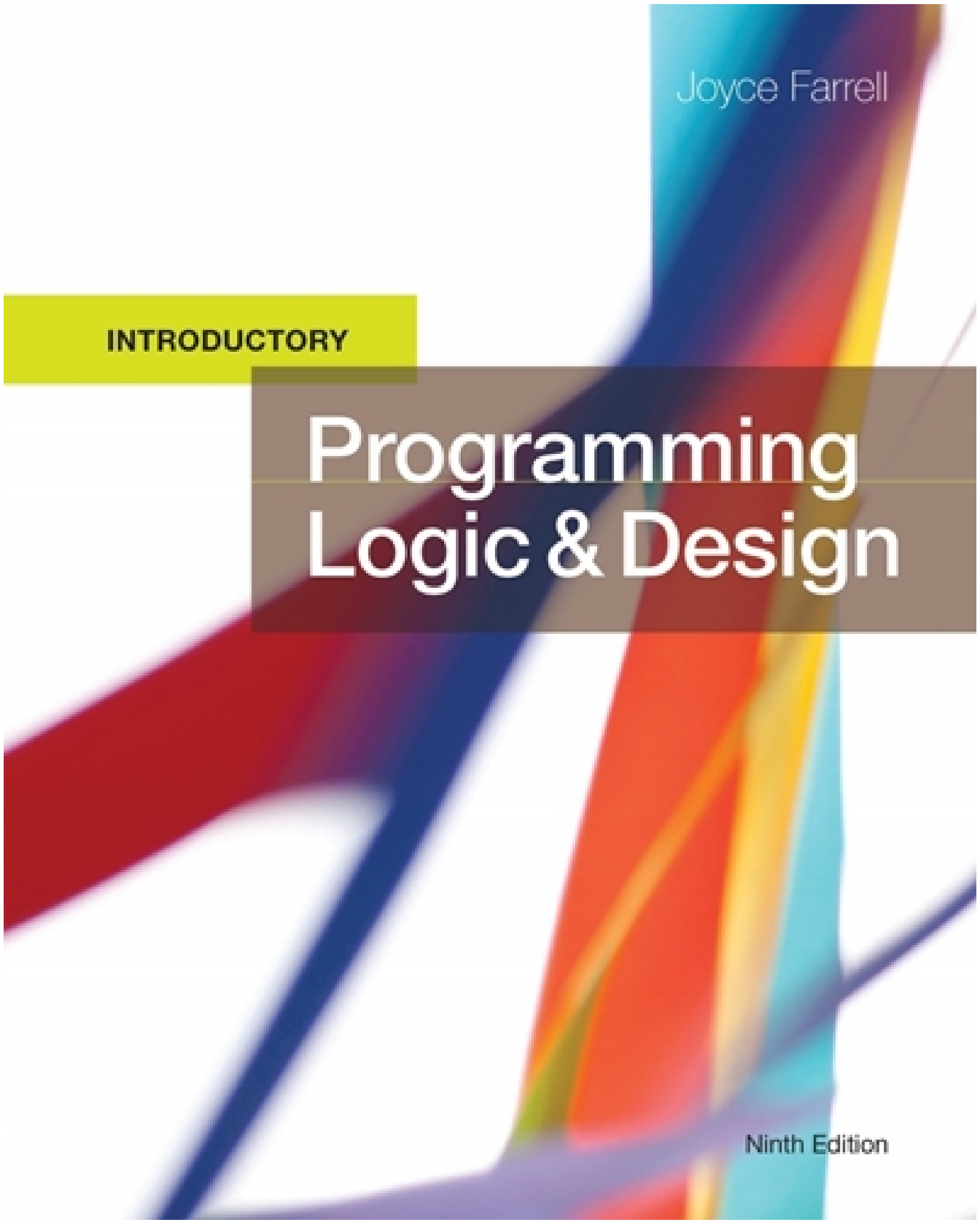

Joyce Farrell
INTRODUCTORY
Programming Logic & Design
Ninth Edition

Solutions

Programming Logic and Design, 9th Edition

Chapter 1

Review Questions

1. Computer programs also are known as____________.
   a. data
   b. hardware
   c. software
   d. information
2. The major computer operations include ____________.
   a. input, processing, and output
   b. hardware and software
   c. sequence and looping
   d. spreadsheets, word processing, and data communications
3. Visual Basic, C++, and Java are all examples of computer ____________.
   a. operating systems
   b. programming languages
   c. hardware
   d. machine languages
4. A programming language's rules are its ____________.
   a. syntax
   b. logic
   c. format
   d. options

5. The most important task of a compiler or interpreter is to ____________.
   a. create the rules for a programming language
   b. translate English statements into a language such as Java
   c. translate programming language statements into machine language
   d. execute machine language programs to perform useful tasks
6. Which of the following is temporary, internal storage?
   a. CPU
   b. hard disk
   c. keyboard
   d. memory
7. Which of the following pairs of steps in the programming process is in the correct order?
   a. code the program, plan the logic
   b. test the program, translate it into machine language
   c. put the program into production, understand the problem
   d. code the program, translate it into machine language
8. A programmer's most important task before planning the logic of a program is to ____________.
   a. decide which programming language to use
   b. code the problem
   c. train the users of the program
   d. understand the problem
9. The two most commonly used tools for planning a program's logic are _____.

