Preface

Welcome to *A Balanced Introduction to Computer Science*. There are a number of reasons why you might be reading this text. Perhaps you have had limited experience with computers and would like to know more about how they work and how to control them. Or perhaps you recognize the marketability of programming and computer literacy, and would like to expand your skills for the future job market. Or perhaps you are just curious about the World Wide Web and want to know how you can create your own Web presence. In any case, you are embarking on what I hope will be an exciting, challenging, and rewarding experience.

Balancing Breadth and Depth

This text is different from most introductory computing texts in that it attempts to maintain a balance between computing breadth and programming depth. Traditionally, introductory texts have focused almost exclusively on one approach or the other. Breadth-based texts have emphasized a broad understanding of computers and computer science. By surveying a wide range of topics such as computer organization, graphics, networking, and technology in society, the intent is for students to experience the breadth of the field and develop a perspective to later understand and appreciate the role of technology in their lives. Alternatively, depth-based texts have focused more deeply on the role of programming in computing. The discipline of programming not only develops problem-solving skills, but also is central to many areas of computer science and thus important to appreciating their significance.

Each of these approaches has merit, but there are potential weaknesses to either extreme. A breadth-based survey of computing can be too superficial, presenting a broad perspective to students who lack the context or experience to fully comprehend it. And although programming depth can provide experience with many computing concepts, developing proficiency as a programmer and problem-solver requires extensive hands-on experience (especially when learning a complex language such as C++ or Java), and may not be directly relevant to all students.

The approach taken by this text is to balance breadth and depth. Chapters are included on concepts and issues in computing that are most relevant to the beginning student, including computer terminology, the Internet and World Wide Web, the history of computing, the organization and manufacture of computer technology, and technology's impact on society. Mixed among these breadth topics are chapters that introduce fundamental programming concepts and skills in a hands-on, tutorial format. Using the programming language JavaScript, students will develop skills in designing and implementing interactive Web pages. JavaScript's simplicity, natural interfaces, and seamless integration with the Web make it possible for novices to develop interesting and engaging programs quickly. In addition, JavaScript is always available (for free!) for anyone with a Web browser, making it easy to apply programming skills learned from this text to everyday problems.

In striking a balance between breadth and depth, the intent of this text is not to be a complete and exhaustive survey of computing or a reference on Web development. Breadth chapters focus on key ideas and concepts that are relevant to beginning students as they attempt to understand computing technology and the field of computer science. Likewise, programming chapters focus on JavaScript features that demonstrate fundamental programming concepts while also allowing for interesting and engaging applications. Links to other sources are provided for the interested reader, including supplemental material and exercises at the end of each programming chapter. This provides a broad perspective on computing as well as enough problem solving and programming depth to appreciate the significance of computer science.

Text Goals

There are three main goals to this text and its accompanying resources. First, it serves to expose the student to the breadth that is the field of computer science. Computer science is more than just the study of computers—it focuses on all facets of computation, from the design and analysis of algorithms (step-by-step sequences of instructions for carrying out tasks), to the engineering and manufacture of computer components, to the development of software systems. Through readings and the use of online resources, the student will study topics such as the history of computer technology, the underlying architecture of modern computers, the translation and execution sequence of programs, and the capabilities and limitations of computation. Using software simulators, the student will build virtual components of a computer and watch the flow of information as a program is translated and executed on the low-level machinery. Through this combination of reading and experimentation, hopefully these concepts will come alive for the student and provide a sense of what computer science is all about.

The second main goal of this text is to teach the student the fundamentals of programming. *Programming is the process of solving problems on the computer* , that is, devising solutions to specific tasks and formalizing those solutions in a language the computer can understand and execute. Programming is the central activity in computer science, providing an inroad to many of the interesting facets and challenges of the field. In learning to program, the student will be learning to analyze problems, think logically, formalize his or her thoughts, and solve problems. It is a discipline, because a systematic approach must be learned, but it is also a creative process, since novel approaches must be found to attack new problems. And because many of the skills developed in programming apply to problem solving in general, experience gained through this text should carry over to other disciplines as well. Finally, the fact that programs are written in the context of interactive Web pages highlights the relevance of programming to applications students use everyday.

