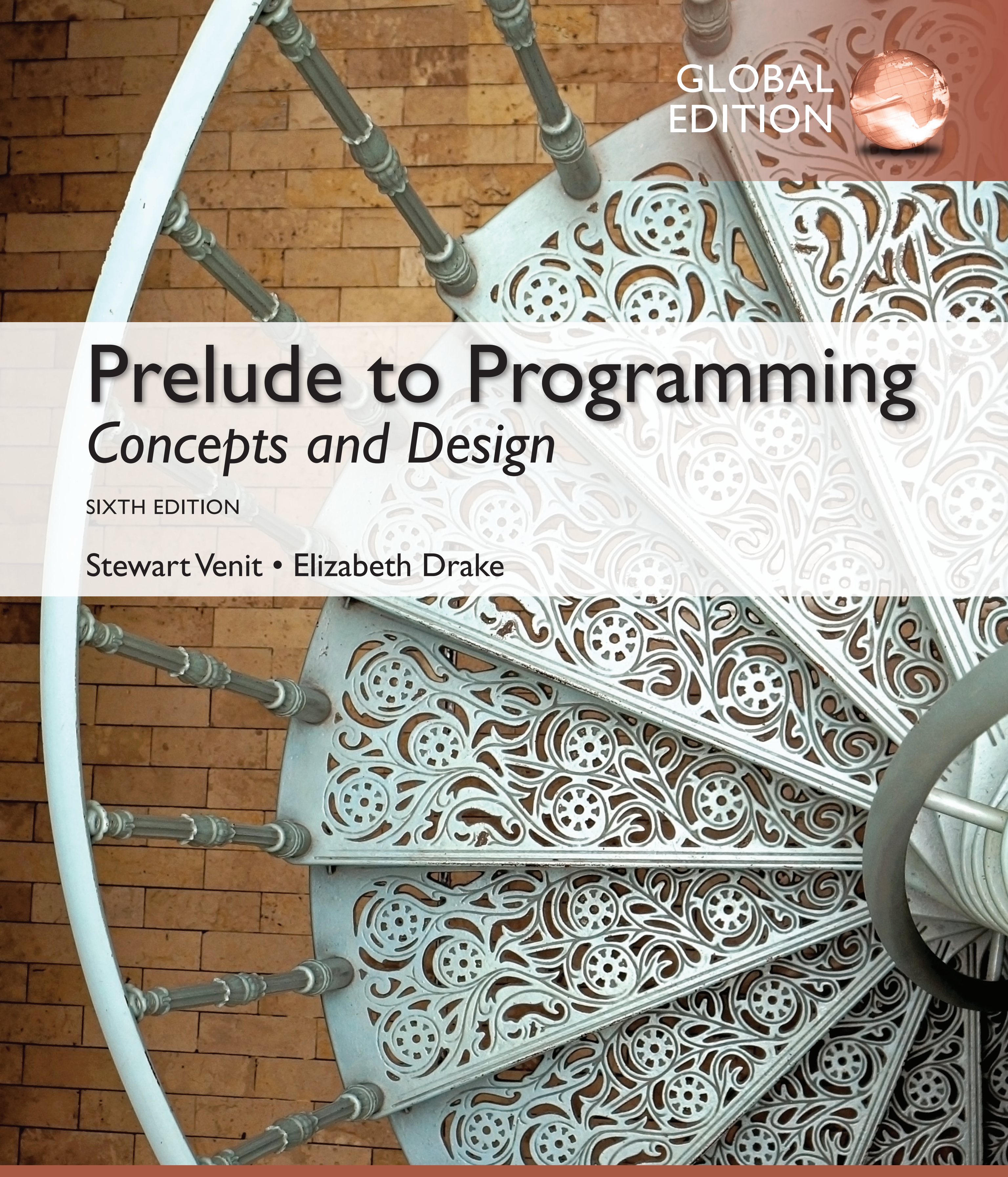


GLOBAL EDITION

Prelude to Programming

Concepts and Design

SIXTH EDITION

Stewart Venit • Elizabeth Drake

ALWAYS LEARNING

PEARSON

ONLINE ACCESS

Thank you for purchasing a new copy of ***Prelude to Programming: Concepts and Design,*** **Sixth Edition, Global Edition.** Your textbook includes one year of prepaid access to the book's Companion Website. This prepaid subscription provides you with full access to the following student support areas:

- Video notes
- Answers to Review Questions
- Answers to Self-Check Questions

Use a coin to scratch off the coating and reveal your student access code.
Do not use a knife or other sharp object as it may damage the code.

To access the ***Prelude to Programming: Concepts and Design*****, Sixth Edition, Global Edition,** Companion Website for the first time, you will need to register online using a computer with an Internet connection and a web browser. The process takes just a couple of minutes and only needs to be completed once.

1. Go to **www.pearsonglobaleditions.com/Venit**
2. Click on **Companion Website.**
3. Click on the **Register** button.
4. On the registration page, enter your student access code* found beneath the scratch-off panel. Do not type the dashes. You can use lower- or uppercase.
5. Follow the on-screen instructions. If you need help at any time during the online registration process, simply click the **Need Help?** icon.
6. Once your personal Login Name and Password are confirmed, you can begin using the *Prelude to Programming: Concepts and Design* Sixth Edition, Global Edition Companion Website!

To log in after you have registered:

You only need to register for this Companion Website once. After that, you can log in any time at **www.pearsonglobaleditions.com/Venit** by providing your Login Name and Password when prompted.

*Important: The access code can only be used once. This subscription is valid for one year upon activation and is not transferable. If this access code has already been revealed, it may no longer be valid. If this is the case, you can purchase a subscription by going to **www.pearsonglobaleditions.com/Venit** and following the on-screen instructions.

