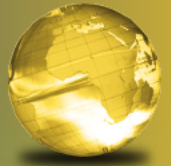


GLOBAL  
EDITION



# Visual C#

## *How to Program*

SIXTH EDITION

Paul Deitel • Harvey Deitel



 Pearson

# Visual C#<sup>®</sup>



## HOW TO PROGRAM

This page intentionally left blank

# Visual C#<sup>®</sup>



## HOW TO PROGRAM

SIXTH EDITION  
GLOBAL EDITION

**Paul Deitel**  
**Harvey Deitel**  
*Deitel & Associates, Inc.*



Boston Columbus Hoboken Indianapolis New York San Francisco  
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal  
Toronto Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Vice President, Editorial Director: *Marcia Horton*  
Acquisitions Editor: *Tracy Johnson*  
Editorial Assistant: *Kristy Alaura*  
Acquisitions Editor, Global Editions: *Sourabh Maheshwari*  
VP of Marketing: *Christy Lesko*  
Field Marketing Manager: *Demetrius Hall*  
Marketing Assistant: *Jon Bryant*  
Director of Product Management: *Erin Gregg*  
Team Lead, Program and Project Management: *Scott Disanno*  
Program Manager: *Carole Snyder*  
Project Manager: *Robert Engelhardt*  
Project Editor, Global Editions: *K.K. Neelakantan*  
Manufacturing Buyer, Higher Ed | RR Donnelley: *Maura Zaldivar-Garcia*  
Senior Manufacturing Controller, Global Editions: *Trudy Kimber*  
Media Production Manager, Global Editions: *Vikram Kumar*  
Cover Design: *Lumina Datamatics*  
R&P Manager: *Ben Ferrini*  
Inventory Manager: *Ann Lam*  
Cover Photo Credit: © *Pichugin Dmitry/Shutterstock.com*

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on page 6.

The authors and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The authors and publisher make no warranty of any kind, expressed or implied, with regard to these programs or to the documentation contained in this book. The authors and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Pearson Education Limited  
Edinburgh Gate  
Harlow  
Essex CM20 2JE  
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:  
[www.pearsonglobaleditions.com](http://www.pearsonglobaleditions.com)

© Pearson Education Limited 2018

The rights of Paul Deitel and Harvey Deitel to be identified as the authors of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

*Authorized adaptation from the United States edition, entitled Visual C# How to Program, 6th Edition, ISBN 978-0-13-460154-0 by Paul Deitel and Harvey Deitel published by Pearson Education © 2017.*

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6-10 Kirby Street, London EC1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

ISBN-10: 1-292-15346-6

ISBN-13: 978-1-292-15346-9

Typeset by GEX Publishing Services

Printed and bound in Malaysia

*In memory of William Siebert, Professor Emeritus of  
Electrical Engineering and Computer Science at MIT:*

*Your use of visualization techniques in  
your Signals and Systems lectures inspired  
the way generations of engineers, computer  
scientists, educators and authors present  
their work.*

*Harvey and Paul Deitel*

## Trademarks

DEITEL and the double-thumbs-up bug are registered trademarks of Deitel and Associates, Inc.

Microsoft® Windows®, and Microsoft Visual C#® are registered trademarks of the Microsoft Corporation in the U.S.A. And other countries. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

UNIX is a registered trademark of The Open Group.

Throughout this book, trademarks are used. Rather than put a trademark symbol in every occurrence of a trademarked name, we state that we are using the names in an editorial fashion only and to the benefit of the trademark owner, with no intention of infringement of the trademark.



# Contents

<b>Preface</b>	<b>23</b>
<b>Before You Begin</b>	<b>37</b>
<b>I Introduction to Computers, the Internet and Visual C#</b>	<b>41</b>
1.1 Introduction	42
1.2 Computers and the Internet in Industry and Research	42
1.3 Hardware and Software	45
1.3.1 Moore's Law	45
1.3.2 Computer Organization	46
1.4 Data Hierarchy	47
1.5 Machine Languages, Assembly Languages and High-Level Languages	50
1.6 Object Technology	51
1.7 Internet and World Wide Web	53
1.8 C#	55
1.8.1 Object-Oriented Programming	56
1.8.2 Event-Driven Programming	56
1.8.3 Visual Programming	56
1.8.4 Generic and Functional Programming	56
1.8.5 An International Standard	57
1.8.6 C# on Non-Windows Platforms	57
1.8.7 Internet and Web Programming	57
1.8.8 Asynchronous Programming with async and await	57
1.8.9 Other Key Programming Languages	58
1.9 Microsoft's .NET	60
1.9.1 .NET Framework	60
1.9.2 Common Language Runtime	60
1.9.3 Platform Independence	61
1.9.4 Language Interoperability	61
1.10 Microsoft's Windows® Operating System	61
1.11 Visual Studio Integrated Development Environment	63
1.12 <b>Painter</b> Test-Drive in Visual Studio Community	63



<b>2</b>	<b>Introduction to Visual Studio and Visual Programming</b>	<b>73</b>
2.1	Introduction	74
2.2	Overview of the Visual Studio Community 2015 IDE	74
2.2.1	Introduction to Visual Studio Community 2015	74
2.2.2	Visual Studio Themes	75
2.2.3	Links on the Start Page	75
2.2.4	Creating a New Project	76
2.2.5	<b>New Project</b> Dialog and Project Templates	77
2.2.6	Forms and Controls	78
2.3	Menu Bar and Toolbar	79
2.4	Navigating the Visual Studio IDE	82
2.4.1	<b>Solution Explorer</b>	83
2.4.2	<b>Toolbox</b>	84
2.4.3	<b>Properties</b> Window	84
2.5	Help Menu and Context-Sensitive Help	86
2.6	Visual Programming: Creating a Simple App that Displays Text and an Image	87
2.7	Wrap-Up	96
2.8	Web Resources	97
<b>3</b>	<b>Introduction to C# App Programming</b>	<b>105</b>
3.1	Introduction	106
3.2	Simple App: Displaying a Line of Text	106
3.2.1	Comments	107
3.2.2	<code>using</code> Directive	108
3.2.3	Blank Lines and Whitespace	109
3.2.4	Class Declaration	109
3.2.5	Main Method	111
3.2.6	Displaying a Line of Text	111
3.2.7	Matching Left ( { ) and Right ( } ) Braces	112
3.3	Creating a Simple App in Visual Studio	112
3.3.1	Creating the Console App	112
3.3.2	Changing the Name of the App File	114
3.3.3	Writing Code and Using <i>IntelliSense</i>	114
3.3.4	Compiling and Running the App	116
3.3.5	Errors, Error Messages and the <b>Error List</b> Window	117
3.4	Modifying Your Simple C# App	117
3.4.1	Displaying a Single Line of Text with Multiple Statements	118
3.4.2	Displaying Multiple Lines of Text with a Single Statement	118
3.5	String Interpolation	120
3.6	Another C# App: Adding Integers	121
3.6.1	Declaring the <code>int</code> Variable <code>number1</code>	122
3.6.2	Declaring Variables <code>number2</code> and <code>sum</code>	123