a. ASCII and EBCDIC

b. Java and Visual Basic

c. flowcharts and pseudocode

d. word processors and spreadsheets

10. Writing a program in a language such as C++ or Java is known as _______ the program.

a. translating

b. coding

c. interpreting

d. compiling

11. An English-like programming language such as Java or Visual Basic is a ________ programming language.

a. machine-level

b. low-level

c. high-level

d. binary-level

12. Which of the following is an example of a syntax error?

a. producing output before accepting input

b. subtracting when you meant to add

c. misspelling a programming language word

d. all of the above

13. Which of the following is an example of a logical error?

a. performing arithmetic with a value before inputting it

b. accepting two input values when a program requires only one

c. dividing by 3 when you meant to divide by 30

d. all of the above

14. The parallelogram is the flowchart symbol representing ____________.

a. input

b. output

c. either a or b

d. none of the above

15. In a flowchart, a rectangle represents ____________.

a. input

b. a sentinel

c. a question

d. processing

16. In flowcharts, the decision symbol is a ____________.

a. parallelogram

b. rectangle

c. lozenge

d. diamond

17. The term *eof* represents ____________.

a. a standard input device

b. a generic sentinel value

c. a condition in which no more memory is available for storage

d. the logical flow in a program

18. When you use an IDE instead of a simple text editor to develop a program, ____________.

    a. the logic is more complicated

    b. the logic is simpler

    c. the syntax is different

    d. some help is provided

19. When you write a program that will run in a GUI environment as opposed to a command-line environment, ____________.

    a. the logic is very different

    b. some syntax is different

    c. you do not need to plan the logic

    d. users are more confused

20. As compared to procedural programming, with object-oriented programming, ____________.

    a. the programmer's focus differs

    b. you cannot use some languages, such as Java

    c. you do not accept input

    d. you do not code calculations; they are created automatically

Programming Exercises

1. Match the definition with the appropriate term.

1.	Computer system devices	a.	compiler
2.	Another word for *program*	b.	syntax
3.	Language rules	c.	logic
4.	Order of instructions	d.	hardware
5.	Language translator	e.	software

Answer:

1.	Computer system equipment	→	d.	hardware
2.	Another word for *program*	→	e.	software
3.	Language rules	→	b.	syntax
4.	Order of instructions	→	c.	logic
5.	Language translator	→	a.	compiler

2. In your own words, describe the steps to writing a computer program.

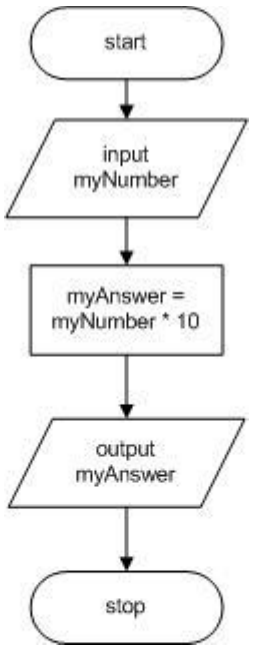
Answer:

The programmer must understand the problem that the user is trying to solve. Next, the programmer plans the logic, often using a flowchart or pseudocode. Then, the program is coded in a language, such as Visual Basic or Java, and translated to machine language using a compiler or interpreter. Finally, the program is tested and then put into production and maintained over the ensuing months or years.

3. Draw a flowchart or write pseudocode to represent the logic of a program that allows the user to enter a value. The program multiplies the value by 10 and outputs the result.

Answer:

Flowchart



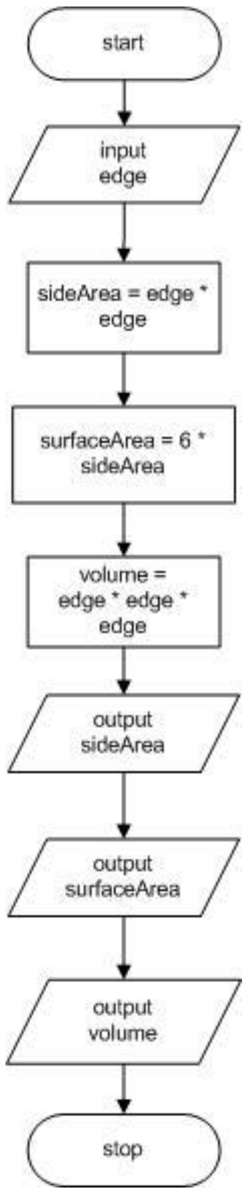
Pseudocode

```
start
   input myNumber
   myAnswer = myNumber * 10
   output myAnswer
stop
```

4. Draw a flowchart or write pseudocode to represent the logic of a program that allows the user to enter a value for one edge of a cube. The program calculates the surface area of one side of the cube, the surface area of the cube, and its volume. The program outputs all the results.

