa sổ **Watch**. Chú ý rằng cửa sổ **Watch** có thể tính toán các biểu thức toán học và trả lại giá trị là `15D`. Thêm biểu thức `wage = 3` vào trong cửa sổ **Watch** - Biểu thức chứa ký hiệu = được xử lý như là biểu thức `Boolean` thay vì xử lý như lệnh gán. Giá trị được trả lại là `False` bởi vì hiện tại `wage` không có giá trị là 3. Thêm biểu thức `variableThatDoesNotExist` vào cửa sổ **Watch**. Định danh này không tồn tại trong ứng dụng hiện tại cho nên nó không có kết quả trả về, thay vào đó là một thông báo sẽ hiển thị ở trường **Value**. Cửa sổ **Watch** trông như Hình 7.27.



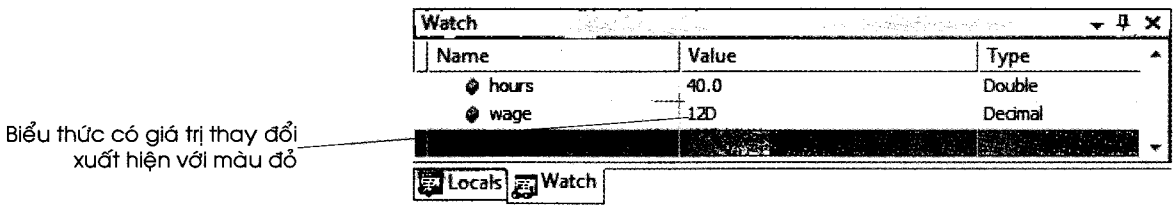
Hình 7.27 Tính toán giá trị các biểu thức.

5. ***Xóa biểu thức ra khỏi cửa sổ Watch***. Bây giờ, chúng ta muốn xóa biểu thức cuối ra khỏi cửa sổ **Watch**. Để xóa biểu thức, bạn chỉ cần nhấn chuột phải vào biểu thức muốn xóa ở trong cửa sổ **Watch** và chọn **Delete Watch** (Hình 7.28). Hoặc là bạn có thể nhấn vào một biến ở trong cửa sổ **Watch** và nhấn phím *Delete* để xóa biểu thức này. Hãy xóa toàn bộ các biểu thức mà bạn đã thêm ở *Bước 4*.



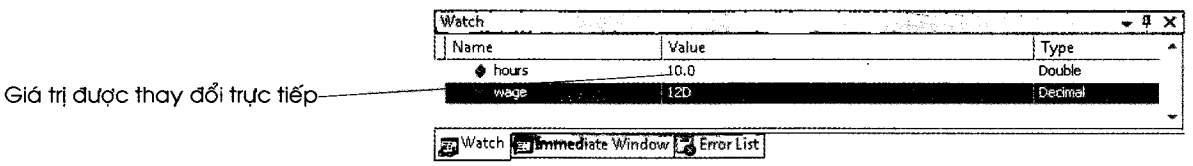
Hình 7.28 Xóa một watch.

6. ***Theo dõi sự thay đổi giá trị trong cửa sổ Watch.*** Tiếp tục gỡ lỗi bằng cách lựa chọn **Debug > Continue**. Ứng dụng tiếp tục thực thi cho đến khi gặp điểm dừng tiếp theo (dòng 20). Khi dòng 15 thực thi, giá trị người dùng nhập vào (giá trị 12) được gán vào biến `wage`. Đến dòng 18, điều kiện `If... Then` đạt giá trị `True` và một lần nữa ứng dụng tạm dừng thực thi ở dòng 20. Chú ý rằng giá trị biến `wage` không chỉ thay đổi trong ứng dụng, mà còn thay đổi trong cửa sổ **Watch**. Vì giá trị này bị thay đổi so với lần tạm dừng trước, do đó nó được hiển thị màu đỏ (Hình 7.29).



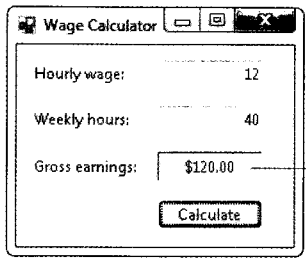
Hình 7.29 Các giá trị bị thay đổi trong cửa sổ **Watch**.

7. ***Thay đổi giá trị trực tiếp trong cửa sổ Watch.*** Cửa sổ **Watch** có thể được dùng để thay đổi giá trị của một biến trực tiếp bằng cách nhập giá trị mới vào cột **Value**. Nhấn đúp vào trường **Value** của biến `hours`, thay giá trị `40.0` bởi giá trị `10.0`, sau đó nhấn phím *Enter*. Giá trị bị thay đổi xuất hiện với màu đỏ (Hình 7.30). Chức năng này cho phép bạn kiểm tra các giá trị khác nhau để chắc chắn rằng ứng dụng của bạn có thể hoạt động đúng với các giá trị khác nhau.



Hình 7.30 Thay đổi giá trị trong cửa sổ **Watch**.

