LEARNING MADE EASY

2nd Edition

Beginning Programming ALL-IN-ONE

for dummies®
A Wiley Brand

7 Books in one!

Wallace Wang

Created his first computer game written in BASIC using a teletype printer

Beginning Programming

ALL-IN-ONE

2nd Edition

by Wallace Wang

for dummies®
A Wiley Brand

Beginning Programming All-in-One For Dummies®, 2nd Edition

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Copyright © 2022 by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

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, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at http://www.wiley.com/go/permissions.

Trademarks: Wiley, For Dummies, the Dummies Man logo, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: WHILE THE PUBLISHER AND AUTHORS HAVE USED THEIR BEST EFFORTS IN PREPARING THIS WORK, THEY MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES REPRESENTATIVES, WRITTEN SALES MATERIALS OR PROMOTIONAL STATEMENTS FOR THIS WORK. THE FACT THAT AN ORGANIZATION, WEBSITE, OR PRODUCT IS REFERRED TO IN THIS WORK AS A CITATION AND/OR POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE PUBLISHER AND AUTHORS ENDORSE THE INFORMATION OR SERVICES THE ORGANIZATION, WEBSITE, OR PRODUCT MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING PROFESSIONAL SERVICES. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR YOUR SITUATION. YOU SHOULD CONSULT WITH A SPECIALIST WHERE APPROPRIATE. FURTHER, READERS SHOULD BE AWARE THAT WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ. NEITHER THE PUBLISHER NOR AUTHORS SHALL BE LIABLE FOR ANY LOSS OF PROFIT OR ANY OTHER COMMERCIAL DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR OTHER DAMAGES.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit https://hub.wiley.com/community/support/dummies.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at http://booksupport.wiley.com. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2022936302

ISBN 978-1-119-88440-8 (pbk); ISBN 978-1-119-88441-5 (ebk); ISBN 978-1-119-88442-2 (ebk)

Contents at a Glance

Introduction 1

Book 1: Getting Started with Programming 5

CHAPTER 1: Getting Started Programming a Computer 7
CHAPTER 2: Different Methods for Writing Programs 29
CHAPTER 3: Types of Programming Languages 53
CHAPTER 4: Programming Tools 77

Book 2: Programming Basics 101

CHAPTER 1: How Programs Work 103
CHAPTER 2: Variables, Data Types, and Constants 119
CHAPTER 3: Manipulating Data 141
CHAPTER 4: Making Decisions by Branching 163
CHAPTER 5: Repeating Commands by Looping 181
CHAPTER 6: Breaking a Large Program into Subprograms 197
CHAPTER 7: Breaking a Large Program into Objects 215
CHAPTER 8: Reading and Saving Files 243
CHAPTER 9: Documenting Your Program 263
CHAPTER 10: Principles of User Interface Design 277
CHAPTER 11: Debugging and Testing 297

Book 3: Data Structures 311

CHAPTER 1: Structures and Arrays 313
CHAPTER 2: Sets and Linked Lists 333
CHAPTER 3: Collections and Dictionaries 351
CHAPTER 4: Stacks, Queues, and Deques 367
CHAPTER 5: Graphs and Trees 381

Book 4: Algorithms 397

CHAPTER 1: Sorting Algorithms 399
CHAPTER 2: Searching Algorithms 415
CHAPTER 3: String Searching 429
CHAPTER 4: Data Compression Algorithms 441
CHAPTER 5: Encryption Algorithms 451

Book 5: Web Programming . . . 469

CHAPTER 1: HyperText Markup Language . . . 471
CHAPTER 2: CSS . . . 487
CHAPTER 3: JavaScript . . . 495
CHAPTER 4: PHP . . . 509
CHAPTER 5: Ruby . . . 523

Book 6: Programming Language Syntax . . . 537

CHAPTER 1: C and C++ . . . 539
CHAPTER 2: Java and C# . . . 557
CHAPTER 3: Perl and Python . . . 577
CHAPTER 4: Kotlin . . . 595
CHAPTER 5: Swift and SwiftUI . . . 613
CHAPTER 6: Flutter and Dart . . . 637