3.6.3	Prompting the User for Input	123
3.6.4	Reading a Value into Variable <code>number1</code>	123
3.6.5	Prompting the User for Input and Reading a Value into <code>number2</code>	124
3.6.6	Summing <code>number1</code> and <code>number2</code>	124
3.6.7	Displaying the sum with <code>string</code> Interpolation	125
3.6.8	Performing Calculations in Output Statements	125
3.7	Memory Concepts	125
3.8	Arithmetic	126
3.8.1	Arithmetic Expressions in Straight-Line Form	127
3.8.2	Parentheses for Grouping Subexpressions	127
3.8.3	Rules of Operator Precedence	127
3.8.4	Sample Algebraic and C# Expressions	128
3.8.5	Redundant Parentheses	129
3.9	Decision Making: Equality and Relational Operators	129
3.10	Wrap-Up	134

## 4 Introduction to Classes, Objects, Methods and strings

146

4.1	Introduction	147
4.2	Test-Driving an Account Class	148
4.2.1	Instantiating an Object—Keyword <code>new</code>	148
4.2.2	Calling Class <code>Account</code> 's <code>GetName</code> Method	149
4.2.3	Inputting a Name from the User	149
4.2.4	Calling Class <code>Account</code> 's <code>SetName</code> Method	150
4.3	<code>Account</code> Class with an Instance Variable and <code>Set</code> and <code>Get</code> Methods	150
4.3.1	<code>Account</code> Class Declaration	150
4.3.2	Keyword <code>class</code> and the Class Body	151
4.3.3	Instance Variable name of Type <code>string</code>	151
4.3.4	<code>SetName</code> Method	152
4.3.5	<code>GetName</code> Method	154
4.3.6	Access Modifiers <code>private</code> and <code>public</code>	154
4.3.7	<code>Account</code> UML Class Diagram	155
4.4	Creating, Compiling and Running a Visual C# Project with Two Classes	156
4.5	Software Engineering with <code>Set</code> and <code>Get</code> Methods	157
4.6	<code>Account</code> Class with a Property Rather Than <code>Set</code> and <code>Get</code> Methods	158
4.6.1	Class <code>AccountTest</code> Using <code>Account</code> 's Name Property	158
4.6.2	<code>Account</code> Class with an Instance Variable and a Property	160
4.6.3	<code>Account</code> UML Class Diagram with a Property	162
4.7	Auto-Implemented Properties	162
4.8	<code>Account</code> Class: Initializing Objects with Constructors	163
4.8.1	Declaring an <code>Account</code> Constructor for Custom Object Initialization	163
4.8.2	Class <code>AccountTest</code> : Initializing <code>Account</code> Objects When They're Created	164

4.9	Account Class with a Balance; Processing Monetary Amounts	166
4.9.1	Account Class with a decimal balance Instance Variable	166
4.9.2	AccountTest Class That Uses Account Objects with Balances	169
4.10	Wrap-Up	173

## 5 Algorithm Development and Control Statements: Part I 182

5.1	Introduction	183
5.2	Algorithms	184
5.3	Pseudocode	184
5.4	Control Structures	185
5.4.1	Sequence Structure	185
5.4.2	Selection Statements	186
5.4.3	Iteration Statements	187
5.4.4	Summary of Control Statements	187
5.5	if Single-Selection Statement	187
5.6	if...else Double-Selection Statement	188
5.6.1	Nested if...else Statements	189
5.6.2	Dangling-else Problem	191
5.6.3	Blocks	191
5.6.4	Conditional Operator (?:)	192
5.7	Student Class: Nested if...else Statements	193
5.8	while Iteration Statement	196
5.9	Formulating Algorithms: Counter-Controlled Iteration	197
5.9.1	Pseudocode Algorithm with Counter-Controlled Iteration	197
5.9.2	Implementing Counter-Controlled Iteration	198
5.9.3	Integer Division and Truncation	200
5.10	Formulating Algorithms: Sentinel-Controlled Iteration	201
5.10.1	Top-Down, Stepwise Refinement: The Top and First Refinement	201
5.10.2	Second Refinement	202
5.10.3	Implementing Sentinel-Controlled Iteration	204
5.10.4	Program Logic for Sentinel-Controlled Iteration	205
5.10.5	Braces in a while Statement	206
5.10.6	Converting Between Simple Types Explicitly and Implicitly	206
5.10.7	Formatting Floating-Point Numbers	207
5.11	Formulating Algorithms: Nested Control Statements	208
5.11.1	Problem Statement	208
5.11.2	Top-Down, Stepwise Refinement: Pseudocode Representation of the Top	209
5.11.3	Top-Down, Stepwise Refinement: First Refinement	209
5.11.4	Top-Down, Stepwise Refinement: Second Refinement	209
5.11.5	Complete Second Refinement of the Pseudocode	210
5.11.6	App That Implements the Pseudocode Algorithm	211
5.12	Compound Assignment Operators	213

5.13	Increment and Decrement Operators	213
5.13.1	Prefix Increment vs. Postfix Increment	214
5.13.2	Simplifying Increment Statements	215
5.13.3	Operator Precedence and Associativity	216
5.14	Simple Types	216
5.15	Wrap-Up	217

## 6 Control Statements: Part 2 232

6.1	Introduction	233
6.2	Essentials of Counter-Controlled Iteration	233
6.3	for Iteration Statement	235
6.3.1	A Closer Look at the for Statement's Header	236
6.3.2	General Format of a for Statement	236
6.3.3	Scope of a for Statement's Control Variable	236
6.3.4	Expressions in a for Statement's Header Are Optional	237
6.3.5	Placing Arithmetic Expressions in a for Statement's Header	237
6.3.6	Using a for Statement's Control Variable in the Statement's Body	238
6.3.7	UML Activity Diagram for the for Statement	238
6.4	Examples Using the for Statement	238
6.5	App: Summing Even Integers	239
6.6	App: Compound-Interest Calculations	240
6.6.1	Performing the Interest Calculations with Math Method pow	241
6.6.2	Formatting with Field Widths and Alignment	242
6.6.3	Caution: Do Not Use float or double for Monetary Amounts	243
6.7	do...while Iteration Statement	244
6.8	switch Multiple-Selection Statement	245
6.8.1	Using a switch Statement to Count A, B, C, D and F Grades	245
6.8.2	switch Statement UML Activity Diagram	249
6.8.3	Notes on the Expression in Each case of a switch	250
6.9	Class AutoPolicy Case Study: strings in switch Statements	251
6.10	break and continue Statements	253
6.10.1	break Statement	253
6.10.2	continue Statement	254
6.11	Logical Operators	255
6.11.1	Conditional AND (&&) Operator	256
6.11.2	Conditional OR (  ) Operator	256
6.11.3	Short-Circuit Evaluation of Complex Conditions	257
6.11.4	Boolean Logical AND (&) and Boolean Logical OR (  ) Operators	257
6.11.5	Boolean Logical Exclusive OR (^)	258
6.11.6	Logical Negation (!) Operator	258
6.11.7	Logical Operators Example	259
6.12	Structured-Programming Summary	261
6.13	Wrap-Up	266