Prelude to Programming

Concepts and Design

Sixth Edition

Global Edition

Stewart Venit | Elizabeth Drake

PEARSON

Boston Columbus Indianapolis New York San Francisco
Upper Saddle River Amsterdam Cape Town Dubai London Madrid Milan
Munich Paris Montréal Toronto Delhi Mexico City Sáo Paulo Sydney
Hong Kong Seoul Singapore Taipei Tokyo

Editorial Director: Marcia Horton
Acquisitions Editor: Matt Goldstein
Program Manager: Kayla Smith-Trabox
Director of Marketing: Christy Lesko
Marketing Manager: Yez Alayan
Marketing Coordinator: Kathryn Ferranti
Marketing Assistant: Jon Bryant
Senior Managing Editor: Scott Disanno
Senior Project Manager: Marilyn Lloyd
Operations Supervisor: Vincent Scelta
Operations Specialist: Linda Sager
Head of Learning Asset Acquisition, Global Edition: Laura Dent
Assistant Acquisitions Editor, Global Edition: Aditee Agarwal
Project Editor, Global Edition: Amrita Naskar
Manager, Media Production, Global Edition: Vikram Kumar
Senior Manufacturing Controller, Production, Global Edition: Trudy Kimber
Text Designer: Gillian Hall
Cover Designer: Lumina Datamatics Ltd.
Manager, Visual Research: Karen Sanatar
Permissions Supervisor: Michael Joyce
Permission Administrator: Jenell Forschler
Cover Image: aimy27feb/Shutterstock
Media Project Manager: Renata Butera

Credits:

Figure 0.1 U. S. Army Center of Military History; Figure 0.2 dule964/Fotolia; Figure 0.3 Shutterstock/Stu49; Figure 0.4a Jultud/Fotolia; Figure 0.4b Giuseppe Lancia/Fotolia; Figure 0.5 Fotosearch/Publitek, Inc.; Figure 0.7 National Center for Computational Sciences; Figure 6a Chuck/Alamy; Figure 6b Marian Stanca/Alamy; Figure 11.01a Shutterstock; Figure 11.01b Shutterstock

Screenshots throughout the entire text: RAPTOR is provided free courtesy of the United States Air Force Academy, http://raptor.martincarlisle.com/

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

ISBN 10: 1-292-06153-7
ISBN 13: 978-1-292-06153-5

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

Typeset by Laserwords Pvt. Ltd

Brief Contents

Preface 15

0 Introduction 23
1 An Introduction to Programming 47
2 Data Representation 89
3 Developing a Program 135
4 Selection Structures: Making Decisions 201
5 Repetition Structures: Looping 277
6 More about Loops and Decisions 351
7 Arrays: Lists and Tables 429
8 Searching and Sorting Arrays 487
9 Program Modules, Subprograms, and Functions 549
10 Sequential Data Files 621
11 Object-Oriented and Event-Driven Programming 677

Appendix A: Study Skills 749
Appendix B: The ASCII Character Set: Printable Characters 757
Appendix C: Answers to Self Checks 761

Index 801

Contents

Preface 15

0 Introduction 23

In the Everyday World: You Are Already a Programmer! 24

0.1 A Brief History of Computers 24
- What Is a Computer? 24
- Personal Computers 26
- The Internet 29

0.2 Computer Basics 30
- The Central Processing Unit 31
- Internal Memory 31
- Mass Storage Devices 32
- Input Devices 34
- Output Devices 34

0.3 Software and Programming Languages 36
- Types of Software 36
- Types of Programming and Scripting Languages 37

Chapter Review and Exercises 41

1 An Introduction to Programming 47

In the Everyday World: You Are Already a Programmer! 48

1.1 What Is Programming? 48
- A General Problem-Solving Strategy 49
- Creating Computer Programs: The Program Development Cycle 50

1.2 Basic Programming Concepts 51
- A Simple Program 51
- Data Input 54
- Program Variables and Constants 56

1.3 Data Processing and Output 59
- Processing Data 59
- Data Output 63

1.4 Data Types 67
- The `Declare` Statement 67

Character and String Data 67

1.5 **Integer Data 70**

Operations on Integers 71

1.6 **Floating Point Data 72**

The Declare Statement Revisited 73

Types of Floating Point Numbers 75

1.7 **Running With RAPTOR (Optional) 80**

Introducing RAPTOR 80

Chapter Review and Exercises 82

2 **Data Representation 89**

In the Everyday World: **It Isn't Magic—It's Just Computer Code 90**

2.1 **Decimal and Binary Representation 90**

Bases and Exponents 90

The Binary System 92

2.2 **The Hexadecimal System 95**

Hexadecimal Digits 95

Using Hexadecimal Notation 99

2.3 **Integer Representation 102**

Unsigned Integer Format 102

Sign-and-Magnitude Format 103

One's Complement Format 106

Two's Complement Format 108

2.4 **Floating Point Representation 113**

Floating Point Numbers: the Integer Part 113

Floating Point Numbers: the Fractional Part 113

Converting a Decimal Fraction to Binary 115

Putting the Two Parts Together 118

2.5 **Putting it All Together 119**

Scientific Notation 119

Exponential Notation 120

Base 10 Normalization 121

Normalizing Binary Floating Point Numbers 122

The Excess_127 System 122

Base 2 Normalization 123

Single- and Double-Precision Floating Point Numbers 123

Hexadecimal Representation 126

Chapter Review and Exercises 128

3 **Developing a Program 135**

In the Everyday World: **Planning to Program? You Need a Plan 136**

3.1 **The Program Development Cycle 137**

The Process of Developing a Program 137

Additional Steps in the Cycle 140

3.2 **Program Design 144**

Modular Programming 144

3.3 Coding, Documenting, and Testing a Program 153
Coding and Documenting a Program 153
Testing a Program 155
Types of Errors 156
3.4 Commercial Programs: Testing and Documenting 157
The Testing Phase Revisited 158
External Documentation 158
3.5 Structured Programming 160
Flowcharts 160
Control Structures 165
Programming Style 168
3.6 Running With RAPTOR (Optional) 169
Getting Started 170
Introduction to RAPTOR Symbols 171
Variables 173
RAPTOR Symbols 177
Run It: The Sign-In Program 186
Developing the Program 187
Creating the Program in RAPTOR: Input 187
Creating the Program in RAPTOR: Processing 190
Creating the Program in RAPTOR: Output 191
Check It Out 192
Chapter Review and Exercises 194

