CENGAGE Learning 大学计算机教育国外著名教材系列 影印版

Introduction to Computer Science

Fifth Edition

计算机科学导论

（第5版）

G. Michael Schneider
Judith L. Gersting 著

CENGAGE Learning

清华大学出版社

大学计算机教育国外著名教材系列（影印版）

Introduction to Computer Science

Fifth Edition

计算机科学导论

（第5版）

G. Michael Schneider

Judith L. Gersting

著

清华大学出版社
北京

Introduction to Computer Science, Fifth Edition
G. Michael Schneider

978-7-302-23296-4

Cengage Learning Asia Pte Ltd
5 Shenton Way, # 01-01 UIC Building Singapore 068808

图书在版编目(CIP)数据

计算机科学导论=Introduction to Computer Science, Fifth Edition：第 5 版：英文 /（美）施奈德（Schneider, G. M.）等著. --影印本. --北京：清华大学出版社，2010.9
（大学计算机教育国外著名教材系列）
ISBN 978-7-302-23296-4

Ⅰ. ①计…　Ⅱ. ①施…　Ⅲ. ①计算机科学－高等学校－教材－英文　Ⅳ. ①TP3

中国版本图书馆 CIP 数据核字（2010）第 148469 号

责任编辑：龙啟铭
责任印制：王秀菊

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62795954, jsjjc@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 装 者：清华大学印刷厂
发 行 者：全国新华书店
开　　本：185×230　　**印张**：45
版　　次：2010 年 9 月第 1 版　　**印　　次**：2010 年 9 月第 1 次印刷
印　　数：1~3000
定　　价：59.50 元

产品编号：039221-01

出 版 说 明

进入21世纪，世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才，谁就能在竞争中取得优势。高等教育，作为培养高素质人才的事业，必然受到高度重视。目前我国高等教育的教材更新较慢，为了加快教材的更新频率，教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从1996年开始，与国外著名出版公司合作，影印出版了“大学计算机教育丛书（影印版）”等一系列引进图书，受到国内读者的欢迎和支持。跨入21世纪，我们本着为我国高等教育教材建设服务的初衷，在已有的基础上，进一步扩大选题内容，改变图书开本尺寸，一如既往地请有关专家挑选适用于我国高校本科及研究生计算机教育的国外经典教材或著名教材，组成本套“大学计算机教育国外著名教材系列（影印版）”，以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材，以利我们把“大学计算机教育国外著名教材系列（影印版）”做得更好，更适合高校师生的需要。

清华大学出版社

PREFACE

Overview

This text is intended for a one-semester introductory course in computer science. It presents a breadth-first overview of the discipline that assumes no prior background in computer science, programming, or mathematics. It is appropriate for use in a service course for students not majoring in computer science. It is also appropriate for schools that implement their introductory sequence for majors using the breadth-first model described in the ACM/IEEE Computing Curricula 2001 Report. It would be quite suitable for a high school computer science course as well. Previous editions of this text have been used in all these types of courses.

The Non-Majors Course

The introductory computer science service course has undergone many changes over the years. In the 1970s and early 1980s, it was usually a course in FORTRAN, BASIC, or Pascal. At that time it was felt that the most important skill a student could acquire was learning to program in a high-level language. In the mid-to-late '80s, a rapid increase in computer use caused the course to evolve into something called "computer literacy" in which students learned about new applications of computing in such fields as business, medicine, law, and education. With the growth of personal computers and productivity software, a typical early to mid-1990s version of this course would spend a semester teaching students to use word processors, databases, spreadsheets, presentation software, and electronic mail. The most recent change has been its evolution into a Web-centric course where students learn to design and implement Web pages using technology such as HTML, XML, and Java applets.

Most academics feel it is time for the computer science service course to evolve yet again. There are two reasons for this. First, virtually all students in college today are familiar with personal computers and productivity software. They have been using word processors since elementary school and are quite familiar with social networks, online retailing, e-mail, and chat rooms. Many have written Web pages and some even have their own Web sites. In this day and age, a course that focuses on applications of computing will be of little or no interest.

