Instructor's Guide

to Accompany

Computer Science: An Overview

Tenth Edition

J. Glenn Brookshear

This manual is a supplement to the text *Computer Science: An Overview*, tenth edition. It consists mainly of answers to the chapter review problems although it also contains some comments regarding the material in that text. The chapters in the manual are coordinated with the parent text. That is, to find material relating to Chapter 4 in the text, turn to Chapter 4 of this supplement.

Chapter Zero

INTRODUCTION

Chapter Summary

This chapter introduces computer science as the discipline that seeks a scientific foundation for topics such as computer design, computer programming, algorithmic processes, etc. It gives an informal introduction to the concept of an algorithm (more detail is given in Chapter 5) and discusses how this concept forms the foundation of the field known as computer science. The chapter also presents a brief history of computing machinery, introduces the concept of abstraction, and sets the stage for future discussions social/professional/ethical considerations.

A major goal of this chapter is to establish the concept of computer science as being the underpinning for the development of the computer applications with which students are familiar. Most introductory students will have seen computing/computer science only in the context of using application software, Web browsing, and perhaps some programming. Thus, they may not understand the distinction between studying computer science and learning how to use today's computer application software. I find it helpful to explain that computer science deals with the development of tomorrow's application software, rather than learning how to use the applications of today.

Comments

1. This introductory chapter is included to set the stage—not to be the final word on the topics presented. The goal at this point is merely to develop an intuitive understanding of the ideas and the terminology involved.

2. When writing this introduction, I envisioned a chapter that would be used largely as a reading assignment. Students tend to start a new semester with a fresh, enthusiastic attitude. They are eager to get started and have resolved that this semester "I'll keep up and stay organized." I like to take advantage of this enthusiasm. Thus, I assign this chapter as a reading assignment on the first day of class and spend very little time discussing it. In my courses, class presentation usually starts with material from Chapter 1.

3. Those of us who teach introductory computer science courses are always looking for interesting algorithms to use as examples. Along these lines I've drawn from the art of origami (see the bird folding algorithm in Chapter 5) for some time. Introductory students seem to enjoy working with an algorithm that does something "different." I've also drawn from the field of magic for such examples. I hope you like the example in Figure 0.2 of the text and find it useful.

4. Most beginning students don't distinguish between data processing and computer science. They don't understand that there is much more to computer science than Web browsing and writing programs. In this regard, I like to use the following quote from Charles Darwin. "... science consists of grouping facts so that general laws and conclusions may be drawn from them."

5. The topics discussed throughout the text collectively provide an understanding of computer science. There is probably no single topic that a student must know. (Do students really have to know about error correcting codes, two's complement arithmetic, the bootstrap process, or the significance of the halting problem?) So don't hesitate to skip a topic if it doesn't fit your course goals. On the other hand, I encourage you to cover a wide range of topics. The goal is to introduce students to computer science by presenting a variety of topics in enough detail to expose the realities of the issues involved. (Each individual topic may not be necessary on its own, but together they paint an important picture.)

Maintaining this perspective is perhaps more of a challenge when teaching a computer literacy course than a course within a computer science curriculum. In these former cases there is a temptation to skip the more challenging or tedious topics since "they don't need to know that anyway." In contrast, I prefer to go ahead and present such subjects in a manner compatible with the audience and then adjust the level of assignments and exams to match the objectives of the particular course and the abilities of the students. (I think a major problem in today's education is that we avoid challenging topics. In turn, the students have learned to view formal course work as an irrelevant waste of time and treat it accordingly. They perform poorly, we decide we need to simplify the course further, and the cycle continues.)

6. You may want to point out that the discussion of ethical theories in Section 0.6 is there merely to suggest that before one takes the position that “I’m right and you’re wrong,” one should think about the source of his/her opinion. This is not a course on the philosophy of ethics, so don’t let your students get bogged down in the differences between duty-based and contract-based ethics.

Chapter One
DATA STORAGE

Chapter Summary

This chapter presents the rudiments of data storage within digital computers. It introduces the basics of digital circuitry and how a simple flip-flop can be used to store a single bit. It then discusses addressable memory cells and mass storage systems (magnetic disk, compact disks, and flash memory). Having established this background, the chapter discusses how information (text, numeric values, images, and sound) are encoded as bit patterns. The optional sections delve more deeply into these topics by presenting the problems of overflow errors, truncation errors, error detection and correction techniques, and data compression.

Comments

1. Perhaps the most important comment I can make about this chapter (and the next one as well) is to explain its role in the chapters that follow. This involves the distinction between exposing students to a subject and requiring them to master the material—a distinction that is at the heart of the spirit in which the entire text was written. The intention of this chapter is to provide a realistic exposure to a very important area of computer science. It is not necessary for the students to master the material. All that is needed from this chapter in the remaining part of the book are the remnants that remain from a brief exposure to the issues of data storage. Even if the course you teach requires a mastery of these details or the development of manipulation skills, I encourage you to avoid emphasizing bit manipulations and representation conversions. In particular, I urge you to avoid becoming bogged down in the details of converting between base ten and binary notation. I can't think of anything that would be more boring for the students. (I apologize for stating my opinion.)

2. The "required" sections in this chapter cover the composition of main memory (as a background for machine architecture in chapter 2 and data structures in chapter 8), the physical issues of external data storage systems (in preparation for the subjects of file and database systems in chapter 9), and the rudiments of data encoding (that serves as a background for the subject of data types and high-level language declaration statements in chapter 6). The optional sections explore the issues of error handling, including transmission error detection and correction as well as the problem of truncation and overflow errors resulting from numeric coding systems.

3. As mentioned in the preface of the text, there are several themes that run throughout the text, one of which is the role of abstraction. I like to include this theme in my lecture in which I introduce flip-flops. I end up with both flip-flop diagrams from the text on the board, and I emphasize that they represent two ways of accomplishing the same task. I then draw a rectangle around each diagram and erase the circuits within the rectangles leaving only the inputs, outputs, and rectangles showing. At this point the two look identical. I think that this creates a strong visual image that drives home the distinction between an abstract tool's interface with the outside world and the internal details of the tool.

This is a specific example of teaching several topics at the same time—in this case, the concepts of abstraction and encapsulation are taught in the context of teaching digital circuits.

4. Don't forget about the circuits in Appendix B. I used to have students who continued to record an extra bit in the answer to a two's complement addition problem when a carry occurred—even though I had explained that all values in a two's complement system were represented with the same number of bits. Once I started presenting the addition circuit in Appendix B, this problem disappeared. It gave the students a concrete understanding that the carry is thrown away. (Of course, in a later course computing students will learn that it really isn't thrown away but saved as the "carry bit" for potential use in the future, but for now I ignore this.) I have also found that a good exercise is to ask students to extend the circuit in Figure B.3 so that it produces an additional output that indicates whether an overflow has occurred. For example, the output could be 1 in the case of an overflow and 0 otherwise.

5. For most students, seeing the reality of the things they are told is a meaningful experience. For this reason I often find it advantageous to demonstrate the distinction between numeric and character data using a spreadsheet. I like to show them how the manipulation of large numbers can lead to errors.

7. I have found that students respond well to hearing about CD and DVD storage systems, how sound is encoded, and image representation systems such as GIF and JPEG. I have often used these topics as a way of getting non-majors interested in technical issues.

8. For students not majoring in computer science, topics such as two's complement and floating-point notation can get a bit dry. The main point for them to understand is that when information is encoded, some information usually gets lost. This point can be made just as well using audio and video, which are contexts that seem to be more interesting to the non-majors.

Answers to Chapter Review Problems

1. a. 0 b. 0 c. 0

2. a. upper input = 1, lower input = 0

   b. upper input = either 0 or 1, lower input = 1

   c. upper input = 1, lower input = 0

3. a. AND b. OR c. XOR

4. This is a flip-flop that is triggered by 0s rather than 1s. That is, temporarily changing the upper input to 0 will establish an output of 1, whereas temporarily changing the lower input to 0 will establish an output of 0. To obtain an equivalent circuit using NAND gates, simply replace each AND-NOT gate pair with a NAND gate.

5.

Address	Contents
00	02
01	53
02	01
03	53

6. 256 using two hexadecimal digits (16 bits) , 65536 using four hexadecimal digits (32 bits).

7. a. 11001011 b. 01100111 c. 10101001 d. 00010000 e. 11111111

8. a. 0 b. 1 c. 1 d. 0

9. a. AAA b. CB7 c. 0EB

10. The image consists of 1024 x 1024 = 1,048,576 pixels and therefore 3 x 1,048,576 = 3,145,728 bytes, or about 3MB. This means that about 86 images could be stored in the 256MB camera storage system. (By comparing this to actual camera storage capacities, students can gain an appreciation for the benefits of image compression techniques. Using them, a typical 256MB storage system can hold as many as 300 images.)

11. 786,432. (Each pixel would require one memory cell.)

12. Data retrieval from main memory is much faster than from disk storage. Also data in main memory can be referenced in byte-sized units rather than in large blocks. On the other hand, disk storage systems have a larger capacity than main memory and the data stored on disk is less volatile than that stored in main memory.

13. There are 70GB of material on the hard-disk drive. Each CD can hold no more than 700MB. Thus, it will require at least 100 CDs to store all the material. That does not seem practical to me. On the other hand, DVDs have capacities of about 4.7GB, meaning that only about 15 DVDs would be required. This may still be impractical, but its a big improvement over CDs. (The real point of this problem is to get students to think about storage capacities in a meaningful way.)

14. There would be about 5,000 characters on the page requiring two bytes each (Unicode). So the page would require about 10,000 bytes or 10 sectors of size 1024 bytes.

15. The novel would require about 1.4MB using ASCII and about 2.8MB if Unicode were used.

16. The latency time of a disk spinning at 60 revolutions per second is only 0.0083 seconds.

17. About 18.3 milliseconds.

18. About 7 years!

19. What does it say?

20. hexadecimal

21. a.

1	0	0	/	5
00110001	00110000	00110000	00101111	00110101
	=		2	0
00100000	00111101	00100000	00110010	00110000

b.

T	o		b	e
01010100	01101111	00100000	01100010	01100101
	o	r		n
00100000	01101111	01110010	00100000	01101110
o	t		t	o
01101111	01110100	00100000	01110100	01101111
	b	e	?	
00100000	01100010	01100101	00111111	

c.

```
    T          h          e                    t
01010100  01101000  01100101  00100000  01110100
    o          t          a          l
01101111  01110100  01100001  01101100  00100000
    c          o          s          t
01100011  01101111  01110011  01110100  00100000
    i          s                     $          7
01101001  01110011  00100000  00100100  00110111
    .          2          5          .
00101110  00110010  00110101  00101110
```

22. a. 31 30 30 2F 3D 20 3D 20 32 30

    b. 54 6F 20 62 65 20 6F 72 20 6E 6F 74 20 74 6F 20 62 65 3F

    c. 54 68 65 20 74 6F 74 61 6C 20 63 6F 73 74 20 69 73 20 24

    37 2E 32 35 2E8

23. 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 10000

24. a. 00110010 00110110 b. 11010

25. They are the powers of two. 1 10 100 1000 10000 100000

26. a. 7 b. 1 c. 21 d. 17 e. 19 f. 0 g. 4 h. 8 i. 16 j. 25 k. 26 l. 27

27. a. 111 b. 1011 c. 10000 d. 1111 e. 100001

28. a. 0 b. 3 c. -3 d. -1 e. 7

29. a. 100 b. 111 c. 001 d. 011 e. 101

30. a. 15 b. -13 c. 13 d. -16 e. -9

31. a. 0001100 b. 1110100 c. 1111111 d. 0000000 e. 0001000

32. a. 01101 b. 00000 c. 10000 (incorrect) d. 10001 e. 01110

    f. 10011 (incorrect) g. 11110 h. 01101 i. 10000 (incorrect) j. 11111

33. a.

```
  7                 00111
+ 1   becomes     + 00001
                    01000  which represents 8
```

b.

```
  7                 00111                       00111
- 1   becomes     - 00001  which converts to  + 11111
                                                00110   which represents 6
```

c.

```
 12                 01100                       01100
- 4   becomes     - 00100  which converts to  + 11100
                                                01000   which  represents 8
```

d.

```
   8              01000                        01000
 - 7   becomes  - 00111  which converts to   + 11001
                                               00001  which represents 1
```

e.

```
  12              01100
 + 4  becomes   + 00100
                 10000   which represents -16 (overflow)
```

f.

```
   5              00101                        00101
 - 11  becomes  - 01011  which converts to   + 10101
                                               11010  which represents -6
```

34. a. 3 1/4 b. 4 5/16 c. 13/16 d. 1 e. 2 1/8

35. a. 101.01 b. .0001 c. 111.111 d. 1.11 e. 110.101

36. a. 1 1/4 b. -1/2 c. 3/16 d. -9/32

37. a. 01001000 b. 01111111 c. 11101111

d. 00101010 e. 00011111 (truncation)

38. 00111100, 01000110, and 01010011

39. The best approximation of the square root of 2 is 1 3/8 represented as 01011011. The square of this value when represented in floating-point format is 01011111, which is the representation of 1 7/8.

40. The value one-eighth, which would be represented as 00101000.

41. Since the value one-tenth cannot be represented accurately, such recordings would suffer from truncation errors.

42. From left to right the result would be 2 3/4. From right to left the result would be 2 1/2.

43. a. 1 5/8 b. 4 c. 3 1/4

44. a. 01101100 b. 01101000 c. 01111000 (truncation) d. 01101011

45. a. The value is either eleven or negative five.

b. A value represented in two's complement notation can be changed to excess notation by changing the high-order bit, and vice versa.

46. The value is two; the patterns are floating-point, excess, and two's complement, respectively.

47. a. This is the value -5 coded in floating-point, excess 8, and two's complement notation, respectively.

b. This is the value -3 coded in two's complement, excess 128, and floating-point notation, respectively.

c. This is the value 2 coded in excess 8, two's complement (or binary), and floating-point notation, respectively.

48. Only bit patterns of length 5 are valid excess 16 representations. Thus, 101, 010101, 1000, 000000, and 1111 are not valid.

49. b would require too large of an exponent. c would require too many significant digits. d would require too many significant digits.

50. When using binary notation, the largest value that could be represented would change from 15 to 255. When using two's complement notation the largest value that could be represented would change from 7 to 127.

51. 4FFFFF

52. Use the first and second inputs as inputs to an XOR gate. Do the same with the third and fourth inputs. Then, tie the outputs of these two XOR gates to the inputs of a third XOR gate.

53. 1123221343435

54. yyxy xx yyxy xyx xx xyx

55. Starting with the first entries, they would be *x*, *y*, space, *xxy*, *yyx*, and *xxyy*.

56. Not a chance. MPEG requires transfer rates of 40 Mbps.

57. a.

```
    1         0         0         /         5
00110001  10110000  10110000  00101111  10110101
   =         2         0
00011101  00110010  10110000
```

```
b.     T         o                   b         e
01010100  11101111  00100000  01100010  11100101
               o         r                   n
00100000  11101111  11110010  00100000  01101110
    o         t                   t         o
11101111  11110100  00100000  11110100  11101111
               b         e         ?
00100000  01100010  11100101  10111111
```

```
c.     T         h         e                   t
01010100  01101000  11100101  00100000  11110100
     o         t         a         l
11101111  11110100  01100001  11101100  00100000
     i         s                   $         7
11101001  01110011  00100000  10100100  00110111
     .         2         5         .
10101110  00110010  10110101  10101110
```

58. The underlined strings definitely contain errors.

11001 <u>11011</u> 10110 <u>00000</u> 11111 <u>10001</u> 10101 00100 01110

59. The code would have a Hamming distance of 3. Thus, by using it, one could detect up to 2 errors per character and correct up to 1 error per character.

60. a. HE b. FED c. DEAD d. CABBAGE e. CAFE

Chapter Two
DATA MANIPULATION

Chapter Summary

This chapter introduces the role of a computer's CPU. It describes the machine cycle and the various operations (or, and, exclusive or, add, shift, etc.) performed by a typical arithmetic/logic unit. The concept of a machine language is presented in terms of the simple yet representative machine described in Appendix C of the text. The chapter also introduces some alternatives to the von Neumann architecture such as multiprocessor machines and artificial neural networks.

The optional sections in this chapter present a more thorough discussion of the instructions found in a typical machine language (logical and numerical operations, shifts, jumps, and I/O communication), a short explanation of how a computer communicates with peripheral devices, and alternative machine designs.

The machine language in Appendix C involves only direct and immediate addressing. However, indirect addressing is introduced in the last section (Pointers in Machine Language) of Chapter 7 after the pointer concept has been presented in the context of data structures.

Comments

1. Much of Comment 1 regarding the previous chapter is pertinent here also. The development of skills in the subjects of machine architecture and machine language programming is not required later in the book. Instead, what one needs is an image of the CPU/main memory interface, an understanding of the machine cycle and machine languages, an appreciation of the difference in speeds of mechanical motion compared to CPU activities, and an exposure to the limited repertoire of bit manipulations a CPU can perform.

2. To most students at this stage the terms millisecond, microsecond, nanosecond, and picosecond merely refer to extremely short and indistinguishable units of time. In fact, most would probably accept the incorrect statement that activities within a computer are essentially instantaneous. Once a student of mine wrote a recursive routine for evaluating the determinant of a matrix in an interpreted language on a time-sharing system. The student tried to test the program using an 8 by 8 matrix, but kept terminating the program after a minute because "it must be in a loop." This student left with an understanding of microseconds as real units of time that can accumulate into significant periods.

3. A subtle point that can add significantly to the complexity of this material is combining notation conversion with instruction encoding. If, for example, all the material in Chapters 1 and 2 is new to a student, the problem, "Using the language of Appendix C, write an instruction for loading register 14 with the value 124" can be much more difficult than the same problem stated as, "Using the language of Appendix C, write an instruction for loading register D with the (hexadecimal) value 7C." In general, notation conversion is a subject of minor importance and should not be allowed to cloud the more important concerns.

4. If you want your students to develop more than a simple appreciation of machine language programming, you may want to use one of the many simulators that have been developed for the machine in Appendix C. A nice example is included on the Addison-Wesley website at `http://www.aw.com/brookshear` or you can find other simulators by searching the Web.

5. Here are some short program routines in the machine language presented in Appendix C of the text, followed by their C language equivalents. (These examples are easily converted into Java, C++, and C#.) Each machine language routine starts at address 10. I've found that they make good examples for class presentations or extra homework problems in which I give the students the machine language form and ask them to rewrite it in a high-level language.

