READINGS FROM Programming with C++

KYLA McMULLEN
ELIZABETH MATTHEWS
JUNE JAMRICH PARSONS

Programming with C++

KYLA McMULLEN
ELIZABETH MATTHEWS
JUNE JAMRICH PARSONS

Australia • Brazil • Canada • Mexico • Singapore • United Kingdom • United States

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

This is an electronic version of the print textbook. Due to electronic rights restrictions, some third party content may be suppressed. Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. The publisher reserves the right to remove content from this title at any time if subsequent rights restrictions require it. For valuable information on pricing, previous editions, changes to current editions, and alternate formats, please visit www.cengage.com/highered to search by ISBN#, author, title, or keyword for materials in your areas of interest.

Important Notice: Media content referenced within the product description or the product text may not be available in the eBook version.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

CENGAGE

Readings from Programming with C++
Kyla McMullen, Elizabeth Matthews, June Jamrich Parsons

SVP, Higher Education Product Management: Erin Joyner

VP, Product Management: Thais Alencar

Product Team Manager: Kristin McNary

Associate Product Manager: Tran Pham

Product Assistant: Tom Benedetto

Learning Designer: Mary Convertino

Senior Content Manager: Maria Garguilo

Digital Delivery Lead: David O'Connor

Technical Editor: John Freitas

Developmental Editor: Lisa Ruffolo

Vice President, Marketing – Science, Technology, & Math: Jason Sakos

Senior Director, Marketing: Michele McTighe

Marketing Manager: Cassie L. Cloutier

Marketing Development Manager: Samantha Best

Product Specialist: Mackenzie Paine

IP Analyst: Ashley Maynard

IP Project Manager: Cassidie Parker

Production Service: SPi Global

Designer: Erin Griffin

Cover Image Source: echo3005/ShutterStock.com

© 2022 Cengage Learning, Inc.
WCN: 02-300

Unless otherwise noted, all content is © Cengage.

ALL RIGHTS RESERVED. No part of this work covered by the copyright herein may be reproduced or distributed in any form or by any means, except as permitted by U.S. copyright law, without the prior written permission of the copyright owner.

For product information and technology assistance, contact us at
Cengage Customer & Sales Support, 1-800-354-9706
or support.cengage.com.

For permission to use material from this text or product, submit all requests online at **www.cengage.com/permissions.**

Library of Congress Control Number: 2020922802

ISBN: 978-0-357-63775-3

Cengage
200 Pier 4 Boulevard
Boston, MA 02210
USA

Cengage is a leading provider of customized learning solutions with employees residing in nearly 40 different countries and sales in more than 125 countries around the world. Find your local representative at **www.cengage.com.**

To learn more about Cengage platforms and services, register or access your online learning solution, or purchase materials for your course, visit **www.cengage.com.**

Notice to the Reader
Publisher does not warrant or guarantee any of the products described herein or perform any independent analysis in connection with any of the product information contained herein. Publisher does not assume, and expressly disclaims, any obligation to obtain and include information other than that provided to it by the manufacturer. The reader is expressly warned to consider and adopt all safety precautions that might be indicated by the activities described herein and to avoid all potential hazards. By following the instructions contained herein, the reader willingly assumes all risks in connection with such instructions. The publisher makes no representations or warranties of any kind, including but not limited to, the warranties of fitness for particular purpose or merchantability, nor are any such representations implied with respect to the material set forth herein, and the publisher takes no responsibility with respect to such material. The publisher shall not be liable for any special, consequential, or exemplary damages resulting, in whole or part, from the readers' use of, or reliance upon, this material.

Printed in Mexico
Print Number: 01 Print Year: 2020

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

BRIEF CONTENTS

PREFACE **XIII**

MODULE 1
Computational Thinking 1

MODULE 2
Programming Tools 15

MODULE 3
Literals, Variables, and Constants 35

MODULE 4
Numeric Data Types and Expressions 49

MODULE 5
Character and String Data Types 63

MODULE 6
Decision Control Structures 83

MODULE 7
Repetition Control Structures 103

MODULE 8
Arrays 125

MODULE 9
Functions 145

MODULE 10
Recursion 165

MODULE 11
Exceptions 185

MODULE 12
File Operations 205

MODULE 13
Classes and Objects 231

MODULE 14
Methods 245

MODULE 15
Encapsulation 271

MODULE 16
Inheritance 291

MODULE 17
Polymorphism 309

MODULE 18
Templates 319

MODULE 19
Linked List Data Structures 333

MODULE 20
Stacks and Queues 353

MODULE 21
Trees and Graphs 371

MODULE 22
Algorithm Complexity and Big-O Notation 395

MODULE 23
Search Algorithms 411

MODULE 24
Sorting Algorithms 427

MODULE 25
Processor Architecture 455

MODULE 26
Data Representation 469

MODULE 27
Programming Paradigms 491

MODULE 28
User Interfaces 507

MODULE 29
Software Development Methodologies 525

MODULE 30
Pseudocode, Flowcharts, and Decision Tables 541

MODULE 31
Unified Modeling Language 557

GLOSSARY **569**
INDEX **583**

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

TABLE OF CONTENTS

PREFACE XIII

MODULE 1
COMPUTATIONAL THINKING 1

Algorithms 2
- Algorithm Basics 2
- Programming Algorithms 2
- "Good" Algorithms 3
- Selecting and Creating Algorithms 4

Decomposition 4
- Decomposition Basics 4
- Structural Decomposition 5
- Functional Decomposition 6
- Object-Oriented Decomposition 7
- Dependencies and Cohesion 7

Pattern Identification 8
- Pattern Identification Basics 8
- Repetitive Patterns 8
- Classification Patterns 9

Abstraction 9
- Abstraction Basics 9
- Classes and Objects 10
- Black Boxes 11
- Levels of Abstraction 12

SUMMARY 12

KEY TERMS 13

MODULE 2
PROGRAMMING TOOLS 15

Programming Languages 16
- Hello World! 16
- Programming Language Basics 16
- Syntax and Semantics 17
- Core Elements 19
- Your Toolbox 19

Coding Tools 20
- Program Editors 20
- Basic Structure 21

Build Tools 22
- The Toolset 22
- Compilers 23
- Preprocessors and Linkers 24
- Virtual Machines 25
- Interpreters 26

Debugging Tools 27
- Programming Errors 27
- Syntax Errors 28
- Runtime Errors 29
- Semantic Errors 29
- Debugging Utilities 30

