GLOBAL EDITION

Problem Solving and Program Design in C

EIGHTH EDITION

Jeri R. Hanly • Elliot B. Koffman

ALWAYS LEARNING

PEARSON

EIGHTH EDITION
GLOBAL EDITION

PROBLEM SOLVING AND PROGRAM DESIGN in C

this page intentionally left blank

EIGHTH EDITION
GLOBAL EDITION

PROBLEM SOLVING AND PROGRAM DESIGN in C

Jeri R. Hanly, University of Wyoming
Elliot B. Koffman, Temple University

Global Edition Contributions by
Mohit P. Tahiliani, National Institute of Technology, Surathkal

PEARSON

Boston Columbus Indianapolis New York San Francisco Hoboken
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto Delhi
Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Vice President and Editorial Director, ECS: Marcia Horton
Executive Editor: Tracy Johnson
Product Marketing Manager: Bram van Kempen
Field Marketing Manager: Demetrius Hall
Marketing Assistant: Jon Bryant
Project Management Team Lead: Scott Disanno
Program Manager: Carole Snyder
Senior Project Manager: Camille Trentacoste
Assistant Acquisitions Editor, Global Edition: Aditee Agarwal
Associate Project Editor, Global Edition: Sinjita Basu
Media Production Manager, Global Edition: Vikram Kumar
Senior Manufacturing Controller, Production, Global Edition: Trudy Kimber
Operations Supervisor: Vincent Scelta
Operations Specialist: Maura Zaldivar-Garcia
Full-Service Project Management: Cenveo® Publisher Services
Cover Design: Lumina Datamatics
Cover Photo Source: Shutterstock
Cover Printer: Courier Kendallville

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: 1-292-09881-3
ISBN 13: 978-1-292-09881-4

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

10 9 8 7 6 5 4 3 2 1
14 13 12 11 10

Typeset in New Caledonia 10/12 by Cenveo® Publisher Services.

Printed and bound by Courier Kendallville in the United States of America.

This book is dedicated to

Jeri Hanly's family:
Brian, Kevin, Laura, Grace, and Caleb
Trinity, Alex, Eva, Eli, and Jonah
Eric, Jennifier, Mical, Micah, Josiah, and Rachel

Elliot Koffman's family:
Caryn and Deborah
Richard, Jacquie, and Dustin
Robin, Jeffrey, Jonathan, and Eliana

this page intentionally left blank

PREFACE

Problem Solving and Program Design in C teaches a disciplined approach to problem solving, applying widely accepted software engineering methods to design program solutions as cohesive, readable, reusable modules. We present as an implementation vehicle for these modules a subset of ANSI C—a standardized, industrial-strength programming language known for its power and portability. This text can be used for a first course in programming methods: It assumes no prior knowledge of computers or programming. The text's broad selection of case studies and exercises allows an instructor to design an introductory programming course in C for computer science majors or for students from a wide range of other disciplines.

New to This Edition

Several changes to this edition are listed below. The majority of these changes are in response to the recommendations of our reviewers.

- Chapter 0 information on professional opportunities in computing has been extensively updated.
- Hardware examples in Chapter 1 have been brought up-to-date to reflect current technology.
- Discussion of programming languages in Chapter 1 has been revised to list the most popular languages in use today.
- Chapters on arrays, strings, files, and dynamic data structures have been renamed and reworked to place greater emphasis on the use of pointers.
- Chapter 6 coverage of levels of testing has been updated.
- All chapters contain new programming project problems, and beginning with Chapter 5, some projects are marked as especially appropriate for team programming.
- Three chapters contain new "C in Focus" articles—"Team Programming" (Chapter 5), "Defensive Programming" (Chapter 8), and "Evolving Standards" (Chapter 10).
- Format of many tables, including those that trace execution of code, has been altered to improve readability.
- Exercises on bitwise operations have been added to Appendix C.

Using C to Teach Program Development

Two of our goals—teaching program design and teaching C—may be seen by some as contradictory. C is widely perceived as a language to be tackled only after one has learned the fundamentals of programming in some other, friendlier language. The perception that C is excessively difficult is traceable to the history of the language. Designed as a vehicle for programming the UNIX operating system, C found its original clientele among programmers who understood the complexities of the operating system and the underlying machine and who considered it natural to exploit this knowledge in their programs. Therefore, it is not surprising that many textbooks whose primary goal is to teach C expose the student to program examples requiring an understanding of machine concepts that are not in the syllabus of a standard introductory programming course.

In this text, we are able to teach both a rational approach to program development and an introduction to ANSI C because we have chosen the first goal as our primary one. One might fear that this choice would lead to a watered-down treatment of ANSI C. On the contrary, we find that the blended presentation of programming concepts and of the implementation of these concepts in C captures a focused picture of the power of ANSI C as a high-level programming language, a picture that is often blurred in texts whose foremost objective is the coverage of all of ANSI C. Even following this approach of giving program design precedence over discussion of C language features, we have arrived at coverage of the essential constructs of C that is quite comprehensive.

Pointers and the Organization of the Book

The order in which C language topics are presented is dictated by our view of the needs of the beginning programmer rather than by the structure of the C programming language. The reader may be surprised to discover that there are multiple chapter titles that include the word “Pointers.” This follows from our treatment of C as a high-level language, and our recognition that the critical role of pointers in C is often a challenging concept for students to grasp.