<b>7</b>	<b>Methods: A Deeper Look</b>	<b>277</b>
7.1	Introduction	278
7.2	Packaging Code in C#	279
7.2.1	Modularizing Programs	279
7.2.2	Calling Methods	280
7.3	static Methods, static Variables and Class Math	280
7.3.1	Math Class Methods	281
7.3.2	Math Class Constants PI and E	282
7.3.3	Why Is Main Declared static?	282
7.3.4	Additional Comments About Main	283
7.4	Methods with Multiple Parameters	283
7.4.1	Keyword static	285
7.4.2	Method Maximum	285
7.4.3	Assembling strings with Concatenation	285
7.4.4	Breaking Apart Large string Literals	286
7.4.5	When to Declare Variables as Fields	287
7.4.6	Implementing Method Maximum by Reusing Method Math.Max	287
7.5	Notes on Using Methods	287
7.6	Argument Promotion and Casting	288
7.6.1	Promotion Rules	289
7.6.2	Sometimes Explicit Casts Are Required	289
7.7	The .NET Framework Class Library	290
7.8	Case Study: Random-Number Generation	292
7.8.1	Creating an Object of Type Random	292
7.8.2	Generating a Random Integer	292
7.8.3	Scaling the Random-Number Range	293
7.8.4	Shifting Random-Number Range	293
7.8.5	Combining Shifting and Scaling	293
7.8.6	Rolling a Six-Sided Die	293
7.8.7	Scaling and Shifting Random Numbers	296
7.8.8	Repeatability for Testing and Debugging	297
7.9	Case Study: A Game of Chance; Introducing Enumerations	297
7.9.1	Method RollDice	300
7.9.2	Method Main's Local Variables	300
7.9.3	enum Type Status	301
7.9.4	The First Roll	301
7.9.5	enum Type DiceNames	301
7.9.6	Underlying Type of an enum	301
7.9.7	Comparing Integers and enum Constants	302
7.10	Scope of Declarations	302
7.11	Method-Call Stack and Activation Records	305
7.11.1	Method-Call Stack	305
7.11.2	Stack Frames	305
7.11.3	Local Variables and Stack Frames	306
7.11.4	Stack Overflow	306
7.11.5	Method-Call Stack in Action	306

7.12	Method Overloading	309
7.12.1	Declaring Overloaded Methods	309
7.12.2	Distinguishing Between Overloaded Methods	310
7.12.3	Return Types of Overloaded Methods	311
7.13	Optional Parameters	311
7.14	Named Parameters	312
7.15	C# 6 Expression-Bodied Methods and Properties	313
7.16	Recursion	314
7.16.1	Base Cases and Recursive Calls	314
7.16.2	Recursive Factorial Calculations	314
7.16.3	Implementing Factorial Recursively	315
7.17	Value Types vs. Reference Types	317
7.18	Passing Arguments By Value and By Reference	318
7.18.1	ref and out Parameters	319
7.18.2	Demonstrating ref, out and Value Parameters	320
7.19	Wrap-Up	322

## **8 Arrays; Introduction to Exception Handling 339**

8.1	Introduction	340
8.2	Arrays	341
8.3	Declaring and Creating Arrays	342
8.4	Examples Using Arrays	343
8.4.1	Creating and Initializing an Array	343
8.4.2	Using an Array Initializer	344
8.4.3	Calculating a Value to Store in Each Array Element	345
8.4.4	Summing the Elements of an Array	347
8.4.5	Iterating Through Arrays with foreach	347
8.4.6	Using Bar Charts to Display Array Data Graphically; Introducing Type Inference with var	349
8.4.7	Using the Elements of an Array as Counters	352
8.5	Using Arrays to Analyze Survey Results; Intro to Exception Handling	353
8.5.1	Summarizing the Results	354
8.5.2	Exception Handling: Processing the Incorrect Response	355
8.5.3	The try Statement	355
8.5.4	Executing the catch Block	355
8.5.5	Message Property of the Exception Parameter	356
8.6	Case Study: Card Shuffling and Dealing Simulation	356
8.6.1	Class Card and Getter-Only Auto-Implemented Properties	356
8.6.2	Class DeckOfCards	357
8.6.3	Shuffling and Dealing Cards	360
8.7	Passing Arrays and Array Elements to Methods	361
8.8	Case Study: GradeBook Using an Array to Store Grades	363
8.9	Multidimensional Arrays	369
8.9.1	Rectangular Arrays	369

## 14 Contents

8.9.2	Jagged Arrays	370
8.9.3	Two-Dimensional Array Example: Displaying Element Values	371
8.10	Case Study: GradeBook Using a Rectangular Array	374
8.11	Variable-Length Argument Lists	380
8.12	Using Command-Line Arguments	382
8.13	(Optional) Passing Arrays by Value and by Reference	384
8.14	Wrap-Up	388

## 9 Introduction to LINQ and the List Collection 410

9.1	Introduction	411
9.2	Querying an Array of <code>int</code> Values Using LINQ	412
9.2.1	The <code>from</code> Clause	414
9.2.2	The <code>where</code> Clause	415
9.2.3	The <code>select</code> Clause	415
9.2.4	Iterating Through the Results of the LINQ Query	415
9.2.5	The <code>orderby</code> Clause	415
9.2.6	Interface <code>IEnumerable&lt;T&gt;</code>	416
9.3	Querying an Array of <code>Employee</code> Objects Using LINQ	416
9.3.1	Accessing the Properties of a LINQ Query's Range Variable	420
9.3.2	Sorting a LINQ Query's Results by Multiple Properties	420
9.3.3	<code>Any</code> , <code>First</code> and <code>Count</code> Extension Methods	420
9.3.4	Selecting a Property of an Object	420
9.3.5	Creating New Types in the <code>select</code> Clause of a LINQ Query	420
9.4	Introduction to Collections	421
9.4.1	<code>List&lt;T&gt;</code> Collection	421
9.4.2	Dynamically Resizing a <code>List&lt;T&gt;</code> Collection	422
9.5	Querying the Generic <code>List</code> Collection Using LINQ	426
9.5.1	The <code>let</code> Clause	428
9.5.2	Deferred Execution	428
9.5.3	Extension Methods <code>ToArray</code> and <code>ToList</code>	428
9.5.4	Collection Initializers	428
9.6	Wrap-Up	429
9.7	Deitel LINQ Resource Center	429