Address	Contents	Address	Contents	Address	Contents
0D	00	14	20	1B	0F
0E	00	15	5A	1C	50
0F	00	16	30	1D	12
10	20	17	0F	1E	30
11	5C	18	11	1F	0D
12	30	19	0E	20	C0
13	0E	1A	12	21	00

C language equivalent:

```
{int X,Y,Z;
 X = 92;
 Y = 90;
 Z = X + Y;
}
```

If the contents of the memory cell at address 1C in the preceding table is changed from 50 to 60 the C equivalent becomes:

```
{float X, Y, Z;
 X = 1.5;
 Y = 1.25;
 Z = X + Y;
}
```

Here's another example:

Address	Contents	Address	Contents	Address	Contents
0E	00	19	0F	24	20
0F	00	1A	20	25	01
10	20	1B	04	26	50
11	02	1C	B1	27	01
12	30	1D	2C	28	30
13	0E	1E	12	29	0F
14	20	1F	0E	2A	B0
15	01	20	50	2B	18
16	30	21	12	2C	C0
17	0F	22	30	2D	00
18	11	23	0E		

C equivalent:

```
{int X, Y;
 X = 2; Y =1;
 while (Y != 4) {X = X + Y; Y = Y + 1;}
}
```

6. Here are two C program segments that can be conveniently translated into the machine language of Appendix C.

```
{int X, Limit;
 X = 0;
 Limit = 5;
 do X = X + 1 while (X != Limit);
}
```

Program segment in machine language:

Address	Contents	Address	Contents	Address	Contents
0E	00 (X)	18	22 (R2 = 1)	22	10 (R0 = Limit)
0F	00 (Limit)	19	01	23	0F
10	20 (X = 0)	1A	11 (R1 = X)	24	B1 (go to end
11	00	1B	0E	25	28 if X == Limit)
12	30	1C	50 (X = X+1)	26	B0 (return)
13	0E	1D	12	27	1A
14	20(Limit = 5)	1E	30	28	C0 (halt)
15	05	1F	0E	29	00
16	30	20	11 (R1 = X)		
17	0F	21	0E		

```
{int X, Y, Difference;
 X = 33;
 Y = 34;
 if (X > Y)Difference = X - Y
 else Difference := Y – X}
```

Program segment in machine language:

Address	Contents	Address	Contents	Address	Contents
0D	00 (X)	1D	01	2D	16 (Diff = X-Y)
0E	00 (Y)	1E	24 (R4=FF)	2E	30
0F	00 (Diff)	1F	FF	2F	0F
10	20 (X = 33)	20	96 (R6=not Y)	30	B0 (branch to
11	21	21	24	31	3A halt)
12	30	22	56 (R6= -Y)	32	90 (R0=not X)
13	0D	23	36	33	14
14	20 (Y = 34)	24	50 (R0=X-Y)	34	50 (R0 = -X)
15	22	25	16	35	03
16	30	26	25 (R5=80Hex)	36	50 (R0 = Y-X)
17	0E	27	80	37	02
18	11 (R1=X)	28	80 (mask low)	38	30 (Diff = Y-X)
19	0D	29	50 7 bits)	39	0F
1A	12 (R2=Y)	2A	B5 (if R0=R5	3A	C0 (halt)
1B	0E	2B	32 then Y>X	3B	00
1C	23 (R3=1)	2C	50		

Answers to Chapter Review Problems

1. a. General purpose registers and main memory cells are small data storage cells in a computer.

   b. General purpose registers are inside the CPU; main memory cells are outside the CPU.

(The purpose of this question is to emphasize the distinction between registers and memory cells—a distinction that seems to elude some students, causing confusion when following machine language programs.)

2. a. 0010000100000101

   b. 1010

   c. 001100100100

3. Nine cells with addresses B9, BA, BB, BC, BD, BE, BF, C0, and C1.

4. BA

5.

Program counter	Instruction register	Memory cell at 00
02	2104	21
04	3100	21
06	C000	04

6. To compute x + y - z, each of the values must be retrieved from memory and placed in a register, the sum of x and y must be computed and saved in another register, z must be subtracted from that sum, and the final answer must be stored in memory.

A similar process is required to compute (2x) + y. The point of this example is that the multiplication by 2 is accomplished by adding x to x.

7. a. Move the contents of register 7 to register E.

   b. AND the contents of register 0 with the contents of register 8 and place the result in register 0.

   c. Rotate the contents of register 4 three bits to the right.

   d. Load register 8 with the value (hexadecimal) 35.

   e. Compare the contents of registers 3 and 0. If the patterns are equal, jump to the instruction at address AD. Otherwise, continue with the next sequential instruction.

8. 16 with 4 bits, 256 with 8 bits

9. a. 2766 b. 1766 c. 80F2 d. A403 e. BB31

10. The only change that is needed is that the third instruction should be 6056 rather than 5056.

11. a. Retrieves from memory cell 3B.

    b. Is independent of memory cell 3B.

    c. Changes the contents of memory cell 3B.

    d. Changes the contents of memory cell 3B.

    e. Is independent of memory cell 3B.

12. a. Place the value 05 in register 4. b. 05

13. a. 241B b. 1B34

14. a. Load register 0 with the contents of memory cell 04.
Store the contents of register 3 in memory cell 45.
Halt.

b. C0

c. 06

15. a. 29 b. 0A

16. a. 00, 01, 02, 03, 04, 05

b. 06, 07

17. a. 03 b. 03 c. 0E

18. 05. The program is a loop that is terminated when the value in register 0 (initiated at 00) is finally incremented to the value in register 3 (initiated at 05).

19. 20 microseconds.

20. The point to this problem is that a bit pattern stored in memory is subject to interpretation—it may represent part of the operand of one instruction and the op-code field of another.

a. Registers 0, 1, and 2 will contain 32, 20, and 12, respectively.

b. 12

c. 32

21. The machine will alternate between executing the jump instruction at address AF and the jump instruction at address B0.

22. It would never halt. The first 2 instructions alter the third instruction to read B000 before it is ever executed. Thus, by the time the machine reaches this instruction, it has been changed to read "Jump to address 00." Consequently, the machine will be trapped in a loop forever (or until it is turned off).

23.

a.	b.	c.
148D	148D	2000
34B3	15B3	1145
C000	358D	B10A
	34BD	22DD
	C000	B00C
		22CC
		3288
		C000

24. a. The single instruction B000 stored in locations 00 and 01.

b.

Address	Contents
00,01	2100 Initialize
02,03	2270 counters.
04,05	3109 Set origin
06,07	320B and destination.
08,09	1000 Now move
0A,0B	3000 one cell.
0C,0D	2001 Increment
0E,0F	5101 addresses.
10,11	5202
12,13	2333 Do it again
14,15	4010 if all cells
16,17	B31A have not

18,19 B004 been moved.
1A,1B 2070 Adjust values
1C,1D 3071 that are
1E,1F 2079 location
20,21 3075 dependent.
22,23 207B
24,25 3077
26,27 208A
28,29 3087
2A,2B 2074
2C,2D 3089
2E,2F 20C0
30,31 30A4
32,33 2000
34,35 20A5
36,37 B070 Make the big jump!

c. Address Contents

00,01 2000 Initialize counter.
02,03 2100 Initialize origin.
04,05 2270 Initialize destination.
06,07 2430 Initialize references
08,09 1530 to table.
0A,0B 310D Get origin
0C,0D 1600 value.
0E,0F B522 Jump if value must be adjusted.
10,11 3213 Place value
12,13 3600 in new location.
14,15 2301 Increment
16,17 5003 R0,
18,19 5113 R1, and
1A,1B 5223 R2.
1C,1D 233C Are we done?
1E,1F B370 If so, jump to relocated program.
20,21 B00A Else, go back.
22,23 2370 Add 70 to
24,25 5663 value being
26,27 2301 transferred and
28,29 5443 update R4 and
2A,2B 342D R5 for next
2C,2D 1500 location.
2E,2F B010 Return (from subroutine).
30,31 0305 Table of
32,33 0709 locations that
34,35 0B0F must be
36,37 111F updated for
38,39 212B new location.
3A,3B 2FFF

25.

20A1 21A4
21A2 6001
6001 30A5
21A3 C000
6001

26. The machine would place a halt instruction (C000) at memory location 04 and 05 and then halt when this instruction is executed. At this point its program counter will contain the value 06.

27. The machine would continue to repeat the instruction at address 06 indefinitely.

28. It copies the data from the memory cells at addresses 00, 01, and 02 into the memory cells at addresses 10, 11, and 12.

29. Let R represent the first hexadecimal digit in the operand field;
Let XY represent the second and third digits in the operand field;
If the pattern in register R is the same as that in register 0,
then change the value of the program counter to XY.

30. Let the hexadecimal digits in the operand field be represented by R, S, and T;
Activate the two's complement addition circuitry with registers S and T
as inputs;
Store the result in register R.

31. Same as Problem 24 except that the floating-point circuitry is activated.

32. a. 04 b. A8 c. FC d. 08 e. F4

33.

a.	b.	c.	d.
1066	1034	10A5	10A5
30BB	21F0	210F	210F
	8001	8001	8001
	3034	12A6	4001
		21F0	A104
		8212	7001
		7002	30A5
		30A6	

34. a. 101000 b. 000000 c. 000100 d. 110001 e. 111001 f. 101110

g. 010101 h. 111111 i. 010001 j. 101110 k. 010001 l. 001110

35. a. AND the byte with 11000011.

b. XOR the byte with 11111111.

c. XOR the byte with 10000000.

d. OR the byte with 10000000.

e. OR the byte with 01111111.

36. XOR the input string with 10000001.

37. First AND the input byte with 10000001, then XOR the result with 10000001.

38. a. 11010 b. 00001111 c. 010 d. 001010 e. 10000

39. a. 9F b. 86 c. FF d. BB

40. a. AB03 b. AB05

41.

Address	Contents	
00,01	2008	Initialize registers.
02,03	2101	
04,05	2200	
06,07	2300	
08,09	148C	Get the bit pattern;
0A,0B	8541	Extract the least significant bit;
0C,0D	7335	Insert it into the result.
0E,0F	6212	
10,11	B218	Are we done?
12,13	A401	If not, rotate registers
14,15	A307	
16,17	B00A	and go back;

```
18,19   338C If yes, store the result
1A,1B   C000   and halt.
```

42. The idea is to complement the value at address A1 and then add. Here is one solution:

```
21FF
12A1
7221
13A0
5423
34A2
```

43. Each character would consist of 8 bits so the rate of 300 bps would translate into approximately 37 characters per second. Thus, the printer could just keep up. (In reality, an ASCII character requires about 10 bits when transmitted serially because, in addition to a parity bit, start and stop bits are also added to each pattern. As a rule of thumb, 300 bps is considered to be 30 characters per second.) If the rate were increased to 1200 bps, the printer wouldn't stand a chance.

44. The typist would be typing 30 x 5 = 150 characters per minute, or 1 character every 0.4 seconds (= 400,000 microseconds). During this period the machine could execute 20,000,000 instructions.

45. The typist would be producing characters at the rate of 2.5 characters per second, which translates to 20 bps (assuming each character consists of 8 bits).

46.

```
Address Contents
00,01   2000
02,03   2101
04,05   12FE Get printer status
06,07   8212 and check the ready flag.
08,09   B004 Wait if not ready.
0A,0B   35FF Send the data.
```

47.

```
Address Contents
00,01   20C1 Initialize registers.
02,03   2100
04,05   2201
06,07   130B
08,09   B312 If done, go to halt.
0A,0B   31A0 Store 00 at destination.
0C,0D   5332 Change destination
0E,0F   330B address,
10,11   B008 and go back.
12,13   C000
```

48. 14,400 bps is equivalent to 1,800 bytes/sec. So it would take 2960 hours (over 123 days) to fill the 20MB drive.

49. 144

50. Group the 64 values into 32 pairs. Compute the sum of each pair in parallel. Group these sums into 16 pairs and compute the sums of these pairs in parallel. etc.

51. CISC involves numerous elaborate machine instructions that can be time consuming. RISC involves fewer and simpler instructions, each of which is efficiently implemented.

52. How about pipelining and parallel processing? Increasing clock speed is another answer.

53. In a multiprocessor machine several partial sums can be computed simultaneously.

Chapter Three
OPERATING SYSTEMS

Chapter Summary

This chapter introduces the fundamental concepts associated with operating systems. It begins with a historical look at operating systems, followed by discussions of operating system architecture and internal operation. An optional section covers semaphores and deadlock. The chapter closes with a discussion of security issues.

Comments

1. This chapter provides an excellent opportunity to introduce the particular features of the local operating system (e.g. pertinent issues of file management, any sign-on and sign-off procedures, and perhaps e-mail features), and the utility programs (such as the editor) that will be used later in this or other classes.

2. The image I like to convey to the student is that of the operating system residing between the computer user and the hardware. Once this image is established, it's nice to show how different operating systems can produce different personalities from essentially the same hardware technology. One method of doing this is to compare an icon-based windowing system with a text oriented system.

3. An operating system is an important example of a large software system, and thus this chapter provides an opportunity to set the stage for software engineering in Chapter 7. This is one reason why the modular structure of an operating system is presented in this chapter. Time spent in class on this topic can pay dividends in the form of ready examples and a basis for class discussions when covering Chapter 7—not to mention the fact that it reinforces the organized, modular approach to problem solving that we want our students to appreciate.

4. Don't miss the opportunities to reinforce the concept of abstraction and abstract tools while covering this chapter.

5. A point that many students never stop to consider is that the operating system is itself a program that is being executed on the same machine that it is controlling. In particular, such components as the command processor, file manager, or scheduler must essentially share time with the other processes in the system. Pointing this out to a beginning class increases the complexity of a classroom discussion but has the advantage of conveying the true complexity of a multiprogramming operating system.

Answers to Chapter Review Problems

1. Control data and its access, provide for efficient device access, coordinate the use of the machine's resources, and control access to the machine.

2. Batch processing refers to the process of collecting a program (or programs) together with data and submitting this material to the operating system for execution (perhaps at a later time) without further intervention by the user.

Interactive processing refers to the technique of executing a program in a manner that allows the user to communicate with the program during its execution.

3. R, S, T, X, Y, Z (The items are removed in the same order they were placed in the queue.)

4. Interactive processing allows the user to communicate with a program during its execution. The phrase "real-time processing" means that the time required for the activities of the program being executed must coordinate with activities in the outside world.

5. An operating system that allows several activities to execute "at the same time."

6. Answers will vary. The goal is for students to "experience" multitasking so that it is real rather than theoretical. We wan them to connect material in the text with reality.

7. Answers will vary. They should project an understanding that application software reflects the computer's application, whereas utility software forms part of the system's infrastructure.

8. a. The shell of an operating system handles the communication with the operating system's users.

b. The kernel of an operating system performs the fundamental tasks of the system.

9. X is a directory containing the subdirectory Y, which contains the subdirectory Z.

10. A process is the execution of a program.

11. The status of each process (ready, waiting) and the priority of each process.

12. A process that is ready could make progress if given a time slice, but giving a time slice to a process that is waiting would merely waste time since it cannot progress until some event occurs.

13. Virtual memory is the memory space whose presence is merely simulated by swapping blocks of data back and forth between a disk and the memory actually present in the machine.

14. To create a 1024MB (MiB) virtual memory using 2KB (KiB) pages would require 524,288 pages.

15. If both processes merely need to read from the file, no conflicts will occur. However, if one of the processes is going to modify the file, them it should have exclusive access. (Such problems are discussed in Section 9.5 in the context of databases.)

16. Application software performs tasks that are unique to the use of the particular computer system, whereas system software performs tasks that are required as the software infrastructure of any computer system.

17. Load balancing refers to the task of keeping all the processors busy. Scaling has to do with dividing a task into subtasks that can be performed simultaneously.

18. The machine begins by executing a program, called the bootstrap, at a predetermined location in memory. This program directs the machine to load a program (the operating system) from mass storage into main memory. The original program tells the machine to transfer its attention to the program just loaded.

19. Since most of a computer's main memory is volatile, the operating system must be reloaded each time the machine is turned on.

20. Answers will vary. Most PCs give the user the option of altering parameters before the booting process actually begins--usually by pressing the F1 key. The software controlling this procedure is part of the BIOS stored in the machine's ROM. Students who have floppy drives will hear the bootstrap routine look for the operating system there before trying the hard drive. They should all be able to hear the bootstrap routine reading the operating system from the hard drive.

21. If the machine can execute 5 instructions in a microsecond, it can execute 5,000 instructions in a millisecond or 100,000 instructions in a 20 millisecond time slice. (The point is that a modern machine can do a lot in a single time slice.)

22. The typist would be typing 5 characters per second, or one every 200 milliseconds. Thus, 10 time slices could be allocated during the 200 milliseconds between characters.

23. At least half. This does not include the time required to actually transfer the data. 25 milliseconds = 25000 microseconds. Thus, 250,000 instructions could be executed during this time.

24. Memory space, disk storage space, access to a printer, time slices, and access to files.

25. The I/O-bound process. This allows the controllers to start with the I/O activities. Then the compute-bound process can run while the other is waiting for these slower activities to take place. As a general rule of thumb, priority should be given to the slower activity.

26. A mix of I/O-bound and compute-bound processes will normally produce a higher throughput than a collection of processes with similar characteristics. For example, little is gained by allowing a collection of compute-bound processes to share time. In fact, such a collection will usually get done faster without the delays caused by switching repeatedly among the different processes in the collection. However, in the case of several I/O bound processes, it could be that the relative timing of the I/O requests would produce benefits in a multiprogramming environment.

27. Save the current process' state;
select another process from the process table;
load that process' state;
start the next time slice.

28. A process's state includes the values in the CPU's registers (including the program counter) as well as the contents of its associated memory cells.

29. If a process asks for service from a mass storage device, its time-slice will be terminated because the process must wait for the device to perform the requested operation before continuing.

30. First: Interrupt signal occurs.

Second: Machine completes its current instruction.

Third: Machine saves the current program state.

Fourth: Machine begins executing the interrupt routine.

31. These questions are compatible with any operating system. The answers will vary. The goal is for the student to relate the material in the text to an actual operating system.

32. These questions are intended for a multiuser, multitasking operating system such as UNIX. The answers will vary. The goal is for the student to relate the material in the text to an actual operating system.

33. The test-and-set instruction is often used to implement semaphores. Since its task is executed as a single instruction, no interrupt signal can interfere.

34. The banker has removed the competition for the nonshareable resource.

35. Our approach to the problem is to consider permission from the instructor and the payment of the fee as nonshareable resources for which the students compete.

a. This removes the competition for the nonshareable resources by removing the need for them.

b. This removes the competition for nonshareable resources by adding additional resources (one more permission and one more fee payment privilege).

c. Here the fee payment privilege and the instructor's permission are forcibly retrieved and given to another student.

d. Here the instructor's permission is forcibly retrieved and given to the other student.

36. The window manager forcibly retrieves an area of the screen that has been allocated to one process and reallocates it to another (by pushing a window into the background and bringing another to the foreground).

37. Deadlock cannot occur because each process must request all the resources it will need at a certain level at once.

38. First, one controlling computer reads the common cell and retrieves the value zero.

Second, the other controlling computer reads the common cell and retrieves the value zero.
Third, the first computer places a non-zero value in the common cell and tells its arm to pick up the assembly.
Fourth, the other computer places a non-zero value in the common cell and tells its arm to pick up the assembly.

39. As the processes producing the printed material terminate, their output that has accumulated in mass storage is placed in a queue to wait for the printer. Each time the printer finishes the output of a process, it begins printing the next unit of output in this queue.

40. a. The longer a lone car waits at a red light, the higher its priority becomes. Thus, it will ultimately be given a green light at the expense of the heavier traffic.

b. The process whose time slice has just finished will most likely have the highest priority and therefore be awarded the next time slice. This is why dynamic priority systems are used in multiprogramming systems. That is, as a process waits for a time slice, its priority increases. (In the simplest cases, processes merely wait in a queue for the next time slice. Thus a process' priority is reflected by its position in the queue. As each process completes a time slice, it is placed at the rear of the queue.)

41. In both deadlock and starvation there are processes that are not able to make progress. The difference is that in the case of deadlock, none of the processes are able to execute, whereas in the case of starvation the higher priority processes are able to execute.

42. The point of this problem is as much to introduce students to this piece of computer science folklore as it is to pose the problem itself. Issues include the problem of each philosopher obtaining possession of one fork as well as the problem of a philosopher's neighbors obtaining possession of the forks available to him and never releasing them.

43. As the length of time slices become smaller, the ratio of time spent swapping processes compared to the time spent executing them increases. Thus, a point is reached where the efficiency of the system becomes quite low. On the other hand, if time slices are too long, the illusion of simultaneous operation is lost.

44. Interrupt disable, interrupt enable, and the test-and-set instructions

45. Answers may vary. Possibilities include establishing new accounts, removing accounts, establishing privileges, and monitoring the machine's usage.

46. By loading the current process's memory limits in special purpose registers that the CPU uses to validate all references to main memory. If a reference is outside the bounds established by those registers, an interrupt will occur, causing control to be returned to the operating system.