Whereas other high-level languages have separate language constructs for output parameters and arrays, C openly folds these concepts into its notion of a pointer, drastically increasing the complexity of learning the language. We simplify the learning process by discussing pointers from these separate perspectives where such topics normally arise when teaching other programming languages, thus allowing a student to absorb the intricacies of pointer usage a little at a time. Our approach makes possible the presentation of fundamental concepts using traditional high-level language terminology—output parameter, array, array subscript, string—and makes it easier for students without prior assembly language background to master the many aspects of pointer usage. This approach is also helpful for students studying C as a second programming language, since it facilitates their understanding of the many aspects of pointer use as simply C’s means of implementing constructs they have already met.

Therefore, this text has not one but five chapters that discuss pointer concepts. Chapter 6 (Pointers and Modular Programming) begins with a discussion of pointers, indirect reference, and the use of pointers to files. It then discusses the use of pointers as simple output and input/output parameters, Chapter 7 deals with arrays, Chapter 8 presents strings and arrays of pointers. Chapter 11 discusses file pointers again. Chapter 13 describes dynamic memory allocation after reviewing pointer uses previously covered.

Software Engineering Concepts

The book presents many aspects of software engineering. Some are explicitly discussed and others are taught only by example. The connection between good problem-solving skills and effective software development is established early in Chapter 1 with a section that discusses the art and science of problem solving. The five-phase software development method presented in Chapter 1 is used to solve the first case study and is applied uniformly to case studies throughout the text. Major program style issues are highlighted in special displays, and the coding style used in examples is based on guidelines followed in segments of the C software industry. There are sections in several chapters that discuss algorithm tracing, program debugging, and testing.

Chapter 3 introduces procedural abstraction through selected C library functions, parameterless void functions, and functions that take input parameters and return a value. Chapters 4 and 5 include additional function examples including the use of a function as a parameter, and Chapter 6 completes the study of functions that have simple parameters. The chapter discusses the use of pointers to represent output and input/output parameters.

Case studies and sample programs in Chapters 6, 7, and 10 introduce by example the concepts of data abstraction and encapsulation of a data type and operators. Chapter 12 presents C's facilities for formalizing procedural and data abstraction in personal libraries defined by separate header and implementation files. Chapter 14 (on the textbook website) introduces essential concepts of multiprocessing, such as parent and child processes, interprocess communication, mutual exclusion locking, and deadlock avoidance. Chapter 15 (on the textbook website) describes how object-oriented design is implemented by C++.

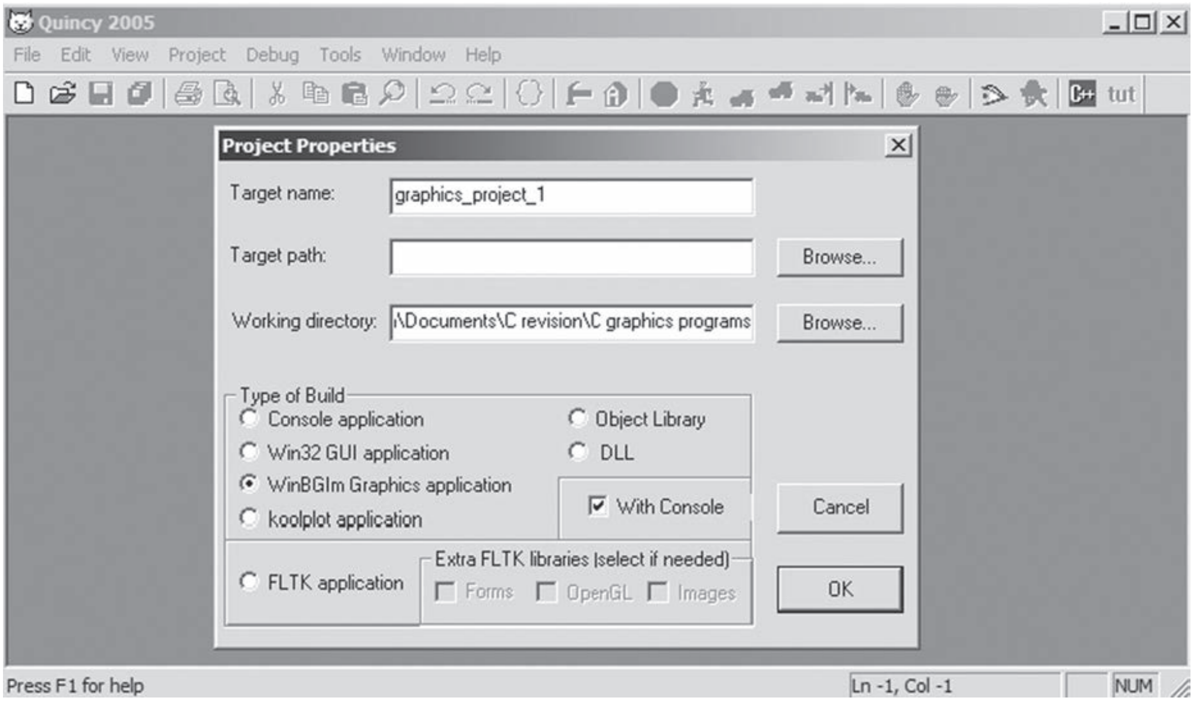
The use of visible function interfaces is emphasized throughout the text. We do not mention the possibility of using a global variable until Chapter 12, and then we carefully describe both the dangers and the value of global variable usage.

