

Preface

Computer science as an academic discipline has evolved to embrace a set of intellectual challenges on a par with other sciences. This fact, combined with the undeniable impact of computer science on the modern world, demands an introductory college textbook comparable with commonly-used textbooks in physics, chemistry, or biology. Accordingly, this book is intended to meet the need for an introductory college text in computer science. The distinctive feature of the book is that it has broader coverage of the field than is found in many texts that are currently in use.

All college students can benefit from exposure to computer science early in the curriculum. Therefore, increasing numbers of colleges and universities are requiring that students (particularly those in the sciences, engineering, mathematics, and even the social sciences) take one or two semester courses in computer science in the first year. In more mature disciplines, it is commonly accepted that, at this critical point in the curriculum, it is important to challenge students with fundamental intellectual issues while surveying the field. This book aims to support an introductory course that does so for computer science.

We have three primary goals. First, we want to *demystify* computer systems by unpeeling levels of abstraction down to the simplest switch, leaving no black boxes. Second, we want to *empower* students by giving them the experience and insight necessary to exploit available technology whenever appropriate. Third, we want to build *awareness* of the substantial intellectual underpinnings of the field and its broad reach into other sciences.

The book is a self-contained treatment intended for people with no previous experience in computer science. While its primary purpose is to serve as a textbook for first-year college students, it may also serve as a broad introduction to the field that may be of interest to anyone who has not been exposed to its fundamentals.

Coverage

The book is organized around four main areas of computer science: programming, architecture, theory, and systems. We introduce fundamental concepts in each of these areas, and pay special attention to relationships among them. A proper introduction to the field must do justice to each of these four areas. We also cover applications in scientific computing throughout the book, to reinforce and supplement the curriculum that is typically taught to students in mathematics, science, and engineering in high school and their first year in college.

Programming is essential to being able to understand and appreciate computer science. We cover basic programming in Java, abstract data types and Java classes, elementary data structures, and the design and analysis of algorithms. Whenever possible, we draw examples from concepts to be covered elsewhere in the book or from scientific or commercial applications. Students who learn the programs in this book will have the solid foundation necessary to prepare them to be able to effectively use computers as they pursue any academic discipline.

Architecture refers both to the art of designing computers and to the process of building them. Our coverage of computer architecture is centered around an imaginary machine that is similar to many real computers. We specify the machine in full detail, consider machine-language programs for familiar tasks, and present a full Java simulator for the machine. We continue with a treatment of circuits and logical design, culminating in a full description of how such a machine might be built from the ground up. Our intent is to demystify this apparently formidable challenge while also providing a context for understanding how Java programming and other familiar high-level tasks are carried out on actual machines.

Theory helps us to understand the limits on what we can accomplish with computers. We present Turing's classical results that show how simple abstract machines can help us to pose fundamental questions about the nature of computation. Some of these are among the most important scientific questions of our time. We also consider practical applications such as how to estimate the running times of our programs and how to design efficient algorithms.

Systems enable us to work with computers at a high level of abstraction. We describe the basic components of computer systems that support programming; operating systems for interacting with our programs and our data; networks that allow interaction among computers; and applications systems that provide specialized support for particular tasks.

Our coverage of the four areas is intertwined and also threaded with descriptions and examples of various classical algorithms, programming languages, scientific computing, and commercial applications. Where relevant, we also provide proper historical perspective.

Generally speaking, we *introduce* material that is covered, at most colleges and universities, in several later computer science courses. For computer science majors, this breadth of coverage provides a foundation to be reinforced in later courses. For students in other fields who have a chance to take only a few courses in computer science, this introduction to programming, architecture, theory and systems in the context of scientific applications gives the background that they need to effectively address the impact of computer science on their chosen fields.

Use in the Curriculum

This book is intended for a college course aimed at teaching novices to program while at the same time introducing them to the field of computer science. As such, its content fulfills the requirements of a first course in computer science in a reasonable way. But it does offer an alternative to many courses that are traditionally taught. Rather than delving deeply into the details of a particular programming language, we put programming in context. Rather than giving scant coverage to programming while surveying

computer science, we are able to address fundamental issues that cannot be understood without programming experience.

One option is to use this book as the basis for a full-year college course, perhaps supplemented with a standard text on computer programming. Depending on the needs of the students and the experience of the instructor, diversions of varying length on basic programming skills are appropriate. For logical consistency, we have put most of the material on programming at the beginning of the book; in practice, it is best to spread the coverage of programming throughout the course, using weekly programming assignments tied to the material in the text.

Another option is to use the book as the basis for a fast-paced one-semester course. This choice is appropriate in situations where students in the sciences and engineering may have only one semester to devote to computer science. The course is best positioned early in the curriculum, because students who take it are sure to be able to use computers much more effectively when necessary in courses in their specialty. If the course is later in the curriculum, students may have a more mature point of view that better enables understanding of the advanced topics (but not much patience with learning the basics of programming).

Simply put, the alternative that this book provides is the following: it provides a way to introduce students to computer science while at the same time teaching them to program, in a single course.

Prerequisites

Our aim is for the book to be suitable for typical first-year science and engineering students. Accordingly, we do not expect preparation significantly beyond what other entry-level science and mathematics courses.