47. 26^9 milliseconds, which is many years. (The point is that milliseconds add up.)

48. To allow the operating system the ability to protect processes from each other. The operating system runs in the highest privilege level but restricts the other processes to lower privilege levels.

49. Two that are identified in the text are changing the contents of memory limit registers and changing the CPU's current privilege level.

50. Answers will vary. Possibilities include accessing data in memory cells outside the process's allocated space, gaining unauthorized access to mass storage, and modifying the operating system itself to gain advantage over other processes.

Chapter Four
NETWORKS AND THE INTERNET

Chapter Summary

This chapter introduces the fundamental concepts associated with networks, internets, and the Internet. It begins with networking fundamentals. It then explores the Internet and Internet applications, with a section devoted to the Web. An optional section discusses the TCP/IP protocol hierarchy. The chapter closes with a section on network/internet security.

Comments

1. It is easy to spend a lot of time in this chapter talking about the latest developments relating to the Internet. Things are changing quickly, and there are numerous topics you may wish to discuss with your students. Some of you may want to expand on the coverage of HTML and/or XML, others may want to delve deeper into the TCP/IP hierarchy, and still others may want to emphasize security. What I have tried to do is provide basic material from which you can build depending on the needs of your course.

2. Having made the previous comment, I encourage you to cover the optional section on the TCP/IP hierarchy even though your students may view it as rather technical. An understanding of this material is basic to truly understanding many of the issues that an Internet user encounters, a specific example being firewalls.

3. My experience has been that I often learn a lot by listing to my students during this part of the course. It's amazing to see what they own in the way of the latest gadgets. (I, myself, have not purchased the latest in "cell phone/camera/PDA/MP3 player" devices, but someone in the class almost always has the newest gizmo to demonstrate.)

4. The text addresses security in the context of guarding a computer system and its contents. You may wish to expand this to include personal protection as well. In particular, an Internet user can easily get into legal trouble (for example, violating copyright laws), financial difficulty (for example, releasing account information), and physical danger (for example, providing personal data to potential predators).

5. If your students have not built a simple Web page, they can learn a lot by doing so. In particular, they learn that posting a Web page is not difficult so they realize that just because information appears on the Web doesn't make it authoritative.

Answers to Chapter Review Problems

1. A terse definition might be "A protocol is a rule or set of rules governing communication." Answers identifying particular protocols in this chapter will vary. Some protocols that are introduced in the non-optional sections are CSMA/CD, CSMA/CA, FTP, telnet, and HTTP.

2. The client/server model is a context in which to envision communication between two processes. One process, called the client, makes requests of the other process, called the server. The server process fills the request and returns an appropriate response.

3. The peer-to-peer model is a context in which to envision communication between two processes. Unlike the client/server model, a process under the peer-to-peer model may provide a service to and receive service from another process.

4. LANs, MANs, and WANs is one. Another is open or proprietary.

5. The internal details of an open network are public knowledge and can be used without specific permission from an owner, which allows different organizations to produce compatible products. Such details of a closed network are proprietary, which restricts the ability of organizations to produce their own versions of the network's components.

6. CSMA/CD requires that members of the network be able to detect if their transmissions interfere with that of others. In a wireless network, one member may not be able to hear all the other members or a member's transmissions may drown out those of a more distant member.

7. Wait for bus to become silent;
   start transmitting;
   if collision occurs, wait before trying again.

8. The hidden terminal problem is that machines in a wireless network may not be able to hear each other's transmissions. One approach to solving the problem is to have each machine send a short request to the AP and wait to receive an acknowledgement before transmitting an entire message. All machines will be able to hear the acknowledgement.

9. A hub connects individual computers to form a bus network, whereas a repeater connects two bus networks to form a larger network.

10. A router connects two networks to form an internet, whereas the other devices connect networks to form a larger network.

11. An internet is a collection of networks that have been linked so that messages can be transferred form one network to another. In an internet, each computer has two addresses associated with it. One is the computer's network address; the other is the computer's internet address. Each network within an internet maintains its own internal characteristics, which may not be the same as those in the other networks.

12. CSMA/CD and CSMA/CA.

13. The population of the world (approximately 7 billion) is between 2^{32} and 2^{33}. Therefore, using 128-bit addresses allows for more than 2^{95} addresses per person. We hope that is adequate.

14. a. 1.2.3

    b. 128.0

    c. 24.12

15. a. 0000000000000000

    b. 000110010001001000000001

    c. 00000101000011000000110100001010

16. The values 134, 48, 4, and 123 (base ten) are written 10000110, 110000, 100, and 1111011 in base two. Therefore the 32-bit address would be 8630047B in hexadecimal.

17. A DNS lookup is the process of using the DNS to translate an address from mnemonic form into the corresponding IP address (or vice versa).

18. The domain name is metropolis.gov. It contains a subdomain called batcave, which contains the machine named batman.

19. The term kermit is the name of the receiver of the message; animals.com identifies the mail server that should receive the email.

20. When transferred from one computer to another, a "text file" may require alterations due to different ways of encoding line breaks, whereas a "binary file" does not required conversions.

21. The mail server collects all incoming email and holds it until the recipient requests to read it. The mail server also receives all email originating within the domain and then forwards it appropriately.

22. N-unicast forces the server to send multiple copies of each message (one to each client), whereas multicast allows the server to send only one copy, which is distributed to each of the clients.

23. a. A name server is a process that provides assistance in converting mnemonic names into IP addresses.

b. An access ISP is an Internet service provider that provides users with connections to the Internet (as opposed to a tier-1 or tier-2 ISP whose task is to provide the Internet's communication system..

c. A router is a machine that connects two networks.

d. An end system is a device connected to the Internet that uses the Internet for communication purposes.

24. a. Hypertext is text in which items are linked to other texts in a manner that allows a reader to move between related materials.

b. HTML (HyperText Markup Language) is a system for describing the structure of a hypertext document and identifying links between its components and other documents.

c. A browser is a program that presents hypertext documents to a reader .

25. The Internet is a world-wide network of computer networks. The World Wide Web is a collection of hypertext documents available on the Internet.

26. Answers will vary. The point is for the students to see an HTML document and understand the role of the tags such as <head> and <body>. This is the first step to learning HTML.

27. Answers will vary. Tags mentioned in the text include <html>, <head>, <body>, <title>, <h1>, , <a>, and <img>.

28. The line My dog's name is Rover. would become

My dog's name is Rover.

29. The level one heading "My Pet Dog" would appear at the top of the page with the image of Rover below. (Note that the title "Example" is not part of the displayed page.

30. Answers should vary widely. They may include such tags as <exp> and </exp> to indicate the beginning and end of an exponent, or perhaps <quo><num>x + y</num> xy </quo> to represent (x + y)/(xy).

31. It should be interesting to see how close students come to HTML. For example, they may use the tags and to identify text that is bold or and to identify paragraphs.

32. The answers will vary. The appearance oriented tags may provide marks indicating line positioning, fonts, and paragraphs whereas the semantics tags might provide marks identifying the picture's title, the director, leading actors and actresses, supporting actors and actresses, and the reviewer's rating.

33. As with the previous problem the answers will vary. If you assign both problems, note that the tags representing the appearance of the text could be the same as those in the previous problem. (Did the students pick up on this?) The semantic tags might provide ways of marking the sport involved, the teams names, the major players, the score, etc.

34. HTTP is the protocol to be used when accessing the document; lifeforms.com is the mnemonic name of the machine holding the document; animals/moviestars is the directory path to the document; and kermit.html is the name of the document.

35. a. Protocol/host address/document

b. Protocol/host address

c. host address

36. The point is that the protocols used by the browser when communicating with the server will differ. In the first case the browser would use HTTP to contact the server at the machine named stargazer.universe.org to retrieve a Web document, whereas in the second case the browser would use HTTPS.

37. Answers will vary. Client-side activities include Web page animation and filling in order forms on a Web page to be sent back to a server. Server-side activities include searching done by search engines, updating reservation information in an airline reservation system, and processing an order received from a client customer.

38. The OSI reference model is a classification of duties that may occur in network communication.

39. One way is to use the CSMA/CD protocol.

40. Application layer: Obtains the IP address for the final destination of a message.
Transport layer: Divides a message into manageable units and attaches sequence numbers to them.
Network layer: Determines the next intermediate destination of a message unit.
Link layer: Handles the task of actually transmitting a message unit to another machine in the network.

41. Small packets will interweave with other traffic in the Internet more easily than large units, leading to a more efficient communication system.

42. When using TCP, the transport layers at the message's origin must contact at the transport layer at the destination to establish a connection. Then the two transport layers carry on a dialog to confirm that the application layer's message is properly transferred. (The point here is for students to understand that the various layers in the hierarchy communicate over the Internet. Not all communication is between application layers.)

43. TCP actually confirms that the entire message made it to the destination, whereas UDP does not. However, UDP is more efficient.

44. A transport layer using UDP does not establish contact with the transport layer at the destination but merely sends the message unannounced.

45. a. At the application layer

b. At the network layer

c. At the transport layer

46. At the domain's mail server because the gateway is only a router and therefore "sees" only individual packets rather than the content of entire messages.

47. A proxy server is an intermediary between a client and a server that shields the client from inappropriate behavior of the server.

48. Public-key encryption is a system by which messages can be encrypted using a publicly known encoding key but deciphered only by using a private decoding key.

49. One is that the PC could be used in a denial of service attack.

50. Without international treaties, legal solutions imposed by a nation can be applied only within that nation's boarders.

Chapter Five
ALGORITHMS

Chapter Summary

In this chapter we turn to an explicit study of algorithms—a topic that was introduced as the core of computer science in Chapter 0. The chapter begins with a formal definition of an algorithm and a discussion of the meaning of that definition. This leads to a discussion of algorithm creation (a creative process that parallels the more generic task of problem solving) and algorithm representation (the process of expressing an algorithm in preparation for, or during, programming). The chapter closes with the subjects of algorithm efficiency (using big theta notation) and correctness.

The search for algorithm development techniques is presented as an ongoing activity—not a settled issue. The chapter discusses some of the ideas proposed by researchers in the field of problem solving and relates these ideas to the problem of algorithm discovery and representation.

This chapter also presents iterative structures by means of the sequential search and insertion sort algorithms as well as recursive structures by means of the binary search algorithm. In each case, emphasis is placed on the components involved in controlling the repetitive process.

For communication purposes, the chapter introduces a pseudocode. This pseudocode is also used at times later in the text, although these later appearances are rather intuitive in that they do not require previous, rigorous coverage of this chapter. So, if you're teaching a course for non-majors, you can treat the pseudocode lightly without creating problems in future chapters.

Comments

1. You will find this text's approach to problem solving somewhat different from that of other texts. The problem solving section in this chapter is intended to be a refreshingly honest discussion. It does not try to convince students that they can become good problem solvers merely by learning a particular problem solving technique. In particular, it does not preach top-down design via stepwise refinement.

Students don't learn to solve problems by practicing explicit techniques in an isolated section of a course. They learn to solve problems unconsciously over an extended period of time in which they are required to do it. Thus, every chapter of the text is designed to develop the students' problem-solving skills. (Maybe I'm old-fashioned, but I think we produce too much rhetoric about problem solving and don't require our students to do enough.)

3. The following is a problem that I have given students to work on. The point is not whether they solve it but that they realize that solving a problem is a creative process that may not be achieved by following a methodology. Actually, you could argue that there are many correct answers to this problem, but I think you'll agree that the one given below is a good one.

Problem: Fill in the blank in the sequence

110, 20, 12, 11, 10, __, ...

Answer: 6. The pattern is

110 (base two) = 6, 20 (base three) = 6, 12 (base four) = 6
11 (base five) = 6, 10 (base six) = 6, so 6 (base seven) = 6

4. Here is a good problem for demonstrating bottom-up design and the use of abstraction in problem solving. Imagine three railroad spurs each opening into a Y that connects to the other spurs. (It is really three stacks, but we have not studied data structures yet.) Cars are on each spur and a locomotive is in the "middle" of the track system. Show that the locomotive can rearrange the cars in any order.

A first step is to show that the locomotive can move the car "on top" of any spur to the top of any other spur. This is actually a two-step process since the locomotive must remain in the "middle" of the track system. The next step is to show that any two cars on a spur can be interchanged. Then show that any two cars in the system can be interchanged. Finally, show that any arrangement can be obtained by a sequence of interchanging two cars at a time.

5. I've debated marking the introduction to recursion as an optional section, but the truth is that recursion is a very important topic in computer science. I even include it when I teach "computer literacy" to non-majors and have always found the class to respond well. (I show the class that solving a problem recursively is like getting something for nothing—we always ask someone else to do the hard work. The students like that.)

Thus, an important goal in this chapter is to begin developing the concept of recursion. I say "begin" because experience has shown that students acquire an understanding of recursion over a period of time. I like to devote a significant part of a class period to walking through an example of the binary search, as done in the book. Each time another activation of the algorithm is called, I mark my place in the current copy of the algorithm, move to a new location on the chalkboard, draw another copy, copy the pertinent portion of the list next to it, and then proceed with execution. (Some form of overhead projection saves time, but working on a chalkboard gives students time to think.) Each time an activation terminates, I transfer its list back into the larger list in the previous activation, erase the terminating activation, and then proceed in the previous one. Such a careful presentation pays numerous dividends. Not only does it clearly present the recursive process, but it also sets the stage for later discussions regarding stacking activation environments, issues of global versus local variables, and parameter passing.

I also like to walk through some short examples such as

```
procedure PrintValues(Input)
if (Input not 0)
     then (Print the value of Input;
           Apply PrintValues to the value (Input - 1))
```

in which I point out the difference in the output obtained by reversing the statements in the then clause.

6. Here are some additional problems you may want to give your students. Each has the characteristic that one can see a solution fairly quickly, but then one must wrestle with the problem of organizing and expressing the solution.

a. Design an algorithm for solving the traveling salesman problem. (Given a network of roads, cities, and mileages, find the shortest route that leads through each city at least once.)

b. Design an algorithm for converting a string of 1s into a string of 0s under the following constraints. The right-most bit can always be complemented. Any other bit can be complemented if and only if the bit to its immediate right is a 1 and all other bits to its right (if there are any) are 0.

c. Design an algorithm for testing a tic-tac-toe board to see if there has been a winner.

d. Design an algorithm for finding a path through a maze.

7. I like to use the problem "Design an algorithm for predicting the sum of the top and bottom faces of four dice." to emphasize the distinction between algorithm discovery and algorithm representation. In this case it's the discovery step that may be tough. (Opposing faces on a die always add to seven. Thus, the opposing faces on four dice must add to 28. Once this is discovered, the underlying algorithm can be expressed as the single statement "print the value 28").

8. A good exercise for "experienced" programmers who think loop control is trivial is to design an iterative algorithm for printing all possible permutations of a list of letters. (This is also a good example of how the use of recursion can simplify matters.)

Answers to Chapter Review Problems

1. A sequence of steps that defines a nonterminating process would do.

2. An instruction such as "Drive to the grocery store." may be ambiguous if there are several grocery stores around, but the underlying algorithm would not be ambiguous. The problem would be in the representation, not the algorithm.

3. The use of primitives establishes a well-defined terminology in which algorithms can be expressed.

4. Students will come up with a variety of answers. (If they don't, they're probably working together.) The point is for them to understand the idea of primitives and to begin to think about programming language design.

5. No. The process described will never terminate because the value of Count will never be 5.

6. The last statement is not executable because the lines drawn in the previous steps do not intersect.

7. One answer would be

```
Count ← 1;
repeat (print the value assigned to Count;
        Count ← Count + 1)
until (Count = 5)
```

8. One answer would be

```
Count ← 1
while (Count < 5) do
      (print the value assigned to Count and
       Count ← Count + 1)
```

9. The conditions appearing in the statements would be negations of each other. That is, the statement repeat (. . .) until (x is zero) is equivalent to do (. . .) while (x is not zero).

10. Here's an outline of one possible solution.

```
Starting from the right end of the input, find the first digit that is
  smaller than the one to its right. (If there isn't such a digit, no the
  input cannot be rearranged to represent a larger value.)
Call the position in the input in which this digit was found the target
  position.
Interchange the digit found above with the smallest digit to its right
  that is still larger than itself.
Sort the digits to the right of the target position in descending order
  from right to left.
```

11. Suppose N is the given integer. Then the following will work. You may want to ask your students how this solution could be made more efficient.

```
X ← 1
while (X ≤ N) do
(if (X divides N) then report X
 X  X + 1
)
```

12. Start with a date whose day of the week is known. Figure out how many days are between that date and the given date (remember leap years). Then divide that total by seven and use the remainder to determine the displacement from the known day.

13. Pseudocode is a relaxed version of a programming language used to jot down ideas. A formal programming language prescribes strict rules of grammar that must be obeyed.

14. Syntax refers to the way something is expressed, whereas semantics refers to what is being expressed.

15. W = 6, X = 9, Y = 5, Z = 1. Most will get their foot in the door by realizing that the carry has to be 1. Then they may figure out that X must be 9 since X + Y = 1Y.

16. V = 0, W = 4, X = 1, Y = 3, Z = 9. Most will get their foot in the door by realizing that X must be 1 since X times XY is XY. Then they may discover that Z must be 9 since it must be a one-digit perfect square (Y times Y) which when added to one produces a carry.

17. X = 1, Y = 0. A simple algorithm would be to try X = 1 and Y = 0 first. If that works, report the solution. Otherwise, try X = 0 and Y = 1. If that works, report the solution. Otherwise, report that there is no solution.

18. Andrews and Blake go through the shaft first (2 minutes), and Andrews returns with the lantern (1 minute). Then, Johnson and Kelly go through (8 minutes), and Blake returns with the lantern (2 minutes). Finally, Andrews and Blake go through again (2 minutes). The total travel time is 2 + 1 + 8 + 2 + 2 = 15 minutes.

19. They are the same. Suppose the volume of the small glass is V and x units of water are poured into the large glass. Then the large glass contains $V + x$ units of liquid—V units of wine and x units of water. When the small glass is filled from the large one, x units of liquid are returned to the small glass of which $V/(V + x)$ is wine. Therefore, the small glass will end up with $xV/(V + x)$ units of wine. Furthermore, $xx/(V + x)$ units of water will be returned to the small glass, meaning that exactly $x - xx/(V + x) = xV/(V + x)$ units of water will be left in the large glass.

20. Approximately 122 meters. Let the distance between the hives be y. Since each bee flies at a constant speed, the ratio of the distances traveled must be the same each time the meet. Therefore, 50/(y – 50) = (y + 20)/(2y – 20). Solving for y implies that y is approximately 7.5 meters or 122 meters.

21. How about this?

```
procedure SubStringSearch(FirstString, SecondString)
P ← 0;
Success ← false;
while (P + length of FirstString) <= (length of SecondString)
        and Success = false) do
  [N ← 1;
   while (P + Nth character in SecondString =
        Nth character in FirstString) do
    (N ← N + 1);
       if (N = length of FirstString)
            then (Success ← true)))
   P ← P + 1]
```

Or, this is a recursive solution.

```
procedure SubStringSearch(FirstString, SecondString)
Success ← false;
if (FirstString empty) then (Success ← true);
P ← 0;
while (P + length of FirstString) <= (length of SecondString)
        and Success = false) do
  (P ← P + 1;
      if (1st entry in FirstString = Pth entry in SecondString)
        then (apply SubStringSearch to the "remainder of FirstString"
                  and "remainder of SecondString";
               if (that search is a success)
                    then (Success ← true))
if (Success = true)
  then (declare this search a success)
  else (declare this search a failure)
```

22. Body: Everything after do
    Initialization: The first two assignment statements
    Modification: The last assignment statement (Some could argue that it is the last three assignment statements.)
    Test: while (Current < 100)

The output will be 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89.