Optional Graphics Sections

Many computer science faculty find that the use of graphics is an excellent motivator in the study of introductory programming and as a vehicle to help students understand how to use libraries and to call functions. This text offers three optional sections with graphics examples:

Section 3.6:	Introduction to Computer Graphics
Section 5.11:	Loops in Graphics Programs
Section 7.10:	Graphics Programs with Arrays

To reduce the overhead required to introduce graphics, we use WinBGIm (Windows BGI with mouse), which is a package based on the Turbo Pascal BGI



Project Properties

Target name: graphics_project_1

Target path:

Working directory: \Documents\C revision\C graphics programs

Type of Build

- ( ) Console application
- ( ) Win32 GUI application
- (●) WinBGIm Graphics application
- ( ) koolplot application
- ( ) Object Library
- ( ) DLL
- [✓] With Console
- ( ) FLTK application

Extra FLTK libraries (select if needed)

- [ ] Forms
- [ ] OpenGL
- [ ] Images

FIGURE 1

(Borland Graphics Interface) library. WinBGIm was created to run on top of the Win32 library by Michael Main and his students at the University of Colorado. Several development platforms appropriate for CS 1 courses have incorporated WinBGIm. Quincy (developed by Al Stevens) is an open-source student-oriented C++ IDE that includes WinBGIm as well as more advanced libraries (http://www.codecutter.net/tools/quincy). Figure 1 shows the Quincy new project window (File → New → Project) with WinBGIm Graphics application selected.

A command-line platform based on the open-source GNU g++ compiler and the emacs program editor is distributed by the University of Colorado (http://www.codecutter.net/tools/winbgim). WinBGIm is also available for Bloodshed Software's Dev-C++ and Microsoft's Visual Studio C++.

Pedagogical Features

We employ the following pedagogical features to enhance the usefulness of this book as a learning tool:

End-of-Section Exercises Most sections end with a number of Self-Check Exercises. These include exercises that require analysis of program fragments as well as short programming exercises.

Examples and Case Studies The book contains a wide variety of programming examples. Whenever possible, examples contain complete programs or functions rather than incomplete program fragments. Each chapter contains one or more substantial case studies that are solved following the software development method. Numerous case studies give the student glimpses of important applications of computing, including database searching, business applications such as billing and sales analysis, word processing, and environmental applications such as radiation level monitoring and water conservation.

Syntax Display Boxes The syntax displays describe the syntax and semantics of new C features and provide examples.

Program Style Displays The program style displays discuss major issues of good programming style.

Error Discussions and Chapter Review Each chapter concludes with a section that discusses common programming errors. The Chapter Review includes a table of new C constructs.

End-of-Chapter Exercises Quick-Check Exercises with answers follow each Chapter Review. There are also review exercises available in each chapter.

End-of-Chapter Projects Each chapter ends with Programming Projects giving students an opportunity to practice what they learned in the chapter.

Appendices

Reference tables of ANSI C constructs appear on the inside covers of the book. Because this text covers only a subset of ANSI C, the appendices play a vital role in increasing the value of the book as a reference. Throughout the book, array referencing is done with subscript notation; Appendix A is the only coverage of pointer arithmetic. Appendix B is an alphabetized table of ANSI C standard libraries. The table in Appendix C shows the precedence and associativity of all ANSI C operators; the operators not previously defined are explained in this appendix. Exercises for practicing some of the bitwise operators are included. Appendix D presents character set tables, and Appendix E lists all ANSI C reserved words.

Supplements

The following supplemental materials are available to all readers of this book at www.pearsonglobaleditions.com/Hanly:

- Source code
- Known errata
- Answers to odd-numbered Self-Check exercises.

The following instructor supplement is available only to qualified instructors at the Pearson Instructor Resource Center. Visit www.pearsonglobaleditions.com/Hanly or contact your local Pearson sales representative to gain access to the IRC.

- Solutions Manual

Acknowledgments

Many people participated in the development of this textbook. The reviewers for this edition, who suggested most of the changes, include Michael Geiger, UMASS Lowell, Lowell, MA; Qi Hao, University of Alabama, Tuscaloosa, AL; Haibing Lu, Santa Clara University, Santa Clara, CA; Susan Mengel, Texas Tech University, Lubbock, TX; Shensheng Tang, Missouri Western State University, St. Joseph, MO; Kevin Mess, College of Southern Nevada, Las Vegas, NV; Samir Iabbassen, Long Island University, Brooklyn, NY; and Ray Lauff, Temple University, Philadelphia, PA. We would also like to acknowledge Michael Main for his assistance with the graphics examples and his students at the University of Colorado who adapted WinBGI to create WinBGIm (Grant Macklem, Gregory Schmelter, Alan Schmidt, and Ivan Stashak).

We also want to thank Charlotte Young of South Plains College for her help in creating Chapter 0, and Jeff Warsaw of WaveRules, LLC, who contributed substantially to Chapter 14. Joan C. Horvath of the Jet Propulsion Laboratory, California Institute of Technology, contributed several programming exercises, and Nelson Max of the University of California, Davis suggested numerous improvements to the text. Jeri acknowledges the assistance of her former colleagues at Loyola University Maryland—James R. Glenn, Dawn J. Lawrie, and Roberta E. Sabin—who contributed several programming projects. We are also grateful for the assistance over the years of several Temple University, University of Wyoming, and Howard University former students who helped to verify the programming examples and who provided answer keys for the host of exercises, including Mark Thoney, Lynne Doherty, Andrew Wrobel, Steve Babiak, Donna Chrupcala, Masoud Kermani, Thayne Routh, and Paul Onakoya.