## 10 Classes and Objects: A Deeper Look 434

10.1	Introduction	435
10.2	Time Class Case Study; Throwing Exceptions	435
10.2.1	<code>Time1</code> Class Declaration	436
10.2.2	Using Class <code>Time1</code>	437
10.3	Controlling Access to Members	439
10.4	Referring to the Current Object's Members with the <code>this</code> Reference	440
10.5	Time Class Case Study: Overloaded Constructors	442
10.5.1	Class <code>Time2</code> with Overloaded Constructors	442
10.5.2	Using Class <code>Time2</code> 's Overloaded Constructors	446

10.6	Default and Parameterless Constructors	448
10.7	Composition	449
	10.7.1 Class Date	449
	10.7.2 Class Employee	451
	10.7.3 Class EmployeeTest	452
10.8	Garbage Collection and Destructors	453
10.9	static Class Members	453
10.10	readonly Instance Variables	457
10.11	Class View and Object Browser	458
	10.11.1 Using the Class View Window	458
	10.11.2 Using the Object Browser	459
10.12	Object Initializers	460
10.13	Operator Overloading; Introducing struct	460
	10.13.1 Creating Value Types with struct	461
	10.13.2 Value Type ComplexNumber	461
	10.13.3 Class ComplexTest	463
10.14	Time Class Case Study: Extension Methods	464
10.15	Wrap-Up	467

## **I I Object-Oriented Programming: Inheritance 475**

11.1	Introduction	476
11.2	Base Classes and Derived Classes	477
11.3	protected Members	479
11.4	Relationship between Base Classes and Derived Classes	480
	11.4.1 Creating and Using a CommissionEmployee Class	481
	11.4.2 Creating a BasePlusCommissionEmployee Class without Using Inheritance	485
	11.4.3 Creating a CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy	490
	11.4.4 CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using protected Instance Variables	493
	11.4.5 CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using private Instance Variables	496
11.5	Constructors in Derived Classes	500
11.6	Software Engineering with Inheritance	500
11.7	Class object	501
11.8	Wrap-Up	502

## **I 2 OOP: Polymorphism and Interfaces 508**

12.1	Introduction	509
12.2	Polymorphism Examples	511
12.3	Demonstrating Polymorphic Behavior	512
12.4	Abstract Classes and Methods	515
12.5	Case Study: Payroll System Using Polymorphism	517
	12.5.1 Creating Abstract Base Class Employee	518



12.5.2	Creating Concrete Derived Class <code>SalariedEmployee</code>	520
12.5.3	Creating Concrete Derived Class <code>HourlyEmployee</code>	522
12.5.4	Creating Concrete Derived Class <code>CommissionEmployee</code>	523
12.5.5	Creating Indirect Concrete Derived Class <code>BasePlusCommissionEmployee</code>	525
12.5.6	Polymorphic Processing, Operator <code>is</code> and Downcasting	526
12.5.7	Summary of the Allowed Assignments Between Base-Class and Derived-Class Variables	531
12.6	<code>sealed</code> Methods and Classes	532
12.7	Case Study: Creating and Using Interfaces	533
12.7.1	Developing an <code>IPayable</code> Hierarchy	534
12.7.2	Declaring Interface <code>IPayable</code>	536
12.7.3	Creating Class <code>Invoice</code>	536
12.7.4	Modifying Class <code>Employee</code> to Implement Interface <code>IPayable</code>	538
12.7.5	Using Interface <code>IPayable</code> to Process Invoices and Employees Polymorphically	539
12.7.6	Common Interfaces of the .NET Framework Class Library	541
12.8	Wrap-Up	542

## 13 Exception Handling: A Deeper Look 547

13.1	Introduction	548
13.2	Example: Divide by Zero without Exception Handling	549
13.2.1	Dividing By Zero	550
13.2.2	Enter a Non-Numeric Denominator	551
13.2.3	Unhandled Exceptions Terminate the App	552
13.3	Example: Handling <code>DivideByZeroExceptions</code> and <code>FormatExceptions</code>	552
13.3.1	Enclosing Code in a <code>try</code> Block	554
13.3.2	Catching Exceptions	554
13.3.3	Uncaught Exceptions	555
13.3.4	Termination Model of Exception Handling	556
13.3.5	Flow of Control When Exceptions Occur	556
13.4	.NET Exception Hierarchy	557
13.4.1	Class <code>SystemException</code>	557
13.4.2	Which Exceptions Might a Method Throw?	558
13.5	<code>finally</code> Block	559
13.5.1	Moving Resource-Release Code to a <code>finally</code> Block	559
13.5.2	Demonstrating the <code>finally</code> Block	560
13.5.3	Throwing Exceptions Using the <code>throw</code> Statement	564
13.5.4	Rethrowing Exceptions	564
13.5.5	Returning After a <code>finally</code> Block	565
13.6	The <code>using</code> Statement	566
13.7	Exception Properties	567
13.7.1	Property <code>InnerException</code>	567
13.7.2	Other Exception Properties	568
13.7.3	Demonstrating Exception Properties and Stack Unwinding	568

13.7.4	Throwing an Exception with an InnerException	570
13.7.5	Displaying Information About the Exception	571
13.8	User-Defined Exception Classes	571
13.9	Checking for null References; Introducing C# 6's ?. Operator	575
13.9.1	Null-Conditional Operator (?.)	575
13.9.2	Revisiting Operators is and as	576
13.9.3	Nullable Types	576
13.9.4	Null Coalescing Operator (??)	577
13.10	Exception Filters and the C# 6 when Clause	577
13.11	Wrap-Up	578

## 14 Graphical User Interfaces with Windows Forms: Part 1 584

14.1	Introduction	585
14.2	Windows Forms	586
14.3	Event Handling	588
14.3.1	A Simple Event-Driven GUI	589
14.3.2	Auto-Generated GUI Code	591
14.3.3	Delegates and the Event-Handling Mechanism	593
14.3.4	Another Way to Create Event Handlers	594
14.3.5	Locating Event Information	595
14.4	Control Properties and Layout	596
14.4.1	Anchoring and Docking	597
14.4.2	Using Visual Studio To Edit a GUI's Layout	599
14.5	Labels, TextBoxes and Buttons	600
14.6	GroupBoxes and Panels	603
14.7	CheckBoxes and RadioButtons	606
14.7.1	CheckBoxes	606
14.7.2	Combining Font Styles with Bitwise Operators	608
14.7.3	RadioButtons	609
14.8	PictureBoxes	614
14.9	ToolTips	617
14.10	NumericUpDown Control	618
14.11	Mouse-Event Handling	621
14.12	Keyboard-Event Handling	623
14.13	Wrap-Up	627

## 15 Graphical User Interfaces with Windows Forms: Part 2 637

15.1	Introduction	638
15.2	Menus	638
15.3	MonthCalendar Control	648
15.4	DateTimePicker Control	649

15.5	LinkLabel Control	652
15.6	ListBox Control	655
15.7	CheckedListBox Control	660
15.8	ComboBox Control	663
15.9	TreeView Control	667
15.10	ListView Control	673
15.11	TabControl Control	679
15.12	Multiple Document Interface (MDI) Windows	683
15.13	Visual Inheritance	691
15.14	User-Defined Controls	696
15.15	Wrap-Up	699