The third main goal of this text is to demonstrate the scientific and interdisciplinary nature of computing. Research in various fields of study, most notably the mathematical and natural sciences, is becoming increasingly dependent on computers and programming. By studying and investigating applications in fields such as biology, physics, psychology, and even economics, the student will learn to apply his or her programming skills to a wide range of problems. In addition, the student will develop empirical skills that are common to all scientific endeavors.

Text Features

The balanced approach to computer science and programming taken by this text is evident in the layout of its chapters. There are two types of chapters in the text, those that use narrative to introduce key concepts of computing (i.e., the computer science breadth chapters: 1, 3, 6, 8, 10, 12, 14, 16, and 18) and those that use a tutorial style to develop problem-solving and programming skills (i.e., the programming depth chapters: 2, 4, 5, 7, 9, 11, 13, 15, and 17). The interleaving of these chapters is both intentional and important. It provides variety in the types of activities students will undertake, and thus may be more accommodating to students with different learning styles. Readings and classroom discussions serve as buffers between programming tutorials, allowing the student more time to assimilate programming concepts and skills before beginning the next tutorial. Finally, and perhaps most importantly, the interleaving of chapters supports the student's understanding and appreciation of the content. For example, after developing their own home pages in Chapter 2, students are better prepared to understand what the Web is and how it works in Chapter 3.

Features of the computer science breadth chapters:

- They focus on topics that are most relevant to a beginning student. The goal is not to inundate the student with details, but instead to emphasize the central ideas of that topic.
- Illustrations are used whenever possible to illuminate key points.

- Web-based visualization tools (accessible at the book's Web page: `http://balance3e.com`) are provided to complement many of the chapters and support active learning. For example, Chapter 14 integrates a suite of simulators that allows the student to explore the internal workings of computers.
- Each chapter ends with a Chapter Summary and Review Questions that encourage reflection and the integration of content from that chapter.

Features of the programming depth chapters (which are highlighted by a color bleed down the outer edge of pages):

- They are presented in a tutorial style, recognizing that the only way to learn programming (and, more generally, problem solving) is to actually do it.
- Exercises follow an incremental approach, allowing students to master programming concepts by first studying existing programs (which are accessible at the book's Web page: `http://balance3e.com`) and then making modifications using new tools and constructs. Eventually, students create new programs for solving interesting and hopefully engaging problems.
- In addition to incremental exercises, each depth chapters contains at least one motivational application—a larger programming example that is familiar and relevant to students (such as rotating banner ads, embedded countdown clocks, and text encryption).
- Common errors and points of confusion are identified and discussed in special sections called "Common Errors to Avoid . . ."
- Problem-solving and program design advice is provided in special sections called "Designer Secrets."
- Each chapter includes a Chapter Summary that presents the key concepts and programming tools in a concise bullet list.
- Supplemental material and exercises are provided at the end of each chapter for further study.

Appendices are provided at the end of this text as references. Appendices A and B provide tutorials on Web browsers and common text editors, which may be useful for students who are not already familiar with computers and the Web. Appendix C is an HTML reference, collecting all of the HTML elements used throughout the text in a table. Appendix D is a JavaScript reference, which similarly collects all of the JavaScript programming constructs. Appendices E through G provide full listings of the library files that are introduced in chapter exercises.

Online Resources

The book's Web site, `http://balance3e.com`, contains many resources that can assist the student using the text. These resources include:

- Source code for all examples listed in the text. These Web pages can be viewed directly through the Web site, or they can be downloaded to allow the student to edit and experiment with the pages.
- All of the visualization pages from the chapters, linked together on a single page for ease of access.
- A collection of additional self-study exercises that complement the existing chapter exercises. Solutions are provided so that the student can check his or her answers and receive hints as needed.