Learning to program is one of our primary goals, so we assume no prior programming experience. Indeed, one of the most important features of our approach is that we integrate teaching programming with teaching the rest of computer science. But what should be done with students who have already taken a programming course, in high school or in college? There is no substitute for experience when learning to program. Anyone writing a program to solve a new problem faces a challenging intellectual task, just as does

anyone writing an essay on a new topic. Students who have written numerous essays in high school still benefit from introductory writing courses in college; just as students who have written numerous programs in high school can benefit from taking an introductory programming course. This analogy breaks down slightly because no one comes to college never having written an essay, but many students come to college never having written a program. But our experience has been that we can get students to the point where they can confidently write a program is relatively quickly and that virtually all students in science and engineering can benefit from taking this as a first course (perhaps classified according to their programming experience).

Mathematical maturity is just as important. While we do not dwell on mathematical material, we do refer to the mathematics curriculum that students have taken in high school, including algebra, geometry, trigonometry, and precalculus. We draw some examples from higher mathematics that can be skipped by students who do not have requisite preparation. Most students in our target audience (those intending to major in the sciences and engineering) automatically meet these requirements. *Discrete mathematics* plays a critical role in computer science but is not always fully covered in mathematics curricula, so we cover topics such as Boolean logic, mathematical induction, basic probability, and discrete sums with the point of view that most students have some familiarity with them but that even some basic concepts need to be reinforced (and we include an Appendix with some basic information on discrete mathematics).

We also describe numerous *scientific applications* but do not assume any preparation beyond that provided by typical high school courses in physics, biology, or chemistry.

Goals

What can instructors of upper-level courses in science and engineering expect of students who have successfully completed a computer science course based on this book?

Anyone who has taught an introductory computer science course knows that expectations are typically high: each instructor expects all students to be familiar with the computing environment and approach that he

or she wants to use. A physics professor might expect some students to learn to program over the weekend to run a simulation; an engineering professor might expect other students to be using a particular package to solve differential equations; or a computer science professor might expect knowledge of the details of a particular programming environment. Is it realistic to expect to be able to meet such diverse expectations? Should there be a different introductory course for each set of students? Colleges and universities have been wrestling with such questions since computers came into widespread use in the latter part of the 20th century.

Our primary goal is to provide an answer to such questions by developing a common introductory course in computer science for all students in science and engineering (including prospective computer science majors) that is analogous to commonly-accepted introductory courses in mathematics, physics, biology, and chemistry. The course may also be suitable for students in the humanities who wish a full introduction to the field. Computer science has matured to the point where we can identify fundamental concepts and educate students about them, so that they can be prepared for numerous and diverse computational challenges.

Students who master the material in this book gain the confidence and knowledge that they need to be able to learn to exploit computers wherever they might appear later in their careers, whether it be using an integrated mathematical software package to attack advanced mathematical problems, using Java to develop innovative applications, controlling sophisticated devices, using simulation to study complex problems, or developing new computational paradigms. People continually need to be able to develop new skills in particular contexts because computers continue to evolve. Instructors teaching students who have studied from this book can expect that they have the background and experience necessary to make it possible for them to acquire such skills, to effectively exploit computers in diverse applications, and to appreciate their limitations.

Booksite

An extensive amount of information that supplements this text may be found on the world-wide web at

<http://www.cs.princeton.edu/IntroCS>

For economy, we refer to this web site as the *booksite* throughout the text. It contains material oriented towards instructors, students, and casual readers of the book. We briefly describe this material here, though, as all web users know, it is best surveyed by browsing. With a few exceptions to support testing, the material is all publicly available.

For *instructors*, the booksite contains information about teaching the course. This information is primarily organized around a teaching style that we have developed over the past decade, where we offer two lectures per week to a large audience supplemented by two sessions per week where students meet with instructors in groups of 15-20. The booksite has presentation slides for the lectures, which set the tone for the course. Students are assigned weekly problem sets and programming assignments, both based on exercises from the book. The programming assignments on the booksite have much more detail than found in the book; they represent a progression that embody our approach to teaching programming. Each assignment is intended to teach a relevant concept in the context of an interesting application. The progression of assignments embody our approach to teaching programming. Also included on the booksite is webware for use in managing student submissions and grading assignments.

For *students*, the booksite contains quick access to much of the material in the book, plus extra material to encourage self-learning. Solutions are provided for many of the book's exercises, including program code and text data. There is a wealth of information associated with programming assignments, including suggested approaches, checklists, and FAQs.

For *casual readers* (including instructors and students!) the booksite is a resource for accessing all manner of extra information associated with the book's content. All of the booksite content provides web links and other routes to pursue to find more information about the topic under consideration. There is far more information accessible than any individual could

fully digest, but our goal is to provide enough to whet any reader's appetite for more information about the book's content.

As with any web site, our *Introduction to Computer Science* booksite is continually evolving, but it is an essential resource for everyone who owns this book. In particular, the supplemental materials supporting teaching and learning within a first-year college course are critical to our goal of making computer science an integral component of the education of all scientists and engineers.

Acknowledgements