8. ***Xem kết quả ứng dụng.*** Chọn **Debug > Continue** để tiếp tục chạy ứng dụng. Sau khi xử lý sự kiện `calculateButton_Click` hoàn thành sự thực thi. Cửa sổ **Wage Calculator** xuất hiện và kết quả cuối cùng được hiển thị (Hình 7.31). Kết quả là `$120.00` vì chúng ta đã thay đổi giá trị biến `hours` là 10 ở bước cuối cùng. `TextBox` phía bên phải **Weekly hours:** vẫn hiển thị giá trị **40** vì chúng ta thay đổi giá trị biến `hours` nhưng không thay đổi thuộc tính `Text` của `TextBox`.



Hình 7.31 Kết quả hiển thị sau quá trình gỡ lỗi.

9. ***Đóng ứng dụng***. Đóng ứng dụng của bạn, bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng đang chạy hoặc chọn **Debug > Stop Debugging**.
10. ***Lưu project***. Chọn **File > Save All** để lưu mã đã được sửa đổi. [*Lưu ý*: các điểm dừng đã được thiết lập cũng sẽ được lưu].
11. ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

TỰ ÔN TẬP

1. Một ứng dụng chuyển sang chế độ ngắt khi_____.
   a) **Debug > Start** được chọn
   b) ứng dụng chạy đến một điểm dừng
   c) cửa sổ **Watch** được sử dụng
   d) xảy ra lỗi cú pháp
2. Cửa sổ **Watch** cho phép bạn_____.
   a) thay đổi giá trị của biến
   b) xem thông tin kiểu dữ liệu của biến
   c) tính toán giá trị biểu thức
   d) Các lựa chọn trên đều đúng

Đáp án: **1)** b. **2)** d.

7.11 Tổng kết

Trong chương này, chúng ta đã thảo luận về các kỹ thuật giải quyết vấn đề khi lập trình. Bạn đã được tìm hiểu về giải thuật, mã giả, UML và cấu trúc điều khiển. Bạn cũng đã được tìm hiểu về các loại cấu trúc điều khiển khác nhau và tình huống sử dụng các loại cấu trúc này.

Bạn đã bắt đầu bằng cách chạy thử ứng dụng sử dụng lệnh `If...Then...Else` để tính tổng thu nhập hàng tuần của nhân viên. Bạn đã học các lệnh điều khiển khác nhau và sử dụng UML để sơ đồ hóa quá trình đưa ra quyết định (decision-making process) của lệnh `If...Then` và lệnh `If...Then...Else`.

Bạn cũng đã biết cách định dạng văn bản sử dụng phương thức `String.Format` và cách rút gọn các câu lệnh gán bằng cách sử dụng toán tử gán.

Trong phần *sử dụng trình gỡ lỗi*, bạn đã học cách sử dụng cửa sổ **Watch** để quan sát dữ liệu của ứng dụng. Bạn đã biết cách thêm các watch, xóa các watch và thay đổi giá trị các biến.

Trong chương tiếp theo, bạn sẽ tìm hiểu cách hiển thị các hộp thoại thông báo dựa trên dữ liệu do người dùng nhập vào. Bạn sẽ tìm hiểu về toán tử lôgíc, toán tử này sẽ cung cấp cho bạn thêm nhiều cách tạo các điều kiện cho lệnh điều khiển và bạn sẽ sử dụng điều khiển `CheckBox` để cho phép người dùng lựa chọn các tùy chọn khác nhau trong ứng dụng.

TỔNG KẾT KỸ NĂNG

Lựa chọn giữa các hành động khác nhau

- Sử dụng lệnh điều khiển `If...Then` hoặc `If...Then...Else`.

Khái niệm hóa ứng dụng trước khi sử dụng Visual Studio 2008

- Sử dụng mã giả.
- Tạo bảng Hành động/Điều khiển/Sự kiện (ACE).

Hiểu về lệnh điều khiển

- Xem sơ đồ UML tương ứng của các lệnh điều khiển.

Thực hiện phép so sánh

- Sử dụng toán tử so sánh bằng và toán tử quan hệ.

Tạo hằng số

- Sử dụng từ khóa `Const`.
- Gán giá trị cho hằng số khi khai báo.

Rút gọn biểu thức gán

- Sử dụng toán tử gán.

Định dạng giá trị dưới dạng tiền tệ

- Sử dụng mã định dạng C trong phương thức `String.Format`.

Kiểm tra dữ liệu trong suốt quá trình thực thi ứng dụng

- Sử dụng trình gỡ lỗi để đặt các điểm dừng và kiểm tra dữ liệu trong cửa sổ **Watch**.