IDEs and SDKs 32
- Integrated Development Environments 32
- Software Development Kits 32

SUMMARY 33

KEY TERMS 34

MODULE 3
LITERALS, VARIABLES, AND CONSTANTS 35

Literals 36
- Numeric Literals 36
- Character and String Literals 37
- Tricky Literals 38

Variables and Constants 38
- Variables 38
- Constants 40
- The Memory Connection 41

Assignment Statements 41
- Declaring Variables 41
- Initializing Variables 42
- Assigning Variables 43

Input and Output 44
- Input to a Variable 44
- Output from a Variable 46

SUMMARY 46

KEY TERMS 47

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

MODULE 4
NUMERIC DATA TYPES AND EXPRESSIONS 49
Primitive Data Types 50
Data Types 50
Primitive Data Types 50
Composite Data Types 51
Numeric Data Types 52
Integer Data Types 52
Floating-Point Data Types 53
Mathematical Expressions 54
Arithmetic Operators 54
Order of Operations 56
Compound Operators 56
Numeric Data Type Conversion 58
Convert Integers and Floating-Point Numbers 58
Rounding Quirks 59
Formatting Output 60
Formatted Output 60
Formatting Parameters 60
SUMMARY 62
KEY TERMS 62

MODULE 5
CHARACTER AND STRING DATA TYPES 63
Character Data Types 64
Working with Character Data 64
Character Memory Allocation 65
Digits 66
Character Output Format 67
Character Manipulation 68
String Data Types 69
Working with String Data 69
Escape Characters 70
String Indexes 71
String Functions 72
String Manipulation 72
String Length 72
Change Case 73
Find the Location of a Character 74
Retrieve a Substring 75
Concatenation and Typecasting 76
Concatenated Output 76
Concatenated Variables 77
Coercion and Typecasting 78
SUMMARY 80
KEY TERMS 81

MODULE 6
DECISION CONTROL STRUCTURES 83
If-Then Control Structures 84
Control Structures 84
Decision Logic 85
If-Then Structures 85
Relational Operators 87
The Equal Operator 87
Using Relational Operators 88
Boolean Expressions and Data Types 89
Multiple Conditions 91
If-Then-Else Structures 91
Nested-If Structures 93
Else If Structures 96
Fall Through 97
Conditional Logical Operators 100
The AND Operator 100
The OR Operator 101
SUMMARY 102
KEY TERMS 102

MODULE 7
REPETITION CONTROL STRUCTURES 103
Count-Controlled Loops 104
Loop Basics 104
Control Statements 105
For-Loops 105
User-Controlled Loops 108
Counters and Accumulators 109
Loops That Count 109
Loops That Accumulate 111
Nested Loops 112
Loops Within Loops 112
Inner and Outer Loops 113
Pre-Test Loops 116
While-Loops 116
Infinite Loops 117
Breaking Out of Loops 118
Post-Test Loops 120
Do-Loops 120
Test Conditions and Terminating Conditions 123
SUMMARY 124
KEY TERMS 124

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

MODULE 8
ARRAYS 125

Array Basics 126
Magic Rectangles 126
Array Characteristics 127
Array Use Cases 128

One-Dimensional Arrays 128
Initialize Numeric Arrays 128
Initialize String Arrays 130

Array Input and Output 130
Output an Array Element 130
Index Errors 131
Traverse an Array 132
Input Array Elements 133

Array Operations 135
Change an Array Element 135
Find an Array Element 135
Sum Array Elements 137

Two-Dimensional Arrays 137
Two-Dimensional Array Basics 137
Initialize a Two-Dimensional Array 138
Output a Two-Dimensional Array 139
Sum Array Columns and Rows 141

SUMMARY 143
KEY TERMS 144

MODULE 9
FUNCTIONS 145

Function Basics 146
Function Classifications 146
Programmer-Defined Functions 146
Flow of Execution 147
Function Advantages 147

Void Functions 148
Void Function Basics 148
Function Pseudocode 149

Functions with Parameters 150
Function Parameters 150
Function Arguments 150
The Handoff 152

Return Values 153
Return Values 153
Return Type 156
Function Signature 157

Scope 157
Scope Basics 157
Pass by Value 160
Pass by Reference 161
Namespaces 162

SUMMARY 163
KEY TERMS 163

MODULE 10
RECURSION 165

Key Components of Recursion 165
The Recursive Mindset 165
Recursion Basics 167
When to Use Recursion 171

Using Recursion to Solve Complex Problems 171
Designing Recursive Structures 171
Linear Recursion 174
Branching Recursion 175

Managing Memory during Recursion 179
Memory Management 179
Stable Recursion 182

SUMMARY 183
KEY TERMS 183

MODULE 11
EXCEPTIONS 185

Defining Exceptions 185
Errors in Code 185
Exception Types 187

Dealing with Exceptions 189
Handling Others' Exceptions 189
Try and Catch Blocks 189

Using Exceptions 198
Throwing Exceptions 198
When to Bail 202

SUMMARY 203
KEY TERMS 203

MODULE 12
FILE OPERATIONS 205

File Input and Output 206
The Purpose of Files 206
Anatomy of a File 210
File Usage 212

Processing a File 214
Accessing Files 214
Streaming and Buffering 214

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Reading from a File 216
- Opening a File for Reading 216
- Reading from a File 218

Closing a File 222
- Closing Files after Use 222
- Trying to Close a File 222

Creating and Writing New Files 222
- Creating a File 222
- Opening a File for Writing 223
- Writing to and Appending a File 224
- Anticipating Exceptions 228

SUMMARY 229

KEY TERMS 230

MODULE 13
CLASSES AND OBJECTS 231

Classes in Object-Oriented Programming 232
- Representing the Real World with Code 232
- Using Classes 232
- Class Components 233

Using Objects 236
- Creating Objects 236
- Objects as Variables 238
- Object-Oriented Features and Principles 238

Using Static Elements in a Class 239
- Static Member Variables 239
- Static Methods 240
- Static Classes 241

Characteristics of Objects in Object-Oriented Programs 242
- Object Identity 242
- Object State 242
- Object Behavior 243

SUMMARY 244

KEY TERMS 244

MODULE 14
METHODS 245

Using Methods 245
- Why Use Methods? 245
- Anatomy of a Method 251
- Using Methods 251