Book 7: Applications . . . 657

CHAPTER 1: Database Management . . . 659
CHAPTER 2: Bioinformatics . . . 675
CHAPTER 3: Computer Security . . . 685
CHAPTER 4: Artificial Intelligence . . . 699
CHAPTER 5: Mobile and Wearable Computing . . . 711
CHAPTER 6: Game Engines . . . 721
CHAPTER 7: The Future of Computer Programming . . . 731

Index . . . 747

Table of Contents

INTRODUCTION 1
About This Book. 1
Foolish Assumptions 2
Icons Used in This Book 2
Beyond the Book 3
Where to Go from Here 3

BOOK 1: GETTING STARTED WITH PROGRAMMING 5

CHAPTER 1: Getting Started Programming a Computer 7
How Computer Programming Works 8
Identifying the problem 8
Defining the steps 9
The History of Computer Programming 10
Talking to a processor in machine language 11
Using assembly language as a shortcut to machine language 12
Hiding the details of a computer with a high-level language 15
Combining the best of both worlds with the C programming language. 15
Weighing the pros and cons of programming languages 16
Figuring Out Programming 18
Desire beats technical training every time 19
Picking a computer and an operating system 19
Writing programs with an editor 21
Converting source code with an assembler or compiler 23
Translating source code with an interpreter 25
Combining a compiler with an interpreter to create p-code 25
Taking the time to understand 26

CHAPTER 2: Different Methods for Writing Programs. 29
Spaghetti Programming 31
Structured Programming 34
The three parts of structured programming 34
Top-down programming 36
Event-Driven Programming 38
Designing a user interface 41
Writing event handlers 42
Writing your program 44
Object-Oriented Programming 44
Isolating data 46
Simplifying modifications 47

Using Protocol-Oriented Programming 49
Design Patterns 50

CHAPTER 3: **Types of Programming Languages** 53
Your First Language 54
BASICally disrespected 55
Visual programming with Scratch 56
Programming robots with LEGO Mindstorms 57
Learning object-oriented programming with Alice 57
Programming a killer robot 58
Curly-Bracket Languages 60
Learning programming with C 60
Adding object-oriented programming with C++ 62
Gaining true portability with Java 63
Programming more safely with C# 64
Choosing a curly-bracket language 66
Artificial Intelligence Languages 67
Scripting Languages 71
Automating a program 72
Customizing a program 73
Transferring data among multiple programs 74
Creating stand-alone programs 74
Database Programming Languages 75
Comparing Programming Languages 76

CHAPTER 4: **Programming Tools** 77
Choosing a Compiler 78
Defining your needs for a compiler 79
Evaluating the technical features of a compiler 80
Finding an Interpreter 84
Compiling to a Virtual Machine 86
Writing a Program with an Editor 88
Stand-alone editors 88
Integrated development environments 90
Fixing a Program with a Debugger 91
Stepping line-by-line 92
Watching variables 95
Saving Time with Third-Party Components 96
Optimizing a Program with a Profiler 97
Managing Source Code 97
Creating a Help File 99
Installing a Program 99
Dissecting Programs with a Disassembler 99

BOOK 2: PROGRAMMING BASICS 101

CHAPTER 1: **How Programs Work** 103

Using Keywords as Building Blocks. 105
Organizing a Program. 108
Dividing a Program into Subprograms. 109
Dividing a Program into Objects 113
Creating a User Interface 115

CHAPTER 2: **Variables, Data Types, and Constants** 119

Declaring Variables 120
Variable naming conventions 121
Creating variables in a command 122
Declaring the data type of a variable 124
Using Different Data Types 126
Storing Data in a Variable. 130
Retrieving Data from a Variable 132
Using Constant Values 133
Defining the Scope of a Variable. 135
Handling global variables with care 136
Restricting scope to a module. 137
Isolating variables in a subprogram 138
Passing data among subprograms. 139

CHAPTER 3: **Manipulating Data** 141

Storing Data with the Assignment Operator. .142
Using Math to Manipulate Numbers 143
Organizing equations with operator precedence. 144
Using built-in math functions 146
Manipulating Strings. 147
Finding Strings with Regular Expressions 148
Pattern-matching with the single character (.) wildcard 149
Pattern-matching for specific characters. .149
Pattern-matching with the multiple-character (*) and plus (+) wildcards. 150
Pattern-matching with ranges. 151
Using Comparison Operators 153
Using Boolean Operators. 156
Using the Not operator. 156
Using the And operator 157
Using the Or operator. 158
Using the Xor operator 159
Converting Data Types 160