## 16 Strings and Characters: A Deeper Look 707

16.1	Introduction	708
16.2	Fundamentals of Characters and Strings	709
16.3	string Constructors	710
16.4	string Indexer, Length Property and CopyTo Method	711
16.5	Comparing strings	712
16.6	Locating Characters and Substrings in strings	716
16.7	Extracting Substrings from strings	719
16.8	Concatenating strings	720
16.9	Miscellaneous string Methods	720
16.10	Class StringBuilder	722
16.11	Length and Capacity Properties, EnsureCapacity Method and Indexer of Class StringBuilder	723
16.12	Append and AppendFormat Methods of Class StringBuilder	725
16.13	Insert, Remove and Replace Methods of Class StringBuilder	727
16.14	Char Methods	730
16.15	Introduction to Regular Expressions (Online)	732
16.16	Wrap-Up	732

## 17 Files and Streams 739

17.1	Introduction	740
17.2	Files and Streams	740
17.3	Creating a Sequential-Access Text File	741
17.4	Reading Data from a Sequential-Access Text File	750
17.5	Case Study: Credit-Inquiry Program	754
17.6	Serialization	759
17.7	Creating a Sequential-Access File Using Object Serialization	760
17.8	Reading and Deserializing Data from a Binary File	764
17.9	Classes File and Directory	767
	17.9.1 Demonstrating Classes File and Directory	768
	17.9.2 Searching Directories with LINQ	771
17.10	Wrap-Up	775

<b>18</b>	<b>Searching and Sorting</b>	<b>782</b>
18.1	Introduction	783
18.2	Searching Algorithms	784
18.2.1	Linear Search	784
18.2.2	Binary Search	788
18.3	Sorting Algorithms	792
18.3.1	Selection Sort	793
18.3.2	Insertion Sort	796
18.3.3	Merge Sort	800
18.4	Summary of the Efficiency of Searching and Sorting Algorithms	806
18.5	Wrap-Up	807
<b>19</b>	<b>Custom Linked Data Structures</b>	<b>812</b>
19.1	Introduction	813
19.2	Simple-Type structs, Boxing and Unboxing	813
19.3	Self-Referential Classes	814
19.4	Linked Lists	815
19.5	Stacks	828
19.6	Queues	832
19.7	Trees	835
19.7.1	Binary Search Tree of Integer Values	836
19.7.2	Binary Search Tree of IComparable Objects	843
19.8	Wrap-Up	849
<b>20</b>	<b>Generics</b>	<b>855</b>
20.1	Introduction	856
20.2	Motivation for Generic Methods	857
20.3	Generic-Method Implementation	859
20.4	Type Constraints	862
20.4.1	IComparable<T> Interface	862
20.4.2	Specifying Type Constraints	862
20.5	Overloading Generic Methods	865
20.6	Generic Classes	865
20.7	Wrap-Up	875
<b>21</b>	<b>Generic Collections; Functional Programming with LINQ/PLINQ</b>	<b>881</b>
21.1	Introduction	882
21.2	Collections Overview	884
21.3	Class Array and Enumerators	886
21.3.1	C# 6 using static Directive	888
21.3.2	Class UsingArray's static Fields	889
21.3.3	Array Method Sort	889

21.3.4	Array Method Copy	889
21.3.5	Array Method BinarySearch	889
21.3.6	Array Method GetEnumerator and Interface IEnumerator	889
21.3.7	Iterating Over a Collection with foreach	890
21.3.8	Array Methods Clear, IndexOf, LastIndexOf and Reverse	890
21.4	Dictionary Collections	890
21.4.1	Dictionary Fundamentals	891
21.4.2	Using the SortedDictionary Collection	892
21.5	Generic LinkedList Collection	896
21.6	C# 6 Null Conditional Operator ?[]	900
21.7	C# 6 Dictionary Initializers and Collection Initializers	901
21.8	Delegates	901
21.8.1	Declaring a Delegate Type	903
21.8.2	Declaring a Delegate Variable	903
21.8.3	Delegate Parameters	904
21.8.4	Passing a Method Name Directly to a Delegate Parameter	904
21.9	Lambda Expressions	904
21.9.1	Expression Lambdas	906
21.9.2	Assigning Lambdas to Delegate Variables	907
21.9.3	Explicitly Typed Lambda Parameters	907
21.9.4	Statement Lambdas	907
21.10	Introduction to Functional Programming	907
21.11	Functional Programming with LINQ Method-Call Syntax and Lambdas	909
21.11.1	LINQ Extension Methods Min, Max, Sum and Average	912
21.11.2	Aggregate Extension Method for Reduction Operations	912
21.11.3	The Where Extension Method for Filtering Operations	914
21.11.4	Select Extension Method for Mapping Operations	915
21.12	PLINQ: Improving LINQ to Objects Performance with Multicore	915
21.13	(Optional) Covariance and Contravariance for Generic Types	919
21.14	Wrap-Up	921

## 22 Databases and LINQ

933

22.1	Introduction	934
22.2	Relational Databases	935
22.3	A Books Database	936
22.4	LINQ to Entities and the ADO.NET Entity Framework	940
22.5	Querying a Database with LINQ	941
22.5.1	Creating the ADO.NET Entity Data Model Class Library	943
22.5.2	Creating a Windows Forms Project and Configuring It to Use the Entity Data Model	947
22.5.3	Data Bindings Between Controls and the Entity Data Model	949
22.6	Dynamically Binding Query Results	955
22.6.1	Creating the Display Query Results GUI	956
22.6.2	Coding the Display Query Results App	957
22.7	Retrieving Data from Multiple Tables with LINQ	959

22.8	Creating a Master/Detail View App	965
22.8.1	Creating the Master/Detail GUI	965
22.8.2	Coding the Master/Detail App	967
22.9	Address Book Case Study	968
22.9.1	Creating the <b>Address Book</b> App's GUI	970
22.9.2	Coding the <b>Address Book</b> App	971
22.10	Tools and Web Resources	975
22.11	Wrap-Up	975