It has been a pleasure to work with the Pearson team in this endeavor. Tracy Johnson (Executive Editor), Carole Snyder (Program Manager), and Camille Trentacoste (Senior Project Manager) provided ideas and guidance throughout the various phases of manuscript revision.

J.R.H.
E.B.K.

Pearson wishes to thank and acknowledge the following reviewers for their work on the Global Edition:

Vikas Deep Dhiman, *Amity University*

Ll Xin Cindy, *The Hong Kong University of Science and Technology*

Piyali Sengupta, *Freelance Writer*

CONTENTS

0. Computer Science as a Career Path 21

Section 1 Why Computer Science May be the Right Field for You 22
Section 2 The College Experience: Computer Disciplines and Majors to Choose From 24
Section 3 Career Opportunities 29

1. Overview of Computers and Programming 33

1.1 Electronic Computers Then and Now 34
1.2 Computer Hardware 37
1.3 Computer Software 45
1.4 The Software Development Method 52
1.5 Applying the Software Development Method 56
Case Study: Converting Miles to Kilometers 56
1.6 Professional Ethics for Computer Programmers 59
Chapter Review 61

2. Overview of C 65

2.1 C Language Elements 66
2.2 Variable Declarations and Data Types 73
2.3 Executable Statements 79
2.4 General Form of a C Program 89
2.5 Arithmetic Expressions 92
Case Study: Supermarket Coin Processor 102
2.6 Formatting Numbers in Program Output 107
2.7 Interactive Mode, Batch Mode, and Data Files 110
2.8 Common Programming Errors 113
Chapter Review 119

3. Top-Down Design with Functions 127

3.1 Building Programs from Existing Information 128
Case Study: Finding the Area and Circumference of a Circle 129
Case Study: Computing the Weight of a Batch of Flat Washers 132
3.2 Library Functions 137
3.3 Top-Down Design and Structure Charts 144
Case Study: Drawing Simple Diagrams 144
3.4 Functions Without Arguments 146
3.5 Functions with Input Arguments 156
3.6 Introduction to Computer Graphics (Optional) 166
3.7 Common Programming Errors 183
Chapter Review 184

4. Selection Structures: if and switch Statements 193

4.1 Control Structures 194
4.2 Conditions 195
4.3 The if Statement 205
4.4 if Statements with Compound Statements 211
4.5 Decision Steps in Algorithms 214
Case Study: Water Bill Problem 215
4.6 More Problem Solving 224
Case Study: Water Bill with Conservation Requirements 225
4.7 Nested if Statements and Multiple-Alternative Decisions 227
4.8 The switch Statement 237
C in Focus: The Unix Connection 241
4.9 Common Programming Errors 243
Chapter Review 244

5. Repetition and Loop Statements 255

5.1 Repetition in Programs 256
5.2 Counting Loops and the while Statement 258
5.3 Computing a Sum or a Product in a Loop 262
5.4 The for Statement 267
5.5 Conditional Loops 276
5.6 Loop Design 281
5.7 Nested Loops 288
5.8 The do-while Statement and Flag-Controlled Loops 293
5.9 Iterative Approximations 296
Case Study: Bisection Method for Finding Roots 298

5.10 How to Debug and Test Programs 307
C in Focus: Team Programming 309
5.11 Loops in Graphics Programs (Optional) 311
5.12 Common Programming Errors 318
Chapter Review 321

6. Pointers and Modular Programming 337

6.1 Pointers and the Indirection Operator 338
6.2 Functions with Output Parameters 342
6.3 Multiple Calls to a Function with Input/Output Parameters 350
6.4 Scope of Names 356
6.5 Formal Output Parameters as Actual Arguments 358
6.6 Problem Solving Illustrated 362
Case Study: Collecting Area for Solar-Heated House 362
Case Study: Arithmetic with Common Fractions 369
6.7 Debugging and Testing a Program System 378
6.8 Common Programming Errors 381
Chapter Review 381

7. Array Pointers 397

7.1 Declaring and Referencing Arrays 398
7.2 Array Subscripts 401
7.3 Using for Loops for Sequential Access 403
7.4 Using Array Elements as Function Arguments 408
7.5 Array Arguments 410
7.6 Searching and Sorting an Array 423
7.7 Parallel Arrays and Enumerated Types 428
7.8 Multidimensional Arrays 436
7.9 Array Processing Illustrated 441
Case Study: Summary of Hospital Revenue 441
7.10 Graphics Programs with Arrays (Optional) 450
7.11 Common Programming Errors 459
Chapter Review 460

8. Strings 475

8.1 String Basics 476
8.2 String Library Functions: Assignment and Substrings 482
8.3 Longer Strings: Concatenation and Whole-Line Input 491

8.4 String Comparison 496
C in Focus: Defensive Programming 498
8.5 Arrays of Pointers 500
8.6 Character Operations 506
8.7 String-to-Number and Number-to-String Conversions 511
8.8 String Processing Illustrated 518
Case Study: Text Editor 518
8.9 Common Programming Errors 527
Chapter Review 529

