REVIEW QUESTIONS

1. The two major components of any computer system are its _____.

a. input and output

b. data and programs

c. hardware and software

d. memory and disk drives

2. The major computer operations include _____.

a. hardware and software

b. input, processing, and output

c. sequence and looping

d. spreadsheets, word processing, and data communications

3. Another term meaning "computer instructions" is _____.

a. hardware

b. software

c. queries

d. data

4. Visual Basic, C++, and Java are all examples of computer _____.

a. operating systems

b. hardware

c. machine languages

d. programming languages

5. A programming language's rules are its _____.

a. syntax

b. logic

c. format

d. options

6. 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

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. The two most commonly used tools for planning a program's logic are _____.

a. flowcharts and pseudocode

b. ASCII and EBCDIC

c. Java and Visual Basic

d. word processors and spreadsheets

9. The most important task a programmer must do before planning the logic to a program is _____.

a. decide which programming language to use

b. code the problem

c. train the users of the program

d. understand the problem

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. A compiler would find all of the following programming errors *except* _____.

a. the misspelled word “prrint” in a language that includes the word “print”

b. the use of an “X” for multiplication in a language that requires an asterisk

c. `newBalanceDue` calculated by adding `customerPayment` to `oldBalanceDue` instead of subtracting it

d. an arithmetic statement written as `regularSales + discountedSales = totalSales`

12. In a flowchart, a terminal symbol looks most like a _____.

a. lozenge

b. circle

c. rectangle

d. parallelogram

13. The parallelogram is the flowchart symbol representing _____.

a. input

b. output

c. both a and b

d. none of the above

14. Which of the following is not a legal variable name in any programming language?

a. `semester grade`

b. `fall2011_grade`

c. `GradeInCIS100`

d. `MY_GRADE`

15. In flowcharts, the decision symbol is a _____.

a. parallelogram

b. rectangle

c. lozenge

d. diamond

Just Enough Programming Logic and Design, 1st Edition

Chapter 1

Exercises

1. Match the definition with the appropriate term.

i.	Computer system equipment	a.	compiler
ii.	Another word for programs	b.	syntax
iii.	Language rules	c.	logic
iv.	Order of instructions	d.	hardware
v.	Language translator	e.	software

Answer:

i.	Computer system equipment	→	d.	hardware
ii.	Another word for programs	→	e.	software
iii.	Language rules	→	b.	syntax
iv.	Order of instructions	→	c.	logic
v.	Language translator	→	a.	compiler

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

Answer:
The student's answer should describe the seven steps in the development process:

1. Understanding the problem
2. Planning the logic
3. Coding the program
4. Using software (a compiler or interpreter) to translate the program into machine language
5. Testing the program
6. Putting the program into production
7. Maintaining the program

3. Consider a student file that contains the following data:

Last Name	First Name	Major	Grade Point Average
Andrews	David	Psychology	3.4
Brown	Chris	Computer Science	4.0
Brogan	Lindsey	Biology	3.8
Carson	Kelly	Computer Science	2.8
Eisfelder	Katie	Mathematics	3.5
Faris	Natalie	Biology	2.8
Fredricks	Zachary	Psychology	2.0
Gonzales	Eduardo	Biology	3.1

Would this set of data be suitable and sufficient to use to test each of the following programs? Explain why or why not.

a. a program that displays a list of Psychology majors

Answer:
Yes, the program can select and display the two records whose major field contains "Psychology".

b. a program that displays a list of Art majors

Answer:
No, there are no records whose major field contains "Art". A test would not determine if this field would process correctly.

c. a program that displays a list of students on academic probation—those with a grade point average under 2.0

Answer:
No, there are no records whose grade point average field contains a value under 2.0. A test would not determine if this field would process correctly.

d. a program that displays a list of students on the dean's list

Answer:
No, we don't know what value grade point average must be to qualify for the dean's list, so this condition cannot be tested.

e. a program that displays a list of students from Wisconsin

Answer:
No, there is no state field, so the program would have no way of knowing what state each student is from.

f. a program that displays a list of female students

Answer:
No, there would be no way the program could determine the gender of the student based on the data in the record. People could look at the current data and guess if a student was male or female based on their name, but they might be wrong. (For example, "Chris" might be short for either Christopher or Christine.) If a gender field contained an "F" or "M", then a program could select and display students based on gender.