4 Selection Structures: Making Decisions 201
In the Everyday World: Decisions, Decisions, Decisions . . . 202
4.1 An Introduction to Selection Structures 203
Types of Selection Structures 203
Single- and Dual-Alternative Structures 204
4.2 Relational and Logical Operators 210
Relational Operators 210
Logical Operators 213
Hierarchy of Operations 218
4.3 ASCII Code and Comparing Strings 221
Representing Characters With Numbers 221
4.4 Selecting from Several Alternatives 225
Using `If` Structures 225
Using `Case`-Like Statements 228
4.5 Applications of Selection Structures 232
Defensive Programming 232
Menu-Driven Programs 236
4.6 Focus on Problem Solving: A New Car Price Calculator 238
Problem Statement 238
Problem Analysis 239
Program Design 239
Program Code 242
Program Test 242

4.7 Running With RAPTOR (Optional) 243
The `Selection` Symbol 243
The `Call` Symbol and Subcharts 246
An Example 248
Run It: The New Car Price Calculator 254
Developing the Program 254
Check It Out 264
Chapter Review and Exercises 265

5 Repetition Structures: Looping 277

In the Everyday World: Doing the Same Thing Over and Over and Knowing When to Stop 278

5.1 An Introduction to Repetition Structures: Computers Never Get Bored! 279
Loop Basics 279
Relational and Logical Operators 283

5.2 Types of Loops 285
Pre-Test and Post-Test Loops 285
Counter-Controlled Loops 290

5.3 The `For` Loop 296
The `For` Statement 297
The `For` Loop in Action 300
The Careful Bean Counter 303

5.4 Applications of Repetition Structures 308
Using Sentinel-Controlled Loops to Input Data 308
Data Validation 312
The `Floor()` and `Ceiling()` Functions 316
Computing Sums and Averages 319

5.5 Focus on Problem Solving: A Cost, Revenue, and Profit Problem 324
Problem Statement 324
Problem Analysis 324
Program Design 326
Program Code 329
Program Test 329

5.6 Running With RAPTOR (Optional) 330
Repetition: The `Loop` Symbol 330
A Short Example 332
Run It: Encryption: The Secret Message Encoder 334
What is Encryption? 335
Problem Statement 335
Developing the Program 335
Developing the Encrypting Algorithms 336
Check It Out 342
Chapter Review and Exercises 343

6 More about Loops and Decisions 351

In the Everyday World: Loops Within Loops 352

6.1 Combining Loops with `If-Then` Statements 352
Exiting a Loop 353

6.2 Combining Loops and Decisions in Longer Programs 363
The `Length_Of()` Function 368
The `Print` Statement and the New Line Indicator 369

6.3 Random Numbers 373
The `Random()` Function 373
Not Really Random: The Pseudorandom Number 378

6.4 Nested Loops 379
Nested `For` Loops 379
Nesting Other Kinds of Loops 384
A Mental Workout: Mind Games 389

6.5 Focus on Problem Solving: A Guessing Game 396
Problem Statement 397
Problem Analysis 397
Program Design 398
Program Code 403
Program Test 403

6.6 Running With RAPTOR (Optional) 405
Two Short Examples 405
Run It: Validating a Password 409
Problem Statement 409
Developing the Program 409
Check the length of the password (26–30 characters) 411
Check the first character of the password (cannot be a number, 0–9) 413
Check that the password contains one of the special characters (#, *, or $) 414

Chapter Review and Exercises 421

7 Arrays: Lists and Tables 429

In the Everyday World: Organize It with Lists and Tables 430

7.1 One-Dimensional Arrays 431
Array Basics 431

7.2 Parallel Arrays 438
Some Advantages of Using Arrays 442
A Word About Databases 445

7.3 Strings as Arrays of Characters 446
Concatenation Revisited 446
String Length versus Array Size 448

7.4 Two-Dimensional Arrays 451
An Introduction to Two-Dimensional Arrays 452
Using Two-Dimensional Arrays 453

7.5 Focus on Problem Solving: The Magic Square 458
Problem Statement 458
Problem Analysis 459
Program Design 460
Program Code 466
Program Test 466
7.6 Running With RAPTOR (Optional) 467
A Short Example 470
Run It: Self-Grading Math Test 472
Problem Statement 472
Developing and Creating the Program 472
Check It Out 478
Chapter Review and Exercises 481

8 Searching and Sorting Arrays 487

In the Everyday World: Searching and Sorting 488
8.1 Introduction to Searching and Sorting 488
The Serial Search Technique 488
Basic Steps in a Serial Search 489
Pseudocode for a Serial Search 490
8.2 The Bubble Sort Technique 493
Swapping Values 494
Using the Bubble Sort Algorithm 496
8.3 The Binary Search 502
Use the Binary Search for Large Arrays 503
8.4 The Selection Sort 508
General Selection Sort Technique 508
Applying the Selection Sort Technique 510
8.5 Focus on Problem Solving: A Grade Management Program 513
Problem Statement 513
Problem Analysis 513
Program Design 515
Program Code 521
Program Test 522
8.6 Running With RAPTOR (Optional) 522
The Serial Search 522
The Bubble Sort 525
The Binary Search 527
The Selection Sort 529
Run It: Soccer Camp 531
Problem Statement 531
Developing and Creating the Program 531
Check It Out 537
Revise and Improve 538
Check It Out 539
Chapter Review and Exercises 541

9 Program Modules, Subprograms, and Functions 549

In the Everyday World: Living and Programming in Manageable Pieces: Subprograms 550