Answer:

Flowchart



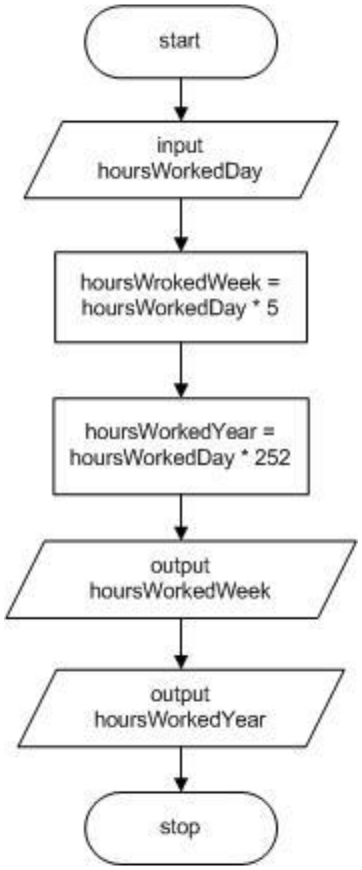
Pseudocode

```
start
   input edge
   sideArea = edge * edge
   surfaceArea = 6 * sideArea
   volume = edge * edge * edge
   output sideArea
   output surfaceArea
   output volume
stop
```

5. Draw a flowchart or write pseudocode to represent the logic of a program that allows the user to enter a value for hours worked in a day. The program calculates the hours worked in a five-day week and the hours worked in a 252-day work year. The program outputs all the results.

Answer:

Flowchart


start
input hoursWorkedDay
hoursWrokedWeek = hoursWorkedDay * 5
hoursWorkedYear = hoursWorkedDay * 252
output hoursWorkedWeek
output hoursWorkedYear
stop

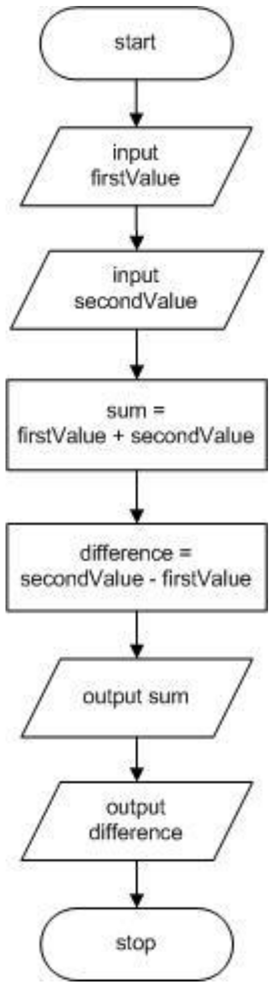
Pseudocode

```
start
   input hoursWorkedDay
   hoursWorkedWeek = hoursWorkedDay * 5
   hoursWorkedYear = hoursWorkedDay * 252
   output hoursWorkedWeek
   output hoursWorkedYear
stop
```

6. Draw a flowchart or write pseudocode to represent the logic of a program that allows the user to enter two values. The program outputs the sum of and the difference between the two values.

Answer:

Flowchart



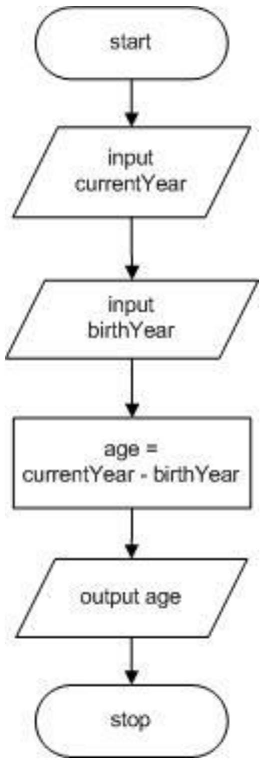
Pseudocode

```
start
   input firstValue
   input secondValue
   sum = firstValue + secondValue
   difference = secondValue - firstValue
   output sum
   output difference
stop
```

7. Draw a flowchart or write pseudocode to represent the logic of a program that allows the user to enter values for the current year and the user's birth year. The program outputs the age of the user this year.

Answer:

Flowchart



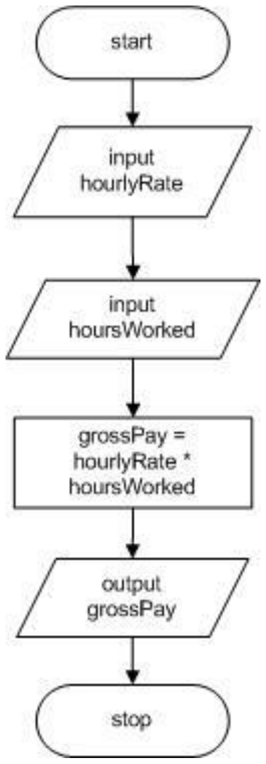
Pseudocode

```
start
   input currentYear
   input birthYear
   age = currentYear - birthYear
   output age
stop
```

8. a. Draw a flowchart or write pseudocode to represent the logic of a program that allows the user to enter an hourly pay rate and hours worked. The program outputs the user's gross pay.