THUẬT NGỮ

biểu đồ hoạt động (activity diagram) - Là biểu đồ UML mô hình hóa hoạt động (hay còn gọi là luồng công việc) cho một phần của hệ thống phần mềm.

biểu thức điều kiện `If` - Biểu diễn rút gọn của lệnh `If...Then...Else`.

biểu thức hành động (action expression trong UML) - Sử dụng một trạng thái hành động trong biểu đồ hoạt động để chỉ ra hành động cụ thể được thực thi.

cấu trúc (lệnh) điều khiển (control structure) - Thành phần ứng dụng chỉ ra thứ tự thực thi lệnh.

cấu trúc (lệnh) điều khiển một đầu vào/một đầu ra (single-entry/single-exit control structure) - Là một lệnh điều khiển có một đầu vào và một đầu ra. Tất cả các lệnh điều khiển của Visual Basic là các lệnh điều khiển một đầu vào/một đầu ra.

cấu trúc (lệnh) lặp (repetition structure) - Cho phép lập trình viên chỉ ra một hành động hay một chuỗi hành động sẽ được lặp lại phụ thuộc vào giá trị của điều kiện.

cấu trúc (lệnh) lựa chọn (selection structure) - Lựa chọn giữa các hành động khác nhau.

cấu trúc (lệnh) tuần tự (sequence structure) - Được cài đặt sẵn trong Visual Basic, máy tính thực thi các lệnh Visual Basic một cách tuần tự.

chú thích (note) - Là một ghi chú mang tính giải thích (được biểu diễn bằng hình vuông với một nếp gấp ở góc trên bên phải) dùng để mô tả mục đích của một ký hiệu được dùng trong biểu đồ hoạt động).

chuỗi điều khiển định dạng (format control string) - Là một chuỗi chỉ ra cách dữ liệu sẽ được định dạng.

chuyển điều khiển (transfer of control) - Xảy ra khi một lệnh không được thực thi ngay sau khi lệnh trước nó được thực thi khi ứng dụng chạy.

cú pháp (syntax) - Chỉ ra định dạng của lệnh để không xảy ra lỗi cú pháp khi biên dịch.

cửa sổ hiện thời (active window) - Cửa sổ đang được sử dụng, hay còn được biết đến như là cửa sổ đang được focus.

cửa sổ Watch - Cửa sổ của IDE Visual Basic cho phép bạn xem và thay đổi giá trị các biến trong khi ứng dụng đang được gỡ lỗi.

điều khiển chương trình (program control) - Có nhiệm vụ sắp xếp các lệnh của ứng dụng theo đúng thứ tự.

điều kiện (condition) - Biểu thức có giá trị `True` hoặc `False` được sử dụng để đưa ra quyết định.

điều kiện canh giữ (guard condition) - Là một biểu thức nằm trong dấu ngoặc vuông ở trên hoặc bên cạnh mũi tên xuất phát từ một ký hiệu quyết định trong biểu đồ hoạt động dùng để xác định xem luồng công việc sẽ được thực hiện theo hướng nào.

định dạng tiền tệ (currency format) - Được sử dụng để hiển thị giá trị dưới dạng tiền tệ.

định dạng văn bản (formatting text) - Thay đổi diện mạo của đoạn văn bản cho mục đích hiển thị.

đường nét đứt (dotted line) - Là một ký hiệu trong biểu đồ hoạt động nối giữa chú thích với thành phần được chú thích.

focus - Dùng để chỉ cửa sổ hiện tại đang được sử dụng.

giải thuật (algorithm) - Là thủ tục để giải quyết vấn đề, chỉ ra các hành động được thực thi và thứ tự thực thi của các hành động này.

hằng (constant) - Là định danh có giá trị không thay đổi sau khi được khởi tạo trong lời khai báo.

hình tròn đặc (trong ngôn ngữ UML) - Là ký hiệu trong biểu đồ hoạt động biểu diễn trạng thái khởi đầu của hoạt động.

hình tròn nhỏ (trong ngôn ngữ UML) - Hình tròn đặc trong biểu đồ hoạt động biểu diễn trạng thái khởi đầu của hoạt động. Hình tròn đặc được bao quanh bởi một hình tròn rỗng biểu diễn trạng thái kết thúc của hoạt động.

khối lệnh (block) - Một nhóm lệnh.

kiểu dữ liệu `Boolean` - Kiểu dữ liệu của biến có giá trị `True` hoặc `False`.

kiểu dữ liệu thập phân (`Decimal` data type) - Được sử dụng để lưu trữ tiền tệ.

ký hiệu hình thoi (diamond symbol) - Ký hiệu hình thoi sử dụng trong biểu đồ hoạt động biểu diễn cho một quyết định được đưa ra.

ký hiệu trạng thái hành động (action-state symbol) - Hình chữ nhật với cạnh bên trái và bên phải được thay bằng đường vòng cung biểu diễn một hành động sẽ được thực thi trong biểu đồ hoạt động.

ký tự chỉ thị định dạng (format specifier) - Mã chỉ ra kiểu định dạng sẽ được áp dụng cho chuỗi.

lập trình cấu trúc (structured programing) - Là kỹ thuật tổ chức các điều khiển chương trình sử dụng cấu trúc lặp, cấu trúc lựa chọn và cấu trúc tuần tự giúp bạn phát triển các ứng dụng dễ hiểu, dễ gỡ lỗi và chỉnh sửa.

