

SCHAUM'S
ouTlines

PRINCIPLES OF COMPUTER SCIENCE

CARL REYNOLDS, Ph.D.

PAUL TYMANN, M.S.

—
Introductions to the theoretical
foundations of computer science

—
250 fully solved problems

—
Hundreds more practice
problems with answers

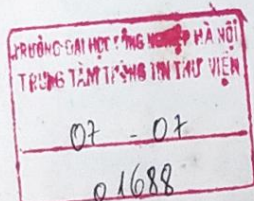
**TRUSTED BY
MORE THAN
40 MILLION
STUDENTS**

Use with these courses: ☒ Introduction to Computer Science

☒ Theory of Computation ☒ Computer Ethics

**SCHAUM'S
OUTLINE OF**

Principles of **COMPUTER SCIENCE**



CARL REYNOLDS

*Department of Computer Science
Rochester Institute of Technology*

PAUL TYMANN

*Department of Computer Science
Rochester Institute of Technology*

**GIFT OF THE ASIA FOUNDATION
NOT FOR RE-SALE**

**QUÀ TẶNG CỦA QUỸ CHÂU Á
KHÔNG ĐƯỢC BÁN LẠI**

Schaum's Outline Series

McGRAW-HILL

New York | Chicago | San Francisco | Lisbon | London | Madrid
Mexico City | Milan | New Delhi | San Juan
Seoul | Singapore | Sydney | Toronto

CARL REYNOLDS teaches courses in database, operating systems, programming, and programming language theory in the Computer Science Department at the Rochester Institute of Technology. He has taught at the college level for 10 years, and in the computer industry for 4 years. Before coming to RIT, Dr. Reynolds spent 19 years in the computer industry working in technical and training capacities for both hardware and software suppliers, and 6 years with a Dow Jones Industrial manufacturer creating expert systems for machine control. His interests include genetic algorithms, expert systems, and image processing.

PAUL TYMANN is Professor and Chair of the Computer Science Department at the Rochester Institute of Technology. He has taught both basic and advanced programming techniques for over 15 years. More recently he has been involved with development of a new bioinformatics program at RIT. Prior to entering academia, Professor Tymann worked in industry developing control software for point-of-sale terminals. For the past 5 years he has worked in the area of bioinformatics and has completed joint software development projects at the University of Rochester and Rutgers University.

The McGraw-Hill Companies

Schaum's Outline of Principles of
COMPUTER SCIENCE

Copyright © 2008 by The McGraw-Hill Companies, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

1 2 3 4 5 6 7 8 9 0 CUS/CUS 0 1 4 3 2 1 0 9 8

ISBN 978-0-07-146051-4
MHID 0-07-146051-9

Library of Congress Cataloging-in-Publication Data

Reynolds, Carl, date.

Schaum's outline of principles of computer science / Carl Reynolds, Paul Tymann.

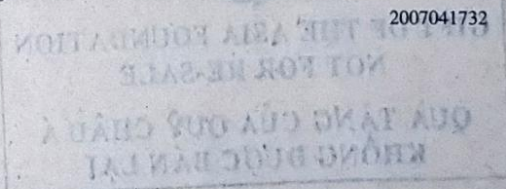
p. cm.—(Schaum's outline series)

Includes index.

ISBN 978-0-07-146051-4

I. Computer science—Problems, exercises, etc. I. Tymann, Paul T. II. Title.
QA76.28.R49 2008
004.076—dc22

2007041732



CONTENTS

| | | |
|------------------|---|-----------|
| CHAPTER 1 | Introduction to Computer Science | 1 |
| | What Is Computer Science? | 1 |
| | Computing History | 4 |
| | A Roadmap | 12 |
| | Review Questions | 13 |
| CHAPTER 2 | Algorithms | 14 |
| | Definition of Algorithm | 14 |
| | Example—Designing a Staircase | 14 |
| | Example—Finding the Greatest Common Denominator | 15 |
| | Representing Algorithms with Pseudocode | 15 |
| | Characterizing Algorithms | 16 |
| | Analyzing Algorithms | 17 |
| | Algorithms as Technology | 25 |
| | Formal Models of Computation | 25 |
| | Church–Turing Thesis | 28 |
| | Unsolvable Problems | 28 |
| | Summary | 29 |
| | Review Questions | 29 |
| CHAPTER 3 | Computer Organization | 31 |
| | Von Neumann Architecture | 31 |
| | Data Representation | 31 |
| | Computer Word Size | 32 |
| | Integer Data Formats | 33 |
| | Real Number Formats | 34 |
| | Character Formats | 35 |
| | CPU/ALU | 36 |
| | Instruction Set | 36 |
| | Memory | 39 |
| | Input and Output (I/O) | 40 |
| | Summary | 41 |
| | Review Questions | 42 |
| CHAPTER 4 | Software | 44 |
| | Generations of Languages | 44 |
| | Compilers and Interpreters | 47 |