CHAPTER 4: **Making Decisions by Branching** 163
Picking One Choice with the IF-THEN Statement 164
Picking Two Choices with the IF-THEN-ELSE Statement 166
Picking Three or More Choices with the IF-THEN-ELSEIF Statement 168
Checking a condition for each set of commands 168
Offering three or more choices 170
Playing with Multiple Boolean Operators 171
Making Multiple Choices with the SELECT CASE Statement 174
Matching multiple values in a SELECT CASE statement 177
Checking a range of values 178
Comparing values 179

CHAPTER 5: **Repeating Commands by Looping** 181
Looping a Fixed Number of Times with the FOR-NEXT Loop 182
Using a FOR-NEXT loop variable 183
Counting by a different range 185
Counting by different increments 186
Counting backward 187
Counting over arrays and other items 188
Looping Zero or More Times with the WHILE Loop 189
Looping at Least Once with the DO Loop 192
Playing with Nested Loops 193
Prematurely Exiting from a Loop 195
Checking Your Loops 195

CHAPTER 6: **Breaking a Large Program into Subprograms** 197
Creating and Using Subprograms 199
Creating a subprogram 200
“Calling” a subprogram 201
Passing Parameters 203
Passing parameters by reference 206
Storing values in a subprogram name 210
Repeating a Subprogram with Recursion 212

CHAPTER 7: **Breaking a Large Program into Objects** 215
How Object-Oriented Programming Works 216
Encapsulation Isolates Data and Subprograms 219
Shielding data inside an object 221
Grouping methods inside of an object 221
Protecting code from other programmers 222
Sharing Code with Inheritance 223
Polymorphism: Modifying Code without Changing Its Name 226
Design Patterns 228

Object-Oriented Languages 230
Hybrid languages 231
Pure languages 231
Disadvantages of object-oriented programming 232
Real-Life Programming Examples 233
Defining an object with a class 233
Creating an object from a class 236
Running methods stored in an object 236
Inheriting an object 238
Using method overloading to rewrite an inherited subprogram 239

CHAPTER 8: **Reading and Saving Files** 243
Storing Data in Text Files 243
Creating a text file 246
Reading a text file 247
Storing Fixed-Size Data in Random-Access Files 250
Writing data 251
Reading data 252
Storing Varying-Size Data in Untyped Files 253
Writing data 254
Reading data 255
Using Database Files 257
Looking at the structure of a database 257
Connecting to a database 259

CHAPTER 9: **Documenting Your Program** 263
Adding Comments to Source Code 264
Identifying the two types of comments 265
Describing code and algorithms 268
Documentation 270
Debugging 271
Writing Software Documentation 272
Documentation types 272
Documentation tools 274
Help files 275

CHAPTER 10: **Principles of User Interface Design** 277
The Evolution of User Interfaces 278
Command-line interface 278
Menus 278
Graphical user interface 280
Elements of a User Interface 281
Displaying commands to a user interface 281
Giving data to the user interface 284

Showing information back to the user 288
Organizing a user interface 290
Designing a User Interface 291
Know the user 291
Hide/disable unusable options 292
Tolerate mistakes 293
Be consistent 294
Give the user freedom to customize the user interface 295
Make navigation easy 295

CHAPTER 11: **Debugging and Testing** 297
Common Types of Programming Errors 297
Debugging with Comments and Print Statements 300
Breakpoints, Stepping, and Watching 302
Stepping through code 304
Watching variables 305
Testing Code. 306
Unit tests. 307
Integration tests. 308
User interface testing 309

BOOK 3: DATA STRUCTURES 311

CHAPTER 1: **Structures and Arrays** 313
Using Structures 314
Storing data 315
Retrieving data. 315
Using an Array 316
Defining the size 317
Storing data 320
Retrieving data. 321
Working with Resizable Arrays 321
BASIC 322
C#. 323
Swift. 323
Working with Multidimensional Arrays 323
Creating a multidimensional array 324
Storing and retrieving data 325
Using Structures with Arrays 325
Drawbacks of Arrays 327
Data types. 328
Searching and sorting. 328
Adding and deleting 329
Identifying the location of data in an array 330