Changing the Default Behavior of an Object 255
- Using Objects as Regular Variables 255
- Overloading Methods 258
- Setting One Object to Equal Another 262

Method Cascading and Method Chaining 263
- Calling Multiple Methods on the Same Object 263

Using Constructors 266
- Specifying How to Construct an Object 266
- Constructing an Object from Another Object 268

SUMMARY 269

KEY TERMS 269

MODULE 15
ENCAPSULATION 271

Components of Class Structure 271
- Data Hiding 271
- Designing Objects 273
- Self-Reference Scope 276

Accessor and Mutator Context 277
- Viewing Data from an Object 277
- Changing Data in an Object 278

Using Constructors 280
- Parameters and Arguments 280
- Default Parameters and Constructor Overloading 281

Encapsulation Enforcement with Access Modifiers 283
- Access Modifiers 283
- Public Variables and Methods 283
- Private Variables and Methods 284

Interfaces and Headers 286
- Interfaces 286
- Programming an Interface 287

SUMMARY 290

KEY TERMS 290

MODULE 16
INHERITANCE 291

Using Inheritance 291
- Creating Classes from Other Classes 291
- Family Trees in OOP 292
- Levels of Access 295

Necessary Components for Inheritance 296
- Defining a Parent Class 296
- Defining a Child Class 297

Creating a Child Class That Inherits from a Parent Class 298
- Inheritance Syntax 298
- Customizing Behavior 301

SUMMARY 307

KEY TERMS 307

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

MODULE 17
POLYMORPHISM 309
The Purpose of Polymorphism 309
Flexibility While Coding 309
Dynamic Binding Under the Hood 314
Polymorphism Basics 314
Classes Within Classes 314
Objects as Other Objects 315
Virtual Functions 316
Anticipating Customization 316
Abstract Classes 317
SUMMARY 318
KEY TERMS 318

MODULE 18
TEMPLATES 319
Template Basics 319
Data Abstraction 319
Template Structure and Use 322
Tricky Templating 328
Advanced Templating 328
Templated Objects as Arguments 330
Templates as a Problem-Solving Approach 331
Designing a Template 331
When to Use Templates 331
SUMMARY 331
KEY TERMS 332

MODULE 19
LINKED LIST DATA STRUCTURES 333
Linked List Structures 334
Data Structure Selection 334
Data Structure Implementation 335
Linked List Basics 336
Types of Linked Lists 337
Singly Linked Lists 337
Doubly Linked Lists 338
Circular Linked Lists 339
Linked List Characteristics 339
Code a Linked List 342
The **Node** Class 342
The **LinkedList** Class 343
The **Append** Method 343
Linked List Traversal 345
The Find Method 346
The Insert Method 347
SUMMARY 350
KEY TERMS 351

MODULE 20
STACKS AND QUEUES 353
Stacks 353
Stack Basics 353
Stack Use Cases 355
Built-in Stacks 356
Code a Stack 357
Queues 362
Queue Basics 362
Queue Use Cases 363
Code a Queue 364
SUMMARY 369
KEY TERMS 369

MODULE 21
TREES AND GRAPHS 371
Nonlinear Data Structures 371
Linear versus Nonlinear Structures 371
Nonlinear Building Blocks 373
Tree Structures 373
Tree Basics 373
Tree Properties 376
Trees as Recursive Structures 376
Solving Problems Using Trees 379
Tree Applications 379
Data Storage in Trees 380
Graph Structures 387
Graph Basics 387
Directed and Undirected Graphs 388
Solving Problems with Graphs 388
Graph Applications 388
Computing Paths 389
SUMMARY 394
KEY TERMS 394

MODULE 22
ALGORITHM COMPLEXITY AND BIG-O NOTATION 395
Big-O Notation 396
Algorithm Complexity 396

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Asymptotic Analysis 397
Asymptotic Notation 398
Time Complexity 398
Big-O Metrics 398
Constant Time 399
Linear Time 399
Quadratic Time 400
Logarithmic Time 401
Space Complexity 403
Memory Space 403
Constant Space Complexity 404
Linear Space Complexity 404
Complexity Calculations 405
Line-by-Line Time Complexity 405
Combine and Simplify 406
A Mystery Algorithm 407
SUMMARY 409
KEY TERMS 409

MODULE 23
SEARCH ALGORITHMS 411
Using Search Algorithms 412
Search Basics 412
Performing a Linear Search 413
Looking for a Needle in a Haystack 413
Evaluating Search Time 416
Performing a Binary Search 416
Shrinking the Search Space 416
Implementing Binary Search 418
Using Regular Expressions in Search Algorithms 423
Specifying a Search Pattern 423
Regular Expression Search Operators 423
SUMMARY 426
KEY TERMS 426

MODULE 24
SORTING ALGORITHMS 427
Qualities of Sorting Algorithms 428
Ordering Items 428
Time Complexity in Sorting Algorithms 428
Sorting Properties 430
Bubble Sort 431
Defining the Bubble Sort Algorithm 431
Bubble Sort Properties 437
Quicksort 438
Defining the Quicksort Algorithm 438
Quicksort Properties 446
Merge Sort 447
Defining the Merge Sort Algorithm 447
Merge Sort Properties 453
SUMMARY 454
KEY TERMS 454

MODULE 25
PROCESSOR ARCHITECTURE 455
Processor Organization 456
Integrated Circuits 456
Moore's Law 458
CPUs 458
Low-Level Instruction Sets 459
Microprocessor Instruction Sets 459
RISC and CISC 460
Machine Language 460
Assembly Language 461
Microprocessor Operations 462
Processing an Instruction 462
The Instruction Cycle 462
High-Level Programming Languages 464
Evolution 464
Teaching Languages 465
The C Family 465
Web Programming Languages 466
Characteristics 466
Advantages and Disadvantages 467
SUMMARY 467
KEY TERMS 468

MODULE 26
DATA REPRESENTATION 469
Bits and Bytes 470
Digital Data 470
Bits 471
Bytes 472
Binary 474
Binary Numbers 474
Binary to Decimal 475
Decimal to Binary 476
Binary Addition 477
Negative Numbers 478
Hexadecimal 480
Colors 480
Hexadecimal Numbers 481

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Binary-Hex-Binary Conversions 481
Hex-Decimal Conversion 482
Information Density 483
ASCII and Unicode 483
ASCII 483
Extended ASCII 484
Unicode 485
Memory Allocation 486
Memory and Storage 486
Storage Devices 487
Memory 487
SUMMARY 489
KEY TERMS 489