lệnh đa lựa chọn (multiple-selection statement) - Là lệnh lựa chọn giữa nhiều hành động khác nhau hoặc giữa nhiều chuỗi hành động khác nhau.

lệnh `If...Then` - Là lệnh lựa chọn thực hiện một hành động (hay một chuỗi hành động) dựa vào điều kiện. Còn được gọi là lệnh lựa chọn đơn.

lệnh `If...Then...Else` - Là lệnh lựa chọn thực hiện một hành động (hay chuỗi hành động) nếu điều kiện có giá trị true và thực hiện một hành động (hay chuỗi hành động) khác khi điều kiện có giá trị false. Lệnh này còn được gọi là lệnh lựa chọn kép.

lệnh lồng (nested statement) - Là một lệnh điều khiển được đặt bên trong một lệnh điều khiển khác.

lệnh lựa chọn đơn - Là lệnh `If...Then` lựa chọn hoặc bỏ qua một hành động đơn hoặc một chuỗi hành động.

lệnh lựa chọn kép - Là một lệnh (ví dụ như lệnh `If...Then...Else`) lựa chọn giữa hai hành động hoặc hai chuỗi hành động khác nhau.

lệnh thực thi (executable statement) - Là hành động được thực hiện khi chạy ứng dụng Visual Basic tương ứng.

lồng cấu trúc (lệnh) điều khiển (control structure nesting) - Đặt một lệnh điều khiển bên trong thân của một lệnh điều khiển khác.

luồng công việc (workflow) - Các hoạt động của một phần hệ thống phần mềm.

mã giả (pseudocode) - Là một loại ngôn ngữ không chính quy giúp bạn phát triển các giải thuật.

mô hình lập trình hành động/quyết định (action/decision model of programming) - Mô phỏng các lệnh điều khiển bằng biểu đồ hoạt động của UML với hình chữ nhật với cạnh bên trái và bên phải được thay bằng đường vòng cung biểu diễn các *hành động* được thực hiện và các ký hiệu hình thoi biểu diễn các *quyết định* được đưa ra.

phương thức `String.Format` - Định dạng chuỗi.

sự chuyển tiếp (transition) - Sự thay đổi từ một trạng thái hành động này sang một trạng thái hành động khác được biểu diễn bởi mũi tên chuyển tiếp trong biểu đồ hoạt động.

thực thi tuần tự (sequential execution) - Các lệnh trong ứng dụng được thực thi từ lệnh này sang lệnh khác tuân theo thứ tự xuất hiện.

toán tử quan hệ (relational operator) - Toán tử < (nhỏ hơn), > (lớn hơn), <= (nhỏ hơn hoặc bằng), >= (lớn hơn hoặc bằng) được dùng để so sánh hai giá trị (còn được gọi là toán tử so sánh).

toán tử so sánh bằng (equality operator) - Là toán tử = (bằng) và <> (khác) sử dụng để so sánh hai giá trị.

trạng thái kết thúc (final state) - Được biểu diễn bằng hình tròn đặc nằm trong một hình tròn rỗng trong biểu đồ hoạt động cho biết đây là điểm kết thúc luồng công việc sau khi ứng dụng thực hiện xong các hoạt động.

trạng thái hành động (action state) - Là hành động thực thi trong biểu đồ hoạt động và được biểu diễn bằng ký hiệu trạng thái hành động.

trạng thái khởi đầu (initial state) - Trong biểu đồ hoạt động, trạng thái khởi đầu được biểu diễn bằng ký hiệu hình tròn đặc, đây chính là điểm bắt đầu luồng công việc trước khi ứng dụng thực hiện các hoạt động đã được mô hình hóa.

từ khóa `Const` - Được dùng để khai báo hằng.

từ khóa `Else` - Chỉ ra rằng lệnh sẽ được thực thi khi điều kiện của lệnh `If...Then...Else` có giá trị `False`.

từ khóa `ElseIf` - Là từ khóa sử dụng cho điều kiện lồng trong lệnh `If...Then...Else` lồng nhau.

UML (Unified Modeling Language - Ngôn ngữ mô hình hợp nhất) - Là tiêu chuẩn công nghiệp được sử dụng để mô hình hóa hệ thống phần mềm bằng các biểu đồ.

xếp chồng cấu trúc (lệnh) điều khiển (control structure stacking) - Tập hợp các lệnh điều khiển có thứ tự. Đầu ra của một lệnh điều khiển này được nối với đầu vào của lệnh điều khiển kế tiếp.

HƯỚNG DẪN THIẾT KẾ GIAO DIỆN

Thiết kế tổng thể

- Định dạng tiền tệ bằng ký tự chỉ thị định dạng C (currency).

TextBox

- Khi sử dụng nhiều `TextBox` theo chiều dọc, hãy căn lề phải các `TextBox` và nên tạo ra các `TextBox` cùng kích thước. Căn lề trái các `Label` mô tả cho các `TextBox` này.