9.1 Data Flow Diagrams, Arguments, and Parameters 551
- A Big Sale: The Sale Price Computation Program 551
- Data Flow Diagrams 552
- An Introduction to Arguments and Parameters 553

9.2 More about Subprograms 559
- Value and Reference Parameters 559
- How to Tell the Difference between Value and Reference Parameters 561
- Two Helpful Functions: `ToUpper()` and `ToLower()` 564
- The Scope of a Variable 567

9.3 Functions 572
- Built-in Functions 572
- User-Defined Functions 575

9.4 Recursion 580
- The Recursive Process 580

9.5 Focus on Problem Solving: A Fitness Plan 585
- Problem Statement 585
- Problem Analysis 585
- Program Design 586
- Program Code 592
- Program Test 592

9.6 Running With RAPTOR (Optional) 595
- RAPTOR Built-In Functions (Procedures) 595
- Creating a New `Procedure` 598
- Run It: The Fitness Plan 603
- Problem Statement 603
- Developing and Creating the Program 604
- Check It Out 612

Chapter Review and Exercises 614

10 Sequential Data Files 621

In the Everyday World: Keeping it On File 622

10.1 An Introduction to Data Files 623
- File Basics 623
- Creating and Reading Sequential Files 625

10.2 Modifying a Sequential File 630
- Deleting Records 631
- Modifying Records 634
- Inserting Records 635
- Using Arrays in File Maintenance 637

10.3 Merging Sequential Files 639

10.4 Focus on Problem Solving: Control Break Processing 642
Problem Statement 642
Problem Analysis 643
Program Design 644
Coding and Testing the Program 647
10.5 Focus on Problem Solving: The Invoice Preparation Program 647
Problem Statement 648
Problem Analysis 648
Program Design 649
Program Code 653
Program Test 653
10.6 Running With RAPTOR (Optional) 654
Creating Data Files with the `Redirect_Output() Procedure` 654
Displaying Data Files with the `Redirect_Input() Procedure` 655
The Limitations 658
Run It: Professor Weisheit's Semester Grades 659
Check It Out 667
Chapter Review and Exercises 669

11 Object-Oriented and Event-Driven Programming 677
In the Everyday World: Objects are Everywhere 678
11.1 Classes and Objects 678
Classes 678
Defining Classes and Creating Objects 681
Creating Objects 683
The Constructor 685
11.2 More Features of Object-Oriented Programming 686
Benefits of Object-Oriented Languages 686
Inheritance and Polymorphism 687
11.3 Object-Oriented Program Design and Modeling 697
Modeling Languages 700
Unified Modeling Language (UML) 700
11.4 Graphical User Interfaces and Event-Driven Programming 703
Window Components 703
Creating GUI Objects in a Program 704
Event-Driven Programming 706
Handling Events 706
Event-Driven Program Design 709
11.5 Focus on Problem Solving: Another Grade Management Program 711
Problem Statement 711
Problem Analysis 711
Program Design 712
Program Code 717
Program Test 718

11.6 Running With RAPTOR (Optional) 719
Object-Oriented Mode 719
Creating a Class 719
The `main` Program 726
Inheritance and Polymorphism 726
Run It: Monster Evasion 727
Problem Statement 727
Developing and Creating the Program 727
The `main` program 731
Using the Classes 735
Check It Out 738
Chapter Review and Exercises 740

Appendix A:
Study Skills 749
A.1 Achieving Success in the Course 749
A.2 Using the Textbook 750
A.3 Doing the Homework 751
A.4 Writing Programs 752
A.5 Preparing for Tests 753
A.6 More about Preparing for Tests 754
A.7 Taking Tests 755
A.8 Overcoming Test Anxiety 756

Appendix B:
The ASCII Character Set: Printable Characters 757

Appendix C:
Answers to Self Checks 761

Index 801

Preface

Prelude to Programming: Concepts & Design provides a language-independent introduction to programming concepts that helps students learn the following:

- General programming topics, such as data types, control structures, arrays, files, functions, and subprograms
- Structured programming principles, such as modular design, proper program documentation and style, and event-driven and object-oriented program design
- Basic tools and algorithms, such as data validation, defensive programming, sums and averages computation, and searching and sorting algorithms
- Real programming experience through the optional use of RAPTOR, a free flowchart-based programming environment
- Data representation of integer and floating point numbers

No prior computer or programming experience is necessary.

Changes to the Sixth Edition

There are significant and exciting changes in this edition. The text continues to strive to enhance learning programming concepts and to provide students with an enriched experience. Throughout the text, concepts build from clear and simple introductory explanations to complex and challenging Examples and Review Exercises. Major improvements include the following:

- Rather than relegating the material on data representation to Appendices, an entire chapter is devoted to these concepts. This chapter is completely independent of the rest of the content and can be skipped with no loss of continuity. However, instructors who want to include the material now have more examples and end-of-chapter Review Exercises.
- Chapter 0 has been revised with up-to-date content relating to new technologies.
- Chapter 1 has been revised and now includes information on the `Boolean` data type.
- The material on arrays, searching, and sorting has been divided into two chapters. Chapter 7 focuses on creating and using both one- and

two-dimensional arrays. Chapter 8 presents algorithms with extensive examples for searching and sorting.
- The text uses RAPTOR, a free flowcharting software application that allows students to create and run programs without focusing on syntax. Each chapter, from Chapter 3 on, includes an optional section devoted to learning RAPTOR and using RAPTOR to develop interesting, executable programs.
- Throughout the text Examples, Self Checks, and Review Exercises have been redesigned when necessary to ensure that they can be worked with or without RAPTOR.
- The Review Exercises in each chapter contain Multiple Choice, True/False, Short Answer, and a Programming Challenges section. All Challenge problems are suitable for RAPTOR.
- When real code is given throughout the text, JavaScript code has been added.
- More built-in functions and properties are introduced including `Length_Of()`, `To_ASCII()`, `To_Character()`, `Indexing[]`, and more.
- The content in Chapter 11 on object-oriented programming has been thoroughly revised and simplified.
- New material on event-driven programming has been added to Chapter 11.