23. This is a recursive version of Problem 13. Its output will be 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89.

24. Move the print statement to the end of the then clause.

25. When searching for J: H, L, J
    When searching for Z: H, L, N, 0

26. Sequential search: 3000
    Binary search: at least 13

27. a. Count ≥ 5 b. Count = 1 c. (Count ≥ 5) or (Total ≥ 56)

28. The body consists of the statements following do. It will be executed twice. The proposed modification would produce a nonterminating loop.

29. It may not terminate because the value in Count may never be exactly 1 (due to round-off errors).

30.

```
procedure Euclidean
if (neither X nor Y is 0)
      then (divide the larger of X and Y by the smaller;
            X ← the value of the remainder;
            Y ← the value of the divisor; and
            apply Euclidean to X and Y)
```

31. Test1 would print 1, 2, 3, 4, whereas Test2 would print 4, 3, 2, 1.

32. The termination condition is Count = 5. The state of the process moves toward this condition (assuming that Count starts with an integer value less than 5) each time a recursive call is made. (Note that the procedures differ merely in that the recursive call and the print statement have been interchanged.)

33. $N \neq 5$

34. 3, 1, 0, 2, 4

35. 2, 0, 1, -1

36.

```
Trial ← 2;
Temp ← Trial;
repeat
       (while (Temp even) do (Temp ← Temp/2)
        while (Temp is an integer) and (Temp not equal to 1)
             (Temp ← Temp/3);
        if (Temp = 1) then (print Trial);
        Trial ← Trial + 1)
until (2 = 3)
```

37. a. binary b. sequential c. sequential d. binary

e. 5 for the sequential search; 1 for the binary search.

38.

```
procedure Factorial (Value)
if (Value is 0)
      then (Return 1 as the answer)
      else (Apply Factorial to (Value - 1),
            multiply the result by Value, and
            X ← the value of this product),
return the number assigned to X as the answer)
```

39. a. Remove the first name from the list, leaving a list of only four names. Use the previous algorithm to sort this shorter list. Then, insert the name removed earlier into this sorted list at its correct position.

b.

```
procedure Sort (List)
if (List has more than one name)
      then (Remove the first name from List,
           Apply sort to the remaining portion of List,
           Insert the name removed earlier into the
                  sorted list at its correct position)
```

40. This algorithm uses the following terminology: The peg on which the rings start is called the initial peg, the peg to which the rings are to be moved is called the destination peg, the third peg is called the auxiliary peg, and the number of rings to be moved is represented by N.

```
procedure Towers (Initial, Destination, Auxiliary, N)
if (N equals 1)
    then (Move the top ring from the Initial peg to the Destination peg)
    else (Apply Towers to move N-1 rings from the current Initial peg to
             the current Auxiliary peg,
          Move the top ring from the Initial peg to the Destination peg,
          Apply Towers to move N-1 rings from the current
             Auxiliary peg to the current Destination peg)
```

41. The solution is almost totally given in the problem.

```
while (the rings are not all on the destination peg) do
      (move the largest numbered ring that can be moved)
```

42. Using a loop structure:

```
Day ← 1
Pay ← 1
while (Day is less than 31) do
      (print the value assigned to Pay,
       Pay ← Pay times 2,
       Day ← Day + 1)
```

Using recursion:

Apply the procedure ComputePay with initial values of Day and Pay being 1.

```
procedure ComputePay (Pay, Day)
Print the value assigned to Pay,
Apply ComputePay using Day + 1 for the new value of Day and
Pay + Pay for the new value of Pay)
```

43. The issue here is the termination condition. It's likely that an approximation method such as this will never produce the exact answer. Thus, we must decide what degree of accuracy is required and write the program accordingly.

```
Accept the number whose square root is to be found and assign
  this value to both Number and Previous;
repeat
   (assign Guess the value (Previous + (Number / Previous)) / 2;
    assign Previous the value of Guess)
until (the square of the value of Guess is close enough to Number)
```

44.

```
procedure Permute (String)
if (String is only one character long)
      then (produce a list containing the only permutation possible)
      else (Remove one character from String and apply a copy of this
            algorithm to obtain a list of permutations of the remaining
            string;
           Produce the desired list by inserting the character removed
           earlier into each possible position in each list obtained
           above)
```

45.

```
if (the list is not empty)
      then(Longest ← the first entry in the list;
           P ← 2;
           while (P <= length of list) do
             (if (the Pth entry in the list is longer than Longest)
                then (Longest ← the Pth entry);
              P ← P + 1)
```

If there are more than one longest entry, this algorithm reports the first one.

46.

```
Place the first five numbers in the set called Largest;
Place the first five numbers in the set called Smallest;
P ← 6;
while (P <= length of list) do
   if (the Pth entry in the list is larger than
      the smallest entry in Largest)
     then (replace the smallest entry in Largest with the Pth entry);
   if (the Pth entry in the list is smaller than
      the largest entry in Smallest)
     then (replace the largest entry in Smallest with the Pth entry);
   P ← P + 1)
```

47. Alphabetical order

48. 12 for the binary search, 4000 for the sequential search.

49. Allowing for carries, the addition of two n-digit values would require 2n – 1 additions. Thus, addition would be in $\Theta(n)$. To multiply two n-digit values requires n^2 multiplications. Thus, multiplication is in $\Theta(n^2)$.

50. To solve the first problem, sort the group by age, and form the subgroups by placing the youngest person in one while putting all the rest in the other. The alternate problem requires the generation of all possible divisions followed by the selection of the proper subgroup pair. Thus, the first problem is a polynomial one, whereas the second is not.

51. 26 + 195 + 793 + 995 + 1156 = 3165. The best solution known for this problem is simply to try different combinations in a systematic manner. If we're lucky we'll find the answer early. However, with n numbers there are 2^n - 1 different combinations, and thus exponential time might be required. Indeed, this is a well-known NP problem called the knapsack problem.

52. Formally, the algorithm should never terminate because the value of X will never become 1. However, when executed on a machine, it will terminate due to truncation errors.

53. No. The algorithm will not compute the correct answer when X = 0.

54. No. The algorithm will not compute the correct answer when X = Y.

55. No. The algorithm will not compute the correct answer if the list contains more than one entry and the largest entry is the last.

56. a. Preconditions: The input list is arranged in ascending order.
Loop invariant: The target value is not equal to any list entry preceding the current entry.

b. Since the input list can contain only a finite number of entries and each time through the body of the loop the current entry is advanced by one, at some point there will no longer remain entries to be considered.

57. J is less than or equal to Y, and Z equals X - J.

Chapter Six
PROGRAMMING LANGUAGES

Chapter Summary

This chapter begins with a short history of programming languages. It presents the classification of languages by generations and also points out that such a linear classification scheme fails to capture the true diversity of languages today. To support this claim, the chapter introduces the declarative, functional, and object-oriented paradigms as alternatives to the more traditional imperative approach. There are separate sections later in the chapter on both the object-oriented and declarative paradigms.

The chapter also presents some common structures found in third-generation imperative and object-oriented languages, drawing examples from Ada, C, C++, C#, FORTRAN, and Java. Topics covered include declaration statements, control statements, procedures (and functions), and parameter passing. An additional section extends this study to include topics associated with object-oriented languages.

To bridge the gap between high-level languages and machine language, this chapter discusses the rudiments of language translation, including the basic steps of lexical analysis, parsing, and code generation.

Optional sections introduce parallel programming, and explore declarative programming via logic programming and the language Prolog.

Comments

1. This chapter is intended to serve one of two roles, depending on the programming background of the audience. For those with previous programming experience, the chapter can be approached as an introduction to the subject of language design and implementation. For those with no programming experience, the chapter can serve as a generic introduction to programming languages or a context in which a particular language can be emphasized.

2. I encourage you not to get bogged down in the details of this chapter. Discourage your students from approaching the material in the context of memorizing facts about specific languages. Their goal should be to answer questions such as "Identify some different programming paradigms," "Describe three generic control structures and indicate how they might be expressed in a high-level programming language," and "Describe the difference between data type and data structure."

3. Although the section on declarative programming is optional, I encourage you to give it some time in your class. It is important that beginning students be exposed to alternatives.

4. An area in programming language research is the identification of useful control structures and the development of syntactic structures to represent them. Along these lines I like to use the example of trying to find a particular value in an array using the equivalent of a "for" statement. The problem is to exit the loop structure as soon as the target value is found without traversing more of the array. This leads to the role of such statements as the "break" in C and its derivatives or "EXIT" in FORTRAN.

Answers to Chapter Review Problems

1. The statements in the language are not made in terms of a particular machine's characteristics.

2. Suppose the value of x is stored in the memory cell whose address is XY and the program begins at address 00.

```
2100
31XY
2003
B110
2201
5112
31XY
B006
```

3. We'll represent the addresses of LENGTH, WIDTH, and HALFWAY by XX, YY, and ZZ, respectively.

```
10XX
11YY
6001
30ZZ
```

4. Suppose the values W, X, Y, and Z are stored at locations WW, XX, YY, and ZZ, respectively. Moreover, we use VV as an address.

```
     2000
     11XX
     12YY
     B1VV
     11WW
VV:  5012
     30ZZ
```

5. The answer to both questions is that this information is needed before the machine code for manipulating the data can be generated. (To add two binary values uses a different op-code than adding two floating-point values.)

6. Imperative paradigm: Programs consist of a sequence of commands.
   Object-oriented paradigm: Programs are organized as active elements called objects.
   Functional paradigm: Programs consist of nested functions.
   Declarative paradigm: Programs consist of declarative statements that describe properties.

7. The smallest of the four values w, x, y, and z.

8. The string dcababcd.

9. The major item of data would be the account balance. The object should be able to respond to deposit and withdrawal messages. Other objects in the program might be saving account objects, mortgage objects, and credit card account objects.

10. An assembly language is essentially a mnemonic form of a machine language.

11. A simple approach would be to assign mnemonics to the opcodes, using R1, R2, and so on to represent the registers, and separating the fields in each instruction with commas. This would produce instructions such as ST R5, F2; MV R4, R5; and ROT R5, 3.

12. This approach is not self-documenting in the sense that it does not indicate that the value of AirportAlt will not change in the program. Moreover, it allows the value to be changed by an erroneous statement.

13. The declarative part of the program contains the programmer-defined terminology; the procedural part contains the steps in the algorithm to be executed.

14. A literal is a specific value appearing as itself; a constant is a name for a fixed value; a variable is a name whose associated value can change as the program executes.

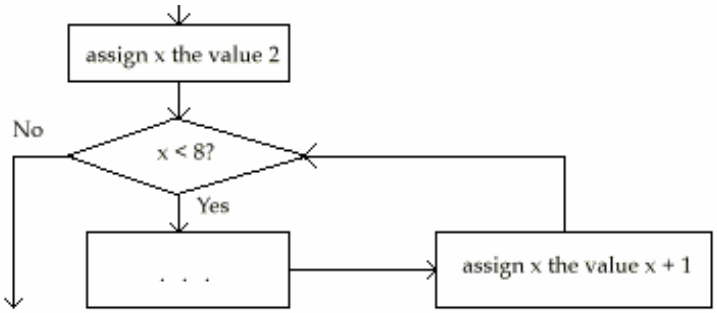
15. a. Operator precedence is a priority system given to operations that determines the order in which operations will be performed unless otherwise specified.

b. Depending on operator precedence the expression could be equal to either 24 or 12.

16. Structured programming refers to an organized method of developing and expressing a program with the goal being to obtain a well-organized program that is easy to read, understand, and modify.

17. In the first case the symbol represents a comparison; in the second case it represents data movement.

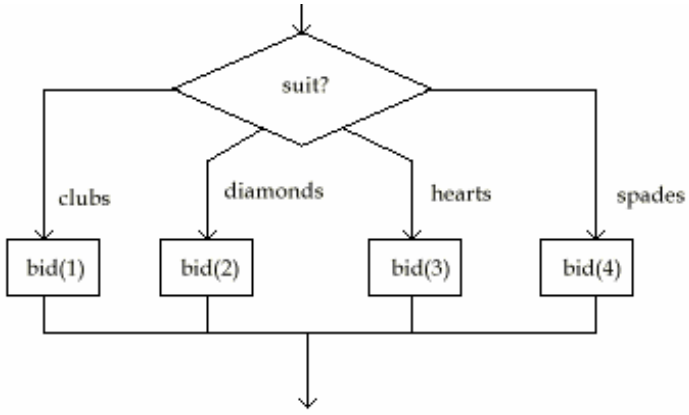
18.



19.
```
x ← 2;
while (x < 8) do
  ( . . .
   x ← x + 2)
```

20. Music is normally written in an imperative language containing loops and "go to" control structures.

21.



22.
```
case(W)
    5: assign Z the value 7
    6: assign Y the value 7
    7: assign X the value 7
```

23.
```
if (X > 5)
   then X = X + 2
   else X = X + 1
```

24. a. if and switch structures
    b. while, repeat, and for structures as well as recursion
    c. the assignment statement

25. A translator merely converts a program form one language to another. An interpreter executes the source program without producing a translated copy.

26. If the required coercion was allowed the value 2.5 would probably be truncated to 2.

27. All operations (including assignments) must be performed without coercion.

28. It would require time to copy the data as well as memory space to hold the copy.

29. When passing by value, the sequence 7, 5 would be printed. When passing by reference the sequence 7, 7 would be printed.

30. When passing by value the sequence 5, 9, 5 would be printed. When passing by reference, the sequence 9, 9, 9 would be printed.

31. 5, 9, 9

32. a. Passing parameters by value protects the calling environment from being altered by the procedure being called.

b. Passing parameters by reference is more efficient than passing them by value.

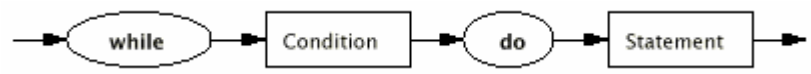
33. Depending on the order of execution, X could be assigned either 25 or 13.

34. It is not clear which of the statements in the first program should be modified, but merely changing the value of the constant NumberOfEmp in the alternate version would update the program without difficulty.

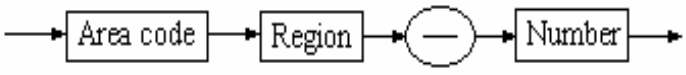
35. a. A formal language is defined by its grammar, whereas the grammar of a natural language is merely an attempt to explain the structure of a natural language.

b. Programming languages such as C, C++, Java, and C# are formal languages. Examples of natural languages include English, Spanish, Italian, etc.

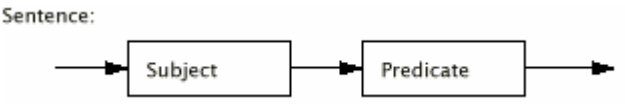
36.



37. As indicated in the text a solution could start like this:



38. The first diagram might be something like this:



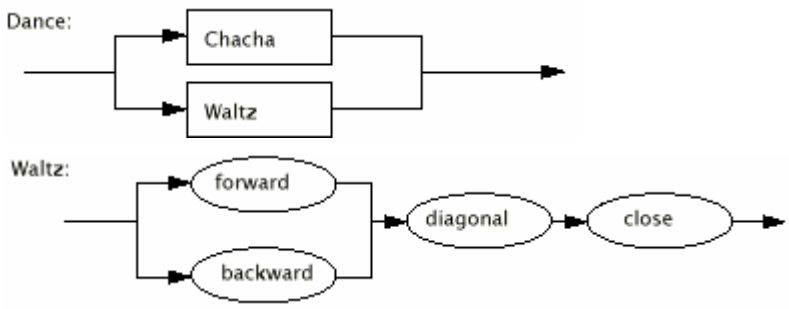
39. Answers will vary. The top level diagram might indicate that the date can be expressed in different ways. Then, other diagrams would reveal the details of each approach.

40. A string would have the structure of "yes no," or the word *yes* followed by a "sentence" followed by the word *no*.

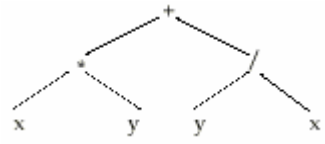
41. This one is a bit tough for beginning students. The point is for them to experience the reality that the tools applied when solving a problem can actually restrict one's ability to find the solution. In this case the problem is to describe the grammatical structure of a simple language. However, the tool is the concept of a syntax diagram that only allows the description of context-free languages, whereas the set to be described does not form a context-free language. Encourage your students to think about this. An extension to the problem would be to ask your students what features could be added to syntax diagrams that would allow such languages to be described.

42. Any string of the form $x^n y x^n$, where n is a nonnegative integer.

43.



44.



45. When performing either assignment statement, the value of X will already be in a register as a result of the comparison performed earlier. Thus, it need not be retrieved from memory.

46. assign Y the value 5
assign Z the value 8

47. assign X the value 5

48. Both types and classes are templates used to describe the underlying composition of "variables." Types, however, are predefined whereas classes are defined by the programmer within the "written program." (We put written program in quotes because the class definition could be imported from a pre-written package as in Java's API.) More about types versus classes is discussed in Section 7.7.

49. There are numerous answers. The idea is for students to isolate basic properties of buildings within a single class and then use inheritance to describe more specialized classes such as houses, hotels, grocery stores, barns, etc. Some students may find that multiple levels of inheritance are helpful.

50. The public parts of a class are those parts that are accessible from the outside; the private parts are those that are not accessible from the outside.

51. Answers will vary.

52. There may be numerous objects representing people. Some would be quests whereas others would be hotel employees (a good place for inheritance). Other objects might include a receptionist's desk, a seating area, and perhaps an elevator.

53. One solution would be

54. No. The following resolution pattern produces the empty clause.

55. Assuming that the sibling relationship is amended so that a person cannot be his or her on sibling as proposed in the answer to Question/Exercise 3, the additional relationships could be defined as:

```
uncle(X, Y) :- male(X), sibling(X, Z), parent(Z, Y).
aunt(X, Y) :- female(X), sibling(X, Z), parent(Z, Y).
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
cousin(X, Y) :- sibling(W, Z), parent(W, X), parent(Z, Y).
parents(X, Y, Z) :- X \= Y, parent(X, Z), parent(Y, Z).
```

56. The last two statements translate to "David likes people who like sports" and "Alice likes things that David likes." Prolog will conclude that Alice likes sports, music, and herself. (Alice likes things that David likes, David likes people who like sports, and Alice likes sports. Therefore, Alice likes Alice.)

57. Due to truncation errors, X may never be exactly equal to 1.00 so the loop may never terminate.

Chapter Seven
SOFTWARE ENGINEERING

Chapter Summary

This chapter begins by considering the broad issues of software engineering and comparing them to those in other engineer disciplines. The chapter then turns to a discussion of the software life cycle and how the discipline has evolved from the rigid water fall model to more flexible methodologies. With this background, the chapter discusses issues of modular design (procedures versus objects, coupling, and cohesion) and then introduces some of the "tools" that have been developed to aid in the software development process (such as dataflow diagrams, structure charts, UML, and design patterns). The remaining sections cover quality assurance, the human-machine interface, documentation, and some issues of software ownership and liability (traditional copyright and patent laws).

Comments

1. Our students in computer science should not study software engineering just to learn skills but should approach the subject as another dynamic area of research in which today's rules are replaced by better policies tomorrow. The subject also offers insights to the question "Why?" in other areas of computer science. For example: Why do programming languages provide for the declaration of constants? Why do programming languages provide for different parameter passing protocols? Why should we approach a problem from a data flow instead of a control-flow point of view?

Thus, the goal of this chapter is not to train students to use certain techniques but rather to give them a background from which they will continue to study computer science. We hope that someday they will discover design techniques that will make the methodologies that our generation preaches obsolete. It is one thing to insist that today's data processing personnel rigorously adhere to current methodologies but another to apply this same restriction to tomorrow's computer scientists.