Answer:

Flowchart



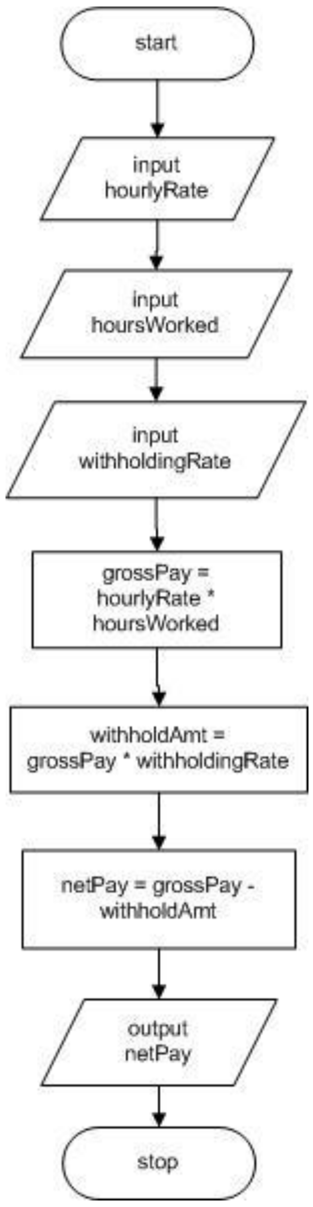
Pseudocode

```
start
   input hourlyRate
   input hoursWorked
   grossPay = hourlyRate * hoursWorked
   output grossPay
stop
```

b. Modify the program that computes gross pay to allow the user to enter the withholding tax rate. The program outputs the net pay after taxes have been withheld.

Answer:

Flowchart



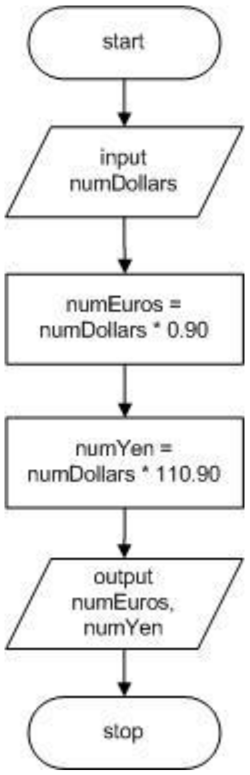
Pseudocode

```
start
   input hourlyRate
   input hoursWorked
   input withholdingRate
   grossPay = hourlyRate * hoursWorked
   withholdAmt = grossPay * withholdingRate
   netPay = grossPay - withholdAmt
   output netPay
stop
```

9. Research current rates of monetary exchange. Draw a flowchart or write pseudocode to represent the logic of a program that allows the user to enter a number of dollars and convert it to Euros and Japanese yen.

Answer:

Flowchart



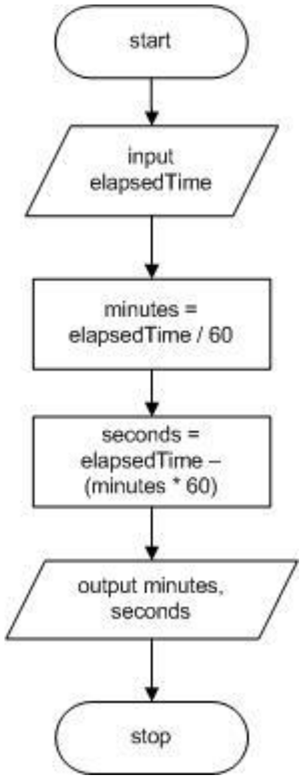
Pseudocode

```
start
   input numDollars
   numEuros = numDollars * 0.90
   numYen = numDollars * 110.90
   output numEuros, numYen
stop

(Please note that these exchange rates are valid as of the
time of this writing)
```

10. A mobile phone app allows a user to press a button that starts a timer that counts seconds. When the user presses the button again, the timer stops. Draw a flowchart or write pseudocode that accepts the elapsed time in seconds and displays the value in minutes and remaining seconds. For example, if the elapsed time was 130 seconds, the output would be 2 minutes and 10 seconds.

Answer: (Please note this solution assumes *minutes* is an integer and has been truncated.)

Flowchart



Pseudocode

```
start
   input elapsedTime
   minutes = elapsedTime / 60
   seconds = elapsedTime – (minutes * 60)
   output minutes, seconds
stop
```

Performing Maintenance

1. In this chapter you learned that some of the tasks assigned to new programmers frequently involve maintenance—making changes to existing programs because of new requirements. A file named MAINTENANCE01-01.txt is included with your downloadable student files. Assume that this program is a working program in your organization and that it needs modifications as described in the comments (lines that begin with two slashes) at the beginning of the file. Your job is to alter the program to meet the new specifications.