MODULE 27
PROGRAMMING PARADIGMS 491
Imperative and Declarative Paradigms 492
Think Outside the Box 492
The Procedural Paradigm 493
Procedural Basics 493
Characteristics of Procedural Programs 494
Procedural Paradigm Applications 496
The Object-Oriented Paradigm 497
Objects, Classes, and Methods 497
Characteristics of Object-Oriented Programs 499
Object-Oriented Applications 501
Declarative Paradigms 501
Declarative Basics 501
Characteristics of the Declarative Paradigm 504
Applications for Declarative Paradigms 504
SUMMARY 505
KEY TERMS 505

MODULE 28
USER INTERFACES 507
User Interface Basics 508
UI and UX 508
UI Components 508
Selecting a UI 510
Command-Line User Interfaces 510
Command-Line Basics 510
Command-Line Program Design 510
Graphical User Interfaces 512
GUI Basics 512
GUI Program Design 514
Voice User Interfaces 515
Voice Interface Basics 515
Speech Recognition 515
Speech Synthesis 516
Designing Programs for Voice User Interfaces 517
Virtual Environment Interfaces 517
Virtual Environments 517
Virtual Environment Interface Components 518
Programming the Virtual Interface 519
Accessibility and Inclusion 520
Accessibility Guidelines 520
Inclusive Design 521
SUMMARY 524
KEY TERMS 524

MODULE 29
SOFTWARE DEVELOPMENT METHODOLOGIES 525
Software Development 526
The Software Development Life Cycle 526
Efficiency, Quality, and Security 527
The Waterfall Model 528
Structured Analysis and Design 528
Waterfall Advantages and Disadvantages 529
The Agile Model 530
Incremental Development 530
Agile Methodologies 530
Agile Advantages and Disadvantages 531
Coding Principles 532
Efficient Coding 532
Modularized Code 533
Clean Coding 534
Secure Coding 534
Success Factors 536
Testing 536
Levels of Testing 536
Unit Testing 537
Integration Testing 538
System Testing 539
Acceptance Testing 539
Regression Testing 539
SUMMARY 540
KEY TERMS 540

MODULE 30
PSEUDOCODE, FLOWCHARTS, AND DECISION TABLES 541
Pseudocode 542
From Algorithms to Pseudocode 542
Pseudocode Basics 544
Pseudocode Guidelines 545
Writing Pseudocode 547

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Flowcharts 548
Flowchart Basics 548
Drawing Flowcharts 548
Flowchart Tools 549
Decision Tables 551
Decision Table Basics 551
List the Conditions 551
List All Possible Alternatives 552
Specify Results and Rules 552
Interpret Rules 553
Optimize the Rules 554
Check for Completeness and Accuracy 555
SUMMARY **555**
KEY TERMS **556**

MODULE 31
UNIFIED MODELING LANGUAGE 557
Purpose of Unified Modeling Language (UML) 557
Communicating Ideas to Other Programmers 557
UML Diagram Parts 558
Class Diagram Basics 558
Use Case Diagram Basics 559
Sequence Diagrams 561
Using UML to Structure Programs 562
UML Associations 562
Translating UML to Code 564
SUMMARY **568**
KEY TERMS **568**

GLOSSARY **569**
INDEX **583**

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

PREFACE

Welcome to *Readings from Programming with C++*. This text includes the stand-alone lessons and readings from MindTap for *Programming with C++* and is intended to be used in conjunction with the MindTap Reader for a complete learning experience.

MindTap Overview

Programming with C++ presents conceptual, language-agnostic narrative with language-specific assets, ungraded C++ coding Snippets, language-agnostic test banks, and additional instructor resources. The goal of this digital product is to develop content around the concepts that are essential for understanding Computer Science from a language-agnostic perspective. Learners will gain a foundational understanding of procedural programming, computer science concepts, and object-oriented programming. Instructors have identified the need for language-agnostic, conceptual content that can be paired with hands-on practice in a specific language. This 31-module text is designed to provide that conceptual content paired with language-specific examples and hands-on learning activities in C++.

Course Objectives:

- Develop a foundational knowledge of coding principles, vocabulary, and core concepts.
- Use new foundational knowledge to learn C++ programming skills.
- Practice emerging coding skills in a low-risk environment.
- Apply learned concepts and skills to assignments/activities that mimic real-world experiences and environments.

C++ Version

We recommend downloading the latest version of C++ before beginning this text. C++14 was used to test all C++ code presented in the module figures.

MindTap Features

In addition to the readings included within this text, the MindTap includes the following:

Course Orientation: Custom videos and readings prepare students for the material and coding experiences they will encounter in their course.

Videos: Animated videos demonstrate new programming terms and concepts in an easy-to-understand format, increasing student confidence and learning.

Coding Snippets: These short, ungraded coding activities are embedded within the MindTap Reader and provide students an opportunity to practice new programming concepts "in-the-moment." Additional language-specific "bridge content" helps transition the student from conceptual understanding to application of C++ code.

Language-specific Examples: Figures within the narrative illustrate the application of general concepts in C++ code.

Instructor & Student Resources

Additional instructor and student resources for this product are available online. Instructor assets include an Instructor's Manual, Teaching Online Guide, PowerPoint® slides, and a test bank powered by Cognero®. Student assets include source code files and coding Snippets ReadMe. Sign up or sign in at **www.cengage.com** to search for and access this product and its online resources.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

ABOUT THE AUTHORS

Dr. Kyla McMullen is a tenure-track faculty member in the University of Florida's Computer & Information Sciences & Engineering Department, specializing in Human-Centered Computing. Her research interests are in the perception, applications, and development of 3D audio technologies. Dr. McMullen has authored over 30 manuscripts in this line of research and is the primary investigator for over 2 million dollars' worth of sponsored research projects.

Dr. Elizabeth A. Matthews is an Assistant Professor of Computer Science at Washington and Lee University. She has taught computer science since 2013 and has been an active researcher in human–computer interaction and human-centered computing. Matthews has published research in the areas of procedural generation, video game enjoyment factors, and freshwater algae identification with HCI.