But a more important reason for rethinking the structure of this course, and the primary reason why we authored this book, is the following observation:

> *Most computer science service courses do not teach students about the foundations of computer science!*

We believe quite strongly that students in a computer science service course must receive a solid grounding in the fundamental intellectual concepts of

computer science in addition to learning about important uses of computing and information technology. The material in such a course would not be limited to "fun" applications such as Web page design and interactive graphics but would also cover issues such as algorithms, hardware design, computer organization, system software, language models, theory of computation, and social and ethical issues of computing. An introduction to these core ideas exposes students to the overall richness and beauty of the field. It allows them to not only use computers and software effectively but to understand and appreciate the basic ideas underlying their creation and implementation.

The CS1 Course

The design of a first course for computer science majors has also come in for a great deal of discussion. Since the emergence of computer science as a distinct academic discipline in the 1960s, the first course has always been an introduction to programming—from BASIC to FORTRAN to Pascal, to C++, Java, and Python today. Related topics have been added to the syllabus (e.g., object-oriented design), but the central focus has remained high-level language programming. However, the ACM/IEEE Computing Curriculum 2001 Report suggested a number of alternative models for the first course, including a breadth-first overview, an approach that has gained in popularity in the last couple of years.

A first course for computer science majors using the breadth-first model emphasizes early exposure to the sub-disciplines of the field rather than placing exclusive emphasis on programming. This gives new majors a more complete and well-rounded understanding of their chosen field of study. As stated in the Curriculum 2001 Report, "[introductory] courses that emphasize only this one aspect [programming] fail to let students experience the many other areas and styles of thought that are part of computer science as a whole."

Our book—intended for either majors or non-majors—is organized around this breadth-first approach, and it presents a wide range of subject matter drawn from many areas of computer science. However, to avoid drowning students in a sea of seemingly unrelated facts and details, a breadth-first presentation must be carefully woven into a fabric, a theme, a "big picture" that ties together these topics and presents computer science as a unified and integrated discipline. To achieve this we have divided the study of computer science into a hierarchy of topics, with each layer in the hierarchy building on and expanding upon concepts from earlier chapters.

A Hierarchy of Abstractions

The central theme of this book is that *computer science is the study of algorithms*. Our hierarchy utilizes this definition by first looking at the algorithmic basis of computer science and then moving upward from this central theme to higher-level issues such as hardware, software, applications, and ethics. Just as the chemist starts from protons, neutrons, and electrons and builds up to atoms, molecules, and compounds, so, too, does our text build from elementary concepts such as algorithms, binary arithmetic, gates, and circuits to higher-level ideas such as computer organization, operating systems, high-level languages, applications, and the social, legal, and ethical problems of information technology.

The six levels in our computer science hierarchy are as follows:

Level 1. The Algorithmic Foundations of Computer Science
Level 2. The Hardware World
Level 3. The Virtual Machine
Level 4. The Software World
Level 5. Applications
Level 6. Social Issues in Computing

Following an introductory chapter, Level 1 (Chapters 2–3) introduces "The Algorithmic Foundations of Computer Science," the bedrock on which all other aspects of the discipline are built. It presents important ideas such as the design of algorithms, algorithmic problem solving, abstraction, pseudocode, iteration, and efficiency. It illustrates these ideas using well-known examples such as searching a list, finding maxima and minima, sorting a list, and matching patterns. It also introduces the concepts of algorithm efficiency and asymptotic growth and demonstrates that not all algorithms are, at least in terms of running time, created equal.