9. Recursion 541

9.1 The Nature of Recursion 542
9.2 Tracing a Recursive Function 548
9.3 Recursive Mathematical Functions 556
9.4 Recursive Functions with Array and String Parameters 562
Case Study: Finding Capital Letters in a String 562
Case Study: Recursive Selection Sort 565
9.5 Problem Solving with Recursion 569
Case Study: Operations on Sets 569
9.6 A Classic Case Study in Recursion: Towers of Hanoi 577
9.7 Common Programming Errors 582
Chapter Review 584

10. Structure and Union Types 591

10.1 User-Defined Structure Types 592
10.2 Structure Type Data as Input and Output Parameters 598
10.3 Functions Whose Result Values Are Structured 604
C in Focus: Evolving Standards 606
10.4 Problem Solving with Structure Types 608
Case Study: A User-Defined Type for Complex Numbers 608
10.5 Parallel Arrays and Arrays of Structures 616
Case Study: Universal Measurement Conversion 618
10.6 Union Types (Optional) 627
10.7 Common Programming Errors 634
Chapter Review 634

11. Text and Binary File Pointers 649

11.1 Input/Output Files: Review and Further Study 650
11.2 Binary Files 660

11.3 Searching a Database 666
Case Study: Database Inquiry 667
11.4 Common Programming Errors 676
Chapter Review 677

12. Programming in the Large 685

12.1 Using Abstraction to Manage Complexity 686
12.2 Personal Libraries: Header Files 689
12.3 Personal Libraries: Implementation Files 694
12.4 Storage Classes 697
12.5 Modifying Functions for Inclusion in a Library 701
12.6 Conditional Compilation 704
12.7 Arguments to Function main 708
12.8 Defining Macros with Parameters 711
12.9 Common Programming Errors 716
Chapter Review 717

13. Pointers and Dynamic Data Structures 725

13.1 Pointers 726
13.2 Dynamic Memory Allocation 731
13.3 Linked Lists 736
13.4 Linked List Operators 742
13.5 Representing a Stack with a Linked List 747
13.6 Representing a Queue with a Linked List 751
13.7 Ordered Lists 757
Case Study: Maintaining an Ordered List of Integers 758
13.8 Binary Trees 769
13.9 Common Programming Errors 779
Chapter Review 780

14. Multiprocessing Using Processes and Threads
(Online at www.pearsonglobaleditions.com/Hanly)

14.1 Multitasking
14.2 Processes
14.3 Interprocess Communications and Pipes
14.4 Threads
14.5 Threads Illustrated
Case Study: The Producer/Consumer Model

14.6 Common Programming Errors
Chapter Review

15. On to C++
(Online at www.pearsonglobaleditions.com/Hanly)

15.1 C++ Control Structures, Input/Output, and Functions
15.2 C++ Support for Object-Oriented Programming
Chapter Review

Appendices

A More about Pointers 789
B ANSI C Standard Libraries 795
C C Operators 813
D Character Sets 819
E ANSI C Reserved Words 821

Answers to Odd-Numbered Self-Check Exercises
(Online at www.pearsonglobaleditions.com/Hanly)

Glossary 823

Index 829

EIGHTH EDITION
GLOBAL EDITION

PROBLEM SOLVING AND PROGRAM DESIGN in C

this page intentionally left blank

CHAPTER

0

Computer Science as a Career Path

CHAPTER OBJECTIVES

- To learn why computer science may be the right field for you
- To become familiar with different computer disciplines and related college majors
- To find out about career opportunities

Introduction

In order to choose a course of study and eventually a desirable career path, we may ask many important questions. Why would we choose this field? Will we be good at it? Will there be jobs for us when we finish our education? Will we enjoy our work? This chapter sheds some light on these types of questions for anyone contemplating a degree in computer science or a related field.

Section 1 Why Computer Science May Be the Right Field for You

Reasons to Major in Computer Science

Millennials Those born from 1982 on are said to be confident, social and team-oriented, proud of achievement, prone to use analytic skills to make decisions, and determined to seek security, stability, and balance for themselves

Almost everything we do is influenced by computing. Today's generation of college students, dubbed the **Millennials**, are not surprised by this statement. They have grown up with computers, the Internet, instant communication, social networking, and electronic entertainment. They embrace new technology and expect it to do fantastic things.

However, previous generations are not as comfortable with technology and try to solve problems without always thinking of technology first. Many people in the workforce resist the changes that technology requires. They often turn to the youngest employees to take over technology issues and to make choices that will have important consequences.

This difference among generations creates a great environment for bright and dedicated students to choose to major in computer science or a related field. The computer industry is one of the fastest-growing segments of our economy and promises to continue to see growth well into the future. In order to be competitive, businesses must continue to hire well-trained professionals not only to produce high-quality products for the present, but also to plan creative scientific and engineering advances for the future.

A person who is part of the computer industry can choose from a wide variety of fields where many interesting and challenging problems will need to be solved. In addition to all the business and communication jobs that may first come to mind, people with degrees in computer science are working on problems from all spectrums of life. A quick review of technical articles highlights such areas as developing electronic balloting for state and national elections, using signals from wireless devices to update vehicle and pedestrian travel times in order to make the best decisions for traffic signals or management of construction zones, and using a supercomputer-powered "virtual earthquake" to study benefits of an early warning system using 3-D models of actual geographic locations and damage scenarios.

Some problems being worked on right now by computer professionals in the medical world include understanding how the human brain works by modeling brain activation patterns with emphasis on helping people impacted by autism or disorders like paranoid schizophrenia; customizing a wide array of helpful devices for the physically impaired, from programmable robotic prostheses to digital "sight"; gathering information from implanted pacemakers in order to make timely decisions in times of crisis; developing a computer system capable of recognizing human emotional states by analyzing a human face in real time; and developing human–computer interfaces that allow a computer to be operated solely by human gestures in order to manipulate virtual objects.