CHAPTER 2: **Sets and Linked Lists** 333
Using Sets 334
Adding and deleting data in a set 335
Checking for membership 336
Avoiding duplicate data 337
Manipulating two sets 337
Using Linked Lists 342
Creating a linked list 343
Modifying a linked list 344
Creating a double linked list 345
Drawbacks of Sets and Linked Lists 346
Problems with pointers 347
Problems with accessing data 347

CHAPTER 3: **Collections and Dictionaries** 351
Using a Collection 352
Adding data to a collection 352
Deleting data from a collection 354
Identifying data with keys 355
Searching and retrieving data 356
Using Dictionaries 358
Adding data to a dictionary 358
Searching and retrieving data from a dictionary 359
Understanding Hash Tables 360
Converting keys with a hash function 360
Hash function collisions 362

CHAPTER 4: **Stacks, Queues, and Deques** 367
Using Stacks 368
Adding data to a stack 369
Removing data from a stack 370
Counting and searching a stack 371
Using Queues 372
Adding data to a queue 373
Removing data from a queue 374
Counting and searching a queue 375
Using Deques 376

CHAPTER 5: **Graphs and Trees** 381
Understanding Graphs 383
Types of graphs 384
Uses for graphs 385
Creating Trees 386
Ordered trees 387

Binary trees 388
B-trees 389
Taking Action on Trees 390
Traversing a tree to search for data 390
Adding new data 392
Deleting data 393
Pruning and grafting sub-trees 394

BOOK 4: ALGORITHMS 397

CHAPTER 1: **Sorting Algorithms** 399
Using Bubble Sort 400
Using Selection Sort 402
Using Insertion Sort 403
Using Shell Sort 405
Using Heap Sort 406
Using Merge Sort 410
Using Quick Sort 411
Comparing Sorting Algorithms 412

CHAPTER 2: **Searching Algorithms** 415
Sequential Search 416
Backward or forward searching 417
Block searching 418
Binary searching 419
Interpolation searching 420
Using Indexes 422
Creating an index 422
Clustered and unclustered indexes 423
Problems with indexes 424
Adversarial Search 424
Depth versus time 426
Alpha-beta pruning 426
Looking up a library of good moves 427

CHAPTER 3: **String Searching** 429
Sequential Text Search 430
The Boyer-Moore algorithm 431
The Rabin–Karp algorithm 431
The Shift Or algorithm 433
The finite automaton algorithm 435
Searching with Regular Expressions 436
Searching for single character patterns 436
Searching for multiple character patterns 437
Searching for alternate patterns 438
Searching Phonetically 438

CHAPTER 4: **Data Compression Algorithms** 441
Lossless Data Compression Algorithms 442
Run-length encoding 442
The Burrows–Wheeler transform algorithm 442
Dictionary encoding 445
Lossy Data Compression 449

CHAPTER 5: **Encryption Algorithms** 451
How Encryption Works 451
The Basics of Encryption 453
Stream ciphers 456
Block ciphers 457
Symmetric/Asymmetric Encryption Algorithms 459
Cracking Encryption 463
Brute-force attacks 463
Dictionary attacks 464
Plaintext and ciphertext attacks 466

BOOK 5: WEB PROGRAMMING 469

CHAPTER 1: **HyperText Markup Language** 471
The Structure of an HTML Document 472
Creating a title 472
Creating the body text 472
Aligning text 474
Emphasizing text 475
Adding color 476
Changing the font size 477
Adding comments 478
Adding Graphics 478
Defining the Background 479
Creating Hyperlinks 480
Defining an anchor point 480
Linking to an anchor point 480
Making Tables 481
Defining a table 481
Defining a table heading 482
Creating table rows and data 483
Displaying a table caption, header, and footer 484

CHAPTER 2: **CSS** 487
The Structure of a Stylesheet 488
Creating Style Classes 489
Separating Styles in Files 491
Cascading Stylesheets 493