The discussions in Level 1 assume that our algorithms are executed by something called a "computing agent," an abstract concept for any entity that can effectively carry out the instructions in our solution. However, in Level 2 (Chapters 4–5), "The Hardware World," we want our algorithms to be executed by "real" computers to produce "real" results. Thus begins our discussion of hardware, logic design, and computer organization. The initial discussion introduces the basic building blocks of computer systems—binary numbers, Boolean logic, gates, and circuits. It then shows how these elementary concepts are used to construct a real computer using the classic Von Neumann architecture, including processors, memory, buses, and input/output. It presents a typical machine language instruction set and explains how the algorithms of Level 1 can be represented in machine language and run on the Von Neumann hardware of Level 2, conceptually tying together these two areas. It ends with a discussion of important new directions in hardware design—multicore and massively parallel machines.

By the end of Level 2 students have been introduced to some basic concepts in logic design and computer organization, and they understand and appreciate the enormous complexity of these areas. This complexity is the motivation for Level 3 (Chapters 6–8), "The Virtual Machine." This section describes how system software produces a more friendly, user-oriented problem-solving environment that hides many of the ugly hardware details just described. Level 3 looks at the same problem discussed in Level 2, encoding and executing an algorithm, but shows how much easier this is in a virtual environment containing software tools like editors, translators, and loaders. This section also discusses the services and responsibilities of operating systems and how operating systems have evolved. It investigates one of the most important virtual environments in current use—a network of computers. It shows how systems such as the Ethernet, Internet, and the Web are created from computers linked together via transmission media and communications software. This creates a virtual environment in which we can seamlessly use not only the computer on our desk but computers located practically anywhere in the world. Level 3 concludes with a look at

one of the most important services provided by a virtual machine, information security, and describes algorithms for protecting the user and the system from accidental or malicious damage.

Once we have created this user-oriented virtual environment, what do we want to do with it? Most likely we want to write programs to solve interesting problems. This is the motivation for Level 4 (Chapters 9–12), "The Software World." Although this book should not be viewed as a programming text, it contains an overview of the features found in modern programming languages. This gives students an appreciation for the interesting and challenging task of the computer programmer and the power of the problem-solving environment created by a modern high-level language. There are many different programming language models, so this level includes a discussion of other language types, including special-purpose languages such as SQL, HTML, and JavaScript, as well as the functional, logic, and parallel language paradigms. This level also describes the design and construction of a compiler and shows how high-level languages can be translated into machine language for execution. This discussion ties together ideas presented in earlier chapters, as we show how an algorithm (Level 1) is translated into a high-level language (Level 4), compiled and executed on a typical Von Neumann machine (Level 2), which makes use of the system software tools of Level 3. These "recurring themes" and frequent references to earlier concepts help reinforce the idea of computer science as an integrated set of related topics. At the conclusion of Level 4, we introduce the idea of computability and unsolvability. A formal model of computing (the Turing machine) is used to prove that there exist problems for which no general algorithmic solution can be found. It shows students that there are provable limits to what computers and computer science can achieve.

We now have a high-level programming environment in which it is possible to write programs to solve important problems. In Level 5 (Chapters 13–16), "Applications," we take a look at a few important uses of computers in our modern society. There is no way to cover even a tiny fraction of the many applications of computers and information technology in a single section. Instead, we focus on a relatively small set that demonstrates some important concepts, tools, and techniques of computer science. This includes applications drawn from the sciences and engineering (simulation and modeling), business and finance (e-commerce, databases), the social sciences (artificial intelligence), and everyday life (computer generated imagery, video gaming, virtual communities). Our goal is not to provide "encyclopedic coverage" of modern computing usage; instead, it is to show students that applications packages are not "magic boxes" whose inner workings are totally unfathomable. Rather, they are the result of utilizing core computer science concepts—e.g., algorithms, hardware, languages—presented in earlier chapters. We hope that our discussions in this section will encourage readers to seek out information on applications and software packages specific to their own areas of interest.