June Jamrich Parsons is an educator, digital book pioneer, and co-author of Texty and McGuffey Award-winning textbooks. She co-developed the first commercially successful multimedia, interactive digital textbook; one that set the bar for platforms now being developed by educational publishers. Her career includes extensive classroom teaching, product design for eCourseware, textbook authoring for Course Technology and Cengage, Creative Strategist for MediaTechnics Corporation, and Director of Content for Veative Virtual Reality Labs.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

ACKNOWLEDGMENTS

The unique approach for this book required a seasoned team. Our thanks to Maria Garguilo who ushered the manuscripts through every iteration and kept tight rein on the schedule; to Mary E. Convertino who supplied her expertise in learning design; to Lisa Ruffolo for her excellent developmental edit; to Courtney Cozzy who coordinated the project; to Kristin McNary for her leadership in Cengage's computing materials; to Rajiv Malkan (Lone Star College) for his instructional input; to Wade Schofield (Liberty University) for his reviewing expertise; and to John Freitas for his meticulous code review. It was a pleasure to be part of this professional and talented team. We hope that instructors and students will appreciate our efforts to provide this unique approach to computer science and programming.

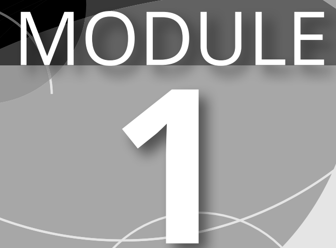
Kyla McMullen: Above all things, I would like to thank God for giving me the gifts and talents that were utilized to write this book. I would like to thank my amazing husband Ade Kumuyi for always being my rock, sounding board, and biggest cheerleader. I thank my parents, Rita and James McMullen for all of their sacrifices to raise me. Last but not least, I thank my spirited friends who help me to remain sane, remind me of who I am, and never let me forget whose I am.

Elizabeth Matthews: I want to thank my parents, Drs. Geoff and Robin Matthews, for their support and understanding in my journey. I would also like to thank my advisor, Dr. Juan Gilbert, for seeing my dream to the end. Finally, I would like to thank my cats, Oreo and Laptop, who made sure that writing this book was interrupted as often as possible.

June Jamrich Parsons: Computer programming can be a truly satisfying experience. The reward when a program runs flawlessly has to bring a smile even to the most seasoned programmers. Working with three programming languages for this project at the same time was certainly challenging but provided insights that can help students understand computational thinking. I've thoroughly enjoyed working with the team to create these versatile learning resources and would like to dedicate my efforts to my mom, who has been a steadfast cheerleader for me throughout my career. To the instructors and students who use this book, my hope is that you enjoy programming as much as I do.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.



COMPUTATIONAL THINKING

LEARNING OBJECTIVES:

1.1 ALGORITHMS

1.1.1 Define the term "algorithm" as a series of steps for solving a problem or carrying out a task.

1.1.2 State that algorithms are the underlying logic for computer programs.

1.1.3 Define the term "computer program."

1.1.4 Provide examples of algorithms used in everyday technology applications.

1.1.5 Confirm that there can be more than one algorithm for a task or problem and that some algorithms may be more efficient than others.

1.1.6 Explain why computer scientists are interested in algorithm efficiency.

1.1.7 List the characteristics of an effective algorithm.

1.1.8 Write an algorithm for accomplishing a simple, everyday technology application.

1.1.9 Write an alternate algorithm for an everyday technology task.

1.1.10 Select the more efficient of the two algorithms you have written.

1.2 DECOMPOSITION

1.2.1 Define the term "decomposition" as a technique for dividing a complex problem or solution into smaller parts.

1.2.2 Explain why decomposition is an important tool for computer scientists.

1.2.3 Differentiate the concepts of algorithms and decomposition.

1.2.4 Identify examples of structural decomposition.

1.2.5 Identify examples of functional decomposition.

1.2.6 Identify examples of object-oriented decomposition.

1.2.7 Provide examples of decomposition in technology applications.

1.2.8 Explain how dependencies and cohesion relate to decomposition.

1.3 PATTERN IDENTIFICATION

1.3.1 Define the term "pattern identification" as a technique for recognizing similarities or characteristics among the elements of a task or problem.

1.3.2 Identify examples of fill-in-the-blank patterns.

1.3.3 Identify examples of repetitive patterns.

1.3.4 Identify examples of classification patterns.

1.3.5 Provide examples of pattern identification in the real world and in technology applications.

1.4 ABSTRACTION

1.4.1 Define the term "abstraction" as a technique for generalization and for simplifying levels of complexity.

1.4.2 Explain why abstraction is an important computer science concept.

1.4.3 Provide an example illustrating how abstraction can help identify variables.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

1.4.4 Provide examples of technology applications that have abstracted or hidden details.

1.4.5 Provide an example illustrating the use of a class as an abstraction of a set of objects.

1.4.6 Explain how the black box concept is an implementation of abstraction.

1.4.7 Identify appropriate levels of abstraction.

1.1 ALGORITHMS

Algorithm Basics (1.1.1, 1.1.4)

A password might not be enough to protect your online accounts. Two-factor authentication adds an extra layer of protection. A common form of two-factor authentication sends a personal identification number (PIN) to your cell phone. To log in, you perform the series of steps shown in **Figure 1-1**.

Figure 1-1 Steps for two-factor authentication

The procedure for two-factor authentication is an example of an algorithm. In a general sense, an **algorithm** is a series of steps for solving a problem or carrying out a task.

Algorithms exist for everyday tasks and tasks that involve technology. Here are some examples:

- A recipe for baking brownies
- The steps for changing a tire
- The instructions for pairing a smart watch with your phone
- The payment process at an online store
- The procedure for posting a tweet

Programming Algorithms (1.1.2, 1.1.3, 1.1.5)

Algorithms are also an important tool for programmers. A **programming algorithm** is a set of steps that specifies the underlying logic and structure for the statements in a computer program. You can think of programming algorithms as the blueprints for computer programs.

A **computer program** is a set of instructions, written in a programming language such as C++, Python, or Java, that performs a specific task when executed by a digital device. A computer program is an implementation of an algorithm.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Q Programming algorithms tell the computer what to do. Can you tell which of these algorithms is a programming algorithm?

Algorithm 1:

Connect to the website's login page.

Enter your user ID.

Enter your password.

Wait for a text message containing a PIN to arrive on your smartphone.

On the website's login page, enter the PIN.

Algorithm 2:

Prompt the user to enter a user ID.

Prompt the user to enter a password.

Make sure that the user ID and password match.

If the user ID and password match:

    Generate a random PIN.