CHAPTER 3: **JavaScript** 495
The Structure of a JavaScript Program 496
Creating Comments 497
Declaring Variables 498
Using Operators 498
Increment and decrement operators 500
Assignment operators 501
Branching Statements 501
Looping Statements 503
Creating Functions 504
Using Arrays 505
Designing User Interfaces 505
Creating dialog boxes 505
Creating windows 507

CHAPTER 4: **PHP** 509
Examining the Structure of a PHP Program 510
Creating Comments 510
Declaring Variables 511
Using Operators 512
Increment and decrement operators 513
Assignment operators 515
Branching Statements 515
Looping Statements 517
Creating Functions 518
Using Arrays 519
Creating Objects 520

CHAPTER 5: **Ruby** 523
The Structure of a Ruby Program 524
Creating Comments 524
Declaring Variables 525
Using Operators 526
Branching Statements 528
Looping Statements 530
Creating Functions 531
Using Data Structures 532
Creating Objects 534

BOOK 6: PROGRAMMING LANGUAGE SYNTAX 537

CHAPTER 1: **C and C++** 539
Looking at the Structure of a C/C++ Program 540
Creating Comments 541

Declaring Variables 542
  Declaring string data types 542
  Declaring integer data types 543
  Declaring floating-point data types. 544
  Declaring Boolean values 545
Using Operators 545
  Increment and decrement operators 546
  Assignment operators 548
Branching Statements 548
Looping Statements 550
Creating Functions 551
Data Structures 553
  Creating a structure 553
  Creating enumerations 554
  Creating an array 554
Using Objects 555

CHAPTER 2: **Java and C#** 557
Looking at the Structure of a Java/C# Program 558
Creating Comments 559
Declaring Variables 559
  Declaring string data types 560
  Declaring integer data types 560
  Declaring floating-point data types. 561
  Declaring Boolean variables 562
Using Operators 562
  Increment and decrement operators 564
  Assignment operators 564
Branching Statements 565
Looping Statements 568
Creating Functions 569
Data Structures 571
  Creating a C# structure. 571
  Creating an array 572
  Creating a Java linked list 573
  Creating C# data structures 574
Using Objects 574

CHAPTER 3: **Perl and Python** 577
Reviewing the Structure of a Perl or Python Program 578
Creating Comments 579
Defining Variables 580
Using Operators 580
  Increment and decrement operators 582
  Assignment operators 583

Branching Statements 584
Looping Statements 586
Creating Functions 588
Making Data Structures 588
    Perl data structures. 589
    Python data structures 590
Using Objects 592

CHAPTER 4: **Kotlin** 595
Looking at the Structure of a Kotlin Program 596
Creating Comments 596
Declaring Variables 597
    Declaring string data types 597
    Declaring integer data types 598
    Declaring floating-point data types. 599
    Declaring Boolean values 599
Declaring Constants 600
Using Operators 600
Branching Statements 601
Looping Statements 605
Creating Functions 606
Creating Data Structures 608
    Creating a list 609
    Creating an array 610
    Creating a set 610
Creating Objects 611

CHAPTER 5: **Swift and SwiftUI** 613
Considering the Structure of a Swift Program 614
    Understanding SwiftUI 614
    Creating a SwiftUI user interface. 616
    Understanding SwiftUI state variables 616
Creating Comments 618
Declaring Variables 618
    Declaring string data types 619
    Declaring integer data types 620
    Declaring decimal data types. 620
    Declaring Boolean values 621
Declaring Constants 621
Using Operators 622
Branching Statements 623
Looping Statements 627
Creating Functions 628

Data Structures 630
Creating an array 630
Creating a dictionary 632
Creating a set 633
Creating Objects 634

CHAPTER 6: **Flutter and Dart** 637
Working with Flutter 638
Understanding the structure of a Flutter program 638
Working with widgets in Flutter 639
Aligning widgets in rows and columns 642
Understanding the Dart Language 645
Creating comments 645
Declaring variables 646
Using operators 646
Using branching statements 649
Using looping statements 650
Creating functions 652
Creating data structures 653
Using objects 656

BOOK 7: APPLICATIONS 657

CHAPTER 1: **Database Management** 659
Understanding the Basics of Databases 659
Free-form databases 660
Flat-file databases 661
Relational databases 663
Manipulating Data 667
Writing database commands 669
The SQL language 670
Data integrity 671
Data mining 672
Database Programming 672