4. Suggest a good set of test data to use for a program that gives an employee a $50 bonus check if the employee has produced more than 1,000 items in a week. For example, one record might include the following:

Last name	First name	Items produced this week
Foster	Samantha	1,315

Answer:

Answers may vary, but included in the record layout should be a field for the number of items produced and a field (or fields) for the employee name (in order to differentiate each record). There should be at least one record where the number of items produced is less than 1000, is 1000, and is greater than 1000. It might also be a good idea to include a record with zero items produced for the week. For example:

Last name	First name	Items produced this week
Brown	John	1,000
Walters	Edna	999
Porter	Kelly	1,001
Davis	Edgar	0

5. Suggest a good set of test data for a program that computes gross paychecks (that is, before any taxes or other deductions) based on hours worked and rate of pay. The program computes gross as hours times rate, unless hours are over 40. If so, the program computes gross as regular rate of pay for 40 hours, plus one and a half times the rate of pay for the hours over 40.

Answer:

Answers may vary, but included in the record layout should be a field for the hours worked and another for the rate of pay. There should be at least one record where the hours worked is less than 40, one where they are 40, and one where they are greater than 40. It might also be a good idea to include a record with zero hours worked. There should also be a way to distinguish one record from another (for example, by employee name or employee number). For example:

Employee number	Hours worked	Pay rate
3347	38	12.55
6299	41	13.50
7218	40	23.00
5030	0	10.25

6. Suggest a good set of test data for a program that is intended to output a student's grade point average based on letter grades (A, B, C, D, or F) in five courses.

Answer:

Answers may vary, but included in the record layout should be the five course grades that will be submitted for averaging. Answers may vary depending on whether the

student will input numeric scores (A = 4, B = 3, C=2, etc.) or input letter grades, with the assumption that the program will convert the letter to a numeric score. There should be a variety of grades, and every grade should be used at least once. There should also be a way to distinguish one record from another (for example, by student last name or student number). For example:

Student number	Course 1 grade	Course 2 grade	Course 3 grade	Course 4 grade	Course 5 grade
00347	A	A	A	A	A
12908	F	F	F	F	F
40981	A	B	C	D	F
66085	B	C	A	C	C

7. Suggest a good set of test data for a program for an automobile insurance company that wants to increase its premiums by $50 per month for every ticket a driver receives in a three-year period.

Answer:

Answers may vary, but included in the record layout should be the insured's premium amount and the number of tickets received in the given three year period. The number of tickets for the given three year period can be given as one field (the total for the entire period), or the student may use three separate fields indicating the number of tickets received in each of the three years. The sample data set should include at least one record where the total number of tickets is 0, one where it is one, and one record where the number of tickets is more than one. There should also be a way to distinguish one record from another (for example, by insured name or policy number). For example:

Insured name	**Premium**	**Total tickets in 3 year period**
Crandell	165.99	3
Byer	237.50	1
Harris	89.75	4
Wu	150.00	0

8. Which of the following names seem like good variable names to you? If a name doesn't seem like a good variable name, explain why not.

Answer:

Answers will vary. A possible solution:

a. `c` – Valid, but probably too short to be descriptive
b. `cost` – Good
c. `costAmount` – Good, but redundant
d. `cost amount` – Invalid, spaces aren't allowed
e. `cstofdngbsns` – Valid, but difficult to read

f. `cost2011` – Good, but if 2011 represents anything other than the year it could be confusing

g. `costOfDoingBusinessThisFiscalYear` – Valid, but long and awkward

9. If `myAge` and `yourRate` are numeric variables, and `departmentName` is a string variable, which of the following statements are valid assignments? If a statement is not valid, explain why not.

Answer:

a. `myAge = 23` – Valid
b. `myAge = yourRate` – Valid
c. `myAge = departmentName` – Invalid: Mismatched types
d. `myAge = "departmentName"` – Invalid: Mismatched types
e. `42 = myAge` – Invalid: Constant on left side of equal sign
f. `yourRate = 3.5` – Valid
g. `yourRate = myAge` – Valid
h. `yourRate = departmentName` – Invalid: Mismatched types
i. `6.91 = yourRate` – Invalid: Constant on left side of equal sign
j. `departmentName = Personnel` – Invalid: Quotes are missing
k. `departmentName = "Personnel"` – Valid
l. `departmentName = 413` – Invalid: Mismatched types
m. `departmentName = "413"` – Valid
n. `departmentName = myAge` – Invalid: Mismatched types
o. `departmentName = yourRate` – Invalid: Mismatched types
p. `413 = departmentName` – Invalid: Constant on left side of equal sign and/or mismatched types
q. `"413" = departmentName` – Invalid: String literal on left of equal sign

10. Draw a flowchart or write pseudocode that represents the directions from your house to your best friend's house.

Answer:
Please note that answer will vary. A sample solution follows.