## 23 Asynchronous Programming with `async` and `await` 982

23.1	Introduction	983
23.2	Basics of <code>async</code> and <code>await</code>	985
23.2.1	<code>async</code> Modifier	985
23.2.2	<code>await</code> Expression	985
23.2.3	<code>async</code> , <code>await</code> and Threads	985
23.3	Executing an Asynchronous Task from a GUI App	986
23.3.1	Performing a Task Asynchronously	986
23.3.2	Method <code>calculateButton_Click</code>	988
23.3.3	Task Method <code>Run</code> : Executing Asynchronously in a Separate Thread	989
23.3.4	<code>awaiting</code> the Result	989
23.3.5	Calculating the Next Fibonacci Value Synchronously	989
23.4	Sequential Execution of Two Compute-Intensive Tasks	990
23.5	Asynchronous Execution of Two Compute-Intensive Tasks	992
23.5.1	<code>awaiting</code> Multiple Tasks with Task Method <code>WhenAll</code>	995
23.5.2	Method <code>StartFibonacci</code>	996
23.5.3	Modifying a GUI from a Separate Thread	996
23.5.4	<code>awaiting</code> One of Several Tasks with Task Method <code>WhenAny</code>	996
23.6	Invoking a Flickr Web Service Asynchronously with <code>HttpClient</code>	997
23.6.1	Using Class <code>HttpClient</code> to Invoke a Web Service	1001
23.6.2	Invoking the Flickr Web Service's <code>flickr.photos.search</code> Method	1001
23.6.3	Processing the XML Response	1002
23.6.4	Binding the Photo Titles to the <code>ListBox</code>	1003
23.6.5	Asynchronously Downloading an Image's Bytes	1004
23.7	Displaying an Asynchronous Task's Progress	1004
23.8	Wrap-Up	1008

## Chapters on the Web 1015

### A Operator Precedence Chart 1016

### B Simple Types 1018

<b>C</b>	<b>ASCII Character Set</b>	<b>1020</b>
	<b>Appendices on the Web</b>	<b>1021</b>
	<b>Index</b>	<b>1023</b>

## Online Topics

PDFs presenting additional topics for advanced college courses and professionals are available through the book's Companion Website:

<http://www.pearsonglobaleditions.com/deitel>

New copies of this book come with a Companion Website access code that's located on the book's inside front cover. If the access code is already visible or there is no card, you purchased a used book or an edition that does not come with an access code.

### **Web App Development with ASP.NET**

#### **XML and LINQ to XML**

#### **Universal Windows Platform (UWP) GUI, Graphics, Multimedia and XAML**

#### **REST Web Services**

#### **Cloud Computing with Microsoft Azure™**

#### **Windows Presentation Foundation (WPF) GUI, Graphics, Multimedia and XAML**

#### **ATM Case Study, Part 1: Object-Oriented Design with the UML**

#### **ATM Case Study, Part 2: Implementing an Object-Oriented Design in C#**

#### **Using the Visual Studio Debugger**



## Preface

6

Welcome to the world of desktop, mobile and web app development with Microsoft's® Visual C#® programming language. *Visual C# How to Program, 6/e* is based on C# 6<sup>1</sup> and related Microsoft software technologies. You'll be using the .NET platform and the Visual Studio® Integrated Development Environment on which you'll conveniently write, test and debug your applications and run them on Windows® devices. The Windows operating system runs on desktop and notebook computers, mobile phones and tablets, game systems and a great variety of devices associated with the emerging "Internet of Things."

We believe that this book and its supplements for students and instructors will give you an informative, engaging, challenging and entertaining introduction to Visual C#. The book presents leading-edge computing technologies in a friendly manner appropriate for introductory college course sequences, based on the curriculum recommendations of two key professional organizations—the ACM and the IEEE.<sup>2</sup>

You'll study four of today's most popular programming paradigms:

- object-oriented programming,
- structured programming,
- generic programming and
- functional programming (new in this edition).

At the heart of the book is the Deitel signature *live-code approach*—rather than using code snippets, we generally present concepts in the context of complete working programs followed by sample executions. We include a broad range of example programs and exercises selected from computer science, business, education, social issues, personal utilities, sports, mathematics, puzzles, simulation, game playing, graphics, multimedia and many other areas. We also provide abundant tables, line drawings and UML diagrams for a more visual learning experience.

---

1. At the time of this writing, Microsoft has not yet released the official C# 6 Specification. To view an unofficial copy, visit <https://github.com/1jw1004/csharpspec/blob/gh-pages/README.md>

2. These recommendations include *Computer Science Curricula 2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*, December 20, 2013, The Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM), IEEE Computer Society.



Read the Before You Begin section after this Preface for instructions on setting up your computer to run the hundreds of code examples and to enable you to develop your own C# apps. The source code for all of the book's examples is available at

<http://www.pearsonglobal editions.com/deitel>

Use the source code we provide to compile and run each program as you study it—this will help you master Visual C# and related Microsoft technologies faster and at a deeper level. Most of the book's examples work in Visual Studio on Windows 7, 8 or 10 (there is no 9). The code examples for the online presentation of the Universal Windows Platform (UWP) and XAML specifically require Windows 10.

## Contacting the Authors

As you read the book, if you have a question, we're easy to reach at

[deitel@deitel.com](mailto:deitel@deitel.com)

We'll respond promptly.

## Join the Deitel & Associates, Inc. Social Media Communities

Subscribe to the *Deitel*<sup>®</sup> *Buzz Online* newsletter

<http://www.deitel.com/newsletter/subscribe.html>

and join the Deitel social media communities on

- Facebook<sup>®</sup>—<http://facebook.com/DeitelFan>
- LinkedIn<sup>®</sup>—<http://linkedin.com/company/deitel-&-associates>
- YouTube<sup>®</sup>—<http://youtube.com/DeitelTV>
- Twitter<sup>®</sup>—<http://twitter.com/Deitel>
- Instagram<sup>®</sup>—<http://instagram.com/DeitelFan>
- Google+<sup>™</sup>—<http://google.com/+DeitelFan>

## Object-Oriented Programming with an Early Objects Approach

The book introduces the basic concepts and terminology of object-oriented programming in Chapter 1. In Chapter 2, you'll *visually* manipulate objects, such as labels and images. In Chapter 3, Introduction to C# App Programming, you'll write Visual C# program code that manipulates preexisting objects. You'll develop your first customized classes and objects in Chapter 4. Presenting objects and classes early gets you “thinking about objects” immediately and mastering these concepts more thoroughly.

Our early objects presentation continues in Chapters 5–9 with a variety of straightforward case studies. In Chapters 10–12, we take a deeper look at classes and objects,

present inheritance, interfaces and polymorphism, then use those concepts throughout the remainder of the book.

## New C# 6 Features

6 We introduce key new C# 6 language features throughout the book (Fig. 1)—each defining occurrence is marked with a “6” margin icon as shown next to this paragraph.

C# 6 new language feature	First introduced in
string interpolation	Section 3.5
expression bodied methods and get accessors	Section 7.15
auto-implemented property initializers	Section 8.6.1
getter-only auto-implemented properties	Section 8.6.1
nameof operator	Section 10.5.1
null-conditional operator (?.)	Section 13.9.1
when clause for exception filtering	Section 13.10
using static directive	Section 21.3.1
null conditional operator (?[])	Section 21.6
collection initializers for any collection with an Add extension method	Section 21.7
index initializers	Section 21.7

Fig. 1 | C# 6 new language features.