Organization of the Text

The text is written and organized to allow flexibility in covering topics. Material is presented in such a way that it can be used in any introductory programming course at any level. Each concept is presented in a clear, easily understood manner and the level of difficulty builds slowly. The **What & Why** sidebars give students the opportunity to think above and beyond the material in the Examples and encourage discussion and student interaction. The **Making it Work** sidebars demonstrate how concepts are applied in the real world. **Examples**, **Self Checks**, and **Review Exercises** increase in difficulty from most basic to very challenging. The **Programming Challenges** sections at the end of each chapter give students a chance to create longer, comprehensive programs from scratch and, if RAPTOR is used, they can run the programs and see the results.

The text has been designed so that instructors can use it for students at various levels. The core of the text consists of Chapter 1 and Chapters 3–7. Chapters 0 and 2 are optional; Chapter 2 in particular covers material that is relatively complex and may be skipped without consequence. Chapters 8–11 are independent of one another except that some material in Chapter 9 is required to understand Chapter 11. Thus, the text lends itself to a custom book adoption.

Chapter Summaries

- Chapter 0 provides an overview of general computer concepts.
- Chapter 1 discusses basic problem solving strategy and the essential components of a computer program (input, processing, and output). A section on data types introduces students to numeric, string, and Boolean types.

- Chapter 2 is dedicated to data representation. Students learn to convert decimal numbers to binary and hexadecimal. The various ways to represent integers (unsigned, signed, two's complement) as well as floating point numbers are covered. IEEE standards are used to represent floating point numbers in single- and double-precision. The material in this chapter is completely independent from the rest of the book.
- Chapter 3 introduces the program development process, the principles of modular design, pseudocode, and flowcharts. Documentation, testing, syntax and logic errors, and an overview of the basic control structures are covered.
- Chapter 4 covers decision (selection) structures including single-, dual- and multiple-alternative structures, relational and logical operators, the ASCII coding scheme, defensive programming, and menu-driven programs.
- Chapters 5 and 6 present a complete coverage of repetition structures (loops). Chapter 5 focuses on the basic loop structures: pre- and post-test loops, sentinel-controlled loops, counter-controlled loops, and loops for data input, data validation, and computing sums and averages. Chapter 6 builds on the basics from the previous chapters to create programs that use repetition structures in combination with decision structures, nested loops, and random numbers.
- Chapter 7 covers one-dimensional, two-dimensional, and parallel arrays. Representation of character strings as arrays is also discussed. The material in this chapter has been expanded from the previous edition, including more examples to assist students in understanding this difficult material.
- Chapter 8 covers searching and sorting. Two search techniques (serial and binary searches) and two sort techniques (bubble and selection sorts) are included with expanded coverage.
- Chapter 9 covers functions and modules, including the use of arguments and parameters, value and reference parameters, passing by reference versus passing by value, and the scope of a variable. Built-in and user-defined functions are covered. Recursion—an advanced topic—is discussed in some depth but can be skipped if desired.
- Chapter 10 is about sequential data files. The discussion covers records and fields and how to create, write, and read from sequential files. Topics also include how to delete, modify, and insert records, and how to merge files. Arrays are used in conjunction with data files for file maintenance. The control break processing technique is demonstrated in a longer program.
- Chapter 11 is an introduction to the concepts of object-oriented programming and event-driven programming. The object-oriented material in this chapter has been revised for better understandability. The material on event-driven programming is new to this edition. A short introduction to modeling languages, including UML is given. Object-oriented design topics include classes (parent and child), objects, inheritance, polymorphism, public versus private attributes and methods, and the use of constructors. The material on event-driven programming includes the graphical user interface and window components. Properties and methods for various window controls are also covered.

Many sections throughout the text are devoted to more advanced applications and are optional. In particular, the Focus on Problem Solving sections develop relatively complex program designs, which some instructors may find useful to illustrate the chapter material and others may elect to skip to save time. RAPTOR can be used as a tool to illustrate concepts by creating examples throughout the text in RAPTOR but can also be used to create longer and more challenging, creative programs.

Running With RAPTOR: A Flowcharting Environment

In this edition, each chapter from Chapter 3 onward contains an optional section entitled **Running With RAPTOR**. The section describes how to use RAPTOR for that chapter's material with screenshots and step-by-step instructions. Short examples demonstrate how RAPTOR is used to work with the chapter's content and a longer program is developed. In many chapters the RAPTOR program is an implementation of the long program developed in the Focus on Problem Solving section. The Running With RAPTOR sections can be skipped with no loss of continuity. However, if used, the longer RAPTOR programs give students a real-life experience by creating interesting, running programs including games, encryption, and more.

Features of the Text

In the Everyday World

Beginning with Chapter 1, each chapter starts with a discussion of how the material in that chapter relates to familiar things (for example, "Arrays in the Everyday World") This material provides an introduction to the programming logic used in that chapter through an ordinary and easily understood topic, and establishes a foundation upon which programming concepts are presented.

Making It Work

Making It Work

The **Making It Work** sidebars provide information about how to implement concepts in an actual high-level language, such as C++, Java, JavaScript, or Visual Basic. These boxed sidebars appear throughout the text and are self-contained and optional.

What &Why

What & Why

Often we conclude an Example with a short discussion about what would happen if the program were run, or what would happen if something in the program were changed. These **What & Why** sidebars help students deepen their understanding of how programs run. They are useful in initiating classroom discussion.

Pointers and Style Pointers

Style Pointer

The concepts of programming style and documentation are introduced in Chapter 3 and emphasized throughout. Other **Pointers** appear periodically throughout the text. These short notes provide insight into the subject or specialized knowledge about the topic at hand.