Flowchart

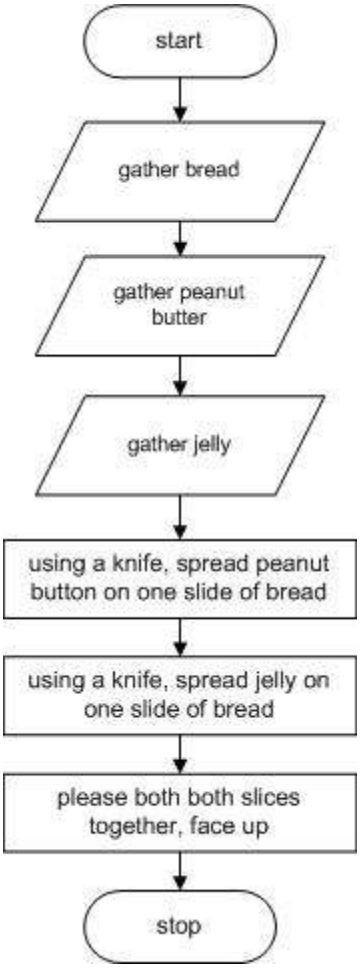
Pseudocode

```
start
        drive south on 136 Ave
        turn left (east) on First Street
        turn right (south) on Palm Ave
        arrive at 15 Palm Ave
stop
```

11. Develop the logic that represents your favorite recipe.

Answer:
Please note that answer will vary. A sample solution follows.

Flowchart



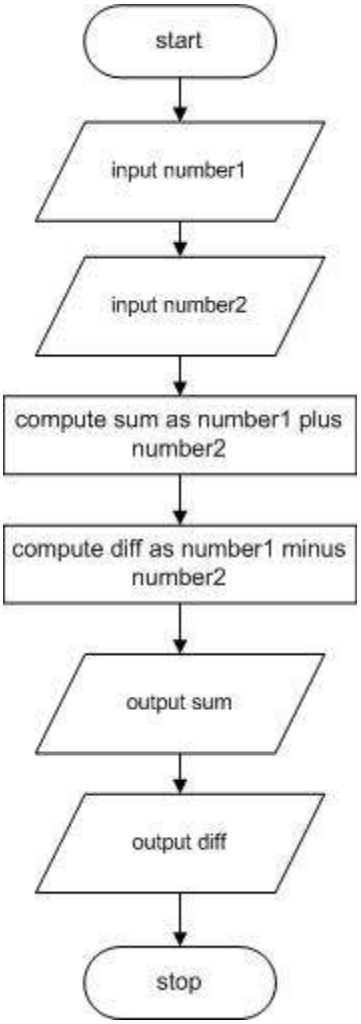
Pseudocode

```
start
        input (gather) bread
        input (gather) peanut butter
        input (gather) jelly
        using a knife, spread peanut butter on one slice of bread
        using a knife, spread jelly on another slice of bread
        place both slices together, face up
stop
```

12. Develop the logic of a program that allows the user to enter two values and displays their sum and difference.

Answer:

Flowchart



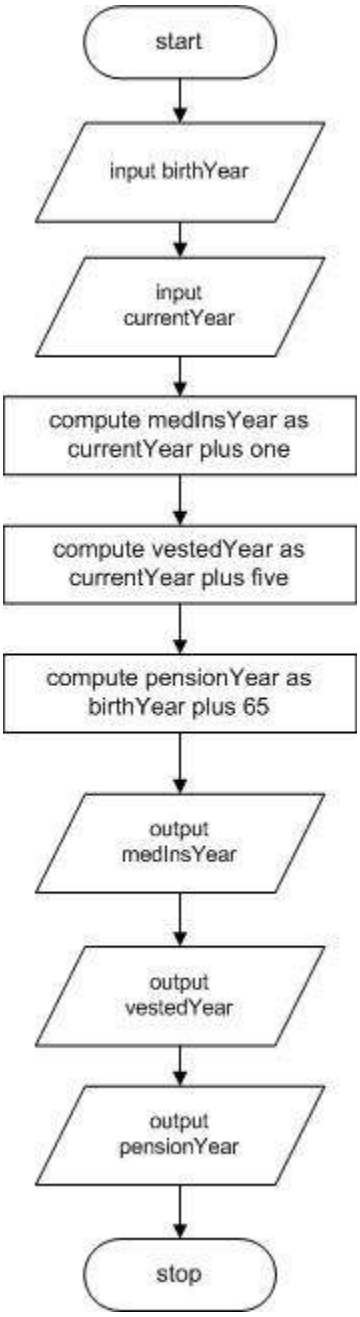
Pseudocode

```
start
        input number1
        input number2
        compute sum as number1 plus number2
        compute diff as number1 minus number3
        output sum
        output diff
stop
```

13. Develop the logic of a program that allows a new employee to enter his or her birth year and the current year. Display the years in which the employee becomes eligible for medical insurance (after one year with the company), is vested in the retirement plan (after five years with the company), and is eligible for a pension (at age 65).