Answer:

```
// This program accepts a user's monthly pay
// and rent, utilities, and grocery bills
// and displays the amount available for discretionary spending
// (which might be negative)
// Modify the program to output the pay and the total bills
// as well as the remaining discretionary amount

start
   input pay
   input rent
   input utilities
   input groceries
   bills = rent + utilities + groceries
   discretionary = pay - bills
   output pay
   output bills
   output discretionary
stop
```

Find the Bugs

1. Your downloadable files for Chapter 1 include DEBUG01-01.txt, DEBUG01-02.txt, and DEBUG01-03.txt. Each file starts with some comments (lines that begin with two slashes) that describe the program. Examine the pseudocode that follows the introductory comments, then find and correct all the bugs.

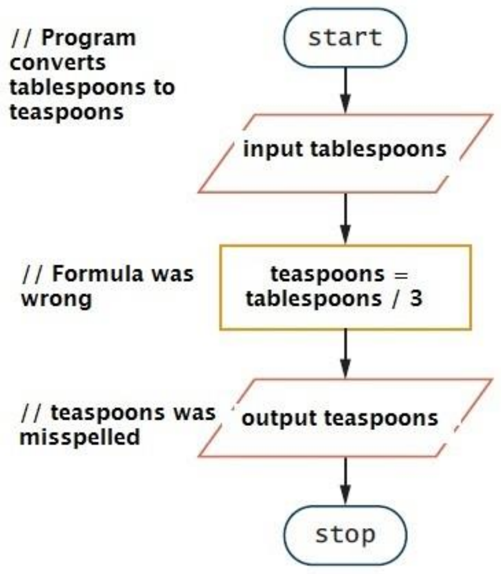
Answer:

```
DEBUG01-01
// This pseudocode is intended to describe
// computing the price of an item on sale for 10% off
start
   input origPrice
   discount = origPrice * 0.10
   finalPrice = origPrice - discount
   output finalPrice
stop

DEBUG01-02
// This pseudocode is intended to compute the number
// of miles per gallon you get with your automobile.
start
   input milesTraveled
   input gallonsOfGasUsed
   milesPerGallon = milesTraveled / gallonsOfGasUsed
```

```
    // milesPerGallon is computed using division
  output milesPerGallon
    // miles is misspelled, and the P in milesPerGallon should be uppercase
stop
  // Program should end with stop

DEBUG01-03
// This pseudocode is intended to describe
// computing the per day cost of your rent
// in a 30-day month
start
  input rent
  costPerDay = rent / 30
    // Comment indicates 30-day month
  output costPerDay
    // output should be costPerDay
stop
```

2. Your downloadable files for Chapter 1 include a file named DEBUG01-04.jpg that contains a flowchart that contains syntax and/or logical errors. Examine the flowchart and then find and correct all the bugs.

Answer:


// Program converts tablespoons to teaspoons
start
input tablespoons
// Formula was wrong
teaspoons = tablespoons / 3
// teaspoons was misspelled
output teaspoons
stop

Game Zone

1. Create the logic for a Mad Lib program that accepts five words from input, then creates and displays a short story or nursery rhyme that uses them.

Answer:

Students' answers will vary. But a simple solution could be:

```
start
   input word1
   input word2
   input word3
   input word4
   input word5
   output "Jack and Jill went up the ", **word1**,
          " to fetch a pail of ", **word2**,
          ". "Jack ", **word3**, " down and broke his ", **word4**,
          " and Jill came ", **word5**, " after."
stop
```

Chapter 1

An Overview of Computers and Programming

A Guide to this Instructor's Manual:

We have designed this Instructor's Manual to supplement and enhance your teaching experience through classroom activities and a cohesive chapter summary.

This document is organized chronologically, using the same headings that you see in the textbook. Under the headings you will find: lecture notes that summarize the section, Teacher Tips, Classroom Activities, and Lab Activities. Pay special attention to teaching tips and activities geared towards quizzing your students and enhancing their critical thinking skills.

In addition to this Instructor's Manual, our Instructor's Resources also contain PowerPoint Presentations, Test Banks, and other supplements to aid in your teaching experience.

At a Glance

Instructor's Manual Table of Contents

Note to Instructors: Retired exercises from the previous edition text are available in the Solutions to Exercises files on the Web.

Lecture Notes

Overview

Chapter 1 provides an introduction to the basic steps involved in the programming process. Students will learn about hardware and software, source and object code, logic, and flowcharting symbols, and become familiar with using sentinel values. Finally, students will learn about different types of programming and user environments.

Chapter Objectives

In this chapter, your students will learn about:

- Computer systems
- Simple program logic
- The steps involved in the program development cycle
- Pseudocode statements and flowchart symbols
- Using a sentinel value to end a program
- Programming and user environments
- The evolution of programming models

Teaching Tips

Understanding Computer Systems

1. Review the major components of a **computer system**:
   a. **Hardware**
   b. **Software**
      i. **Application software**
      ii. **System software**

2. Review the major operations performed in most programs:
   a. **Input** – **data items**
   b. **Processing –** performed by the **central processing unit (CPU)**
   c. **Output –** after data items have been processed it is called **information**
      i. Can send to a printer or monitor
      ii. Can save output on storage devices such as hard drive, flash drive or a **cloud** (**Internet**) based device

3. Discuss **programming languages**, **program code** and **coding the program.**

4. Discuss the difference between **syntax errors** and **logical errors**.

5. Illustrate the syntax for different languages, using Figure 1-1.

6. Discuss the difference between **volatile** and **nonvolatile** memory.

7. Discuss **machine language (binary language)**, **source code** and **object code,** and **compiler** and **interpreter**.