    Send the PIN to user's phone.

    Prompt the user to enter the PIN.

    If the PIN is correct:

        Allow access.

A Algorithm 1 is not a programming algorithm because it outlines instructions for the user. Algorithm 2 is a programming algorithm because it specifies what the computer is supposed to do. When you formulate a programming algorithm, the instructions should be for the computer, not the user.

There can be more than one programming algorithm for solving a problem or performing a task, but some algorithms are more efficient than others.

Q Here are two algorithms for summing the numbers from 1 to 10. Which algorithm is more efficient?

Algorithm 1:

Add 1 + 2 to get a total.

Repeat these steps nine times:

    Get the next number.

    Add this number to the total.

Algorithm 2:

Get the last number in the series (10).

Divide 10 by 2 to get a result.

Add 10 + 1 to get a sum.

Multiply the result by the sum.

A Both algorithms contain four instructions, but Algorithm 2 is more efficient. You can use it to amaze your friends by quickly calculating the total in only four steps. Algorithm 1 is also four lines long, but two of the instructions are repeated nine times. Counting the first step, that's 19 steps to complete this task!

"Good" Algorithms (1.1.6, 1.1.7)

Computer scientists are interested in designing what they call "good" algorithms. A good algorithm tends to produce a computer program that operates efficiently, quickly, and reliably. Good algorithms have these characteristics:

Input: The algorithm applies to a set of specified inputs.
Output: The algorithm produces one or more outputs.
Finite: The algorithm terminates after a finite number of steps.
Precise: Each step of the algorithm is clear and unambiguous.
Effective: The algorithm successfully produces the correct output.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

When formulating an algorithm, you can easily check to make sure it satisfies all the criteria for a good algorithm. You can see how these criteria apply to an algorithm in **Figure 1-2**.

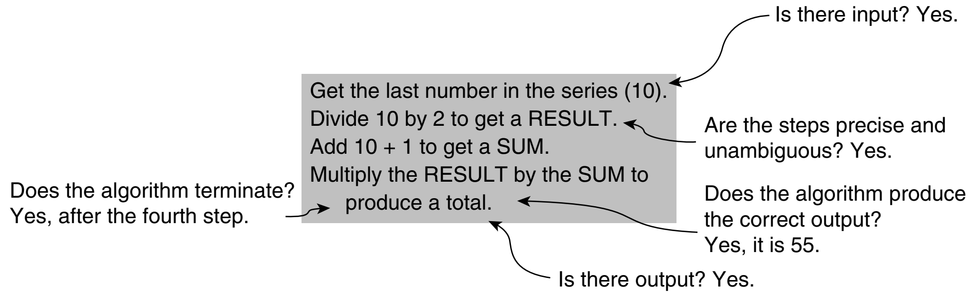


Figure 1-2 Is this a good algorithm?

Selecting and Creating Algorithms (1.1.8, 1.1.9, 1.1.10)

Before coding, programmers consider various algorithms that might apply to a problem. You can come up with an algorithm in three ways:

Use a standard algorithm. Programmers have created effective algorithms for many computing tasks, such as sorting, searching, manipulating text, encrypting data, and finding the shortest path. When you are familiar with these standard algorithms, you can easily incorporate them in programs.

Perform the task manually. When you can't find a standard algorithm, you can formulate an algorithm by stepping through a process manually, recording those steps, and then analyzing their effectiveness.

Apply computational thinking techniques. **Computational thinking** is a set of techniques designed to formulate problems and their solutions. You can use computational thinking techniques such as decomposition, pattern identification, and abstraction to devise efficient algorithms. Let's take a look at these techniques in more detail.

1.2 DECOMPOSITION

Decomposition Basics (1.2.1)

A mobile banking app contains many components. It has to provide a secure login procedure, allow users to manage preferences, display account balances, push out alerts, read checks for deposit, and perform other tasks shown in **Figure 1-3**.

The algorithm for such an extensive app would be difficult to formulate without dividing it into smaller parts, a process called **decomposition**. When devising an algorithm for a complex problem or task, decomposition can help you deal with smaller, more manageable pieces of the puzzle.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

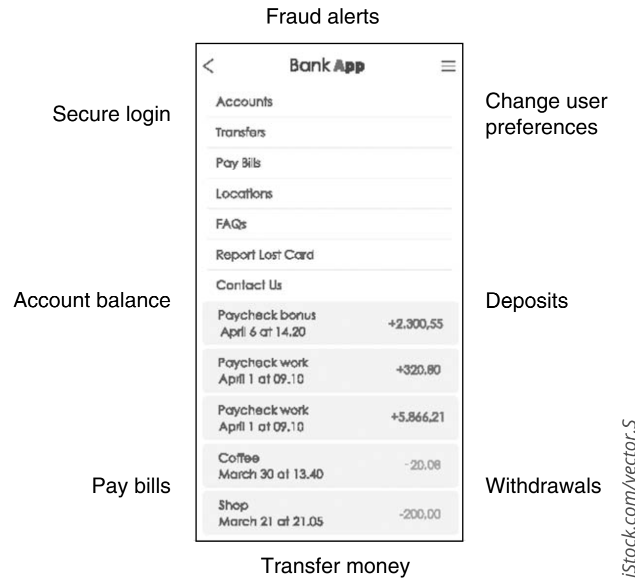


Figure 1-3 A mobile banking app handles many interacting tasks

Structural Decomposition (1.2.2, 1.2.3, 1.2.4, 1.2.7)

The first step in decomposition is to identify structural units that perform distinct tasks. **Figure 1-4** illustrates how you might divide a mobile banking app into structural units, called **modules**.