| | | |
|------------------|---|-----------|
| | Virtual Machines | 49 |
| | Procedural Programming | 49 |
| | Object-Oriented Programming | 51 |
| | Scripting Languages | 53 |
| | Functional Languages | 56 |
| | Language Design | 59 |
| | Language Syntax and Semantics | 61 |
| | Summary | 64 |
| | Review Questions | 64 |
| CHAPTER 5 | Programming in Java | 66 |
| | Introduction | 66 |
| | Java Types | 66 |
| | Arrays | 69 |
| | Java Operators | 70 |
| | Java Identifiers | 71 |
| | Basic Control Structures | 71 |
| | Object-Oriented Programming | 77 |
| | Classes and Objects | 78 |
| | Object State and Behavior | 78 |
| | Inheritance | 79 |
| | Instance and Static Variables and Methods | 79 |
| | Polymorphism | 83 |
| | Interfaces | 84 |
| | Error Handling | 85 |
| | Input and Output | 88 |
| | Scanner | 91 |
| | PrintWriter | 92 |
| | Summary | 93 |
| | Review Questions | 93 |
| CHAPTER 6 | Operating Systems | 95 |
| | Capabilities of the Hardware | 95 |
| | Operating Systems History | 97 |
| | Single-user OS → Network OS | 98 |
| | Multiprocessor Operating Systems | 98 |
| | Real-Time Operating Systems | 99 |
| | Embedded Systems | 99 |
| | Management of Input and Output | 99 |
| | Processes and Scheduling | 101 |
| | Threads | 102 |
| | Synchronization | 103 |
| | Semaphores | 106 |
| | Monitors | 108 |
| | Deadlock | 111 |
| | Scheduling | 114 |
| | Memory Management | 116 |
| | Memory Management with Multiprogramming | 116 |
| | Timesharing and Swapping | 117 |
| | Virtual Memory | 118 |

| | | |
|------------------|--|------------|
| | File Systems | 122 |
| | Summary | 124 |
| | Review Questions | 125 |
| CHAPTER 7 | Networking | 127 |
| | Introduction | 127 |
| | Reference Model | 128 |
| | Subnet (Data-Link) Layer | 130 |
| | Internet (Network) Layer Protocol | 131 |
| | End-to-end (Transport) Layer Protocol | 132 |
| | Application Layer | 133 |
| | Putting It All Together | 133 |
| | The World Wide Web, HTTP, and HTML | 134 |
| | Summary | 137 |
| | Review Questions | 138 |
| CHAPTER 8 | Database | 139 |
| | The Ubiquitous Database | 139 |
| | Database Types | 139 |
| | Advantages of Using a Database | 140 |
| | Modeling the Data Domain | 140 |
| | Building a Relational Database from the Data Model | 144 |
| | Normalization | 145 |
| | SQL—Structured Query Language | 147 |
| | DDL—Data Definition Language | 148 |
| | DML—Data Manipulation Language | 150 |
| | Stored Procedures | 157 |
| | Triggers | 159 |
| | Data Integrity | 160 |
| | Transaction Isolation Levels | 161 |
| | Accessing the Database Programmatically | 162 |
| | Summary | 165 |
| | Review Questions | 166 |
| CHAPTER 9 | Social Issues | 169 |
| | Ethics Theories | 169 |
| | Intellectual Property | 170 |
| | Privacy | 174 |
| | Encryption | 175 |
| | Viruses, Worms, and Trojan Horses | 175 |
| | Hackers | 176 |
| | Can Computers Kill? | 177 |
| | Summary | 178 |
| | Review Questions | 179 |
| APPENDIX | Answers to Review Questions | 180 |
| | Index | 217 |

CHAPTER 1

Introduction to Computer Science

WHAT IS COMPUTER SCIENCE?

Computer Science is defined in different ways by different authors. Wikipedia (http://en.wikipedia.org/wiki/Computer_science) defines computer science as the collection of a variety of disciplines related to computing, both theoretical and practical: theoretical foundations of information and computation, language theory, algorithm analysis and development, implementation of computing systems, computer graphics, databases, data communications, etc.

The US National Coordination Office for *Networking and Information Technology Research and Development* (NITRD) defines computer science in a similarly broad way:

the systematic study of computing systems and computation. The body of knowledge resulting from this discipline contains theories for understanding computing systems and methods; design methodology, algorithms, and tools; methods for the testing of concepts; methods of analysis and verification; and knowledge representation and implementation. (<http://www.nitrd.gov/pubs/bluebooks/1995/section.5.html>)

Another broad definition comes from the *Association for Computing Machinery* (ACM) Model Curriculum. It says that computer science is the “study of computers and algorithmic processes, including their principles, their hardware and software design, their applications, and their impact on society.”

A famous definition of computer science by Gibbs and Tucker (Gibbs and Tucker, “A Model Curriculum for a Liberal Arts Degree in Computer Science,” *Comm. of the ACM*, vol. 29, no. 3, March 1986) emphasizes algorithm development and analysis as the central focus of computer science.

It's also a fair question to ask, “How is computer science a science?” In contrast to physics, biology, and chemistry, computer science is not based on the study of the natural world. In that sense, computer science is more like mathematics than science. Some argue that computer science is really computer art (where “art” means practice). On the other hand, computer scientists do use the scientific method to propose and test hypotheses, and some very nonobvious discoveries in computer science have important real-world implications. An example, which we will discuss later, is the discovery that some important problems simply cannot be solved by computation.

Despite many variations, essentially all definitions of computer science emphasize the study of algorithms. Algorithms, in one form or another, are central to computer science. Computer science combines the theoretical concepts of algorithm design and analysis with the practical considerations of how to implement algorithms on a computer and solve practical problems.