Examples

There are more than 200 numbered worked Examples in the text. The pseudocode in the Examples includes line numbers for easy reference. Detailed line-by-line discussions follow the code with sections entitled **What Happened?**

Focus on Problem Solving

Each chapter from Chapter 4 to the end includes a **Focus on Problem Solving** section which presents a real-life programming problem, analyzes it, designs a program to solve it, discusses appropriate coding considerations, and indicates how the program can be tested. In the process, students not only see a review of the chapter material, but also work through a programming problem of significant difficulty. These sections are particularly useful to prepare students for a language-specific programming course.

Exercises

Many new exercises have been added to this edition to correspond with new material. Many exercises have been revised to permit them to be implemented with RAPTOR. The text contains the following diverse selection:

- **Self Checks** at the end of each section include items that test students' understanding of the material covered in that section (answers to Self Checks are in Appendix C)
- **Review Questions** at the end of each chapter include questions of various types that provide further review of the chapter material (Answers to the questions are available on the instructor resource center).
- **Programming Challenges** at the end of each chapter require students to design programs using the material learned in that chapter and earlier chapters. All Programming Challenges can be implemented with RAPTOR. Solutions to all Programming Challenges in RAPTOR are available on the instructor resource center.

Supplements

Instructor's Supplements

Supplemental materials are available to qualified instructors at www.pearsonglobal editions.com/Venit, including the following:

- PowerPoint Presentations for all Chapters
- Solutions to all Self Checks including RAPTOR implementations of select problems
- Solutions to all Review Exercises including corresponding RAPTOR programs
- RAPTOR programs corresponding to all Programming Challenges
- Testbank

For further information about obtaining instructor supplements, contact your campus Pearson Education sales representative.

Acknowledgments

The **In the Everyday World** essays, a unique feature of this book, were envisioned and drafted by Bill Hammerschlag of Brookhaven College for the second edition, and are expanded and revised in this edition.

The implementations of the code in C++, Visual Basic, Java, and Python from the **Focus on Problem Solving** sections were created by Anton Drake from the University of Florida, presently a software developer at OPIE Technologies.

A special thanks to Martin Carlisle who created RAPTOR and remains eager and generous with his support.

We want to extend our thanks to Matt Goldstein, our most supportive and caring Editor; to Marilyn Lloyd, the most patient and understanding Production Manager ever; to Haseen Khan, the Project Manager at Laserwords who works on the other side of the world but feels like my next-door neighbor; and to the entire team at Pearson Education, including Kayla Smith-Tarbox and Yez Alayan. We also want to extend a special thank you to Michael Hirsch who initially brought us together on this project; without Michael, none of this would have been possible.

—Elizabeth Drake
and Stewart Venit

I want to thank my coauthor, Stewart Venit. It's a pleasure to work with him. Marilyn Lloyd and Haseen Khan are very special people; they answer my questions with unfailing patience. I also want to thank my children, Anton and Severia, who have always encouraged my desire—my need—to write. My grandsons, Justy and Jacob, make me smile by being impressed by my work.

—Elizabeth Drake

I would like to thank my coauthor, Elizabeth Drake, for greatly enhancing and improving this book in each of the last four editions. I am grateful to my wife Corinne, who, over the course of my 35 year writing career, never complained about the countless hours I spent camped in front of a computer screen. I also want to thank the rest of my family for being my family: daughter Tamara, son-in-law Cameron, and grandchildren Evelyn and Damian.

—Stewart Venit

The publishers would like to thank the following for their contribution to the Global Edition:

Contributor
Ramesh Kolluru

Reviewers
Mohit P. Tahiliani
Ela Kashyap
Shivkant Kaushik

Introduction 0

In this introduction, we will discuss the history of computers and computer **hardware** and **software**—the devices and programs that make a computer work.

After reading this introduction, you will be able to do the following:

- Understand the evolution of computing devices from ancient Babylonia to the twenty-first century
- Understand the components that make up a typical computer system: the central processing unit, internal memory, mass storage, and input and output devices
- Know the types of internal memory—RAM and ROM—and understand their functions
- Know the types of mass storage: magnetic, optical, solid state, and online storage
- Know the types of software used by a modern computer: application software and system software
- Know the levels of programming languages: machine language, assembly language, and high-level language
- Know the types of programming and scripting languages used to create software
- Understand the distinction between programming and scripting languages

In the Everyday World

Computers Everywhere

A century ago, a child would listen in wonder as his parents described what life was like before cars, electricity, and telephones. Today, a child listens in wonder as his parents describe what life was like without video games, smart phones, GPS systems, and computers. Seventy years ago, electronic computers didn't exist. Now, we use computers daily. Computers are in homes, schools, and offices; in supermarkets and fast food restaurants; on airplanes and submarines. Computers are in our phones, kitchen appliances, and cars. We carry them in our backpacks, pockets, and purses. They are used by the young and old, filmmakers and farmers, bankers and baseball managers. By taking advantage of a wealth of diverse and sophisticated software (programs and apps), we are able to use computers almost limitlessly for education, communication, entertainment, money management, product design and manufacture, and business and institutional processes.

0.1 A Brief History of Computers

Calculators, devices used to increase the speed and accuracy of numerical computations, have been around for a long time. For example, the abacus, which uses rows of sliding beads to perform arithmetic operations, has roots that date back more than 5,000 years to ancient Babylonia. More modern mechanical calculators, using gears and rods, have been in use for almost 400 years. In fact, by the late nineteenth century, calculators of one sort or another were relatively commonplace. However, these machines were by no means *computers* as we use the word today.

What Is a Computer?

A **computer** is a mechanical or an electronic device that can efficiently store, retrieve, and manipulate large amounts of information at high speed and with great accuracy. Moreover, it can execute tasks and act upon intermediate results without human intervention by carrying out a list of instructions called a **program**.