In addition, password-protected resources are available for the instructor teaching with the text. The password is available from any Pearson sales representative or online through `http://www.pearsonhighered.com/educator`. These resources include:

- Solutions to all of the exercises and review questions in the text.
- Nine laboratory assignments that complement the material in the programming depth chapters. They emphasize experimentation, analysis, and the use of programs for solving interdisciplinary problems.
- Lecture notes in PowerPoint format corresponding to each chapter of the book.

- Images of many of the figures from the book in a ZIP file for easy download.
- Sample syllabi for organizing a course based on the book.

Changes in the Third Edition

The third edition of this text is significantly different from the previous two editions. The structure and main topics of the text are similar, but much of the content has been reorganized and augmented to reflect the current state of computing. The most notable changes are:

- HTML and JavaScript code throughout the book has been revised to match the latest HTML5 draft standard.
- The programming depth chapters have been reorganized to take advantage of changes to HTML5. In particular, Chapters 4 and 5 introduce event-driven pages earlier, using buttons, text boxes, and page divisions for controlling images and text within a page. Chapter 7 focuses on functions and abstraction, using randomness as a common thread through numerous examples.
- In addition to incremental exercises, each programming depth chapter has at least one larger, motivational application that demonstrates programming concepts in a setting familiar to students. These include interacting help buttons (Chapter 4), online form letters (Chapter 5), rotating banner ads (Chapter 7), embedded countdown clocks (Chapter 9), a slot machine simulation (Chapter 11), dice simulations (Chapter 13), text encryption (Chapter 15), and ASCII animations (Chapter 17).
- New material has been added throughout the book on recent developments and important technologies, such as multi-core processors and operating systems (Chapter 1), cascading style sheets (Chapter 2), HTML5 standards (Chapter 3), wireless networking (Chapter 6), parallel processing (Chapter 10), digital media (Chapter 12), and social networking (Chapter 18).
- Statistics on the Internet/Web and computer specifications have been updated to match the current state of technology.

Advice for the Student

This text does not assume any previous experience with computer applications or programming. Certainly, familiarity with computers is a plus (e.g., word processing or email), but is not necessary. Basic computer terminology will be covered in the text as needed, and appendices are provided to assist the novice with simple computer skills, such as using a text editor, saving files to a disk, and browsing the Web. The goal is not to teach you everything you could ever want to know about computers and programming, but instead to provide you with a working set of skills and knowledge. Whenever possible, links to further readings will be provided in case you are interested in a topic and would like to learn more.

Through the use of readings, exercises, and experiments, this text aspires to provide you with a broad sense of what computer science is all about, while simultaneously developing depth as a problem-solver and programmer. The choice of JavaScript as the medium for developing programming skills was specifically made to make this task simpler and more relevant to students. JavaScript was designed to be a simple scripting language for controlling pages on the World Wide Web. Using JavaScript, you can control actions with the click of the mouse or generate dynamic images on a page. As such, learning JavaScript opens the door for many exciting applications on the Web. Similarities between JavaScript and the programming languages Java and C++ also mean that experience with JavaScript programming can be a stepping-stone to larger-scale programming in these industry-strength languages.

Whether you choose to continue studies in computer science, or merely wish to apply computing skills to your everyday life, the balanced coverage of computing topics as found in this text should prove valuable to you. As always, enjoy and learn.

Advice for the Instructor

The layout of the chapters in this text is designed to provide maximum flexibility for the instructor. Depending on the preferences of the instructor and the goals of the particular course, the right balance and order of the material can be determined for your needs. If you are teaching a traditional nonmajor course, then a roughly even balance between breadth and depth probably makes sense. If the students are computer knowledgeable and taking this course as part of a computer science sequence, then some breadth topics may be skipped or shortened to allow for greater programming depth. This text has been used in both nonmajors' (CS0) courses and introductory majors' courses (CS1) at a variety of institutions. Sample syllabi that demonstrate possible course organizations are available online as supplements.