My point is that, in a general computer science course, we should never preach a particular methodology as the only way, or even the right way. Instead, we should present today's methods as merely the state of the art. This is the general theme that runs throughout the text. My goal is to introduce students to computer science—not to train them in particular skills.

2. An important thread that runs through this chapter is the impact that the object-oriented paradigm is having on the subject of software engineering. Indeed, software engineering is more dynamic than ever as it struggles to provide methodologies based on the object-oriented, rather than the imperative, paradigm.

3. Keeping up-to-date with CASE tools presents the same problem for educational institutions as changing hardware technology. We're always on the verge of producing obsolete students. I predict that third generation languages (and most of the subject matter that goes along with them) will soon be viewed in the same way we view assembly language today. In this regard, I think component architecture will play an important role.

4. The Java Application Programming Interface, the C++ Standard Template Library, and the .NET Framework provide examples of how the "theory" of design patterns is finding its way into today's programming environment.

Answers to Chapter Review Problems

1. A good modular design allows alterations to focus on only the pertinent sections of the software. The use of local variables reduces side effects and thus reduces the chance of "corrections" introducing more errors, etc.

2. Evolutionary prototyping is the process of developing the final software system by enhancing the prototype—that is, the prototype evolves into the final product.

3. It results in a subjective discipline in which theories are debated but few precise, definitive conclusions can be reached.

4. No. The complexity of a software system is enhanced by the interaction between its parts and therefore would be greater than merely the sum of the complexities of its parts. (This is why beginning programmers often do not comprehend the problems faced by software engineers. They tend to see large software systems as merely bigger versions of the small programs they have written.) (This question is meant to invite the more mathematically inclined students to consider the mathematical properties that software metrics would have.)

5. Probably not. The structures of the underlying systems would probably be different. Feature X may fit conveniently into the system that already contains Y, but adding feature Y to a system already containing X may be problematic. (Here, again, this question is meant to invite the more mathematically inclined students to consider the mathematical properties that software metrics would have.)

6. Other fields of engineering tend to be based on exact science. Mechanical engineering, for example, has a strong foundation in physics.

7. There are several answers. Here are two that come to mind.

   a. The lack of flexibility means that mistakes can be very expensive.

   b. The waterfall model provides a well-defined process in which progress can be measured.

8. Open-source development is an example of bottom-up design in that the final system is built by expanding a simpler system. In fact, the functionality of the final system is usually not known at the beginning of the project. Instead, it evolves according to the desires of the developers.

9. Using a constant clarifies the role of the value.

10. Coupling refers to the connections between modules; cohesion refers to the connectivity within a module. On the surface, one would like to minimize coupling (because that leads to independent modules that can be maintained individually) and maximize cohesion (because that leads to modules whose activities can be more easily understood).

11. The answer, of course, depends on the object that the student selects. As a general rule an entire device is usually only logically cohesive but individual components tend to be functionally cohesive. An entire automobile is an example of logical cohesion, whereas a component such as a single seat or the steering wheel is more functional cohesive.

12. Although disliked for other reasons, the control coupling obtained by a simple goto statement is less complex than that of a subprogram call. In particular, the transfer is in only one direction. On the other hand, the careless use of goto statements can introduce a tangled mess that may never be understood.

13. Passing parameters by reference would probably be consider the more “complex” form of data coupling because if allows two-way communication between the program units whereas passing parameters by value does not allow the procedure to communicate back to the calling environment.

14. One problem would be that if the structure of one of the global elements was modified, it would be difficult to identify the portions of the program that access that element so they could be adjusted.

15. If an instance variable is private, no data coupling based on that variable can occur between objects.

16. The problem of updating shared data as discussed in Section 5.6 of the previous chapter.

17. a. W

b. W if W called it, X if X called it.

c. no

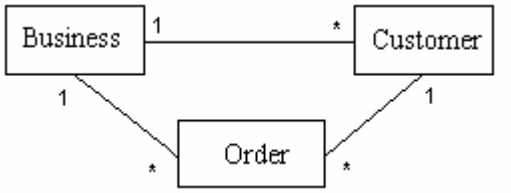
d. yes, through the common use of data element a.

e. element a.

f. data coupling through data element a.

18. Answers will vary. The point is for students to understand structure charts and to think about how their modular designs would hold up against future modifications in the system. (Did they, for instance, minimize data coupling?)

19. How about something like this?



20.

21. UML is a standard for representing object-oriented designs.

22. Answers will vary. The library should be represented as a large rectangle and the patron should be represented as a stick figure outside the rectangle. Two important uses that may be depicted are "check out book" and "return book." Others might include "browse" and "access catalogue."

23. Answers will vary. Perhaps the simplest would should a "solicit payment" message being sent to the customer followed by a "make payment" message being returned. On the other hand, some students may get creative and follow a longer sequence showing the customer refusing to pay and perhaps having his or her service cut off. The point is for the students to indicate that they understand collaboration diagrams.

24. Answers will vary. The important thing is that the students focus on the flow of data. One answer would be a diagram showing an "inventory record" being retrieved from the "inventory database" and merging with a "sales record" to form an "updated inventory record," which flows back to the "inventory database."

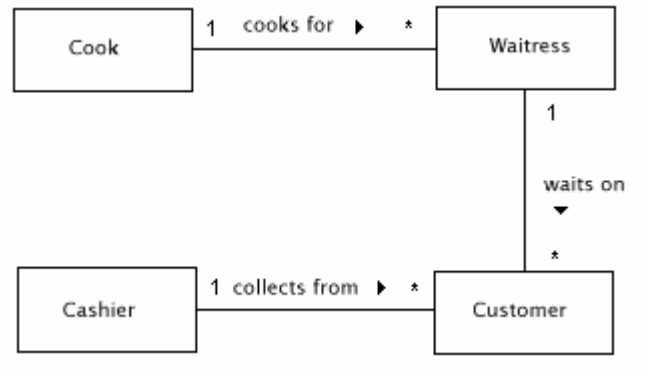
25. A class diagram represents relationships between classes, whereas a sequence diagram represents interaction between objects.

26. In a one-to-many relationship, each occurrence of one of the entities may be related to several occurrences of the other entity, but not vice-versa. In a many-to-many relationship, each occurrence of either entity may be related to several occurrences of the other entity.

27. There are many possible answers. An example of a one-to-many relationship is found between primary residences and individuals. (A residence may be the primary residence of several individuals but, by definition, each individual can have only one primary residence.) An example of a many-to-many relationship is found between stockholders and companies. (One person may own stock in many companies and a single company may have many stockholders.)

28. Answers will vary. The point is for the student to display an understanding of what a sequence diagram represents and the notation used. One sequence might be that the physician sends the patient a "request symptoms" message, the student responds by "reporting symptoms," and the physician replies with "prescribe action."

29.



30. Oops, this is the same problem as number 20. (It must be a really good problem.)

31. Immediately before the bottom most arrow, insert duplicates of the second and third horizontal arrows.

32. a. X = User, Y = Tool, and Z = Manufacturer. A major clue is that tools are used by users.

b. No

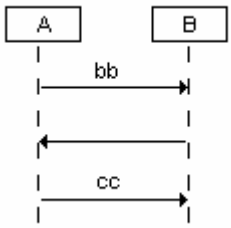
c. No

d. No

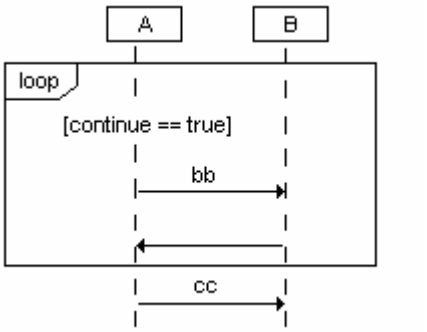
33. a. use case diagram b. class diagram c. sequence diagram

34. a. X b. Z c. No

35. The surrounding details will vary, but the core of the answer should look like this:

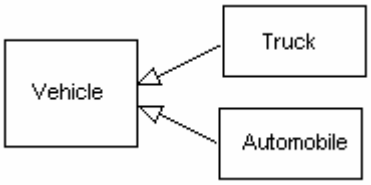


36.



A good way to relate this material to the previous topic of control structures back in chapters 5 and 6 is to point out that this is UML's way of representing a pretest loop. Ask your students to suggest a notation that UML could use to represent a posttest loop. (Move the "test condition" to the bottom of the interaction fragment.)

37.



38. Both reportMedicalHistory and reportPatientAddress.

39. Careless use of inheritance results in a strong coupling between the parent and child classes that may lead to unforeseen errors later in software maintenance or may even need to be unraveled to implement future modifications.

40. One possible answer would be in the context of university schedules. Some follow the two-semester-summer-session pattern while others follow the quarter-system pattern. Another would be in television broadcasting where programs fit a variety of patterns such as 30-minute national news, 30-minute situation comedy, one-hour newsmagazine, etc.

41. By means of design patterns, software engineers hope to construct predefined building blocks from which large systems can be constructed.

42. The point here is for the student to realize that the control structures are exactly small scale design patterns. In chapters 4 and 5 we tended to use flowcharts to represent such patterns. The statement structures in a particular language are frameworks for those patterns. When a programmer writes a program, he or she customizes those frameworks to fit the specific needs of that particular program. In this sense the syntax diagrams for a programming language make up a cookbook. (Note that this is not only true for the control structures in a language. Indeed, each statement structure in a language is a framework for a small design pattern.)

43. All of them. The Pareto principle has a wide range of applications. (This could make an interesting discussion topic.)

44. Software engineers, according to the Pareto principle, expect errors to be concentrated in certain parts of a software system. Thus, they see large software systems as heterogeneous mixtures of errors.

45. Black-box testing tests the performance of software without regard for its internal construction. Glass-box testing is done with the knowledge of the system's internal design.

46. Analogies to black-box testing would be a customer test driving a new automobile and another customer sampling a flavor of ice cream. An analogy to glass-box testing would be a health inspector evaluating the conditions under which an ice cream parlor prepared its ice cream.

47. In beta testing the tester is only allowed to test; in open-source development the "tester" is allowed to test and modify the software. Thus, beta testing is a black-box methodology whereas open-source development is a glass-box methodology.

48. Since half of the known errors were found, it's reasonable to assume that half of all the errors were found. Thus, we conclude that there were a total of 400 errors in the system. 200 of these were found and removed. Another 50 were known but not found. These were removed also. Consequently, we would estimate that 150 errors remain in the system.

49. GOMS is a methodology for measuring the efficiency of a human-machine interface.

50. Ergonomics is the engineering discipline that deals with designing systems that harmonize with the physical capabilities of humans. Cognetics is the engineering discipline that deals with designing systems that harmonize with the mental capabilities of humans.

51. Copyright laws were designed to protect form rather than function, but the investment is software development is often in the function rather than in its form.

52. One cannot patent natural phenomena, which tends to include algorithms. Patents are also expensive and obtained slowly in relation to the speed in which technology is changing.

Chapter Eight
DATA ABSTRACTIONS

Chapter Summary

This chapter contains and introduction to the subject of data structures along with a heavy dose of abstraction. Sections 8.1 through 8.4 provide an introduction to traditional data structures and their implementation. Sections 8.5 and 8.5 trace the evolution of data abstraction beginning with primitive data types, and progressing through user defined types, abstract data types, and classes. The final section discusses how pointers can be implemented in main memory and shows how the machine language of Chapter 2 could be extended to include indirect referencing.

Comments

1. An interesting activity for students is to see what a NIL pointer actually looks like in the system being used. This can be done by running a short program that uses coercion to print the value of a NIL pointer as an integer. (Such techniques are, of course, frowned upon in the software development community, and thus it's getting harder and harder to trick modern programming environments into viewing a "pointer" as an integer.)

2. You may want to point out that the technique of maintaining a vector of pointers that lead to components of interest (as depicted in Figure 8.7 part b) is a very generic idea. For example, it can be used for storing a list so that sorting the list involves merely rearranging the pointers within the vector. Moreover, the list could have two orders associated with it by maintaining two vectors of pointers. The "vector of pointers" idea is also nothing more than an index in disguise. I think that pointing out such generalities encourages students to see a broad picture and to think creatively.

3. I like to return to the concept of structured programming and the evils of the goto statement during the period that I'm teaching data structures. The point that I make is that pointers are to data structures as goto statements are to imperative programs. Each has the potential of creating a tangled mess that cannot be understood. I have never understood why we preach the glories of constructing webs of data using pointers but scorn the goto statement. It seems to me that both should be considered dangerous.

Answers to Chapter Review Problems

1. Row-major order: A B C D E F G H I J K L
   Column-major order: A E I B F J C G K D H L

2. Beginning at address 20, we must jump over (8 x (3 - 1)) + (4 - 1) = 19 more cells. Thus, the address of the third row, fourth column entry would be 39.

3. (6 x (4 - 1)) + (3 - 1) = 20, so the address of the third row, fourth column entry would be 40.

4. One problem described in the text is that additions to the list could force the entire list to be moved to a new location. (In general, arrays are inherently static and thus are problematic when used to implement dynamic structures. On the other hand, this is normally done in the case of stacks.)

5. Use a contiguous block of memory cells in which the planes in the three-dimensional array are stored as consecutive two-dimensional planes in row major order. If each entry requires one memory cell and the first cell of the block is at address x, then the (i, j, k)th entry will be located at the address x + (R × C)(i - 1) + C(j - 1) + (k - 1), where R is the number of rows and C is the number of columns.

6. The letters A, B, and C must be moved forward, or the letters E, F, and G must be moved backward, or the entire list must be moved to another location in memory.

7. The head pointer should contain 17.

Address	Contents	Address	Contents	Address	Contents
11	C	15	E	19	U
12	15	16	21	20	00
13	G	17	B	21	F
14	19	18	11	22	13

8. With N removed:

Address	Contents	Address	Contents	Address	Contents
30	J	34	X	38	K
31	38	35	46	39	40
32	B	36	N	40	P
33	30	37	40	41	34

After inserting G:

Address	Contents	Address	Contents	Address	Contents
30	J	34	X	38	K
31	38	35	46	39	40
32	B	36	G	40	P
33	36	37	30	41	34

9. The list currently spells JANE. To spell JEAN, it should be changed to:

Address	Contents	Address	Contents	Address	Contents
40	N	44	J	48	M
41	00	45	46	49	42
42	I	46	E	50	A
43	40	47	50	51	40

10. Routine 1 is correct. Routine 2 results in the New Entry pointing to itself.

11. Copy the address in the head pointer of one list into the cell containing the NIL pointer in the other list.

12. In either case the problem can be solved using the merge algorithm that the students will see applied to sequential files in Chapter 8. One solution, therefore, is to alter Figure 8-3 to refer to NIL pointers rather than EOF marks, list entries rather than records, and lists A, B, and C rather than transaction, old master, and new master files.

In the case of linked lists, however, the merge can be performed without producing a third list. That is, the entries can remain in the same places in memory while their pointers are altered.

13.
```
Previous ← NIL;
 Current ← value of the head pointer;
 while (Current not NIL) do
  (Next ← value pointed to by Current;
   pointer in the current entry ← Previous;
   Previous ← Current;
   Current ← Next)
 head pointer ← Previous
```

14. a.
```
procedure ReversePrint (List)
CurrentPointer ← Head pointer of List;
while (CurrentPointer is not NIL) do
  (Push the entry pointed to by CurrentPointer onto the stack;
   CurrentPointer ← value of next pointer in entry pointed to
                    by CurrentPointer
  )
while (stack not empty) do
  (pop an entry from the stack and print it)
```

b. The stack structure is hidden in the implementation of the recursive process.
```
procedure ReversePrint (List)
if (head pointer of List not NIL)
      then (apply ReversePrint to the list following the
              first entry of List;
            print the first entry in List)
```

15. The head pointer of the first list should contain 63, while the head pointer of the second list should contain 66.

Address	Contents	Address	Contents	Address	Contents
60	O	66	A	72	R
61	69	67	72	73	60
62	72	68	63	74	00
63	C	69	L		
64	66	70	00		
65	69	71	60		

16. A pop operation will "remove" the entry A and change the stack pointer to 11.

17. The stack pointer would be changed to 13 and the memory would appear as:

Address	Contents
10	F
11	C
12	A
13	D
14	E

18.
```
if (stack not empty) then
     [repeat
       (Pop an entry from the original stack and
        push it on an auxiliary stack)
      until (original stack empty)
 Pop an entry from the auxiliary stack and save if as the target value
 while (auxiliary stack not empty) do
     (Pop an entry from the auxiliary stack and
      push it on the original stack)]
```

19.
```
Equal ← true;
while (neither stack empty) do
     (Pop an entry from each stack
     if (these entries are not equal)
           then (Equal ← false)
      Push the entries from the first and second stacks onto
           auxiliary stacks Aux1 and Aux2, respectively)
if (either stack not empty)
     then (Equal ← false)
while (neither auxiliary stack empty) do
     (Pop entries from auxiliary stacks and
      push them onto the corresponding original stacks)
```

20. With two stacks the fundamental order of the entries cannot be altered. (The situation is analogous to holding the ends of a slinky in your hands.) With three stacks any arrangement is possible. (You may like to point out that this problem is much like the Towers of Hanoi puzzle.)

21. Move all of the entries above the adjacent entries to another stack. Then move the adjacent entries to different stacks and return them to the original stack in the opposite order. Finally retrieve the entries that were moved in the first step.

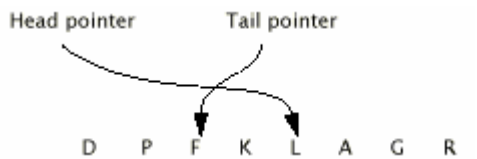
22. If we tried to stack the names themselves, the amount that the stack pointer must be adjusted would vary with the length of each name. By storing the names elsewhere and stacking the pointers, we obtain a stack whose entries are of uniform size.

23. A queue crawls in the direction of its tail. This is where new entries are added.

24. There could be several creative answers here, but most students will probably opt for a linked list.

25. The head pointer will contain 13 and the tail pointer will contain 18.

26. a.

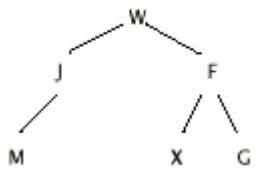


b. The size of the queue would exceed the space allotted to it.

27. Set aside a one-dimensional array, of which each entry is capable of holding one entry in the queue. Then, instead of using memory addresses in the head and tail pointers, each pointer contains an integer representing the index of the appropriate location in the array.

28. Suppose the queues were A and B and the entries to be reversed are in queue A. Cycle the entries in queue A until the first of the target entries comes to the front. Then move this entry to the tail of queue B and move the next entry in queue A to the tail of A. Now cycle queue B until the entry from queue A comes to the front and then move that entry to the tail of queue A. Finally, cycle queue A until all its entries except the reversed entries are back in their original positions.

29.



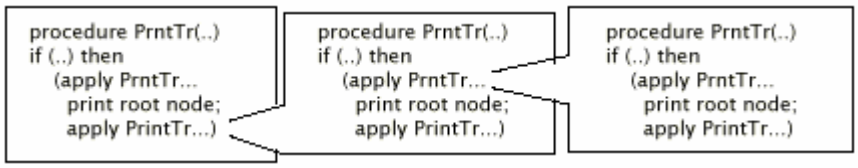
30. The root pointer should contain 42.