ĐIỀU KHIỂN, SỰ KIỆN, THUỘC TÍNH & PHƯƠNG THỨC

`String` - Lớp này biểu diễn một xâu các ký tự được xem như là một đơn vị đơn lẻ.

- ***Phương thức***

  `Format` - Sắp xếp chuỗi theo một định dạng cụ thể.

CÂU HỎI TRẮC NGHIỆM

7.1 Trong biểu thức điều kiện, toán tử_____trả lại giá trị `False` khi toán hạng bên trái lớn hơn toán hạng bên phải.

a) =
b) <
c) <=
d) Các lựa chọn trên đều đúng

7.2 _____xảy ra khi một lệnh không được thực thi ngay sau khi lệnh trước nó được thực thi theo thứ tự xuất hiện trong ứng dụng.

a) Sự chuyển tiếp
b) Luồng
c) Lỗi lôgíc
d) Chuyển điều khiển

7.3 Bạn có thể tương tác với cửa sổ **Watch** trong_____.

a) chế độ chạy
b) chế độ gỡ lỗi
c) chế độ ngắt
d) Các lựa chọn trên đều đúng

7.4 Lệnh `If...Then` được gọi là một lệnh_____bởi vì nó lựa chọn hoặc bỏ qua một hành động.

a) lựa chọn đơn
b) đa lựa chọn
c) lựa chọn kép
d) lặp

7.5 Có ba loại lệnh điều khiển là lệnh tuần tự, lệnh lựa chọn và lệnh_____.

a) lặp lại
b) vòng lặp
c) thực hiện lại
d) lặp

7.6 Trong một biểu đồ hoạt động, một hình chữ nhật có hai cạnh bên được thay bằng đường vòng cung biểu diễn _____.

a) một giải thuật hoàn chỉnh
b) một chú thích
c) một hành động
d) điểm kết thúc của ứng dụng

7.7 Lệnh lựa chọn `If...Then...Else` kết thúc với từ khóa_____.

a) `End If Then Else`
b) `End If Else`
c) `End Else`
d) `End If`

7.8 Giá trị của biến kiểu `Boolean` được gán bởi từ khóa_____hoặc từ khóa _____.

a) `True, False`
b) `Off, On`
c) `True, Not True`
d) `Yes, No`

7.9 Định danh có giá trị không thay đổi sau khi được khởi tạo ở lệnh khai báo gọi là _____.

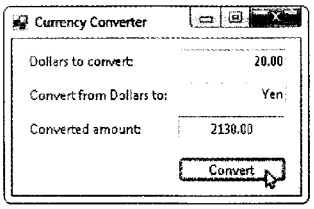
a) `Double`
b) hằng
c) tiêu chuẩn
d) `Boolean`

7.10 Toán tử_____gán kết quả phép cộng toán hạng bên trái với toán hạng bên phải cho toán hạng bên trái.

a) +
b) =+
c) +=
d) Các lựa chọn trên đều sai

BÀI TẬP

7.11 (***Ứng dụng Currency Converter***) Phát triển ứng dụng có chức năng như một bộ chuyển đổi tiền tệ, như trên Hình 7.32. Người dùng phải cung cấp một số tiền đô la vào `TextBox` **Dollars to convert:** và tên một loại tiền tệ (dạng văn bản) trong `TextBox` **Convert from dollars to:**. Nhấn vào `Button` **Convert** sẽ chuyển số tiền trên thành số tiền tính theo loại tiền tệ được nhập và hiển thị nó trên một `Label`. Bạn có thể giới hạn các loại tiền tệ cho người dùng nhập vào gồm các loại sau: Ơ-rô (Euros), Yên (Yen) và Pê-sô (Pesos). Sử dụng tỷ giá hối đoái: 1 đô la = 0.69 Ơ-rô, 106.5 Yên, 11 Pê-sô.



Chú thích hình

- ***Dollars to convert:*** *Số tiền đô la*
- ***Convert from dollars to:*** *Chuyển từ đô la sang*
- ***Converted amount:*** *Số tiền sau khi chuyển đổi*

Hình 7.32 Giao diện ứng dụng **Currency Converter**.

a) ***Copy template của ứng dụng vào thư mục làm việc***. Copy thư mục `C:\Examples\Tutorial07\Exercises\CurrencyConverter` tới thư mục `C:\SimplyVB2008`.

b) ***Mở file template của ứng dụng***. Nhấn đúp vào file `CurrencyConverter.sln` trong thư mục `CurrencyConverter` để mở ứng dụng.

c) ***Thêm xử lý sự kiện cho sự kiện `Click` của `Button` Converter***. Nhấn đúp vào `Button` **Converter** để tạo ra một xử lý sự kiện rỗng cho sự kiện `Click` của `Button` này. Đoạn mã từ *Bước d-f* nằm trong xử lý sự kiện này.

d) ***Lấy giá trị do người dùng nhập vào***. Định nghĩa biến `amount` kiểu `Decimal`. Sử dụng hàm `Val` để chuyển giá trị trong `TextBox` **Dollars to convert:** do người dùng nhập vào thành kiểu `Double`. Gán kết quả này cho biến `amount`. Visual Basic sẽ ngầm chuyển giá trị này từ kiểu `Double` sang `Decimal`.