CHAPTER 2: **Bioinformatics** 675
The Basics of Bioinformatics 676
Representing molecules 676
Manipulating molecules in a computer 677
Database Searches 679
Bioinformatics Programming 681

CHAPTER 3: **Computer Security** 685
Stopping Malware 686
Viruses. 687
Worms 687
Trojan horses 688
Spyware. 689
Distributed denial-of-service attacks 690
Stopping Hackers. 691
Intrusion detection systems. 692
Rootkit detectors 693
Forensics 694
Secure Computing. 695
Patching as an afterthought 695
Security in coding 696
Security by design 697

CHAPTER 4: **Artificial Intelligence** 699
Problem Solving 700
Game-playing 701
Natural language processing 702
Speech recognition 704
Image recognition 705
Machine Learning 706
Bayesian probability 707
Neural networks 707
Applications of Artificial Intelligence. 710

CHAPTER 5: **Mobile and Wearable Computing** 711
Understanding the Different Generations of Computing 712
Giving Data to the User 714
Getting Data from the User 716
Tracking Motion and Location 717
Tracking Real-Time Health Data 718
Looking to the Future of Augmented Reality and Wearable Computers 718

CHAPTER 6: **Game Engines** 721
Understanding Game Engines. 722
Picking a Game Engine 723
Programming a Game Engine 724
Exploring the Future Uses of Game Engines 726
Filmmaking 726
Architecture and engineering simulations. 727
Marketing and advertising 728

CHAPTER 7: **The Future of Computer Programming** 731
Picking a Programming Language 732
Picking an Operating System 733
Doing Cross-Platform Programming 735
The portability of C 735
Cross-platform languages 736
Virtual machines 737
Software as a service. 738
Data science 739
Website programming 740
Macro programming 740
Robotics programming 741
Blockchain programming 742
Defining Your Future in Programming 743
Teaching yourself other languages 743
Combining other interests besides programming 744
Getting experience and knowledge 744

INDEX 747

Introduction

f you enjoy using a computer, you may have even more fun learning to control a computer by writing your own programs. To learn how to program a computer, you need to:

- **Understand that computer programming is nothing more than problem solving.** Before you even think about writing a program, you need to know what problem you want your program to solve and how it will solve it.
- **Learn the basic ideas behind computer programming that work with all programming languages on any computer.** Although programming a Windows computer is different from programming a Mac, a smartphone, a smart watch, or a super computer, the general principles remain the same. By learning what these common programming principles are and why they exist, you can learn different ways to tell a computer what to do, step-by-step.
- **Learn a specific programming language.** A programming language represents just one way to express your ideas in a language that the computer can understand. By combining your knowledge of a programming language with programming principles and the type of problem you want the computer to solve, you can create your own computer programs for fun or profit.

About This Book

If you have any interest in programming but don't know where to start, this book can give you a nudge in the right direction. You won't learn how to write programs in a specific programming language, but you'll learn the basics of computer programming so you'll have no trouble learning more on your own.

If you already know something about programming, this book can still help you learn more by introducing you to the variety of programming languages available and make it easy for you to pick up different programming languages quickly. The more you understand the advantages and disadvantages of different programming languages, the better you'll be able to choose the language that's best suited for a particular task.

Whether you're a novice or an intermediate programmer, you'll find this book can work as a tutorial to teach you more and as a reference to help refresh your memory on programming topics you may not normally use every day. This book won't turn you into an expert overnight, but it will open the doors to more information about programming than you may have ever known even existed.

This book is a reference — you don't need to read the chapters in order from front cover to back and you don't have to commit anything you read here to memory. Also, *sidebars* (text in gray boxes) and anything marked with the Technical Stuff icon are skippable.

Finally, within this book, you may note that some web addresses break across two lines of text. If you're reading this book in print and want to visit one of these web pages, simply key in the web address exactly as it's noted in the text, pretending as though the line break doesn't exist. If you're reading this as an e-book, you've got it easy — just click the web address to be taken directly to the web page.

Foolish Assumptions

When writing this book, I made two assumptions about you, the reader:

- You may have no experience in computer programming or a limited amount of experience, but you're eager to learn.
- You have a computer (whether it's the latest model on the market or simply an older model that still works). Ideally, your computer can connect to the Internet.