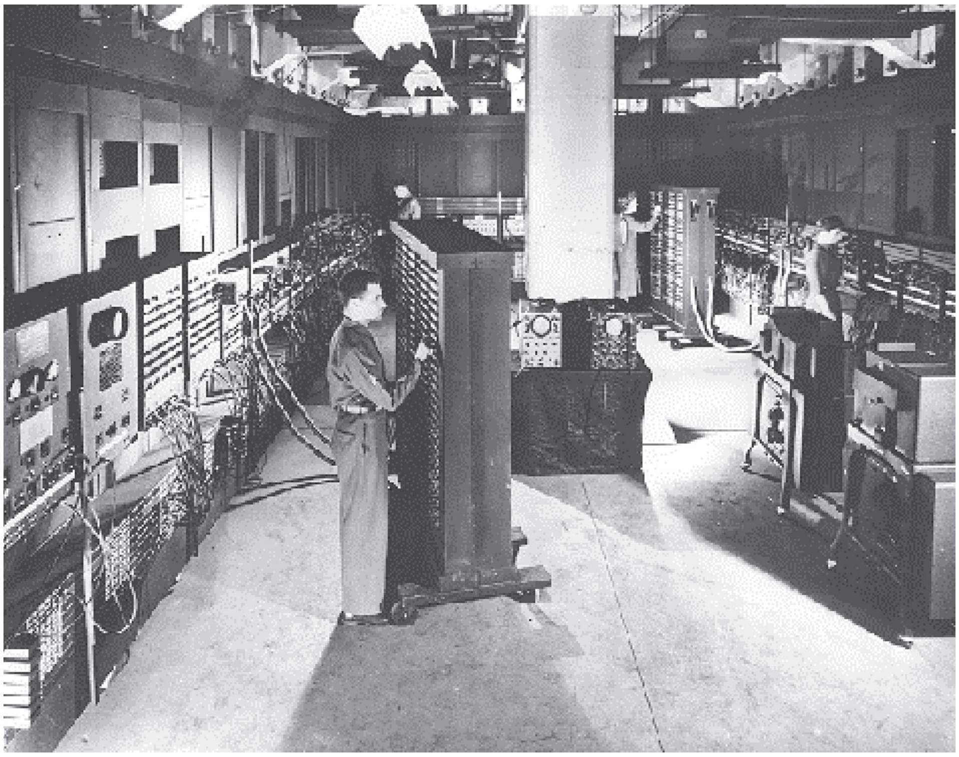
Although we tend to think of the computer as a recent development, Charles Babbage, an Englishman, designed and partially built a true computer in the mid-1800s. Babbage's machine, which he called an *Analytical Engine*, contained hundreds of axles and gears and could store and process 40-digit numbers. Babbage was assisted in his work by Ada Augusta Byron, the daughter of the poet Lord Byron. Ada Byron grasped the importance of the invention and helped to publicize the project. A major programming language (Ada) was named after her. Unfortunately, Babbage never finished his Analytical Engine. His ideas were too advanced for the existing technology, and he could not obtain enough financial backing to complete the project.

Serious attempts to build a computer were not renewed until nearly 70 years after Babbage's death. Around 1940, Howard Aiken at Harvard University, John Atanasoff, and Clifford Berry at Iowa State University built machines that came close to being true computers. However, Aiken's Mark I could not act independently on its intermediate results, and the Atanasoff-Berry computer required the frequent intervention of an operator during its computations.

Just a few years later in 1945, a team at the University of Pennsylvania, led by John Mauchly and J. Presper Eckert, completed work on the world's first fully operable electronic computer. Mauchly and Eckert named it ENIAC, an acronym for Electronic Numerical Integrator and Computer. ENIAC (see Figure 0.1) was a huge machine. It was 80 feet long, 8 feet high, weighed 33 tons, contained over 17,000 vacuum tubes in its electronic circuits, and consumed 175,000 watts of electricity. For its time, ENIAC was a truly amazing machine because it could accurately perform up to 5,000 additions per second. However, by current standards, it was exceedingly slow. A modern run-of-the-mill personal computer can exceed 100 million operations per second!

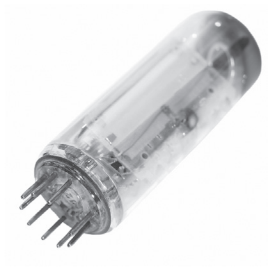
For the next decade or so, all electronic computers used **vacuum tubes** (see Figure 0.2) to do the internal switching necessary to perform computations. These machines, which we now refer to as first-generation computers, were large by modern standards, although not as large as ENIAC. They required a climate-controlled environment and a lot of tender love and care to keep them operating. By 1955, about 300 computers—built mostly by IBM and Remington Rand—were being used, primarily by large businesses, universities, and government agencies.

Figure 0.1 The ENIAC computer



Source: U.S. Army

Figure 0.2 A vacuum tube



By the late 1950s, computers had become much faster and more reliable. The most significant change at this time was that the large, heat-producing vacuum tubes were replaced by relatively small transistors. The **transistor** (see Figure 0.3) is one of the most important inventions of the twentieth century. It was developed at Bell Labs in the late 1940s by William Shockley, John Bardeen, and Walter Brattain, who later shared a Nobel Prize for their achievement. Transistors are small and require very little energy, especially compared to vacuum tubes. Therefore, many transistors can be packed close together in a compact enclosure.

In the early 1960s, Digital Equipment Corporation (DEC) took advantage of small, efficient packages of transistors called **integrated circuits** to create the **minicomputer**, a machine roughly the size of a four-drawer filing cabinet. Because these computers not only were smaller but also less expensive than their predecessors, they were an immediate success. Nevertheless, sales of larger computers, now called **mainframes**, also rapidly increased. The computer age had clearly arrived and the industry leader was the IBM innovative System 360.