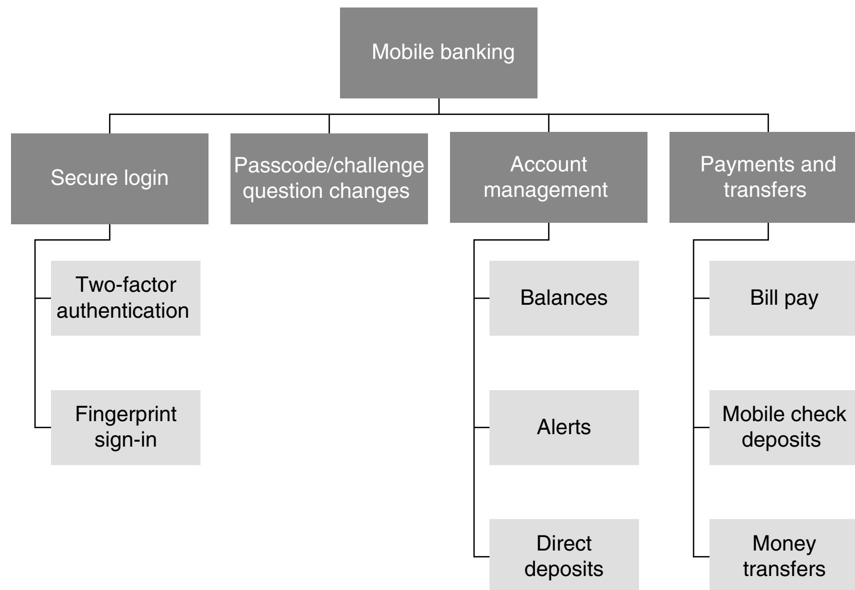


Figure 1-4 Structural decomposition diagram

Structural decomposition is a process that identifies a hierarchy of structural units. At the lowest levels of the hierarchy are modules, indicated in yellow in Figure 1-4, that have a manageable scope for creating algorithms.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Q Which module of the hierarchy chart is not fully decomposed?

A The module for modifying passwords and challenge questions could be further decomposed into two modules: one module that allows users to change their passwords and one for changing their challenge questions.

Here are some tips for creating a structural decomposition diagram:

- Use a top-down approach. The nodes at the top break down into component parts in the nodes below them.
- Label nodes with nouns and adjectives, rather than verbs. For example, "Account management" is the correct noun phrase, rather than a verb phrase, such as "Manage accounts."
- Don't worry about sequencing. Except for the actual login process, the components in a mobile banking system could be accessed in any order. This is a key difference between an algorithm and decomposition. An algorithm specifies an order of activities, whereas decomposition specifies the parts of a task.

Functional Decomposition (1.2.5)

Functional decomposition breaks down modules into smaller actions, processes, or steps. **Figure 1-5** illustrates a functional decomposition of the two-factor authentication module.

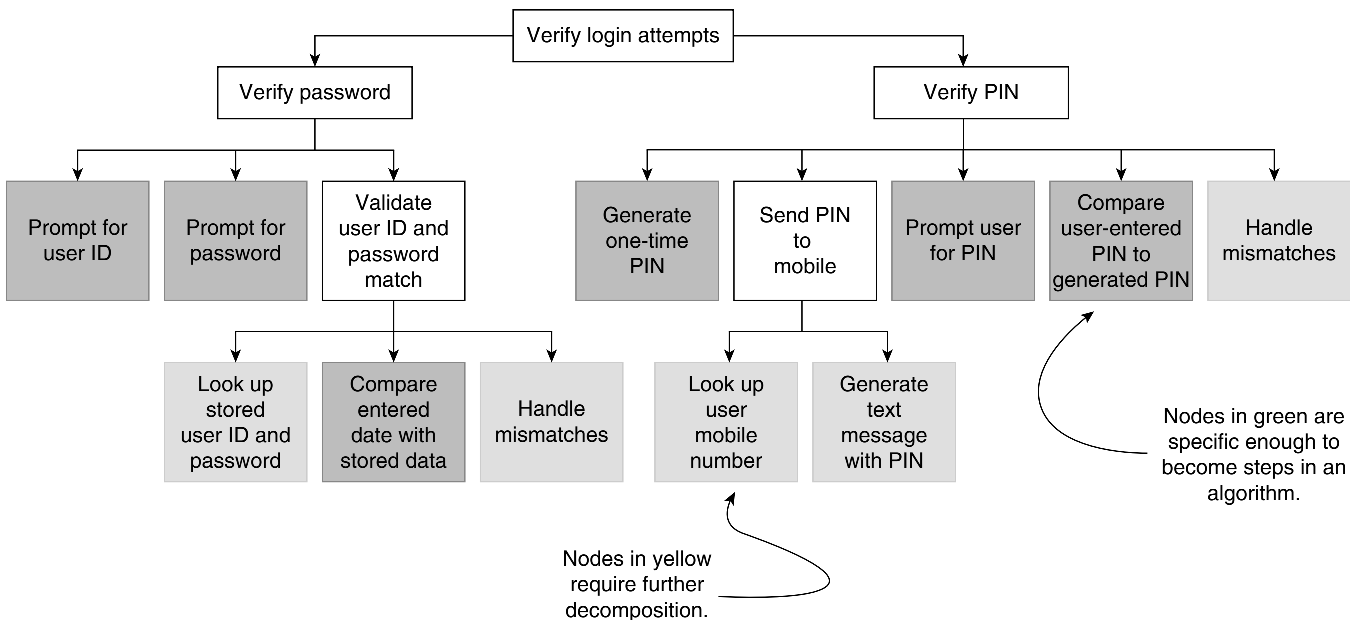


Figure 1-5 Functional decomposition diagram

Notice how the levels of the functional decomposition diagram get more specific until the nodes in the lowest levels begin to reveal instructions that should be incorporated in an algorithm.

Here are some tips for constructing functional decomposition diagrams and deriving algorithms from them:

- Label nodes with verb phrases. In contrast to the nodes of a structural decomposition diagram, the nodes of a functional decomposition are labeled with verb phrases that indicate "what" is to be done.
- Sequence from left to right. Reading left to right on the diagram should correspond to the sequence in which steps in the algorithm are performed.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Object-Oriented Decomposition (1.2.6)

Another way to apply decomposition to a module is to look for logical and physical objects that a computer program will manipulate. **Figure 1-6** illustrates an **object-oriented decomposition** of the two-factor authentication module.