One of the strengths of this book is the flexibility that it provides the instructor. An instructor might choose to emphasize the breadth chapters, while only covering the first few programming chapters, or emphasize the programming chapters while selecting a few breadth chapters of interest, or perhaps find a balance in between. The interleaving of breadth and depth chapters throughout the text is specifically designed to support the content and vary the types of learning activities students engage in. Although the relative order in which the programming depth chapters are covered is constrained by their content, it is certainly possible to omit or move some of the computer science breadth chapters. For example, some instructors might prefer covering the history of computers (Chapter 6) earlier in the course, or perhaps covering the chapters on how computers work (Chapter 14) and are built (Chapter 16) together.

As always, enjoy, teach, and learn.

Acknowledgments

This book is the culmination of a long process of thinking, experimenting, implementing, classroom testing, rethinking, reexperimenting, and so on. There are numerous people who have contributed to this book during its development, both tangibly and with moral support. The people at Prentice Hall have been outstanding, including editors Petra Recter, Kate Hargett, and Tracy Dunkelberger. I would also like to thank the following outside reviewers, whose insightful comments were a great help:

Debra Burhans, *Canisius College*
Martin Chetlen, *Moorpark College*
Donald Costello, *University of Nebraska*
Lionel Craddock, *Bluefield State College*
Scott Dexter, *Brooklyn College*
Linda DuHadway, *Utah State University*
Buster Dunsmore, *Purdue University*
Erica Eddy, *University of Wisconsin — Parkside*
Karen Ehrlich, *SUNY College at Fredonia*
Chaya Gurwitz, *Brooklyn College*
Dmitri Gusev, *Edinboro University of Pennsylvania*
Paul Helmer, *Hampden-Sydney College*
Michael Hennessy, *University of Oregon*
Mark Holliday, *Western Carolina University*
Ralph Hooper, *University of Alabama — Tuscaloosa*
Nancy Kinnersley, *University of Kansas – Main*
Hank Korth, *Lehigh University*
Rowan Lindley, *Westchester Community College*
Ronald Marsh, *University of North Dakota*
David Middleton, *Arkansas Tech University*
Arnie Miles, *Georgetown University*

Jenna Miley, *Bainbridge College*
Vince Offenback, *North Seattle Community College*
Jeff Parker, *Merrimack College*
Roger Priebe, *University of Texas — Austin*
Mary Ann May-Pumphrey, *DeAnza College*
Charles Riedesel, *University of Nebraska—Lincoln*
Anton Riedl, *Christopher Newport University*
Jerry Ross, *Lane Community College*
Jim Schmolze, *Tufts University*
Patrick Sebrechts, *California State University—San Marcos*
Gene Sheppard, *Georgia Perimeter College*
David Valentine, *Slippery Rock University*
Mark Williams, *Lane Community College*

The concepts behind this book were initially formed while I was at Dickinson College in the late 1990s. Numerous colleagues at Dickinson taught early versions of the material and contributed new ideas, especially Grant Braught and Craig Miller. In addition to those early colleagues, I would like to thank colleagues at Creighton University, who have also been supportive of the course and its material, especially Brian Kokensparger. As a general inspiration and source for great ideas, I gratefully acknowledge my good friend Owen Astrachan, who served as my early role model in teaching as well as embarrassing me in public on numerous occasions.

On a personal note, I would like to thank my parents who raised me to value knowledge and learning, and supported me in my studies. My wife Laura has probably suffered the most during the development of this book—losing me to many late nights of work and having to listen to endless stories about what did and didn't work in class. Her willingness to listen, provide keen insights, and pick up the slack for me at home has made this book possible. Finally, special thanks go to our boys, Charlie and Jack, who have grown up along with the evolution of this book. While I am proud of this book, they are without a doubt my best work.

Dave Reed
Creighton University