The fields of security and law enforcement present many challenges to the computer professional, and include the following: The U.S. government is performing observational studies on normal behavior in online worlds in hopes of developing techniques for uncovering online activities of terrorist groups. Advancements in voice biometrics technology allow speech to be analyzed by computer software to determine identity, truthfulness, and emotional states. Electronic protection against malicious software is of great concern to national economies and security interests.

Some of our world's most challenging problems will be worked on by teams of professionals from many disciplines. Obviously, these teams will include computer professionals who are creative and possess the knowledge of how to best use technology. In the near future, we will see much innovation in the areas of the human genome project, environmental monitoring, AIDS vaccine research, clean fuels, tracking weather changes by using robots in potentially dangerous areas, and using supercomputers to simulate the earth's architecture and functions in order to predict natural disasters. A way to make a positive difference in the world would be to study computing, either as your primary focus or as a means of advancing technology in another field.

Traits of a Computer Scientist

An individual's personality and character traits typically influence the field he or she chooses to study and eventually in which he or she will work. The demands of certain fields are met by individuals with certain capabilities. It makes sense that people who are successful computer science students will have many common traits. Read the following description and decide if it sounds like you.

Foremost, you must love the challenge of solving problems. Computer science is more about finding solutions to problems than it is about using the current computer hardware or programming language. Solving problems requires being creative and "thinking outside the box." You must be willing to try things that are different from the "accepted" solution.

You enjoy working with technology and enjoy being a lifelong learner. You enjoy puzzles and work tenaciously to find solutions. You probably don't even notice that the hours have flown by as you are narrowing in on the answer. You enjoy building things, both in the actual world and in a "virtual world." You can see how to customize a particular object to make it work in a specific environment. You like to tackle large projects and see them to completion. You like to build things that are useful to people and that will have a positive impact on their lives.

To be successful in the workplace, you must also be a good communicator. You should be able to explain your plans and solutions well to both technical and non-technical people. You must be able to write clearly and concisely in the technical environment. Since most projects involve multiple people, it is important to work well in a group. If you plan to become a manager or run your own company, it is very important to be able to work with different personalities.

Frederick P. Brooks, famous for leading the team that developed the operating system for the IBM System/360, wrote a book in the 1970s titled *The Mythical Man Month—Essays on Software Engineering*. Even though much has changed in the computing world since he wrote the book, his essays still hold a lot of relevance today. He listed the "Joys of the Craft" as the following: First is the sheer joy of making things of your own design. Second is the pleasure of making things that are useful to and respected by other people. Third is the joy of fashioning complex puzzle-like entities into a system that works correctly. Fourth is the joy of always learning because of the nonrepetitive nature of the work. Finally, there is the joy of working with a very tractable medium. The programmer can create in his or her imagination and readily produce a product that can be tested and easily changed and reworked. Wouldn't the sculptor or civil engineer enjoy such easy tractability!

The IBM System/360 was a mainframe computer system family announced by IBM in 1964. It was the first family of computers making a clear distinction between architecture and implementation, allowing IBM to release a suite of compatible designs at different price points. The design is considered by many to be one of the most successful computers in history, influencing computer design for years to come (see Figure 0.1).

FIGURE 0.1

IBM introduced the system/360 family of business mainframe computers in 1964. (©2012 akg-images/Paul Almasy/Newscom. Unauthorized use not permitted.)

Section 2 The College Experience: Computer Disciplines and Majors to Choose From

Most professionals in the computing industry have at least an undergraduate degree in mathematics, computer science, or a related field. Many have advanced degrees, especially those involved primarily in research or education.

Computing is a broad discipline that intersects many other fields such as mathematics, science, engineering, and business. Because of such a wide range of choices, it is impossible for anyone to be an expert in all of them. A career involving computing requires the individual to focus his or her efforts while obtaining a college degree.

There are many different degrees that involve computing offered at institutions of higher learning. These degrees can even be from different departments within the same institution. Although computing degrees can share some of the same courses, they can also be quite different from each other. Choosing among them can be confusing.

To ease this confusion, it is wise for students to consult with academic advisors in the computer science department, the computer or electrical engineering department, and the business school to explore the options available in their specific institution. Next we summarize some of the degree programs that your institution may offer.

Computer Science

Computer science as a discipline encompasses a wide range of topics from theoretical and algorithmic foundations to cutting-edge developments. The work computer scientists are trained to do can be arranged into three categories:

- Designing and implementing useful software
- Devising new ways to use computers
- Developing effective ways to solve computing problems

A computer science degree consists of courses that include computing theory, programming, and mathematics. These courses ultimately develop the logic and reasoning skills integral to becoming a computer scientist. The math sequence includes calculus I and II (and in many cases, calculus III) as well as discrete mathematics. Some students also study linear algebra and probability and statistics. A computer science degree offers a comprehensive foundation that permits graduates to understand and adapt to new technologies and new ideas. Computer science departments are often found at universities as part of the science, engineering, or mathematics divisions.