8. Discuss **scripting languages**.

Understanding Simple Program Logic

1. Review the example of the cake recipe on page 5.

2. Review the simple example program to double a number on page 6 and include a definition of the term **variable**.

Understanding the Program Development Cycle

1. List the seven steps of the **program development cycle** (Figure 1-2) and discuss each one in detail.

Teaching Tip	Students want to get going as soon as possible and thing that they should start coding the program right away. Stress the need for understanding the problem and planning the logic. It may be a good idea to force them to do flowcharts or pseudocode before they are allowed to turn the computer on.

Understanding the Problem

1. Point out that the failure to understand the problem to be solved is the major cause of software project failures. The most important sources for information about the problem are the **users** or **end users** and **documentation**.

Teaching Tip	A programmer's ability to understand a user's needs by observing the user's job functions and questioning the user is a key skill. In many organizations, the step of understanding the problem is performed by application analysts, not programmers. In other organizations, programmers perform both analysis and programming, and may hold the job title of programmer/analyst.

Planning the Logic

1. Explain that pseudocode allows a description of the program logic to be created without worrying about the details or syntax of a particular programming language.

2. Define the term **algorithm**.

Teaching Tip	The term **desk-checking** is also commonly used to describe the process of checking code for syntactical and logical correctness before compiling. This process was essential in earlier times when computing power was limited and scheduled, and a programmer may have been able to complete only one compilation a day.

Coding the Program

1. Stress the fact that the actual coding of the program does not occur until after the logic has been completely described and checked.

Using Software to Translate the Program into Machine Language

1. Describe the repetitive process of coding, compiling, and fixing syntax errors that must occur until the program is free of syntax errors. This is shown in Figure 1-2.

2. Explain the difference between a **high-level programming language** and a **low-level programming language.**

Testing the Program

1. Point out the importance of selecting appropriate test data to ensure that all of the code is tested. Consider boundary conditions such as equal, minimum, and maximum values.

Putting the Program into Production

1. Explain that after testing and documentation are completed, a program is put into production.

2. Define the term **conversion**, and explain that it may take months or years to complete. Recall the Y2K (Year 2000) issue.

Maintaining the Program

1. Describe the process of program **maintenance**.

Teaching Tip	Software testing, or software quality assurance, can be a career of its own. Today, automated testing tools are available to attempt a thorough review of all of the code in a program. The following article on how to choose test automation software can serve as a starting point for a discussion of test automation issues: http://www.softwaretestinghelp.com/choosing-automation-tool-for-your-organization/

Teaching Tip	Point out that an error-free test of the program does not guarantee that all errors have been corrected. The most one can say is that "all errors that have been identified have now been corrected." Stress the fact that students should try to "break" their program by entering incorrect values.

Quick Quiz 1

1. What are the two major components of a computer system?
   Answer: hardware and software

2. What are the three basic operations of a computer?
   Answer: input, processing, and output

3. What is the most important phase in creating a program to solve a problem?
   Answer: understanding the problem

4. A computer program must be free of ____ errors before you can execute it.
   Answer: syntax

Using Pseudocode Statements and Flowchart Symbols

1. Review the definitions of **flowchart** and **pseudocode**.

Writing Pseudocode

1. Walk through the example of pseudocode given on page 15 and discuss the flexibility of pseudocode. Note that most programmers use `start` or `begin` to begin a program and `end` or `stop` at the end of the program.

Teaching Tip	Explain that there are no “rules” for writing pseudocode. Developers can use any combination of English, slang, code, whatever they want to explain the algorithms.

Drawing Flowcharts

1. Describe each flowchart symbol and its purpose, using Figures 1-4 through 1-6 for illustration.
2. Describe **flowlines** and the depiction of direction with arrowheads on flowlines.
3. Describe the use of **terminal symbols**.
4. Use Figure 1-7 to show the flowchart depiction of each line of pseudocode. This will help the students understand how much detail should appear in pseudocode.

Repeating Instructions

1. Discuss the need for repetitive processing of certain sets of program statements. This is achieved using a **loop**. Stress the importance of avoiding an **infinite loop**.
2. Describe how the use of directed flowlines indicates repetition in a flowchart using Figure 1-9.

Teaching Tip	Explain how the directed flowlines in Figure 1-9 show the code that forms the loop. Point out the necessity for a decision somewhere inside the loop to avoid an infinite loop.

Using a Sentinel Value to End a Program

1. Describe an infinite loop and methods of stopping a loop during processing.
2. Describe **making a decision** in a program, and introduce the flowchart **decision symbol**. Ensure that students understand the need to document both the Yes and No paths in a decision symbol.
3. Introduce the concept of a **sentinel value** or **dummy value** to be used as a signal to stop the loop processing. Point out that if the loop is used to read files, the end-of-file (`eof`) marker is often used for this purpose.
4. Use Figures 1-10 and 1-11 to show the use of a sentinel value in the decision point of the loop.