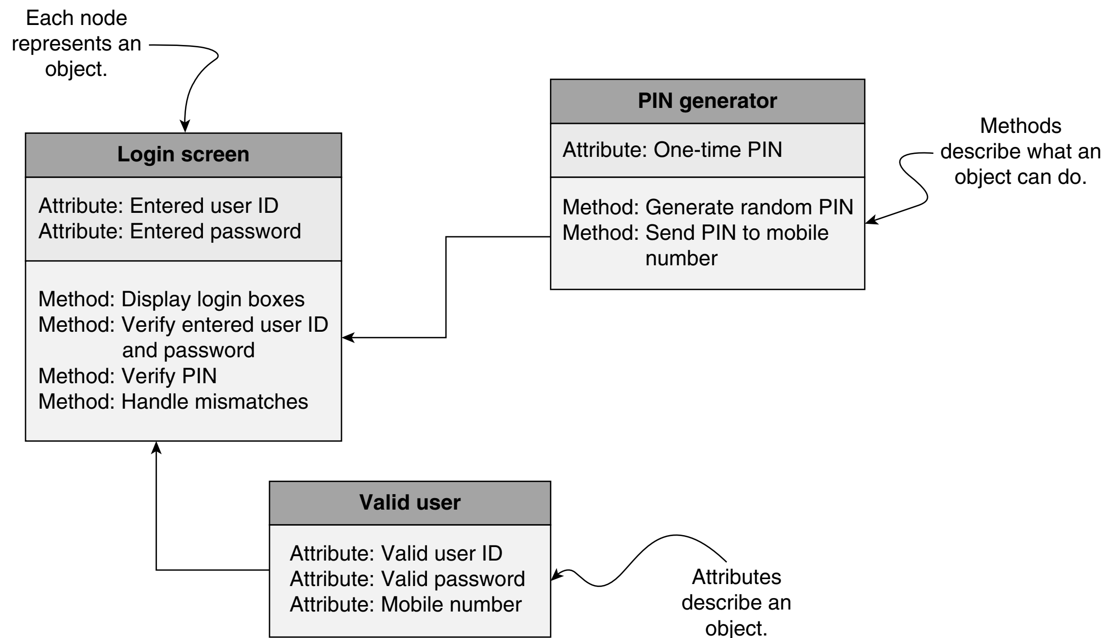


Figure 1-6 Object-oriented decomposition diagram

An object-oriented decomposition does not produce a hierarchy. Instead it produces a collection of **objects** that can represent people, places, or things.

Tips for object-oriented decomposition:

- Node titles are nouns. Each node in the object-oriented decomposition diagram is labeled with a noun.
- Attributes are nouns. A node can contain a list of **attributes**, which relate to the characteristics of an object.
- Methods are verb phrases. An object can also contain **methods**, which are actions that an object can perform. You may need to devise an algorithm for each method.
- Sketch in connection arrows. Connection arrows help you visualize how objects share data.

Dependencies and Cohesion (1.2.8)

You might wonder if there is a correct way to decompose a problem or task. In practice, there may be several viable ways to apply decomposition, but an effective breakdown minimizes dependencies and maximizes cohesion among the various parts.

The principles of decomposition are:

- Minimize dependencies. Although input and output may flow between nodes, changing the instructions in one module or object should not require changes to others.
- Maximize cohesion. Each object or module contains attributes, methods, or instructions that perform a single logical task or represent a single entity.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

1.3 PATTERN IDENTIFICATION

Pattern Identification Basics (1.3.1, 1.3.2)

The Amaze-Your-Friends math trick for quickly adding numbers from 1 to 10 is very simple:

Get the last number in the series (10).
Divide 10 by 2 to get a result.
Add 10 + 1 to get a sum.
Multiply the result by the sum.

Q Try the algorithm yourself. What is your answer?

A If your math is correct, your answer should be 55.

Now, what if the challenge is to add the numbers from 1 to 200? That algorithm looks like this:

Get the last number in the series (200).
Divide 200 by 2 to get a result.
Add 200 + 1 to get a sum.
Multiply the result by the sum.

Notice a pattern? This fill-in-the-blank algorithm works for any number:
Get the last number in the series (____________________).
Divide ____________________ by 2 to get a result.
Add ____________________ + 1 to get a sum.
Multiply the result by the sum.

The process of finding similarities in procedures and tasks is called **pattern identification**. It is a useful computational thinking technique for creating algorithms that can be used and reused on different data sets. By recognizing the pattern in the Amaze-Your-Friends math trick, you can use the algorithm to find the total of any series of numbers.

Repetitive Patterns (1.3.3)

In addition to fill-in-the-blank patterns, you might also find repetitive patterns as you analyze tasks and problems. Think about this algorithm, which handles logins to a social media site:

Get a user ID.
Get a password.
If the password is correct, allow access.
If the password is not correct, get the password again.
If the password is correct, allow access.
If the password is not correct, get the password again.
If the password is correct, allow access.
If the password is not correct, get the password again.
If the password is correct, allow access.
If the password is not correct, lock the account.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Q How many repetition patterns do you recognize?

A Two lines are repeated three times:
If the password is not correct, get the password again.
If the password is correct, allow access.

Recognizing this repetition, you can streamline the algorithm like this:
Get a password.
Repeat three times:
    If the password is correct, allow access.
    If the password is not correct, get the password again.
If the password is correct, allow access.
If the password is not correct, lock the account.

Classification Patterns (1.3.4, 1.3.5)

Everyone who subscribes to a social media site has a set of login credentials. Here are Lee's and Priya's:

Lee's login credentials:
Lee's user ID: LeezyBranson@gmail.com
Lee's password: MyCat411
Lee's mobile number: 415-999-1234

Priya's login credentials:
Priya's user ID: PriyaMontell@gmail.com
Priya's password: ouY52311v
Priya's mobile number: 906-222-0987

The series of attributes that define each user's login credentials have a pattern of similarities. Each user has three attributes: a user ID, a password, and a mobile number. By recognizing this pattern, you can create a template for any user's login credentials like this:

User ID: ____________________
Password: ____________________
Mobile number: ____________________

You can often discover **classification patterns** in the attributes that describe any person or object. Identifying classification patterns can help you design programs that involve databases because the template identifies fields, such as User ID, that contain data.

Classification patterns also come in handy if you want to design programs based on the interactions among a variety of objects, rather than a step-by-step algorithm. In some programming circles, templates are called **classes** because they specify the attributes for a classification of objects. For example, people classified as social media subscribers have attributes for login credentials. Vehicles classified as cars have attributes such as color, make, model, and VIN number. Businesses classified as restaurants have a name, hours of operation, and a menu.

1.4 ABSTRACTION

Abstraction Basics (1.4.1, 1.4.2, 1.4.3)

Think back to the Amaze-Your-Friends math trick. By identifying a pattern, you formulated a general algorithm that works for a sequence of any length, whether it is a sequence of 1 to 10 or 1 to 200.

Get the last number in the series (____________________).
Divide ____________________ by 2 to get a result.

Copyright 2022 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s).
Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.