That's it! As long as you have a computer and the desire to learn, you have everything you need to learn computer programming.

Icons Used in This Book

Icons highlight important or useful information that you may want to know about. Here's a guide to the icons:

TIP

The Tip icon highlights information that can save you time or make it easier for you to do something.

REMEMBER

The Remember icon emphasizes information that's so important you should commit it to memory.

WARNING

Look out! The Warning icon highlights something dangerous that you need to avoid before making an irreversible mistake that could make you curse your computer forever.

TECHNICAL STUFF

The Technical Stuff icon highlights interesting technical information that you can safely ignore, but which may provide additional background about programming a computer.

Beyond the Book

In addition to what you're reading right now, this product also comes with a free access-anywhere Cheat Sheet that summarizes different types of programming principles, common ways to store and organize data, and lists of suggested software to use. To get this Cheat Sheet, simply go to `www.dummies.com` and type **Beginning Programming All-in-One For Dummies Cheat Sheet** in the Search box.

Where to Go from Here

You can use this book as a tutorial or a reference. Although you can just flip through this book to find the information you need, programming novices should start with Book 1 before tackling any other books. After you understand the basics of programming from Book 1, you can freely jump around to read only the information that interests you.

Programming is more than learning a particular programming language or even knowing how to program a particular type of computer. Basically, programming is about tackling difficult problems and breaking them down into smaller problems until you ultimately solve one much bigger problem. If you like the idea of solving problems, this may be the perfect book to introduce you to the wonderful world of computer programming!

1 Getting Started with Programming

Contents at a Glance

CHAPTER 1: **Getting Started Programming a Computer** 7

CHAPTER 2: **Different Methods for Writing Programs** 29

CHAPTER 3: **Types of Programming Languages** 53

CHAPTER 4: **Programming Tools** 77

IN THIS CHAPTER

» Understanding how computer programming works

» Diving into the history of computer programming

» Discovering programming

Chapter 1 Getting Started Programming a Computer

Believe it or not, if you can write a recipe on an index card, you can program a computer! At the simplest level, computer programming is nothing more than writing instructions for a computer to follow, step-by-step. The most important part of programming isn't knowing how to write a program or how to use a particular programming language, but knowing what to create in the first place.

Some of the most popular and useful computer programs were created by people who didn't have any formal training in math or computer science. Dan Bricklin invented the spreadsheet while studying for his MBA at Harvard. Scott Cook, who worked in marketing and product development at Procter & Gamble, created the popular money-management program Quicken after hearing his wife complain about the tedium of paying bills. Nineteen-year-old Shawn Fanning created Napster, the first peer-to-peer file-sharing network, after hearing a friend complain about the difficulty of finding his favorite songs on the Internet. Game developer Dona Bailey wanted to create a video game that would appeal to both men and women; as the only woman working at Atari's coin-op division, she created the video game Centipede, which became Atari's second best-selling coin-op game.

The point is that anyone can figure out how to program a computer. What's more important than knowing how to program a computer is knowing what to do with your programming skills. As Albert Einstein said, "Imagination is more important than knowledge." After you have an idea for a program, you can use programming to turn your idea into reality.

How Computer Programming Works

Computer programming is nothing more than problem solving. Every program is designed to solve a specific problem. The more universal the problem (calculating formulas in a spreadsheet, managing your money, searching for music files over the Internet, or keeping people amused playing a game creating virtual buildings), the more useful and popular the program will be.

Identifying the problem

Before you even touch a computer, identify the specific problem you want the computer to solve. For example, spreadsheets eliminate the tedium of writing and calculating formulas manually. Word processors make editing and formatting text fast and easy. Even video games solve the problem of challenging people with puzzles, obstacles, and battles.

Although the most popular programs solve universal problems, literally thousands of programs are designed to solve specific problems in niche markets, such as hotel reservation software, construction billing and invoice management programs, and dental office management programs. If you can identify a problem that a computer can solve or simplify, you have an idea for a computer program.

REMEMBER

You must know exactly what you want your program to do before you start designing and writing it. One of the most common reasons programs fail is because the program doesn't solve the right problem that people really need.