Address	Contents	Address	Contents	Address	Contents
30	C	36	K	42	G
31	00	37	33	43	30
32	39	38	45	44	36
33	H	39	E	45	P
34	00	40	00	46	00
35	00	41	00	47	00

31.
```
procedure BinarySearch (Tree, TargetValue)
 CurrentPointer ← root pointer of Tree;
*The node pointed to by CurrentPointer will be called the current node*
 Found ← false;
 while (Found is false and CurrentPointer is not NIL) do
   (Perform the activity associated with the appropriate
    case below:
     case 1, TargetValue = current node:
            (Found ← true)
     case 2, TargetValue < current node:
            (CurrentPointer ← current node's left child pointer)
     case 3, TargetValue > current node:
            (CurrentPointer ← current node's right child pointer)
   )
 if (Found = false) then (declare the search a failure)
                    else (declare the search a success)
```

32.
```
CurrentPointer ← root pointer;
*The node pointed to by CurrentPointer will be called the current node*
 while (CurrentPointer is not NIL) do
  (while (the left child pointer in the current node is not NIL) do
    (Push the address in CurrentPointer on a stack;
     CurrentPointer ← left child pointer in the current node)
   Print the current node;
   while (the stack is not empty and the right child
            pointer in the current node is NIL) do
        (Pop an address from the stack;
         CurrentPointer ← this address;
         Print the current node)
   CurrentPointer ← right child pointer in the current node)
```

33.

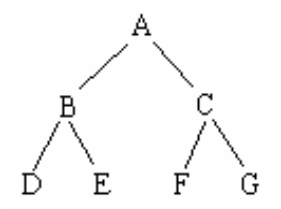


34.

Address	Contents	Address	Contents	Address	Contents
40	G	46	J	52	F
41	52	47	00	53	00
42	46	48	49	54	00
43	X	49	M	55	W
44	00	50	00	56	40
45	00	51	00	57	43

35.

36.



37. If you needed to search the list often, you might want to implement it as a binary tree as discussed in the text. If the tree was to be static, you might want to implement it as a contiguous list as described in the text.

38. This is a fairly straightforward pointer exercise. You may want your students to write an algorithm for inserting a new child in a system that contains pointers from parents to children as well as from sibling to sibling.

39. One's first inclination would probably be an 8 by 8 matrix. However, other structures may be more compatible with developing a program for playing chess. For example, toward the end of a game, it may be advantageous to maintain a list of merely the remaining pieces and their locations.

40. All three of them.

41. Just interchange the terms left and right.

42. How about a linked structure with parent, child, sibling, and spouse pointers? New nodes will need to be added but none deleted. You may ask your students how the structure of the tree would be different if it were constructed from a given individual backward (the individual as the root and the children of each node representing parents) or from an individual forward (with the children of each node representing the children of the person).

43. Use the normal tree search algorithm to find the node to be deleted, but keep a record of the previous node each time a new node is investigated. If the node to be deleted has a left child, trace the right child pointers below that child until a NIL pointer is found, and change this pointer to point to the right child of the node to be deleted. Then, replace the node to be deleted by its left child. If the node to be deleted does not have a left child, then merely replace the node to be deleted by its right child (which may be represented by the NIL pointer). Note: This approach may increase the depth of the tree to an unacceptable degree. Ask your students to explain how this could happen and how the algorithm could be changed to avoid this problem.

44. One approach would be to attach children to a node as a linked list, with the head pointer contained in the parent node.

45. How about something like:

```
define type EmployeeType to be
  {LastName    char[30];
   FirstName   char[20];
   MiddleName  char[20];
   PayScale    int[5];
       .
       .
       .
  }
```

46. Answers will vary. The point is that the abstract data type should include procedures for inserting and deleting entries from the list and perhaps additional operations.

47. Answers will vary. The point is that the abstract data type should include insert and delete procedures. If the type is called QueueType then instances of the type would be created by statements such as

```
QueueType WaitingLine;
```

and an entry would be inserted in the queue via a statement such as

```
WaitingLine.insert(entry)
```

48. a. A primitive type is defined as a part of the language, a user-defined type is defined in a program. Also a primitive type has operations associated with it, whereas a programmer cannot attach new operations to a user-defined type.

b. An abstract data type includes procedures for manipulating instances of the type.

49. The major data structure would be a list containing the names, addresses, etc. of the entries. This, of course, could be implemented as a dense list (probably not), a linked list, or perhaps a tree. The procedures would include those for inserting, altering, deleting, and retrieving entries from the list.

50. The data structures might include a heterogeneous array containing the location, velocity, and other data relating to the status of the spacecraft. (Knowing today's games, this would include its weaponry, and state of damage.) The procedures would include those for changing its velocity, firing weapons, and responding to hits by other weapons.

51. A simple version (that is not exactly Java or C#) would be

```
class QueueOfIntegers
{private int[] Entries = new int[20];
 private int head = 0;
 private int tail = 0;
 public void insert(int new)
 {Entries[tail] = new;
  tail = (tail + 1)mod 20;
 }
 public int remove()
 {int value = Entries[head];
  head = (head + 1)mod 20;
  return value;
 }
}
```

52. A class is not a *data* type in the sense that it may not involve data. Moreover, the use of classes usually incorporates encapsulation and inheritance.

53. E50F, 2001, 5F0F

54. 20B6, 21A1, 22A0, D300, E301, E200

55. The pointer being used can be altered via arithmetic operations more quickly if the pointer is held in a register (as with the form DR0S) than if held in main memory (as with the form DRXY). The form DRXS has the additional advantage of allowing easy access to items within a heterogeneous array. (If S points to the beginning of the array, then an item within the array can be identified merely by setting X to the proper displacement.)

Chapter Nine
DATABASE STRUCTURES

Chapter Summary

This chapter begins by introducing a database as an integrated data system that provides multiple views of the information within it. The distinction between conceptual and actual organizations is emphasized along with the layered approach to database systems (application software, database management system, actual data) and the goal of achieving data independence.

The second section is devoted to the relational database model. The conceptual organization of the model is described in terms of how an employee information database might be constructed. Then, the rudiments of data retrieval (the select, project, and join operations) are explored, and finally some simple SQL queries are presented.

The following sections present the basic concepts involved in object-oriented database design, some issues relating to the concurrent execution of transactions, the rudiments of sequential, indexed, and hash files, and the effects that database technology is having on society.

Comments

1. In addition to presenting the concept of a database, this chapter once again emphasizes the importance of abstract tools and how they are provided by a layered (or modular) organization. Thus, the chapter provides another example reinforcing the modular approach to software development. You may wish to exploit this opportunity more heavily than the text does.

2. There are several database systems available for PCs, most of which are based on the relational model. You may wish to use one of these for demonstration purposes or for student activities. In particular, there is Microsoft's Access. Using such systems gives non-majors the feeling that they are learning something that may be useful to them in the future.

3. Database systems play a major role in today's Web. Indeed, most of the more popular websites today are essentially interfaces to a database. Getting your students to recognize this should make this chapter more relevant to them.

Answers to Chapter Review Problems

1. A simple file presents its data in a single format and tends to be used for a single application. A database allows its data to be referenced via a variety of formats and tends to be used in numerous applications.

2. Data independence means that the application software is independent of the actual format in which the data is stored.

3. The database management system provides abstract tools that allow the database to be manipulated as though it was actually organized according as described by the database model.

4. A schema describes the structure of the entire database, whereas a subschema describes only that part of the database that relates to a particular user.

5. Here are three reasons: It simplifies the design of application software, it provides an opportunity for data independence, and it places the database management system in position to enforce the restrictions of subschema.

6. Both abstract data types and database models describe an abstract image of the data that simplifies the users interaction with the data. This interaction is supported by the procedures defined in the abstract data type or in the DBMS. In a sense, an instance of an abstract data type is a miniature database.

7. a. Designer of the database management system

b. End user

c. Designer of the database management system (although some systems allow the application programmer to have a say in this decision)

d. Programmer of application software

e. Designer of the database management system

8. All of them (a, b, and c).

9.

Airline	Flight
Clear Sky	CS205
Clear Sky	CS37
Clear Sky	CS102
Long Hop	LH67
Long Hop	LH89
Tree Top	TT331
Tree Top	TT809

Flight	Passenger	Seat
CS205	Smith	12B
CS37	Smith	18C
LH89	Smith	14A
CS37	Baker	18B
LH89	Baker	14B
LH67	Clark	5A
TT331	Clark	4B

10. If the PROJECT operation preserves the attribute used in the SELECT "where clause," either operation can be applied first.

11. All one must do is simply following the JOIN operation with a SELECT operation that retrieves the tuples that would have been selected by the "where" clause.

12. a.

U	W
A	5
B	3
C	5

b.

U	V	W
A	Z	5
C	Q	5

c.

S
J
K

d.

X.U	X.V	X.W	Y.R	Y.S
A	Z	5	3	J
A	Z	5	4	K
B	D	3	3	J
B	D	3	4	K
C	Q	5	3	J
C	Q	5	4	K

13. a.
```
TEMP ← SELECT from MANUFACTURER where PartName = "bolt 2Z";
RESULT ← PROJECT CompanyName from TEMP.
```

b.
```
TEMP ← SELECT from MANUFACTURER where CompanyName = COMPANY X;
RESULT ← PROJECT PartName, Cost from TEMP.
```

c.
```
TEMP1 ← JOIN PART and MANUFACTURER
    where PART.PartName = MANUFACTURER.PartName;
TEMP2 ← SELECT from TEMP1 where PART.Weight = 1;
RESULT ← PROJECT MANUFACTURER.CompanyName from TEMP2.
```

14. a.
```
select CompanyName
    from MANUFACTURER
    where PartName = "bolt 2Z"
```

b.
```
select PartName, Cost
    from MANUFACTURER
    where CompanyName = COMPANY X
```

c.
```
select CompanyName
    from PART, MANUFACTURER
    where PART.PartName = MANUFACTURER.PartName
      and PART.Weight = 1
```

15. a.
```
RESULT ← PROJECT Name, Address from EMPLOYEE.
```

b.
```
TEMP1 ← JOIN ASSIGNMENT and JOB where ASSIGNMENT.JobId = JOB.JobId;
TEMP2 ← SELECT from TEMP1 where Dept = Personnel;
TEMP3 ← PROJECT EmplId from TEMP2;
TEMP4 ← JOIN TEMP4 and EMPLOYEE;
TEMP5 ← SELECT from TEMP4 where TEMP3.EmplId = EMPLOYEE.EmplId;
RESULT ← PROJECT Name, Address from TEMP5.
```

c.
```
TEMP1 ← JOIN ASSIGNMENT and JOB where ASSIGNMENT.JobId = JOB.JobId;
TEMP2 ← SELECT from TEMP1
          where (Dept = Personnel) AND (TermDate = '*');
TEMP3 ← PROJECT EmplId from TEMP2;
TEMP4 ← JOIN TEMP3 and EMPLOYEE
          where TEMP3.EmplId = EMPLOYEE.EmplId;
RESULT ← PROJECT Name, Address from TEMP4.
```

16. a.
```
select Name, Address
      from EMPLOYEE
```

b.
```
select Name, Address
      from EMPLOYEE, JOB, ASSIGNMENT
      where ASSIGNMENT.JobId = JOB.Job.Id
            JOB.Dept = Personnel
            and EMPLOYEE.EmplId = ASSIGNMENT.EmplId
```

c.
```
select Name, Address
      from EMPLOYEE, JOB, ASSIGNMENT
      where JOB.Dept =Pesonnel
            ASSIGNMENT.TermDate = '*'
            and ASSIGNMENT.EmplId = EMPLOYEE.EmplId
```

17. A good approach would be to use three relations. One containing information about composers; one containing information about compositions; and another linking composers to compositions.

18. One solution would use three relations: Performers, Recordings, and Composers. The Recordings relation would link the three together by containing the performers and composers of the various recordings.

19. A good approach would be to use three relations—one containing information about manufacturers; one containing information about products; and another linking manufacturers to products.

20. Use two relations: one called Publish, the other Subscribe. Publish contains two attributes: Publisher and Magazine. Subscribe contains two attributes: Magazine and Subscriber.

21. Use three relations named PARTS, SUPPLIERS, and CUSTOMERS to store information about the parts, suppliers, and customers. Use a fourth relation called ORDERS, with attributes Part, Supplier, Customer, and Quantity, to store information about each order.

22.
```
TEMP1 ← JOIN JOB and ASSIGNMENT where JOB.JobId = ASSIGNMENT.JobId;
TEMP2 ← SELECT from TEMP1 where Dept = Accounting;
RESULT ← PROJECT JobId, StartDate, TermDate from TEMP2.
```

23.
```
select JobId, StartDate, TermDate
    from JOB, ASSIGNMENT
    where JOB.JobId = ASSIGNMENT.JobId
    and JOB.Dept = Accounting
```

24.
```
TEMP1 ← SELECT from ASSIGNMENT where TermDate = "*";
TEMP2 ← JOIN JOB and TEMP1 where JOB.JobId = TEMP1.JobId;
TEMP3 ← PROJECT EmplId, JobTitle, Dept from TEMP2;
TEMP4 ← JOIN EMPLOYEE and TEMP3 where EMPLOYEE.EmplId = TEMP3.EmplId;
RESULT ← PROJECT Name, Address, JobTitle, Dept from TEMP4.
```

25.
```
select Name, Address, JobTitle, Dept
    from JOB, EMPLOYEE, ASSIGNMENT
    where ASSIGNMENT.TermDate = "*"
    and EMPLOYEE.EmplId = ASSIGNMENT.EmplId
```

26.
```
TEMP1 ← SELECT from ASSIGNMENT where TermDate = "*";
TEMP2 ← JOIN JOB and TEMP1 where JOB.JobId = TEMP1.JobId;
TEMP3 ← PROJECT EmplId, JobTitle, from TEMP2;
TEMP4 ← JOIN EMPLOYEE and TEMP3 where EMPLOYEE.EmplId = TEMP3.EmplId;
RESULT ← PROJECT Name, JobTitle from TEMP4.
```

27.
```
select Name, JobTitle
    from ASSIGNMENT, JOB, EMPLOYEE
    where ASSIGNMENT.TermDate = "*"
    and EMPLOYEE.EmplId = ASSIGNMENT.EmplId
```

28. In contrast to the single relation system, one cannot precisely determine the telephone number of either Jones or Smith in the two relation system.

29. One needs only a simple relation with attributes Part and Subpart. In this case, a part could well be presented in both columns: in the left column for each of its subparts and in the right column for each part in which it is used.

30. Answers will vary. The point is for the students to show an understanding of the relational database model, and observe the role that databases play in today's Web.

31. Find the dates at which each current position within the company was last filled.

32.
```
select JobId, StartDate
    from ASSIGNMENT
    where TermDate = "*"
```

33. Find the dates at which the current employee started his or her current position.

34.
```
select Name, StartDate
    from EMPLOYEE, ASSIGNMENT
    where EMPLOYEE.EmplId = ASSIGNMENT.EmplID
      and TermDate = "*"
```

35. Find the names of those people who have been employed in the sales department.

36.
```
select Name
      from EMPLOYEE, JOB
      where EMPLOYEE.EmplId = JOB.EmplId
      and Dept = "Sales"
```

37.
```
TEMP1 ← JOIN ASSIGNMENT and JOB where ASSIGNMENT.JobId = JOB.JobId
TEMP2 ← SELECT from TEMP1 where JOB.EmplId = "34Y70"
RESULT ← PROJECT jobTitle from TEMP2
```

38.
```
TEMP1 ← JOIN ASSIGNMENT and EMPLOYEE
            where ASSIGNMENT.EmplId = EMPLOYEE.EmplId
TEMP2 ← SELECT from TEMP1 where EMPLOYEE.Name = "Joe E. Baker"
RESULT ← PROJECT StartDate from TEMP2
```

39. It would add a tuple (containing the values Company Z, Bolt 2X, and .03) to the Manufacturer relation.

40. It would change the cost of part Bolt 2X from Company Y to .03.

*41. One class of objects might be shelves. Each of these objects, could receive messages to hold or release items. Each of these items would also be objects. Another class of objects would be suppliers, each of which would receive orders.

*42. As in the grocery store example, one class of objects might be shelves, which would receive messages to hold or release items. In this case the items would be books. Each book object might respond to messages to check itself in or out. Over objects might include library customers who may have to respond to the message to pay a fine.

*43. The sum computed by T1 would be $100 too large because it added the balance of A before the transfer made by T2 to the balance of B after the transfer.

*44. T2 would require an exclusive lock on the balances of both A and B. This would keep T1 from using any of these values until T2 had performed its task. The result would be that the transactions would not be interwoven.

*45. If T1 were the younger transaction, the transfer between accounts A and B would be performed before the sum was computed. If T2 were the younger transaction, the sum would be computed before the transfer was performed.

*46. Call the transaction that adds $100 T1 and the other transaction T2. Then, if T1 is performed in the middle of T2, the final balance will be $100. However, if T2 is performed in the middle of T1, the balance will be $300.

*47. Many transactions can have shared access to an item at the same time, but only one transaction can have exclusive access. This allows transaction to alter the database in a safe manner.

*48. Allowing two word processors to update the same document at the same time leads to the same lost update errors that can occur in database systems. The problem statement hints at the following experiment using Windows and Microsoft Word: Start an activation of Word and open a document. Then start another activation of Word and try to open the same document. The system will block the second attempt to open the document because the first activation of Word has exclusive access to it.

*49. 125 seconds (over 2 minutes).

*50. Steps 3 and 5 would be skipped.

*51. Change the else clause in the major while loop to read as follows:

```
else (write the current transaction record on the new master file;
      if (key field of current transaction record equals the key
            field of the current master record)
        then (if (EOF on old master file)
                  then (declare old master file empty)
                  else (read the next record from the old master file))
      if (EOF on transaction file)
        then (declare transaction file empty)
        else (read the next record from the transaction file))
```

*52. Records could be stored individually with a dual pointer system just like a doubly linked list. Another technique would be to store the file as an inverted file, which would allow records to be retrieved in the order listed in either index.

*53. First, design a standard format for the information about a subscriber in which each field is assigned a specific size. Fields would include such items as name, street address, city, expiration date, etc. Then encode the information about each subscriber according to this format and store these logical records one after the other.

*54. The point here is that the logical records cannot be separated by dividing the file into blocks of a fixed length. Instead, some other technique for separating records must be used. One approach would be to use a special symbol or symbol sequence to indicated the end of one logical record and the beginning of the next. (Note that this is the same idea used in text files to separate lines of text.)

*55. An indexed file does not suffer from clustering but a hashed file does not require the overhead of index maintenance.

*56. A directory maintained by an operating system may contain entries representing files and other entries representing subdirectories. Thus, there is not the clear distinction between the "index" and the "records" as there is in the case of an indexed file. Moreover, the directory system maintained by an operating system contains more information about each entry than merely its location. For example, a multiuser operating system might record the owner of each file in its entry in the directory system.

*57. The probability of all three records hashing to different locations would be (10/10)(9/10)(8/10) = .72, so the probability of at least two hashing to the same location would be .28. If a fourth record were added, the chances of at least two hashing to the same location would increase to .496, and a fifth record would increase this to .6976. Thus, with the five records, it is more likely for clustering to have occurred than not.

*58. The probability of at least two records hashing to the same bucket would be .0298. It is more likely for clustering to have occurred than not when 12 or more records are stored.

*59. 124/23 produces the quotient 5 with a remainder of 9. So, we should search bucket number 9.

*60. The point here is that the homogeneous array is stored as a hashed file with the address polynomial being the hash function.

*61. a. Overhead of maintaining index is not required.
b. Records can be processed easily in sequential order.
c. Individual records can be accessed quickly.
d. Records can be processed sequentially.
e. Individual records can be accessed quickly.
f. Overhead of maintaining index is not required.

*62. Both are accessed in a sequential manner, starting at the first and traversing each entry in the ordered in which they are stored.

Chapter Ten
COMPUTER GRAPHICS

Chapter Summary

This chapter focuses on 3-D graphics, with emphasis on the process of modeling objects and rendering scenes. Most of the material views the subject from the perspective of video games and animated films. These two applications provide contrasting real-time constraints and are therefore useful when discussing topics such as local versus global lighting models. The last section introduces the rudiments of animation.