Quick Quiz 2

1. A symbol shaped like a(n) ____________ represents a decision in a flowchart.
   Answer: diamond

2. A loop that runs forever is called a(n) _______________ loop.
   Answer: infinite

3. What purpose does a sentinel value have?
   Answer: A sentinel value is used as a signal value to indicate that processing should stop.

4. A code stored in a file that marks the end of the data is called a(n) ____ marker.
   Answer: `eof` or end-of-file

Understanding Programming and User Environments

1. Note that there are many different options when it comes to programming and user environments. These will be covered in the following sections.

Understanding Programming Environments

1. Describe the two main options for programming environments:
   a. Plain **text editor** (Figure 1-12)
   b. Text editor that is part of an **integrated development environment** (**IDE**) (Figure 1-13)

2. Review the features of an IDE listed on page 24.

Understanding User Environments

1. Introduce the two main ways a user may interact with a program:
   a. **Command line** (Figure 1-14)
   b. **Graphical user interface** (**GUI**) (Figure 1-15)

Understanding the Evolution of Programming Models

1. Describe early programming, which required the use of direct memory addresses. Point out the high probability of errors due to the fact that the code did not resemble a natural language.

Teaching Tip	Note that low-level languages also required a much deeper understanding of how the computer worked internally, as there was close to a one-to-one relationship between the instruction and the machine action it caused. It's also a good idea to review the binary numbering system and how it relates to memory locations.

2. Point out that higher-level languages provide more than just ease of understanding the language. Many instructions in high-level languages perform several low-level operations automatically.

3. Point out that programming design in early days tended to be monolithic, while newer programming design builds reusable modules of code.

4. Describe and contrast the two major programming techniques: **procedural programming** and **object-oriented programming**.

Teaching Tip	Many students think that object-oriented techniques are unique in that they employ reusable code. Point out that the more important difference is the focus on objects as a programming technique.

Quick Quiz 3

1. (True/False) Many approaches can be used to write and execute a computer program.
   Answer: True

2. A(n) ____ is a program that you use to create simple text files.
   Answer: text editor

3. The ____ is a location on your computer screen where you type text entries to communicate with the computer's operating system.
   Answer: command line

4. ____ programming focuses on objects, or "things," and describes their features (or attributes) and their behaviors.
   Answer: Object-oriented

Class Discussion Topics

1. Discuss types of input and output devices other than the ones described in this text. You may wish to describe some of the devices used in the early days of computing (e.g., card

punches and card readers). Can you name any devices that are used for both input and output (e.g., touch-sensitive monitors)?

2. Software testing occurs near the end of the programming process. Because of this, if a project falls behind schedule, testing time is often reduced. Discuss the possible impacts of less-than-thorough testing. Topics might include company reputation, customer dissatisfaction, the company's ability to sell new or updated products, etc. Discuss which is worse: releasing a product late or releasing it on time without fully testing it.

3. Which is the better tool for learning programming—flowcharts or pseudocode? Cite any educational research you can find.

   Answers will vary. Many educators like the visual feedback flowcharts provide.

4. What is the image of the computer programmer in popular culture? Is the image different in books than in TV shows and movies? Would you like that image for yourself

   The programmer is often seen as an anti-social nerd with tape around his glasses, a pocket protector, and the inability to form social relationships. In some movies however, the computer-savvy programmer can save the day. High school aged programmers, in particular, are often portrayed as Robin Hood-type rebels. Movie plots tend to concentrate on "good" programmers versus "evil" programmers. Whether good or evil, the programmer is virtually always seen as smart.

Additional Projects

1. Do some research about early programming using low-level machine languages. Write a short paper about what these languages looked like and what types of problems were encountered when using them.

2. Do some research about object-oriented programming techniques. Write a short paper describing the advantages and disadvantages of using object-oriented techniques.

3. Separate students into "programmers" and "robots." Send the robots out of the room and instruct the programmers to write the steps necessary to perform simple tasks around the classroom (turn off the lights, throw a piece of paper in the garbage, shake the professor's hand, etc.). Give the programmers about 15 minutes to write their steps. Explain to the robots that when they re-enter the classroom, they are not to assume anything—just follow the instructions.

4. On the board, draw a closet with six shelves. Ask the students to draw a box on the third shelf. Did they draw the box on the third shelf from the bottom or the third shelf from the top? Explain how important it is to be very specific in your instructions.