Computer scientists take on challenging programming jobs, supervise other programmers, and advise other programmers on the best approaches to be taken. Computer science researchers are working with scientists from other fields to perform such tasks as using databases to create and organize new knowledge, making robots that will be practical and intelligent aides, and using computers to help decipher the secrets of human DNA. Their theoretical background allows them to determine the best performance possible for new technologies and their study of algorithms helps them to develop creative approaches to new (and old) problems.

Computer Engineering

For students who are more interested in understanding and designing the actual computing devices, many opportunities are available in computer engineering, which is concerned with the design and construction of computers and computer-based systems. A computer engineering degree involves the study of hardware, software, communications, and the interaction among them, and is a customized blend of an electrical engineering degree with a computer science degree.

The computer engineering curriculum includes courses on the theories, principles, and practices of traditional electrical engineering as well as mathematics through the standard calculus sequence and beyond. This knowledge is then applied in courses dealing with designing computers and computer-based devices. In addition, programming courses are required so that the computer engineer can develop software for digital devices and their interfaces.

Currently, an important area for computer engineers involves embedded systems. This involves the development of devices that have software and hardware embedded in them such as cell phones, digital music players, alarm systems, medical diagnostic devices, laser surgical tools, and so on. The devices a computer engineer might work with are limitless as he or she applies his or her knowledge of how to integrate hardware and software systems.

Information Systems

The information systems (IS) field focuses on integrating technology into businesses and other enterprises to manage their information in an efficient and secure manner. In this area, technology is viewed as an instrument for generating, processing, and distributing information. Therefore, the focus in this field is on business and organizational principles.

Most IS programs are located in the business school of a university or college, and IS degrees combine business and computing coursework, and the math that is required has a business application focus. These degrees may be found under such programs as Computer Information Systems (CIS) or Management Information Systems (MIS). Degree program names are not always consistent, but they all have their focus on business principles and applications of technology with less emphasis on the theory of computer science or the digital design of computer engineering.

IS specialists must understand both technical and organizational factors, and must be able to help an organization determine how to use information and technology to provide a competitive edge. These professionals serve as a bridge between the technical community and the management community within an organization. They are called on to determine the best way to use technology, organize information, and communicate effectively.

Information Technology

An Information Technology (IT) program prepares students to meet the computer technology needs of business, government, healthcare, schools, and other organizations. IT has its emphasis on the technology itself, more than on the information handled, the theory behind it, or how to design hardware or software. IT professionals work with computer systems to ensure they work properly, are secure, are upgraded and maintained, and are replaced as appropriate.

Because computers have become integral parts of the work environment for all employees at all levels of the organization, many enterprises must maintain departments of IT workers. Organizations of every kind are dependent on information technology on a daily basis and the need for qualified workers is great.

Degree programs in IT are commonly found in business or information management departments, or as an alternate degree in a computer science department. IT programs in business departments focus on using applications to meet the requirements of networking, systems integration, and resource planning. The emphasis is less on programming and more on using programs already written to the best advantage. IT programs in computer science departments often have more emphasis on programming for computer users, with a focus on writing software for interactive web pages, multimedia, and cloud computing.

IT specialists select appropriate hardware and software products for an organization and then integrate these products within the existing infrastructure. They install and customize and maintain the software as needed. Other examples of responsibilities include network administration and security, design and implementation of web pages, development of multimedia resources, oversight of e-mail systems, and installation of communication components. User support and training are often important responsibilities for the IT professional as well.

Software Engineering

Software engineering (SE) is the discipline of developing and maintaining large software systems. These systems must behave reliably and efficiently, be affordable, and satisfy all requirements defined for them. SE seeks to integrate the theory of computer science and mathematics with the practical engineering principles developed for physical objects.

An SE degree program is closely related to the computer science degree program, and they are usually offered within the same department. In fact, most computer science curricula require one or more software engineering courses. An SE degree can be considered a specialized degree within the confines of the field of computer science.

SE students learn more about software reliability and maintenance of large systems and focus more on techniques for developing and maintaining software that is engineered to be correct from its inception. Most programs require SE students to participate in group projects for the development of software that will be used in earnest by others. Students assess customer needs, develop usable software, test the product thoroughly, and analyze its usefulness.

Professionals who hold a software engineering degree expect to be involved with the creation and maintenance of large software systems that may be used by many different organizations. Their focus will be on the design principles that make the system viable for many people and through many years.

Although an SE degree has a recognized description, the term *software engineer* is merely a job label in the workplace. There is no standard definition for this term when used in a job description, and its meaning can vary widely among employers. An employer may think of a programmer or an IT specialist as a software engineer.

Mixed Disciplinary Majors

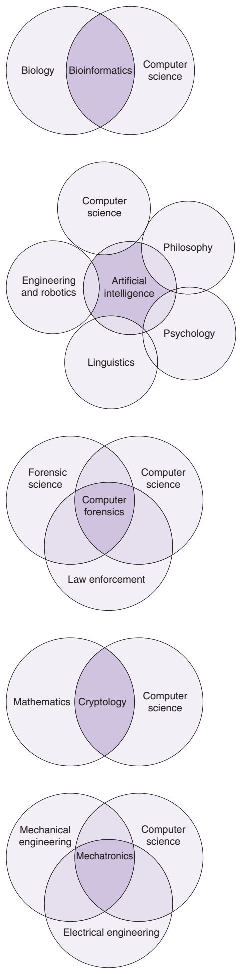
Technology is opening doors for fields of study that combine different sciences or engineering fields with computing as illustrated by Figure 0.2. Institutes of higher learning have responded by offering courses or programs for multidisciplinary majors. Some examples follow.