Comments

1. An interesting class presentation would be to demonstrate a current video game and then discuss the underlying graphics. For example, you could discuss the objects appearing in the scene and help the class speculate on how they were created. You could also point out applications of such techniques as texture mapping and help the class identify features that reflect real-time rendering constraints (such as a local lighting model). Another approach would be to compare the graphics in an older video game system to a current one.

2. There is a lot of material on the Web regarding graphics. You might want to take advantage of it--perhaps by means of a research assignment.

3. If you want your course to be popular next semester, give your students the assignment to see the latest animated motion picture. (Of course, you should have them produce some form of a report, which will probably squash their enthusiasm for the assignment.)

4. The discussion of video games is a natural environment to bring up the subject of ethics in general and violence in particular. I've found two types of students. One group takes the subject seriously; the other brushes it off as "no big deal."

Answers to Chapter Review Problems

1. a. 2D b. 2D c. 3D

2. a. Film b. Rectangle in view finder c. Scene being photographed

3. Only when the straight line from the center of projection to the center of the sphere is perpendicular to the projection plane. Otherwise the image will be an oval.

4. No. The justification will vary depending on the mathematical sophistication of your students. (The projection is a linear process, and therefore straight lines will remain straight.) What we're really looking for here is for students to think and express their thoughts.

5. 2 feet. Students with some knowledge of trigonometry should realize that the triangle formed by the center of projection and the pole and by the center of projection and the image of the pole are similar triangles. Thus the ratios of their sides are equal. Students without a trigonometry background might solve the problem by drawing a sketch.

6. A parallel projection is constructed along parallel projectors, whereas a perspective projection is constructed along projectors that converge at the center of projection or view point.

7. The frame buffer contains a bit map representation of the image projected into the image window.

8. Answers can vary, but the one I have in mind is that animation for an interactive video game must be generated under real time constraints.

9. Answers will vary. Prominent properties that are traditionally modeled include the shape of the object, the surface characteristics of the object (color/color pattern, rough/smooth), and the object's material composition (refraction properties, light absorption characteristics). Properties that would not be modeled (traditionally) would include non-physical characteristics such as the emotion of an actor (although we're beginning to see models that incorporate artificial intelligence techniques that allow models to react in ways determined by their mental states).

10. One property would be texture, which could be added to the model by means of a texture map.

11. No, the points must lie on the same plane.

12. A pyramid

13. A triangular prism

14. Answers will vary. A natural approach would be to represent each face of the solid by a single patch. If a student uses patches that are not triangular, make sure that the patches are planar.

15. Answers will vary. The one that I have in mind is the shape whose vertices are located at the points (1, 0, 0), (-1, 0, 0), (0, 1, 0), (0, -1, 0), (0, 0, 1), and (0, 0, -1).

16. They do not lie in a common plane

17. b and c

18. One would be by applying a grammar to build the shape in the same manner as a parser builds a parse tree. Another would be to use a collection of particles that interact with one another to simulate the internal structure of the object being modeled.

19. a. Rendering—because since the rendering process is "standardized" by the rendering pipeline.
b. Rendering—because such steps as clipping, scan conversion, shading, etc. are computationally intense.
c. Modeling—because the process involves describing objects.

20. a, b, c, and d (All of them)

21. It marks the end of the creative modeling process and the beginning of the computationally complex rendering process.

22. Answers may vary. Points to be made are that models may be viewed from different directions and from different distances, meaning that an object may need to be modeled on all sides and in more detail than if viewed only from large distances. Moreover, objects that are not in the original field of view may need to be included in the scene graph.

23. c

24. 5 feet from the observer.

25. a

26. a. Inside b. Outside c. Outside d. Outside

27. Answers will vary. One would be that the object may be reflected by a mirror inside the view volume.

28. For each pixel in the image, the z-buffer contains the distance from the camera to the surface currently represented by the pixel in the frame buffer. As new surfaces are considered for rendering, this distance can be used to tell if the new surface is in front of or behind the currently represented surface.

29.
```
if (position in frame buffer empty OR
      entry in z-buffer > distance to current object)
    then (render point,
          store results in frame buffer,
          store distance to current object in z-buffer
         )
```

30. The object will appear to be either solid orange or solid blue. (This is an example of aliasing.)

31. Answers will vary. Points that might be mentioned include the fact that texture mapping associates a predetermined "pattern" with a surface (predetermined in the sense that the pattern is selected as a part of the modeling process before rendering begins), whereas bump mapping is a random process performed entirely during rendering. Moreover, texture mapping deals with color patterns whereas bump mapping deals with surface orientation.

32. Clipping—Identifying the part of the scene within the view volume.
Scan conversion/rasterization—Associating points in the scene with pixel positions.
Hidden-surface removal—Identifying surfaces that are blocked from view.
Shading—"Rounding off" the flat patches.

33. Advantages include less complex application software and greater (time) efficiency. A significant disadvantage is that the traditional rendering pipeline implements only a local lighting model rather than a global lighting model.

34. The answer that would be most pertinent to this chapter is that a video game machine contains hardware/firmware for implementing the rendering pipeline.

35. A local lighting model renders an object without regard for other objects. (Thus, it does not properly handle shadows.) A global lighting model renders objects in the context of the other objects in the scene. (It properly handles shadows.)

36. Ray tracing is a means of implementing a global lighting model, whereas the traditional rendering pipeline implements only a local lighting model. However, ray tracing is much more computationally intensive than the traditional rendering pipeline.

37. Distributed ray tracing tends to avoid the shinny appearance of objects inherent in traditional ray tracing. However, it is more computationally intensive.

38. Radiosity is a means of implementing a global lighting model, whereas the traditional rendering pipeline implements only a local lighting model. However, radiosity is much more computationally intensive than the traditional rendering pipeline.

39. Objects in the image produced by ray tracing would appear to have shinny surfaces, whereas the same objects in the image produced by radiosity would appear to have dull surfaces.

40. 129,600 frames = (90 minutes)(60 seconds/minute)(24 frames/second)

41. Answers will vary. One would be to generate particles at the base of the fire, apply dynamics to move the particles upward as though they were being pushed by rising air currents and being blown by wind, and finally delete the particles randomly toward the top of the system to simulate the cooling process. Note that this solution involves the creation and destruction of particles, which is not mentioned in the text.

42. Information in the z-buffer could be used to tell whether the moving object was passing in front of or behind objects that have already been rendered.

43. Digitizing and motion capture both involve identifying locations on objects. However, with digitizing the locations are identified directly by touching the surface, whereas in motion capture the locations are identified by indirect means. Moreover, the goal of digitizing is to capture the shape of an object, whereas the goal of motion capture is to capture motion.

44. An animator today works with three-dimensional virtual worlds, whereas an animator in the past worked with two-dimensional drawings. Moreover, the task of in-betweening is largely automated today (and performed in three-dimensional virtual worlds rather than two-dimensional drawings.)

Chapter Eleven
ARTIFICIAL INTELLIGENCE

Chapter Summary

This chapter begins by formulating research in artificial intelligence in the context of developing agents that exhibit rational (intelligent) behavior. It then proceeds to investigate the topics of perceiving and reasoning in the context of building an "intelligent" agent for solving the eight puzzle. Two additional areas in artificial intelligence, knowledge and learning, are explored in the Section 11.4, artificial neural networks are investigated in Section 11.5, and robotics is discussed in Section 11.6. The chapter closes by considering the social consequences of advances in artificial intelligence.

Comments

1. I think your students will be interested in the material on artificial neural networks. I tried to include enough to allow you to show how such networks are programmed and to explain some of the current research problems in the field. I find that students respond well to the material on associative memory.

2. A good example of a depth first search strategy is the approach to the problem of traversing a maze by staying next to the wall on your right until either the goal or a dead end is reached. (We'll assume that no loops can occur.) In the latter case one backtracks to the last possible choice that has not been tried, selects a new option at that point, and once again proceeds forward while staying against the wall on the right. This process is also a good instrument for demonstrating how a stack structure is embedded in the backtracking process.

3. You should be aware that the algorithm for solving the eight-puzzle that is developed in this chapter is not the same as the A*-algorithm. You may want to discuss the A*-algorithm in the classroom.

4. The eight-puzzle solving algorithm provides an example of how data structures can be extended from the traditional systems described in Chapter 8. In particular, a convenient technique for finding the "left-most leaf node with the smallest projected cost" is to link the leaf nodes together to form a linked list ordered by projected cost. Thus, each leaf node is in both a tree and a linked list. Depending on the emphasis you wish to place on data structures, the development of this structure might constitute the core of a meaningful class lecture that could be followed by the assignment of implementing the system.

5. This chapter is exceptionally suited to individual research projects using the Web as a source. If you're looking for an assignment requiring a written report or a class presentation, this is a good opportunity.

Answers to Chapter Review Problems

1. "Would I tell on you?" might be used to reassure. "Do you see that snake?" might be a warning. "Can't you ever do anything right?" might be used to criticize.

2. The agent has two sensors: one detects money, the other detects the selection made. Its actuator is its dispensing mechanism. Its responses would be considered reflexive.

3. a. Reflex b. Knowledge based c. Goal based

4. No. A computer can memorize a list of names perfectly whereas a human cannot.

5. Knowing your name and address would be examples of declarative knowledge. Being able to tie your shoes is procedural knowledge.

6. Instance variables hold declarative knowledge. Methods hold procedural knowledge.

7. a. simulation oriented
   b. performance oriented
   c. performance oriented
   d. simulation oriented
   e. performance oriented

8. Not in my opinion. On the other hand, I have to admit that on rare occasions I have initially thought the recorded message was being spoken live--but only for a few words. (The point here is for students to think about the Turing test and what it entails.)

9. Closed loop, open loop, short straight line.

10. In both cases one selects the best match as being the correct interpretation of the input.

11. If the corner is concave, the drawing would be interpreted as a single large block with a corner cut out of it. If the corner is convex, the drawing would be interpreted as a small block floating in front of a larger block.

12. "By the barn" tells where and "by the farmer" tells how.

13. The parsing process would consider the two sentences as being completely different. For example, the subject in the first is Theodore, while the subject in the second is zebra. However, semantic analysis would recognize that the two sentences actually have the same meaning.

14. The then and else clauses are different but the two sentences have the same meaning.

15. The question might be either a request for the time or a complaint that too much time has elapsed.

16. This is an example of how the meaning of a sentence can vary depending on its context. Here the significance of the action can vary from a simple act of childishness to one of abuse.

17. There should be entries for the people Donna, Jack, and the fielder, as well as the actions of hitting and catching associated with their correct objects and agents. Note that a problem surfaces as to how time relationships should be represented.

18. Database A could conclude that the employee did not belong to the company's health insurance program whereas database B could not.

18. The standard example is a database consisting of a single statement of the form "A or B." In this case the closed-world assumption leads to the contradictory conclusions ¬A and ¬B.

20. We normally apply the closed-world assumption when we deal with a list. If an item is not on the list, we assume that it is not supposed to be on the list, that is, we assume the list is complete. We also use the closed-world assumption when driving a car on the expressway—we assume that there is not construction around the bend unless we are told that there is.

21. The state graph is a representation of all possible states and paths, whereas the search tree represents only those states and paths under consideration.

22. The states are the various positions in which the cube may be. The productions are the various rotations that can be made. You are the control system.

23. a. There could be $2^{11} - 1 = 2047$ nodes in the tree when the goal is finally found.

b. Both search trees would contain at most $2^6 - 1 = 63$ nodes. Thus, the total number of nodes would be at most 126. (The point here is that the two trees would be half as deep as the single tree in part a, which significantly reduces the number of nodes considered.)

24. Students at this stage probably don't know the terminology of logical deduction or the predicate calculus, but I'm continually surprised by the quality of their own approaches to questions like this. The following is one approach to the problem using resolution. Let the predicates B, S, and T represent the predicates "is a basketball player," "is short," and "is tall," respectively. Then, each of the statements given in the problem can be expressed by clauses as follows:

`B(John` = John is a basketball player
`For all X, (¬S(X) OR ¬B(X))` = Basketball players are not short
`T(John) OR S(John)` = John is either tall or short

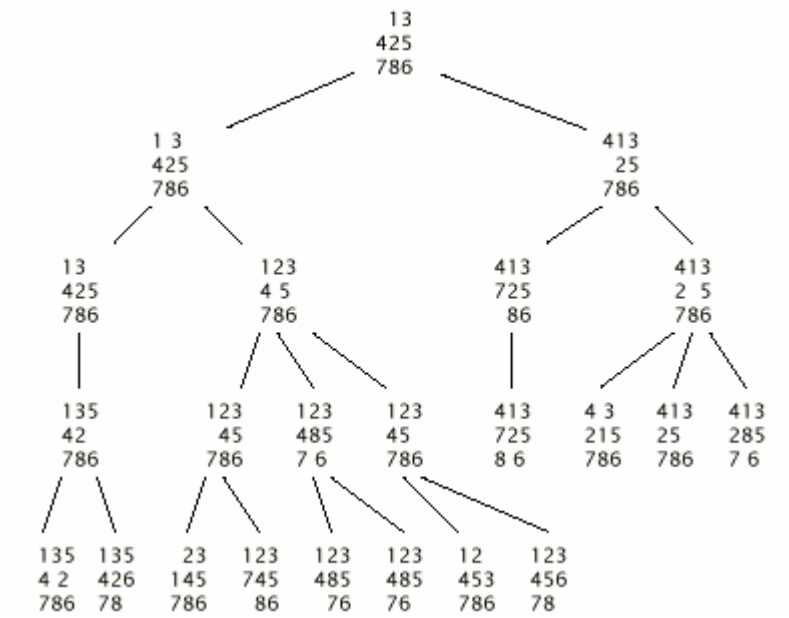
By universal specialization, the second clause can be modified to read ¬S(John) OR ¬B(John). Resolving this with the first clause gives ¬S(John), which resolves with the third clause to give T(John), representing the goal statement John is tall. Thus, the productions used are universal specialization and resolution.

25. Player X should select move B. This means that X cannot win but assures X of a tie. Note that following the production system examples in the text, one would be inclined to pick move A since this leads to leaf nodes that are most advantageous to X. However, Y will not allow X to reach these nodes. This is an example of the minimax search strategy common in searching game trees.

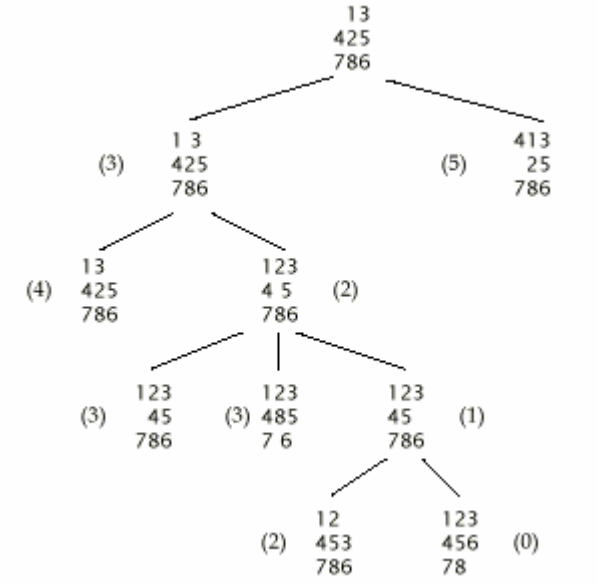
26. States would be board configurations and productions would be checker moves. A (perhaps too) simple heuristic would be the number of the opponent's checkers still on the board. Since this is a competitive game, the control system must compete with an adversary to move the system toward its goal. (You may want to show your students the mini-max process.)

27. Productions would include multiplying both sides of the equation by the same value, combining terms of similar type, and adding the same value to both sides of the equation. Heuristic rules would include moving all terms involving the variable for which the equation is being solved to the same side of the equation.

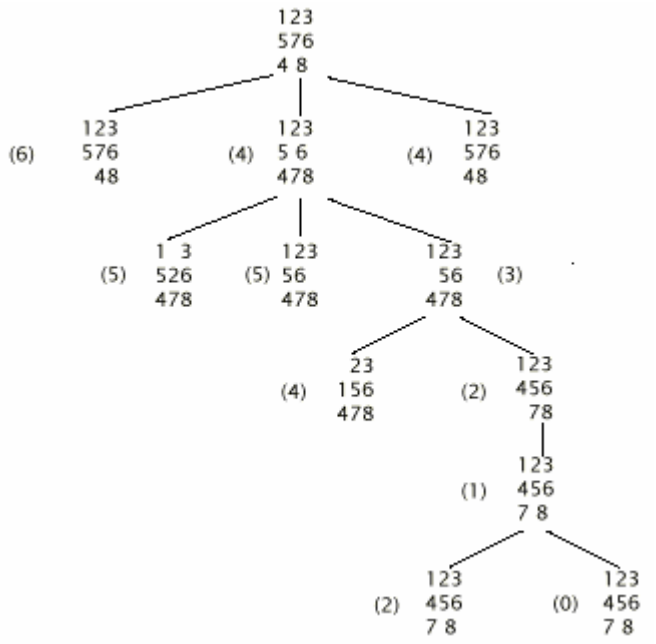
28.


13 425 786
1 3 425 786
413 25 786
13 425 786
123 4 5 786
413 725 86
413 2 5 786
135 42 786
123 45 786
123 485 7 6
123 45 786
413 725 8 6
4 3 215 786
413 25 786
413 285 7 6
135 4 2 786
135 426 78
23 145 786
123 745 86
123 485 76
123 485 76
12 453 786
123 456 78

29.


13 425 786
(3)
1 3 425 786
(5)
413 25 786
(4)
13 425 786
123 4 5 786
(2)
(3)
123 45 786
(3)
123 485 7 6
123 45 786
(1)
(2)
12 453 786
123 456 78
(0)

30.



31. It does not distinguish between different states as well.

32. In the case of the binary search, there is no doubt as to which way to go, whereas in the case of searching for a goal state in a production system one can only guess which direction is correct.

33. As the state in question is revisited over and over, the cost accumulated in reaching it will grow larger and larger until the projected cost ultimately exceeds that of another option. At that point the control system will shift its attention to the other option. Thus, the algorithm will ultimately find its ways to the correct path.

34. One normally follows the road leading in the direction of the destination.

35. It should be easy to calculate and be an approximation of the actual distance to the goal.

36. Each state consists of the buckets at a particular degree of fullness. Productions would be emptying a bucket, filling a bucket, or pouring the contents of one bucket into another.

37. One approach to this problem would be to distribute the heaviest crates between the two trucks and then try to distribute the lighter ones. This policy would fail if two of the crates weighed 5 tons each and the others were 2 tons each. Another approach might be to load the first truck with the heaviest possible crates and try to place the remaining crates on the other trucks. This policy would fail if the crate weights were 7, 6, 3, 3, 3, 3, and 3.

38. b and d

39. By solving the frame problem, a human is able to figure out where a lost article may have gone. ("I left the glue on the kitchen table yesterday, but the children were playing at the table this morning. Perhaps I should look in the children's room.")

40. a. Both involve using a model (a teacher) that demonstrates correct behavior.

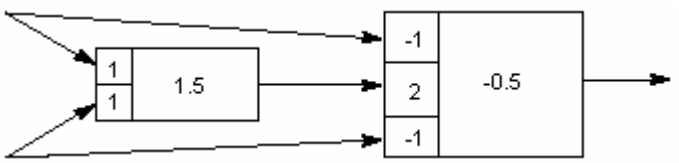
b. In the case of learning by imitation, the behavior of the model is strictly followed, but in learning by supervised training the model is used only as a guide.

41. If the network is initialized with only two units (separated by a single unit) excited, the network will complete the alternating pattern around the ring. If the network is initialized with all its units inhibited, the entire network will blink between excited and inhibited.