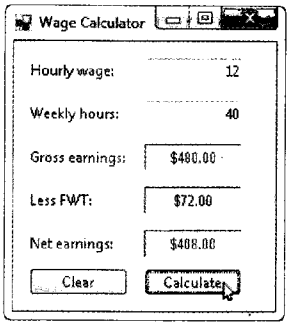
e) ***Thực hiện chuyển đổi***. Sử dụng lệnh lồng `If...Then...Else` để xác định tên loại tiền tệ người dùng nhập vào. Gán kết quả chuyển đổi cho biến `amount`. Sử dụng phương thức `String.Format` với ký tự chỉ thị định dạng là `F` để hiển thị kết quả.

f) ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng. Nhập số tiền đô la và tên loại tiền tệ bạn muốn chuyển đổi. Nhấn vào `Button` **Convert**, hãy sử dụng tỷ giá hối đoái như ở trên để kiểm tra xem kết quả hiển thị đúng chưa.

g) ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

h) ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

7.12 (***Ứng dụng Wage Calculator thực hiện tính thu nhập sau thuế***) Phát triển ứng dụng tính thu nhập của nhân viên như Hình 7.33. Người dùng cung cấp thông tin về tiền thù lao mỗi giờ và số giờ làm việc một tuần. Khi người dùng nhấn vào `Button` **Calculate**, tổng thu nhập chưa chịu thuế (gross earnings) sẽ được hiển thị trên `TextBox` **Gross Earnings:**. `TextBox` **Less FWT** hiển thị số tiền thuế bị giảm trừ và `TextBox` **Net earnings:** hiển thị thu nhập sau thuế hay thu nhập thực tế (là chênh lệch giữa số thu nhập chưa chịu thuế và số tiền thuế bị giảm trừ). Giả sử rằng tiền thù lao làm ngoài giờ gấp `1,5` lần tiền thù lao mỗi giờ và mức thuế bằng `15%` thu nhập chưa chịu thuế. `Button` **Clear** sẽ xóa dữ liệu nhập trong các `TextBox`.



Chú thích hình

- ***Hourly wage:*** *Tiền thù lao mỗi giờ*
- ***Weekly hours:*** *Số giờ trong tuần*
- ***Gross earnings:*** *Tổng thu nhập chưa chịu thuế*
- ***Less FWT:*** *Tiền thuế*
- ***Net earnings:*** *Thu nhập thực tế*

Hình 7.33 Giao diện ứng dụng **Wage Calculate**.

a) ***Copy template của ứng dụng vào thư mục làm việc.*** Copy thư mục `C:\Examples\Toturial07\Exercises\ExpandedWageCalculator` vào thư mục `C:\SimplyVB2008`.

b) ***Mở file template của ứng dụng.*** Nhấn đúp vào file `WageCalculator.sln` trong thư mục `ExpandedWageCalculator` để mở ứng dụng.

c) ***Sửa đổi xử lý sự kiện cho sự kiện `Click` của `Button` Calculate.*** Thêm đoạn mã từ *Bước d-f* vào `CalculateButton_Click`.

d) ***Thêm một biến và một hằng mới.*** Khai báo biến `federalTaxes` để chứa số tiền thuế bị trừ. Khai báo hằng TAX_RATE và gán cho hằng này giá trị `0.15` (15%).

e) ***Tính và hiển thị kết quả số tiền thuế bị trừ.*** Nhân tổng thu nhập (`earnings`) với tỷ lệ thuế (TAX_RATE) để tính số tiền thuế bị trừ. Gán kết quả này cho biến `federalTaxes`. Hiển thị giá trị biến này sử dụng phương thức `String.Format` với ký tự chỉ thị định dạng là C.

f) ***Tính và hiển thị thu nhập thực tế của nhân viên.*** Lấy hiệu giữa biến `earnings` và `federalTaxes` để tính thu nhập thực tế (net earnings). Hiển thị giá trị sử dụng phương thức `String.Format` với ký tự chỉ thị định dạng là C.

g) ***Thêm xử lý sự kiện cho `Button` Clear.*** Nhấn đúp vào `Button` **Clear** để tạo ra xử lý sự kiện rỗng cho sự kiện `Click` của `Button` này. Xử lý sự kiện này sẽ xóa dữ liệu người dùng nhập vào các `TextBox` và xóa kết quả hiển thị trên ba `Label`.