## Interesting, Entertaining and Challenging Exercises

The book contains hundreds of exercises to practice the skills you learn. Extensive self-review exercises and answers are included for self-study. Also, each chapter concludes with a substantial set of exercises, which generally include

- simple recall of important terminology and concepts,
- identifying the errors in code samples,
- writing individual program statements,
- writing methods to perform specific tasks,
- writing C# classes,
- writing complete programs and
- implementing major projects.

Figure 2 lists only a sample of the book’s hundreds of exercises, including selections from our *Making-a-Difference* exercises set, which encourage you to use computers and the Internet to research and work on significant social problems. We hope you’ll approach these exercises in the context of your own values, politics and beliefs. The solutions to most of the book’s exercises are available only to college instructors who have adopted the book for their courses. See “Instructor Supplements” later in this Preface.

## A sampling of the book's exercises

Carbon Footprint Calculator	Calculating the Value of $\log(x)$	ComplexNumber Class
Body-Mass-Index Calculator	Pythagorean Triples	Vehicle Inheritance Hierarchy
Attributes of Hybrid Vehicles	Global Warming Facts Quiz	Payroll System
Gender Neutrality	Tax Plan Alternative: The “FairTax”	Accounts Payable System
Stopwatch GUI	Rounding to a Specific Decimal Place	Polymorphic Banking Program
Login Form GUI	Hypotenuse of a Right Triangle	CarbonFootprint Interface: Polymorphism
Calculator GUI	Displaying a Hollow Right- Isosceles Triangle of Any Character	Length Conversions
Alarm Clock GUI	Separating Digits	Painter
Radio GUI	Temperature Conversions	Guess the Number Game
Displaying Shapes	Amicable Numbers	Ecofont
Odd or Even?	Prime Numbers	Typing Tutor
Is a Number a Multiple of Another?	Reversing Digits	Restaurant Bill Calculator
Separating an Integer's Digits	Letter Grades to a Four-Point Scale	Story Writer
Multiplication Table of a Number	Coin Tossing	Pig Latin
Account Class	Guess-the-Number Game	Cooking with Healthier Ingredients
Student Record Class	Distance Between Two Points	Spam Elimination
Asset Class	Craps Game with Betting	SMS Language
Coaching Class	Towers of Hanoi	File of Product Details
Removing Duplicated Code	Computer-Assisted Instruction	Telephone-Number Words
Target-Heart-Rate Calculator	Wages Rate	Student Poll
Computerizing Health Records	Identifying Multiples of Eight	Phishing Scanner
Inventory Level Calculator	Dice Game of Craps	Bucket Sort
Sales-Commission Calculator	Airline Reservations System	Palindromes
Discount Calculator	Knight's Tour Chess Puzzle	Evaluating Expressions with a Stack
Find the Two Largest Numbers	Eight Queens Chess Puzzle	<b>Building Your Own Compiler</b>
Dangling-else Problem	Sieve of Eratosthenes	Generic Linear Search
Expanded Form	Tortoise and the Hare	SortedDictionary of Colors
Decimal Equivalent of a Binary Number	Merging Arrays	Prime Factorization
Type of Parallelogram	<b>Building Your Own Computer (Virtual Machine)</b>	Bucket Sort with LinkedList<int>
Permutations	Polling	Sieve of Eratosthenes with BitArray
Infinite Series: Mathematical Constant $e$	Querying an Array of Invoice Objects	Credit-Inquiry Program
World Population Growth	Name Connector	Rolling a Die 60,000,000 Times
Enforcing Privacy with Cryptography	Hemisphere Class	Baseball Database App
Bar Chart Display	Depreciating-Value Class	Parsing with LINQ to XML
Prime Numbers	Set of Integers	I/O-Bound vs. Compute- Bound Apps
Calculating Sales	RationalNumber Class	Recursive Fibonacci
Car-Pool Savings Calculator	HugeInteger Class	
Gas Mileage Calculator	Tic-Tac-Toe	

Fig. 2 | A sampling of the book's exercises.

## A Tour of the Book

This section discusses the book's modular organization to help instructors plan their syllabi.

### *Introduction to Computing, Visual C# and Visual Studio 2015 Community Edition*

The chapters in this module of the book

- Chapter 1, Introduction to Computers, the Internet and Visual C#
- Chapter 2, Introduction to Visual Studio and Visual Programming

introduce hardware and software fundamentals, Microsoft's .NET platform and Visual Programming. The vast majority of the book's examples will run on Windows 7, 8 and 10 using the *Visual Studio 2015 Community* edition with which we test-drive a fun **Painter** app in Section 1.12. Chapter 1's introduction to object-oriented programming defines key terminology and discusses important concepts on which the rest of the book depends.

### *Introduction to C# Fundamentals*

The chapters in this module of the book

- Chapter 3, Introduction to C# App Programming
- Chapter 4, Introduction to Classes, Objects, Methods and Strings
- Chapter 5, Algorithm Development and Control Statements: Part 1
- Chapter 6, Control Statements: Part 2
- Chapter 7, Methods: A Deeper Look
- Chapter 8, Arrays; Introduction to Exception Handling

present rich coverage of C# programming fundamentals (data types, operators, control statements, methods and arrays) and introduce object-oriented programming through a series of case studies. These chapters should be covered in order. Chapters 5 and 6 present a friendly treatment of control statements and problem solving. Chapters 7 and 8 present rich treatments of methods and arrays, respectively. Chapter 8 briefly introduces exception handling with an example that demonstrates attempting to access an element outside an array's bounds.

### *Object-Oriented Programming: A Deeper Look*

The chapters in this module of the book

- Chapter 9, Introduction to LINQ and the `List` Collection
- Chapter 10, Classes and Objects: A Deeper Look
- Chapter 11, Object-Oriented Programming: Inheritance
- Chapter 12, OOP: Polymorphism and Interfaces
- Chapter 13, Exception Handling: A Deeper Look

provide a deeper look at object-oriented programming, including classes, objects, inheritance, polymorphism, interfaces and exception handling. An optional online two-chapter case study on designing and implementing the object-oriented software for a simple ATM is described later in this preface.

Chapter 9 introduces Microsoft's Language Integrated Query (LINQ) technology, which provides a uniform syntax for manipulating data from various data sources, such as

arrays, collections and, as you'll see in later chapters, databases and XML. Chapter 9 is intentionally simple and brief to encourage instructors to begin covering LINQ technology early. Section 9.4 introduces the `List` collection, which we use in Chapter 12. Later in the book, we take a deeper look at LINQ, using LINQ to Entities (for querying databases) and LINQ to XML. Chapter 9's LINQ coverage can be deferred if you're in a course which either skips LINQ or defers coverage until later in the book—it's required for one example in Chapter 17 (Fig. 17.6) and many of the later chapters starting with Chapter 22, Databases and LINQ.