Personal Computers

Despite the increasing popularity of computers, it was not until the late 1970s that the computer became a household appliance. This development was made possible by the invention of the **microchip** (see Figure 0.4) in the 1960s. A microchip is a piece of silicon about the size of a postage stamp, packed with thousands of electronic components. The microchip and its more advanced cousin, the **microprocessor**, led to the creation of the world's first **personal computer (PC)** in 1974. The PC

Figure 0.3 An early transistor

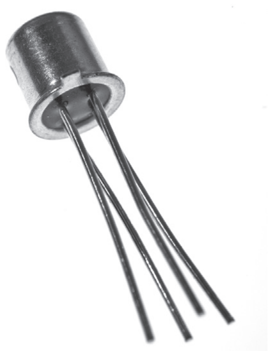
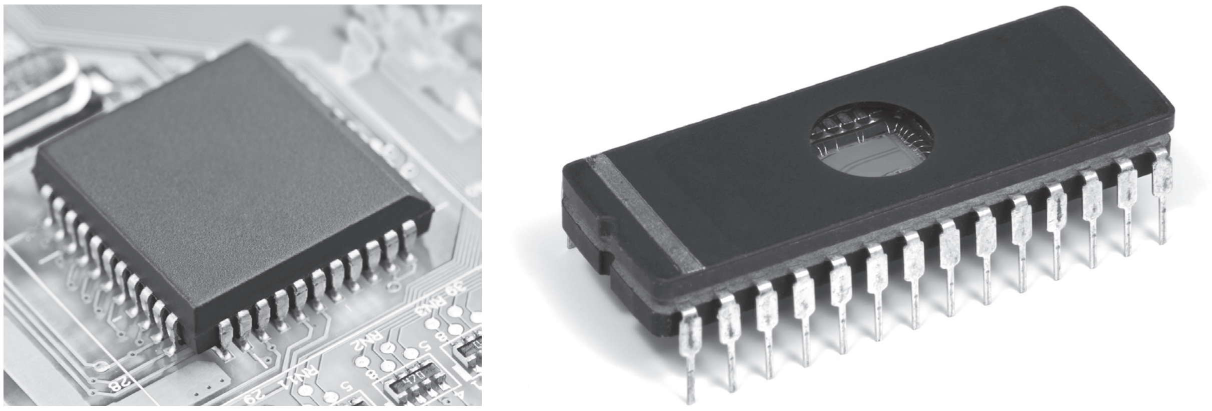


Figure 0.4 The microchip

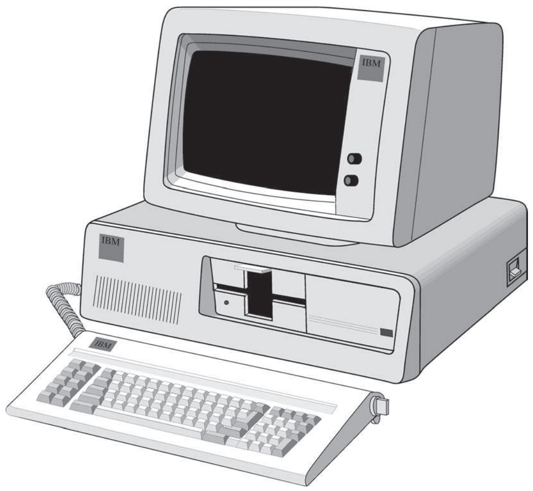


was relatively inexpensive compared to its predecessors and was small enough to fit on a desktop. This landmark computer, the Altair 8800 microcomputer, was unveiled in 1975. Although it was a primitive and not a very useful machine, the Altair inspired thousands of people, both hobbyists and professionals to become interested in PCs. Among these pioneers were Bill Gates and Paul Allen, who later founded Microsoft Corporation, now one of the world's largest companies.

Apple Computers and the IBM PC

The Altair also captured the imagination of two young Californians, Stephen Wozniak and Steven Jobs. They were determined to build a better, more useful computer. They founded Apple Computer, Inc., and in 1977 they introduced the Apple II, which was an immediate hit. With the overwhelming success of this machine and Tandy Corporation's TRS-80, companies that were manufacturing larger minicomputers and mainframes began to notice. In 1981, IBM introduced the popular IBM PC (see Figure 0.5), and the future of the PC was assured.

Figure 0.5 The IBM PC, introduced in 1981, is an antique now!



Many companies hoping to benefit from the success of the IBM PC, introduced computers that could run the same programs as the IBM, and these "IBM compatibles" soon dominated the market. Even the introduction of Apple's innovative and easy-to-use Macintosh in 1984 could not stem the tide of the IBM compatibles. These computers, virtually all of which make use of Microsoft's Windows operating system, have also spawned a huge array of software (computer programs) never dreamed of by the manufacturers of the original mainframes. This software includes word processors, photo editing programs, Web browsers, spreadsheet programs, database systems, presentation graphics programs, and a seemingly infinite variety of computer games. However, while in 2000 the Windows operating system commanded more than 95% of the market share, today's mobile devices, such as smart phones and tablets, have reduced Microsoft's domination drastically with Google's Android operating system and the Apple operating system providing strong competition.

Today's Computers

Today the computer market comprises a vast array of machines. Personal computers are everywhere and range in price from a few hundred to a few thousand dollars. For the most part, their manufacturers are billion dollar companies like IBM, Dell, Hewlett-Packard, and Apple. Although PCs are small and inexpensive, they produce a remarkable amount of computing power. Today's tablets, which can weigh less than a pound and fit into a handbag, are far more powerful than the most advanced mainframes of the mid-1970s (see Figure 0.6).

Minicomputers have also found their niche. Unlike PCs, these machines can be used by a number of people (typically 16 or more) working simultaneously at separate and remote **terminals**. Each terminal consists of a keyboard and a display screen. Minicomputers have become the mainstay of many small businesses and universities, but mainframe computers are by no means dinosaurs. These relatively large and costly machines supply users with tremendous power to manipulate information. **Supercomputers** (see Figure 0.7) are even more powerful than mainframes and can process well over 1 billion instructions per second. For a special effects company like Industrial Light and Magic or a government agency like the Internal Revenue Service, there is no substitute for a large mainframe or supercomputer.

Figure 0.6 Today's laptop and tablet computers