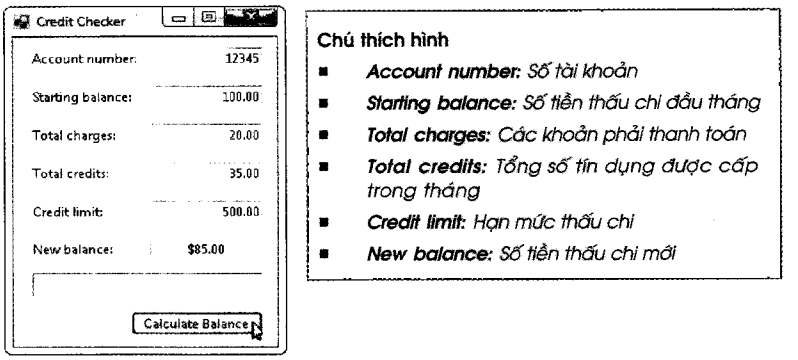
h) ***Chạy ứng dụng.*** Chọn **Debug > Start Debugging** để chạy ứng dụng. Nhập tiền thù lao mỗi giờ và số giờ làm việc. Nhấn vào `Button` **Calculate** và kiểm tra lại kết quả thu nhập chưa chịu thuế, số tiền thuế bị trừ và thu nhập sau thuế đã hiển thị đúng chưa. Nhấn vào `Button` **Clear** và kiểm tra xem dữ liệu các trường có bị xóa hay không.

i) ***Đóng ứng dụng.*** Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

j) ***Đóng IDE.*** Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

7.13 (***Ứng dụng Customer Charge Account Analyzer***) Phát triển ứng dụng (như trong Hình 7.34) xác định xem liệu khách hàng của một cửa hàng tạp hóa có vượt quá hạn mức tín dụng của tài khoản thấu chi hay không. Mỗi khách hàng nhập vào số tài khoản (một số `Integer`), số tiền thấu chi trong tài khoản đầu tháng (một số `Decimal`), số tiền bạn đã vay nợ trong tháng một số `Decimal`), tổng số tín dụng được cấp cho tài khoản của khách hàng trong tháng (một số `Decimal`), và hạn mức tín dụng được cho phép của khách hàng (một số `Decimal`). Nhập các thông số thực tế và tính số tiền thấu chi mới (= *số tiền thấu chi đầu tháng – tổng số tín dụng được cấp trong tháng + các khoản phải thanh toán*), hiển thị tiền số thấu chi mới và kiểm tra số tiền mới này có vượt quá hạn mức tín dụng của khách hàng hay không. Nếu vượt quá giới hạn này, ứng dụng sẽ hiển thị thông

báo (trên một `Label` nằm ở cuối `Form`) cho khách hàng. Nếu người dùng nhập một số tài khoản khác, ứng dụng sẽ tự động xóa các `TextBox`, `Label` thông báo lỗi và `Label` kết quả.



Hình 7.34 Giao diện ứng dụng **Credit Checker**.

a) ***Copy template của ứng dụng vào thư mục làm việc***. Copy thư mục `C:\Examples\Tutorial07\Exercises\CreditChecker` vào thư mục `C:\SimplyVB2008`.

b) ***Mở file template của ứng dụng***. Nhấn đúp vào file `CreditChecker.sln` trong thư mục `CreditChecker` để mở ứng dụng.

c) ***Thêm xử lý sự kiện cho sự kiện `Click` của `Button` Calculate Balance***. Nhấn đúp vào `Button` **Caculate Balance** để tạo ra một xử lý sự kiện rỗng cho sự kiện `Click` của `Button` này. Thêm đoạn mã từ *Bước d-g* vào xử lý sự kiện này.

d) ***Khai báo biến***. Khai báo bốn biến kiểu `Decimal` để lưu thông tin số thấu chi đầu tháng, các khoản phải thanh toán, tín dụng được cấp trong tháng và hạn hạn mức tín dụng. Khai báo biến kiểu `Decimal` thứ năm để lưu số tiền thấu chi sau khi đã tính khoản vay và tín dụng.

e) ***Lấy dữ liệu người dùng nhập vào***. Lấy dữ liệu do người dùng nhập vào từ thuộc tính `Text` của `TextBox`.

f) ***Tính và hiển thị số tiền thấu chi mới***. Số dư mới bằng số thấu chi đầu tháng trừ tổng tín dụng được cấp trong tháng và cộng với các khoản phải thanh toán. Gán kết quả này cho biến. Hiển thị kết quả này dưới định dạng tiền tệ.

g) ***Xác định xem hạn mức tín dụng có bị vượt quá không***. Nếu số thấu chi mới vượt quá hạn mức tín dụng cho phép thì thông điệp sẽ được hiển thị trên `Label errorLabel`.

h) ***Xử lý sự kiện `TextChanged` của `TextBox` Account number:***. Nhấn đúp vào `TextBox` **Account Number:** để tạo xử lý sự kiện cho sự kiện `TextChanged`. Xử lý sự kiện này xóa giá trị các `TextBox` còn lại, cùng với `Label` thông báo lỗi và `Label` kết quả.