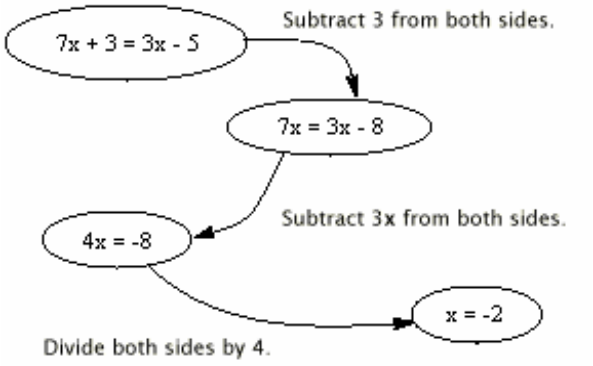
42. Any pattern in which at least three perimeter units are excited and the center unit is inhibited is associated with the pattern in which the perimeter units are excited and the center unit is inhibited. If the network is given an initial configuration in which only two opposing units are excited, it will blink—alternating between vertical and horizontal "lines."

43. There are numerous answers. One would be a rectangular array of processing units, each with a threshold value of 0.5. Each unit is connected to its immediate horizontal neighbors with a weight of -1 and to its immediate vertical neighbors with a weight of 1. (The "immediate neighbors" of a unit on the perimeter includes the unit on the perimeter on the opposite side.)

44. How about this?

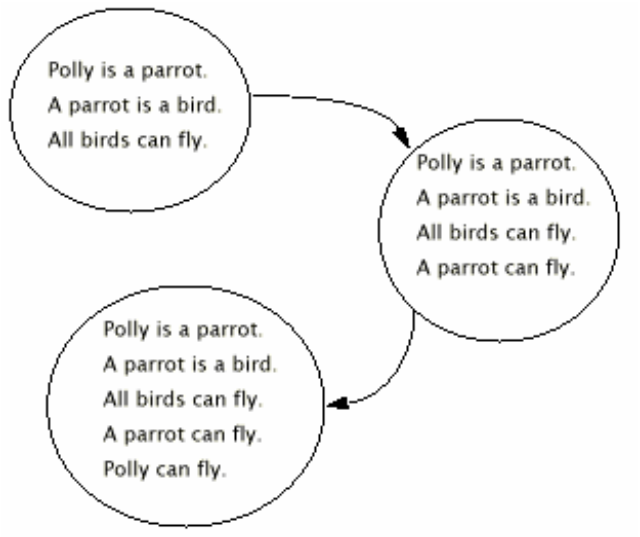


45.



46. One could begin by subtracting 3x from both sides, adding 5 to both sides, or dividing both sides by 4. Moreover, there are numerous less productive steps.

47.



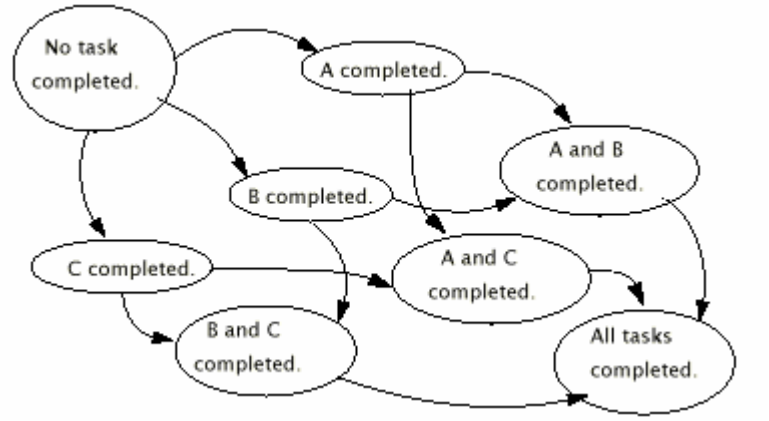
48. We normally use the closed-world assumption and assume that a bird can fly unless we know otherwise.

49. Is it a law for a new tax or a new law for an existing tax?

50. The state of being in a particular city is a state of the production system. The productions are the processes by which one moves from one city to another.

51. The states would be the various sets of tasks completed. Thus, the state graph would be the following:

52.



53. a. The second—which is actually the bubble sort algorithm.

b. Start with a large number of strings of pairs of integers in the range of one to ten. Test each string to see how well it sorts lists. Then, pick the best performers, cut each into two pieces, and reconnect the pieces to form another generation of potential sorting programs. Then repeat this process until a good sorting program is found.

54. The experiments may progress as follows: Try different sensor placements to see which perform the best. Then, analyze the findings to learn what works and what does not. Then experiment with new improved designs.

55. Answers will vary. The point is for the student to consider the distinction between and the roles for reactive and plan-based responses.

Chapter Twelve
THEORY OF COMPUTATION

Chapter Summary

This chapter introduces the subjects of computability as well as problem classification according to (time) complexity. It begins by presenting the concept of computing functions and introducing Turing machines as a (universal) means of performing such computations. Then, a simple universal programming language (which we call Bare Bones) is introduced and used to prove that the halting problem is unsolvable.

The discussion of problem complexity applies the material from Chapter 5 regarding the efficiency of the sequential and binary search algorithms and the insertion sort algorithm. Another example developed here is the merge sort algorithm.

The final section presents the RSA public key encryption system as a means of taking advantage of problems with high complexity.

Comments

1. The theory of computation is an important area of computer science that should not be omitted from an introductory survey course. It is important that beginning students understand that there is a rigorous background to the discipline—an insight they do not gain by seeing only Internet and data processing topics. Keep in mind that the goal is not for the students to master this material but for them to gain an appreciation for its existence and subject matter. They should understand that not every problem that appears to be algorithmically solvable is, in fact, so and that some problems that are solvable are too complex to be realistically solvable.

2. The early sections of this chapter (12.1 through 12.4) approach the theory of computation from the direction of computing functions. There is, of course, another popular approach known as formal languages, in which Turing machines are thought of as recognizing sentences instead of computing function values. I chose the function approach because I've found it to be more natural for beginning students. If, however, your students show special interest in such topics, Chomsky's hierarchy of grammars should make an interesting and relatively self-contained subject for further study.

3. The subjects of computability (Sections 12.1 through 12. 4) and complexity (Section 12.5) are treated separately. Thus, Section 12.5 can be covered without first discussing the earlier sections. In fact, the only reference to the subject of computability in these last sections is to the fact that there are problems that algorithmic systems cannot solve (referred to as unsolvable problems).

4. When discussing the halting problem with students who lack mathematical maturity, I like to have fun with the "if it does, it doesn't; but if it doesn't, then it does" style of argument. I give several examples such as the barber who shaves "all those and only those who do not shave themselves" (Does the barber shave himself?) just before I present the argument that the halting problem is unsolvable. This gets them in the spirit before launching into the less familiar context of self-terminating programs.

Answers to Chapter Review Problems

1. Using the shorthand notation of "copy X to W" developed in the text, we could write something like this.

```
copy X to W;
invert W; (See Section 12-1, Exercise 1)
while W not 0 do;
    .
    .
    .
copy X to W;
invert W; (See Section 12-1, Exercise 1)
end;
```

2.
```
clear Z;
copy Y to YY;
while YY not 0 do;
  incr Z;
  clear YY;
end;
copy X to XX;
copy Y to YY;
while XX not 0 do;
  clear Z;
  copy YY to YYY;
  while YYY not 0 do;
    incr Z;
    clear YYY;
  end;
  decr YY;
  decr XX;
end;
```

3.
```
clear Z;
incr Z;
copy X to XX;
while XX not 0 do;
  copy Z to ZZ;
  while ZZ not 0 do;
    incr Z;
    decr ZZ;
  end;
  decr XX;
end;
```

4. a .
```
clear Z;
  copy X to Y;
  while Y not 0 do;
    invert Z; (See Section 12-3, Exercise 1)
    decr Y;
  end;
```

b.
```
clear SUM;
copy X to Y;
while Y not 0 do;
  copy Y to Z;
  while Z not 0 do;
    incr SUM;
    decr Z;
  end;
  decr Y;
end;
```

5. The following routine places the quotient of X and Y in Z:
```
clear Z;
copy X to XX;
while XX not 0 do;
  copy Y to YY;
  while YY not 0 do;
    decr XX;
    decr YY;
  end;
  incr Z;
end;
copy X to XX;
while XX not 0 do;
  decr Z;
  clear XX;
end;
```

6. It computes X – Y when X ≥ Y.

7. It computes X AND Y.

8.
```
copy X to Aux1;
copy Y to Aux2;
clear Z;
while Aux1 not 0 do;
  incr Z;
  while Aux2 not 0 do;
    decr Z;
    clear Aux2;
  end;
  clear Aux1;
end;
```

Continued on next page.

```
while Aux2 not 0 do;
  incr Z;
  while Aux1 not 0 do;
    decr Z;
    clear Aux1;
  end;
  clear Aux2;
end;
```

9. All that is needed is to show that the while loop structure can be simulated in the new language.

```
while X not 0 do;
  S
end;
```

is equivalent to

```
     incr Aux;
10   if Aux not 0 goto 30;
20   S
30   if X not 0 goto 20;
```

10. One approach would be

```
incr name1;
clear name2;
repeat
  incr name2;
  decr name1;
until (name1 equals 0);
decr name2;
```

11. Begin with the results of the previous problem. Then, any loop of the form

```
while X not 0 do;
  B
end;
```

can be simulated by first copying all the variables involved into auxiliary variables. Then, increment the auxiliary copy of X and perform the loop

```
repeat
  S
until (AuxX equals 0);
```

where S is the sequence of statements that first copies all the auxiliary variables into their corresponding originals and the performs the original body using the auxiliary variables. (The overall effect is that the new body will be executed one more time than the original, but the effect on the original variables will be the same as the original pretest loop.)

12. An example would be a machine that, regardless of its state or the contents of the tape, writes a zero on the tape, does not move to another position on the tape, and remains in the same state.

13.

current state	current cell contents	value to write	direction to move	new state to enter
START	0	0	left	SEARCH
START	1	1	left	SEARCH
START	*	*	left	SEARCH
SEARCH	0	0	left	SEARCH
SEARCH	1	0	left	SEARCH
SEARCH	*	*	no move	HALT

14.

current state	current cell contents	value to write	direction to move	new state to enter
START	*	*	left	FIRST
FIRST	*	*	right	HALT
FIRST	0	0	left	CARRY 0
FIRST	1	1	left	CARRY 1
CARRY 0	0	0	left	CARRY 0
CARRY 0	1	0	left	CARRY 1
CARRY 0	*	*	right	RETURN 0
CARRY 1	0	1	left	CARRY 0
CARRY 1	1	1	left	CARRY 1
CARRY 1	*	*	right	RETURN 1
RETURN 1	1	1	right	RETURN 1
RETURN 1	0	0	right	RETURN 1
RETURN 1	*	*	left	WRITE 1
RETURN 0	0	0	right	RETURN 0
RETURN 0	1	1	right	RETURN 0
RETURN 0	*	*	left	WRITE 0
WRITE 0	anything	0	right	HALT
WRITE 1	anything	1	right	HALT

15.

current state	current cell contents	value to write	direction to move	new state to enter
START	*	*	left	BEGIN
BEGIN	0	*	left	SAVE 0
BEGIN	1	*	left	SAVE 1
BEGIN	*	*	right	HALT
GO HOME	0	0	right	GO HOME
GO HOME	1	1	right	GO HOME
GO HOME	*	*	no move	HALT
SAVE 0	0	0	left	SAVE 0
SAVE 0	1	1	left	SAVE 0
SAVE 0	*	*	right	GET LEFT 0
SAVE 1	0	0	left	SAVE 1
SAVE 1	1	1	left	SAVE 1
SAVE 1	*	*	right	GET LEFT 1
GET LEFT 0	0	*	right	CARRY RIGHT 00
GET LEFT 0	1	*	right	CARRY RIGHT 01
GET LEFT 0	*	0	right	GO HOME
GET LEFT 1	0	*	right	CARRY RIGHT 10
GET LEFT 1	1	*	right	CARRY RIGHT 11
GET LEFT 1	*	1	right	GO HOME
CARRY RIGHT 00	0	0	right	CARRY RIGHT 00
CARRY RIGHT 00	1	1	right	CARRY RIGHT 00
CARRY RIGHT 00	*	0	left	GET RIGHT 0
CARRY RIGHT 01	0	0	right	CARRY RIGHT 01
CARRY RIGHT 01	1	1	right	CARRY RIGHT 01
CARRY RIGHT 01	*	1	left	GET RIGHT 0
CARRY RIGHT 10	0	0	right	CARRY RIGHT 10
CARRY RIGHT 10	1	1	right	CARRY RIGHT 10
CARRY RIGHT 10	*	0	left	GET RIGHT 1
CARRY RIGHT 11	0	0	right	CARRY RIGHT 11
CARRY RIGHT 11	1	1	right	CARRY RIGHT 11
CARRY RIGHT 11	*	1	left	GET RIGHT 1
GET RIGHT 0	0	*	left	CARRY LEFT 00
GET RIGHT 0	1	*	left	CARRY LEFT 01
GET RIGHT 0	*	0	right	GO HOME
GET RIGHT 1	0	*	left	CARRY LEFT 10
GET RIGHT 1	1	*	left	CARRY LEFT 11

```
  GET RIGHT 1   *         1      right     GO HOME
CARRY LEFT 00   0         0      left      CARRY LEFT 00
CARRY LEFT 00   1         1      left      CARRY LEFT 00
CARRY LEFT 00   *         0      right     GET LEFT 0
CARRY LEFT 01   0         0      left      CARRY LEFT 01
CARRY LEFT 01   1         1      left      CARRY LEFT 01
CARRY LEFT 01   *         0      right     GET LEFT 1
CARRY LEFT 10   0         0      left      CARRY LEFT 10
CARRY LEFT 10   1         1      left      CARRY LEFT 10
CARRY LEFT 10   *         1      right     GET LEFT 0
CARRY LEFT 11   0         0      left      CARRY LEFT 11
CARRY LEFT 11   1         1      left      CARRY LEFT 11
CARRY LEFT 11   *         1      right     GET LEFT 1
```

16. The term Church-Turing thesis today refers to the conjecture that the concept of a Turing machine (and all its equivalent concepts) captures the meaning of "to compute." In other words, it is the conjecture that Turing machines have the capability of solving any algorithmically solvable problem.

17. No. The program halts only when the initial value of X ends with the digit 0. But, since the ASCII representation for a semicolon is 00111011, the encoded representation of the program ends with the digit 1.

18. No. The program does not halt for any positive input value and thus would not halt if the input value were its own Gödel number.

19. Yes, the program terminates for all input values.

20. This is a paradox much like that generated by the assumption that the halting problem is solvable. If the first statement is assumed to be true, it must be false and vice versa.

21. This is a paradox. Whether one assumes that the cook does or does not cook for himself or herself, one must conclude the opposite.

22. If I were to ask you if you are a truth teller, would you answer "yes"? (A truth teller would answer this question by saying "yes," a liar would answer "no.")

23. Turing machines provide a definition of "computable" in that their computing power is believed to be as great as any algorithmic system.

24. The halting problem is an example of a problem that does not have an algorithmic solution and therefore cannot be solved by a computer program. Such problems establish a bound on the capabilities of computing machines.

25. If the answer was yes, you would find out by discovering a person with that birthday. If the answer was no, you would find out by exhausting all possibilities without finding a person with that birthday. In the case of testing positive integers, you could never exhaust the list and therefore would never obtain an answer if the ultimate answer was no. The difficulty here is that many students have trouble understanding that you would never know the answer was no.

26. Yes, both the sequential and binary search algorithms solve the problem in polynomial time.

27. The sieve of Eratosthenes is a well-known approach. The number of divisions required is a polynomial of the input; but, because the time required for each division depends on the length of the input's representation, the algorithm runs in exponential time.

28. No. For small inputs, an exponential algorithm might be faster. For example, the value of the exponential expression 2^x is less than the polynomial expression x^2 in the range between 2 and 4.

29. No. Even a polynomial expression becomes large for extremely large inputs.

30. Mary's solution is much better. Given a committee of size $2n$ (an even value), Charlie's solution would require generating $(2n)!/(n!n!)$ subcommittees, which is more than 2^n . Thus, Charlie's solution would require exponential time. On the other hand, Mary's solution involves little more than a sorting algorithm and can therefore be performed in polynomial time. Consequently, the problem itself is a polynomial problem.

31. This is not a polynomial time solution. (In fact, its space complexity is not satisfactory either.)

32. (The point here is to get students to realize how fast the time requirements of a problem outside of P can grow.) In the case of four numbers, 6,250,000 tickets would have to be purchased requiring 72 days and 8 hours. In the case of five numbers, the number of tickets required jumps to 312,500,000, which would require 3,616 days and 21 hours to purchase--almost ten years!

33. No, it is not deterministic. The value less than five is not specified. (This is essentially a random number generator. Note that in actual applications only pseudo-random number generators are used. If you ran such a generator again, it would produce the same set of numbers. This "algorithm," however, would not necessarily produce the same numbers if repeated.)

34. Yes, it is deterministic. The person following the directions never has to make a choice. At each step, that person merely does what he or she is told.

35. The nondeterminism is in the two select statements.

36. This is a nondeterministic algorithm that runs in polynomial time.

37. a and c.

38. A polynomial problem can be solved by a deterministic algorithm in polynomial time. A nondeterministic polynomial problem can be solved by a nondeterministic algorithm in polynomial time.

39. The problem of sorting a list is in both P and NP.

40. The solution with time complexity n^4 would be more efficient than the solution with time complexity 4^n for values of n greater than 4. For values of n between 0 and 4, the one with time complexity 4^n would be more efficient.

41. There would be 14! different paths that do not visit cities multiple times$14!/(10^6)$ = 63,154 seconds = 1052.5 minutes = 17.5 hours.

42. The merge sort would perform 4 name comparisons when sorting the list Alice, Bob, Carol, and David. The precise number performed when sorting the list Alice, Bob, Carol, David, and Elaine depends on the point at which the odd entry is considered.

43. The problem of searching for an entry in a list is in P. The traveling salesman problem falls in the questionable region. The problem of producing all possible subcommittees of a given committee is nonpolynomial. The halting problem is unsolvable.

44. One approach would be to try all combinations of values for x and y less than or equal to n. This would have complexity on the order of n^2. Another would be to try combinations of values less than or equal to the square root of n. This would have complexity on the order of n. There are still "better" solutions though. Here is one that is on the order of the square root of n.

```
SqRootN ← the largest integer no greater than n;
for x = 0 to SqRootN do
  (y ← the largest integer no greater than n - x^2;
   if (x^2 + y^2 = n) then record a solution)
```

45. 1723 = 257 + 771 + 391 + 304

46. Both involve considering a large number of cases in a systematic manner. In the Traveling Salesman Problem the cases are permutations of cities, in knapsack problems the cases are subsets of a given set of numbers. In both problems the number of cases to be considered grows rapidly as the size of the input increases.

47. $(n - 1) + (n - 2) + ... + 1$

48. The message 110 is the binary representation for 6. $6^5 = 7776$ and $7776 \pmod{91} = 41$, which in binary notation is 101001. Thus, the encrypted version of the message is 101001.

49. The message 111 is the binary representation for 7. $7^5 = 16807$ and $16807 \pmod{133} = 49$, which in binary notation is 110001. Thus, the encrypted version of the message is 110001.

50. $d = 23$. Since n = 77 is easily factored as 7×11, we know that $p = 7$ and $q = 11$. This allows us to determine that $k(p - 1)(q - 1) + 1$ is evenly divided by $e = 7$ when $k = 2$. The quotient 23 is the value of d.

51. $367 \times 293 = 107531$.

52. The integer n must be prime. To find the factors of a positive integer n, one needs to consider only values less than or equal to the square root of n.