- **Bioinformatics** is the use of computer science to maintain, analyze, and store biological data as well as to assist in solving biological problems—usually on the molecular level. Such biological problems include protein folding, protein function prediction, and phylogeny (the history, origin, and evolution of a set of organisms). The core principle of bioinformatics involves using computing resources to help solve problems on scales of magnitude too great for human observation.
- **Artificial Intelligence** (AI) is the implementation and study of systems that can exhibit autonomous intelligence or behaviors. AI research draws from many fields including computer science, psychology, philosophy, linguistics, neuroscience, logic, and economics. Applications include robotics, control systems, scheduling, logistics, speech recognition, handwriting recognition, understanding natural language, proving mathematical theorems, data mining, and facial recognition.
- **Computer Forensics** is a branch of forensic science pertaining to legal evidence that may be found in computers and digital storage devices. The collection of this evidence must adhere to standards of evidence admissible in a court of law. Computer forensics involves the fields of law, law enforcement, and business.
- **Cryptology** (or cryptography) is the practice and study of hiding information and involves mathematics, computer science, and engineering. Electronic data security for commerce, personal uses, and military uses continue to be of vast importance.
- **Mechatronics** is the combination of mechanical engineering, electronic engineering, and software engineering in order to design advanced hybrid systems. Examples of mechatronics include production systems, planetary exploration rovers, automotive subsystems such as anti-lock braking systems, and autofocus cameras.

Even when the definitions are given for the different computing disciplines mentioned in this chapter, it is easy to see that there is great overlap among all of them.

FIGURE 0.2

Illustrations of the overlapping fields within mixed disciplinary majors



In fact, many professionals who have earned a computer science degree may be working in jobs that are closer to an information systems description or vice versa. The student is encouraged to choose a computing field that seems closest to his or her personal goals.

Section 3 Career Opportunities

The Bureau of Labor Statistics is the principal fact-finding agency for the U.S. Federal Government in the field of labor economics and statistics. This agency publishes *The Occupational Outlook Handbook*, which is a nationally recognized source of career information, designed to provide valuable assistance to individuals making decisions about their future work lives. The *Handbook* is revised every two years. Table 0.1 gives an overview of some of the major computer occupations tracked by the U.S. Bureau of Labor Statistics.

The Demand in the United States and in the World

According to the *BLS Occupational Outlook Handbook*, software developer, database administrator, and network/computer systems administrator are three of the occupations projected to grow at the fastest rates over the 2010–2020 decade. Strong employment growth combined with a limited supply of qualified workers will result in excellent employment prospects. Those with practical experience and at least a bachelor’s degree in one of the computing fields described in Section 2 should have the best opportunities. Employers will continue to seek computer professionals with strong programming, systems analysis, interpersonal, and business skills.

The growing need for computer professionals is increased by the ongoing retirement of a generation of baby boomers, and all of this is occurring as the U.S. government projects continued rapid growth in many computer science and IT occupations.

Today’s students need not worry about the impact that outsourcing computer jobs to other countries will have on their ability to find a job. The fact is that many companies have been disappointed in the results when outsourcing entire projects. Some of the more mundane aspects of coding can be outsourced, but the more creative work is best kept in house. For example, during the design and development of a new system, interaction with specialists from other disciplines and communication with other team members and potential system users are of utmost importance. These activities are negatively impacted by communication difficulties across cultures and long distances. Many companies are rethinking outsourcing and doing more system development at home.

The number of graduates from the computing fields will not meet the demand in the marketplace in the foreseeable future. Projections and statistics show that there will be plenty of jobs to be offered to the qualified computer professional and the salaries will be higher than the average full-time worker earns in the United States.

TABLE 0.1 Computer, Computer Engineering, and Information Technology Occupations

Occupation	Job Summary	Entry-Level Education
Computer and Information Research Scientists	Computer and information research scientists invent and design new technology and find new uses for existing technology. They study and solve complex problems in computing for business, science, medicine, and other uses.	Doctoral or professional degree
Computer Programmers	Computer programmers write code to create software programs. They turn the program designs created by software developers and engineers into instructions that a computer can follow.	Bachelor's degree
Computer Support Specialists	Computer support specialists provide help and advice to people and organizations using computer software or equipment. Some, called technical support specialists, support information technology (IT) employees within their organization. Others, called help-desk technicians, assist non-IT users who are having computer problems.	Some college, no degree
Computer Systems Analysts	Computer systems analysts study an organization's current computer systems and procedures and make recommendations to management to help the organization operate more efficiently and effectively. They bring business and information technology (IT) together by understanding the needs and limitations of both.	Bachelor's degree
Database Administrators	Database administrators use software to store and organize data, such as financial information and customer shipping records. They make sure that data are available to users and are secure from unauthorized access.	Bachelor's degree
Information Security Analysts, Web Developers, and Computer Network Architects	Information security analysts, web developers, and computer network architects all use information technology (IT) to advance their organization's goals. Security analysts ensure a firm's information stays safe from cyber attacks. Web developers create websites to help firms have a public face. Computer network architects create the internal networks all workers within organizations use.	Bachelor's degree
Network and Computer Systems Administrators	Network and computer systems administrators are responsible for the day-to-day operation of an organization's computer networks. They organize, install, and support an organization's computer systems, including local area networks (LANs), wide area networks (WANs), network segments, intranets, and other data communication systems.	Bachelor's degree