i) ***Chạy ứng dụng***. Chọn **Debug > Start Debugging** để chạy ứng dụng. Nhập vào số tài khoản, số thấu chi đầu tháng, các khoản phải thanh toán, tín dụng được cấp trong tháng và giới hạn mức tín dụng tài khoản của bạn. Nhấn vào `Button` **Calculate Balance** và kiểm tra xem số dư mới của tài khoản đã hiển thị đúng chưa. Nhập vào một giá trị tổng chi phí vượt quá hạn mức tín dụng của bạn. Nhấn vào `Button` **Calculate Balance** và đảm bảo rằng một thông điệp được hiển thị ở `Label` phía bên dưới. Thay đổi số tài khoản và kiểm tra xem tất cả các trường đã bị xóa chưa.

j) ***Đóng ứng dụng***. Đóng ứng dụng đang chạy bằng cách nhấn vào nút x ở trên cùng, bên phải của ứng dụng.

k) ***Đóng IDE***. Đóng cửa sổ Visual Basic IDE bằng cách nhấn vào nút x ở trên cùng, bên phải của IDE.

Đoạn mã này làm gì? ▶ **7.14** Giả sử rằng `ageTextBox` là một điều khiển `TextBox` và người dùng nhập giá trị 27 vào `TextBox` này. Xác định xem hành động nào được thực hiện trong đoạn mã sau.

```
Dim age As Integer

age = Val(ageTextBox.Text)

If age < 0 Then
    ageLabel.Text = "Enter a value greater than or equal to zero."
ElseIf age < 13 Then
    ageLabel.Text = "Child"
ElseIf age < 20 Then
    ageLabel.Text = "Teenager"
ElseIf age < 30 Then
    ageLabel.Text = "Young Adult"
ElseIf age < 65 Then
    ageLabel.Text = "Adult"
Else
    ageLabel.Text = "Senior Citizen"
End If
```

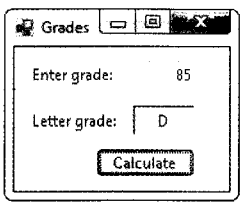
Đoạn mã này có gì sai? ▶ **7.15** Giả sử rằng `ampmLabel` là một điều khiển `Label`. Tìm lỗi trong đoạn mã dưới đây.

```
Dim hour As Integer

hour = 14

If hour < 11 Then
    If hour > 0 Then
        ampmLabel.Text = "AM"
    End If
Else
    ampmLabel.Text = "PM"
ElseIf hour > 23 Then
    ampmLabel.Text = "Time Error."
End If
```

Sử dụng trình gỡ lỗi ▶ **7.16** (***Ứng dụng Grade Calculator***) Copy thư mục `C:\Examples\Tutorial07\Debugger` tới thư mục làm việc của bạn. Thư mục này chứa ứng dụng `Grades`, ứng dụng này lấy một số do người dùng nhập vào và hiển thị một chữ cái tương ứng với điểm số. Ví dụ, với giá trị 90-100 hiển thị **A**; 80-89, **B**, 70-79, **C**, 60-69, **D** và các giá trị thấp hơn hiển thị **F**. Chạy ứng dụng này. Nhập giá trị 85 vào `TextBox` và nhấn vào **Calculate**. Chú ý rằng ứng dụng hiển thị **D** thay vì hiển thị **B**. Chọn **View > Code** để mở chế độ soạn thảo mã và đặt các điểm dừng cần thiết. Chọn **Debug > Start Debugging** để sử dụng trình gỡ lỗi giúp bạn tìm lỗi. Hình 7.35 hiển thị kết quả sai khi giá trị 85 được nhập.



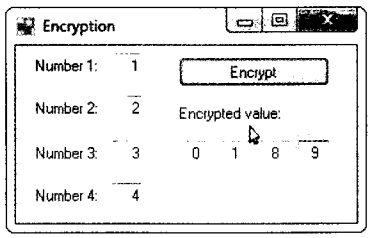
Hình 7.35 Kết quả lỗi của ứng dụng **Grade**.

Bài tập nâng cao ▶

7.17 (***Ứng dụng Encryption***) Một công ty truyền dữ liệu qua điện thoại nhưng lo lắng rằng điện thoại có thể bị nghe trộm. Tất cả dữ liệu được truyền dưới dạng số nguyên (`Integer`) có bốn chữ số. Công ty này yêu cầu bạn viết một ứng dụng mã hóa dữ liệu để truyền dữ liệu an toàn hơn. Mã hóa là quá trình biến đổi dữ liệu cho mục đích bảo mật. Tạo một `Form` tương tự Hình 7.36. Ứng dụng này đọc 4 chữ số do người dùng nhập vào và mã hóa thông tin. Giả sử rằng, người dùng nhập từng chữ số đơn lẻ vào mỗi `TextBox`. Sử dụng các kỹ thuật mã hóa số dưới đây:

a) Thay thế mỗi chữ số bởi (*tổng của chữ số đó và 7*) `Mod 10`.

b) Hoán đổi chữ số đầu tiên với chữ số thứ ba, hoán đổi chữ số thứ hai với chữ số thứ tư.



Hình 7.36 Ứng dụng **Encryption**.