Finally, we reach the highest level of study, Level 6 (Chapter 17), "Social Issues in Computing," which addresses the social, ethical, and legal issues raised by the applications presented in Level 5. This section (written by contributing author Prof. Keith Miller of the University of Illinois at Springfield) examines such thorny problems as the ownership of intellectual property in the electronic age, national security concerns aggravated by information technology, and the erosion of individual privacy caused by the use of online

databases. This section does not attempt to provide quick solutions to these complex problems. Instead, it focuses on techniques that students can use to think about these ethical issues and reach their own conclusions. Our goal in this final section is to make students aware of the enormous impact that information technology is having on everyone's lives and to give them tools that will allow them to make more informed decisions.

This, then, is the hierarchical structure of our text. It begins with the algorithmic foundations of the discipline and works its way from low-level hardware concepts through virtual machine environments, languages, software, and applications to the social issues raised by computer technology. This organizational structure, along with the use of recurring themes, enables students to view computer science as a unified, integrated, and coherent field of study. While the social issues material in Chapter 17 can be presented at any time, the rest of the material is intended to be covered sequentially.

What's New

The fifth edition of *Invitation to Computer Science* represents the single biggest rewrite of this best-selling text. It includes two new chapters that address important emerging areas of computer science. In an age where personal, financial, and medical data is all online, Chapter 8, "Information Security," deals with the growing problem of keeping that data safe from improper access and inappropriate modification. Chapter 16, "Computer Graphics and Entertainment: Movies, Games, and Virtual Communities," looks at how computers, once the domain of the military, government, and business, are now being used to entertain, amaze, and enthrall. It concludes with a discussion of how these same visualization algorithms are also used to address more important problems, such as medical imaging.

In addition to these two chapters, new material and exercises have been added to existing chapters on Computer Organization (multicore and cluster computing), Computer Networks (wireless computing), and Artificial Intelligence (robotics) as well as the addition of new Practice Problems and boxed features.

However, the single biggest change has been to move all programming-language-specific materials, once placed into their own chapter in the text itself, to the Cengage Web site. For the first four editions we produced two distinct versions of the text, one for C++ and the other for Java. As new languages began to enter the computer science curriculum, e.g., Python, Ada, C#, it became infeasible to produce a separate chapter and a separate edition for each one. Instead, Chapter 9, "Introduction to High-Level Language Programming," is now a general description of the features common to modern programming languages. Detailed discussions of a particular language are available to instructors for distribution to students under the Instructor Download section of www.cengage.com. (Currently the Cengage Web site includes online language modules for C++, Java, Python, Ada, and C#, with additional modules possible in the future.) Using this approach we can respond much more quickly to new developments in programming language design as well as proposals for curricular change. In addition, instructors and students are not limited to exposure to a single language but are invited to download (or request from instructors) the modules for any and all languages in which they are interested.

Other Textbook Features

To challenge the more advanced students, each chapter includes, along with a regular set of exercises, some "Challenge Problems." These more complex questions could be used for longer assignments done either individually or by teams of students. Finally, if a student is interested in a topic and wants more detail, there is a section at the end of each chapter titled "For Further Reading" with references to texts and Web sites containing additional material on the topics covered in that chapter.

Summary

Computer science is a young and exciting discipline, and we hope that the material in this text, along with the laboratory projects and online modules, will convey this feeling of excitement. By presenting the field in all its richness—algorithms, hardware, software, systems, applications, and social issues—we hope to give students a deeper appreciation for the many diverse and interesting areas of research and study within the discipline of computer science.

Reviewers

The following reviewers, along with the many users of previous editions who have provided helpful comments, have contributed to the writing of this new edition, and we want to thank them all:

James Aman
Saint Xavier University

Phillip Barry
University of Minnesota

Robert Beasley
Franklin College

Doug Edwards
Central Texas College

S. Jane Fritz
St. Joseph's College—New York

Barry Kolb
Ocean County College

Mike Scherger
Texas A&M University, Corpus Christi

Stewart Shen
Old Dominion University

—G. Michael Schneider
Macalester College
schneider@macalester.edu

—Judith L. Gersting
University of Hawaii - Hilo
gersting@hawaii.edu

CONTENTS