### *Windows Forms Graphical User Interfaces (GUIs)*

The chapters in this module of the book

- Chapter 14, Graphical User Interfaces with Windows Forms: Part 1
- Chapter 15, Graphical User Interfaces with Windows Forms: Part 2

present a detailed introduction to building GUIs using Windows Forms—instructors teaching Visual C# still largely prefer Windows Forms for their classes. Many of the examples in GUI Chapters 14–15 can be presented after Chapter 4. We also use Windows Forms GUIs in several other print and online chapters.

There are two other GUI technologies in Windows—Windows Presentation Foundation (WPF) and Universal Windows Platform (UWP). We provide optional online treatments of both.<sup>3</sup>

### *Strings and Files*

The chapters in this module of the book

- Chapter 16, Strings and Characters: A Deeper Look
- Chapter 17, Files and Streams

present string processing and file processing, respectively. We introduce strings beginning in Chapter 4 and use them throughout the book. Chapter 16 investigates strings in more detail. Most of Chapter 16's examples can be presented at any point after Chapter 4. Chapter 17 introduces text-file processing and object-serialization for inputting and outputting entire objects. Chapter 17 requires Windows Forms concepts presented in Chapter 14.

### *Searching, Sorting and Generic Data Structures*

The chapters in this module of the book:

- Chapter 18, Searching and Sorting
- Chapter 19, Custom Linked Data Structures
- Chapter 20, Generics
- Chapter 21, Generic Collections; Functional Programming with LINQ/PLINQ

introduce searching, sorting and data structures. Most C# programmers should use .NET's *built-in* searching, sorting and generic collections (prepackaged data structures) capabilities, which are discussed in Chapter 21. For instructors who wish to present how to implement customized searching, sorting and data structures capabilities, we provide Chapters 18–20, which require the concepts presented in Chapters 3–8 and 10–13. Chapter 18 presents sev-

---

3. As of Summer 2016, Windows Forms, WPF and UWP apps all can be posted for distribution via the Windows Store. See <http://bit.ly/DesktopToUWP> for more information.

eral searching and sorting algorithms and uses Big O notation to help you compare how hard each algorithm works to do its job—the code examples use especially visual outputs to show how the algorithms work. In Chapter 19, we show how to implement your own custom data structures, including lists, stacks, queues and binary trees. The data structures in Chapter 19 store references to objects. Chapter 20 introduces C# generics and demonstrates how to create type-safe generic methods and a type-safe generic stack data structure.

### *Functional Programming with LINQ, PLINQ, Lambdas, Delegates and Immutability*

In addition to generic collections, Chapter 21 now introduces functional programming, showing how to use it with LINQ to Objects to write code more concisely and with fewer bugs than programs written using previous techniques. In Section 21.12, with one additional method call, we'll demonstrate how PLINQ (Parallel LINQ) can improve LINQ to Objects performance substantially on multicore systems. The chapter's exercises also ask you to reimplement earlier examples using functional-programming techniques.

### *Database with LINQ to Entities and SQL Server*

The chapter in this module

- Chapter 22, Databases and LINQ

presents a novice-friendly introduction to database programming with the ADO.NET Entity Framework, LINQ to Entities and Microsoft's free version of SQL Server that's installed with the Visual Studio 2015 Community edition. The chapter's examples require C#, object-oriented programming and Windows Forms concepts presented in Chapters 3–14. Several online chapters require the techniques presented in this chapter.

### *Asynchronous Programming*

The chapter in this module

- Chapter 23, Asynchronous Programming with `async` and `await`

shows how to take advantage of multicore architectures by writing applications that can process tasks asynchronously, which can improve app performance and GUI responsiveness in apps with long-running or compute-intensive tasks. The `async` modifier and `await` operator greatly simplify asynchronous programming, reduce errors and enable your apps to take advantage of the processing power in today's multicore computers, smartphones and tablets. In this edition, we added a case study that uses the Task Parallel Library (TPL), `async` and `await` in a GUI app—we keep a progress bar moving along in the GUI thread in parallel with a lengthy, compute-intensive calculation in another thread.

## A Tour of the Online Content

The printed book contains the core content (Chapters 1–23) for introductory and intermediate course sequences. Several optional online topics for advanced courses and professionals are available on the book's password-protected Companion Website

<http://www.pearsonglobal editions.com/deitel>

New copies of this book come with a Companion Website access code that's located on the book's inside front cover. If the access code is already visible or there isn't an access code, you purchased a used book or an edition that does not come with an access code. Figure 3 lists the online topics, and Figure 4 lists a sample of the associated exercises.

### Online topics

Web App Development with ASP.NET  
 XML and LINQ to XML  
 Universal Windows Platform (UWP) GUI, Graphics, Multimedia and XAML  
 REST Web Services  
 Cloud Computing with Microsoft Azure™  
 Using the Visual Studio Debugger  
 (Optional) Windows Presentation Foundation (WPF) GUI, Graphics, Multimedia and XAML  
 (Optional) ATM Case Study, Part 1: Object-Oriented Design with the UML  
 (Optional) ATM Case Study, Part 2: Implementing an OO Design in C#

**Fig. 3** | Online topics on the *Visual C# How to Program, 6/e* Companion Website.

### A sampling of the online chapters' exercises

Guestbook App	College Loan Payoff	Favorite Flickr Searches App
Web-Based Address Book	Calculator App	Flag Quiz App
Enhanced Painter App	Car Payment Calculator App	Phone Book App with Data
PhotoViewer App	Mileage Calculator App	Binding
Data Bindings to LINQ queries	Body Mass Index Calculator	Enhanced UsingGradients App
Snake PolyLine App	App	Enhanced DrawStars app
Drawing App	Target-Heart-Rate Calculator	Image Reflector App
Enhanced Tip Calculator App	App	Accessibility: Speech-
Mortgage Calculator App	Phone-Book Web Service	Controlled Drawing App

**Fig. 4** | A sampling of the online chapters' exercises.

#### *Web App Development with ASP.NET*

Microsoft's .NET server-side technology, ASP.NET, enables you to create robust, scalable web-based apps. You'll build several apps, including a web-based guestbook that uses ASP.NET and the ADO .NET Entity Framework to store data in a database and display data in a web page.

#### *Extensible Markup Language (XML)*

The Extensible Markup Language (XML) is pervasive in the software-development industry, e-business and throughout the .NET platform. It's used in most of this book's online topics. XML is required to understand XAML—a Microsoft XML vocabulary that's used to describe graphical user interfaces, graphics and multimedia for Universal Windows Platform (UWP) GUI, graphics and multimedia apps, Windows 10 Mobile apps and Windows Presentation Foundation (WPF) apps. We present XML fundamentals, then discuss LINQ to XML, which allows you to query XML content using LINQ syntax.

#### *Universal Windows Platform (UWP) for Desktop and Mobile Apps*

The Universal Windows Platform (UWP) is designed to provide a common platform and user experience across all Windows devices, including personal computers, smartphones,