Additional Resources

1. Short history of the computer:
   http://www.softlord.com/comp/

2. Article on the software development process:
   http://en.wikipedia.org/wiki/Software_development_process

3. Article on pseudocode:
   http://en.wikipedia.org/wiki/Pseudocode

4. Tutorial on flowcharting:
   http://files.acdsystems.com/english/support/tutorials/canvas11/CreatingFlowcharts_tutorial.pdf

5. Article on IDEs:
   http://en.wikipedia.org/wiki/Integrated_development_environment

Key Terms

- **Algorithm** – the sequence of steps necessary to solve any problem.
- **Application software** – comprises all the programs you apply to a task.
- An **app** is a piece of application software; the term is frequently used for applications on mobile devices.
- **Binary language** – is represented using a series of 0s and 1s.
- **Central processing unit** (**CPU**) – the hardware component that processes data.
- The **cloud** refers to remote computers accessed through the Internet.
- **Coding the program** – the act of writing programming language instructions.
- **Command line** – a location on your computer screen where you type text entries to communicate with the computer's operating system.
- **Compiler/interpreter** – translates a high-level language into machine language and tells you if you have used a programming language incorrectly.
- **Computer memory** – the temporary, internal storage within a computer.
- **Computer system** – a combination of all the components required to process and store data using a computer.
- **Conversion** – the entire set of actions an organization must take to switch over to using a new program or set of programs.
- **Data items** – include all the text, numbers, and other information processed by a computer.
- **Debugging** – the process of finding and correcting program errors.

- **Decision symbol –** shaped like a diamond and used to represent decisions in flowcharts.
- **Desk-checking –** the process of walking through a program's logic on paper.
- **Documentation –** consists of all the supporting paperwork for a program.
- **Dummy value –** a preselected value that stops the execution of a program.
- **`eof`** – means "end of file."
- **Flowchart –** a pictorial representation of the logical steps it takes to solve a problem.
- **Flowlines –** arrows used to connect the steps in a flowchart.
- **Graphical user interface** (**GUI**) – allows users to interact with a program in a graphical environment.
- **Hardware –** the equipment or physical devices that are associated with a computer.
- **High-level programming language –** supports English-like syntax.
- **Infinite loop –** occurs when repeating logic never ends.
- **Information –** data that has been processed.
- **Input –** describes the entry of data items into computer memory using hardware devices such as keyboards and mice.
- **Input symbol –** indicates an input operation and is represented by a parallelogram in flowcharts.
- **Input/output symbol** or **I/O symbol –** represented by a parallelogram in flowcharts.
- **Integrated development environment** (**IDE**) – a software package that provides an editor, compiler, and other programming tools.
- **Logic** of a computer program – when you give instructions to the computer in a specific sequence, without omitting any instructions or adding extraneous instructions.
- **Logical errors –** when instructions are performed in the wrong order, too many times, or not at all.
- **Loop –** a repetition of a series of steps.
- **Low-level machine language –** made up of 1s and 0s that the computer understands; also called machine language.
- **Machine language –** a computer's on/off circuitry language.
- **Maintenance –** consists of all the improvements and corrections made to a program after it is in production.
- **Making a decision –** the act of testing a value.
- **Microsoft Visual Studio IDE –** a software package that contains useful tools for creating programs in Visual Basic, C++, and C#.
- **Nonvolatile –** describes storage whose contents are retained when power is lost.
- **Object code –** translated machine language.
- **Object-oriented programming –** a programming model that focuses on objects, or "things," and describes their features (or attributes) and their behaviors.
- **Output –** describes the operation of retrieving information from memory and sending it to a device, such as a monitor or printer, so people can view, interpret, and use the results.
- **Output symbol –** indicates an output operation and is represented by a parallelogram in flowcharts.
- **Procedural programming –** a programming model that focuses on the actions that are carried out.
- **Processing** data items – may involve organizing them, checking them for accuracy, or performing calculations with them.

- **Processing symbol –** indicates a processing operation and is represented by a rectangle in flowcharts.
- **Program code** – the set of instructions a programmer writes in a programming language.
- **Program development cycle** – A checklist of at least seven steps to ensure the programmer has thought out, coded, and tested the program correctly.
- **Programming** – the act of developing and writing programs.
- **Programming language** – includes languages such as Visual Basic, C#, C++, Java; used to write programs.
- **Programs** – the instruction sets written by programmers.
- **Pseudocode** – an English-like representation of the logical steps it takes to solve a problem.
- **Random access memory** (**RAM**) – temporary, internal computer storage.
- **Runs/executes** – carrying out a program's instructions.
- **Scripting languages** (also called **scripting programming languages** or **script languages**) – such as Python, Lua, Perl, and PHP; used to write programs that are typed directly from a keyboard and stored as text rather than as binary executable files.
- **Sentinel value** – a preselected value that stops the execution of a program.
- **Software** – the computer instructions that tell the hardware what to do.
- **Source code** – the statements a programmer writes in a programming language.
- **Storage devices** – types of hardware equipment, such as disks or flash media, that hold information for later retrieval.
- **Syntax** – grammar rules of a language.
- **Syntax errors** – errors in language or grammar.
- **System software** – comprises the programs that you use to manage your computer.
- **Terminal symbols** (or start/stop symbols) – shaped like an oval and used at each end of a flowchart.
- **Text editor** – a program that you use to create simple text files; similar to a word processor, but without as many features.
- **Users** (or **end users**) – people who benefit from using computer programs.
- **Variable** – a named memory location whose value can vary.
- **Volatile –** describes storage whose contents are lost when power is lost.