Answer:

Flowchart



Pseudocode

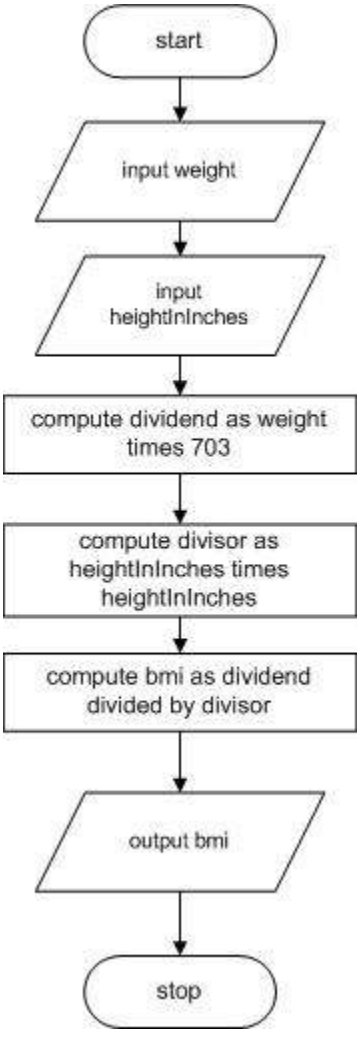
```
start
        input birthYear
        input currentYear
        compute medInsYear as currentYear plus one
        compute vestedYear as currentYear plus five
        compute pensionYear as birthYear plus 65
        output medInsYear
```

```
        output vestedYear
        output pensionYear
stop
```

14. Body mass index (BMI) is a statistical measurement that compares a person's weight and height. To calculate BMI, you multiply your weight in pounds by 703 and divide the result by the square of your height in inches. Develop the logic for a BMI calculator.

Answer:

Flowchart



Pseudocode

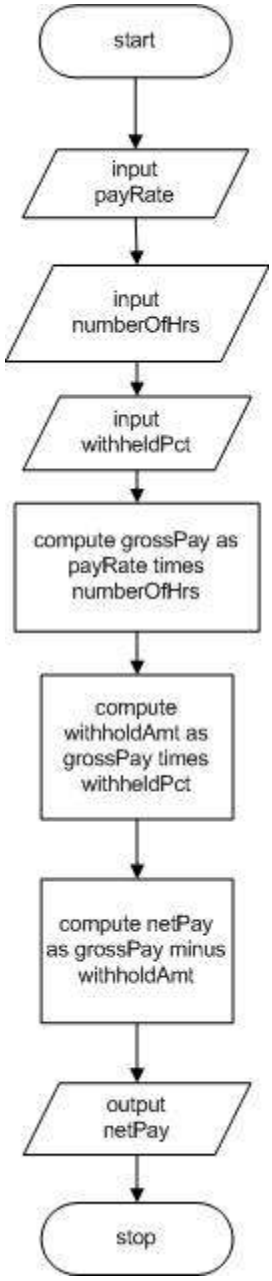
```
start
        input weight
        input heightInInches
        compute dividend as weight times 703
```

```
        compute divisor as heightInInches times heightInInches
        compute bmi as dividend divided by divisor
        output bmi
stop
```

15. Develop the logic of a program that allows the user to enter his or her hourly pay rate, the number of hours worked this pay period, and the percentage of gross salary that is withheld. The program multiplies the hourly pay rate by the number of hours worked, giving the gross pay; then, it multiplies the gross pay by the withholding percentage, giving the withholding amount. Finally, it subtracts the withholding amount from the gross pay, giving the net pay after taxes. The program displays the net pay.

    Answer:

    Flowchart



Pseudocode

```
start
        input payRate
        input numberOfHrs
        input withheldPct
        compute grossPay as payRate times numberOfHrs
        compute withholdAmt as grossPay times withheldPct
        compute netPay as grossPay minus withholdAmt
        output netPay
stop
```

16. Create the logic for a Mad Lib program that displays a message asking the user to provide five words, and then accept those words and create and display a short story or nursery rhyme that uses them.

Answer:

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

```
start
   output "Please enter a noun"
   input word1
   output "Please enter a noun"
   input word2
   output "Please enter a past-tense verb"
   input word3
   output "Please enter a noun"
   input word4
   output "Please enter an -ing verb"
   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 Logic

At a Glance

Instructor's Manual Table of Contents

- Overview
- Chapter Objectives
- Teaching Tips
- Quick Quizzes
- Class Discussion Topics
- Additional Projects
- Additional Resources
- Key Terms

Lecture Notes

Overview

Chapter 1 explores computer components and operations, the programming process, the use of pseudocode and flowcharts to document the logic of a process, naming variables and constants, data types and variable declaration, sentinel values, and the evolution of programming techniques.

Objectives

After completing this chapter, students will be able to:

- Explain computer components and operations
- Discuss the steps involved in the programming process
- Use pseudocode statements and flowchart symbols
- Use and name variables and constants
- Explain data types and declare variables
- End a program by using sentinel values
- Discuss the evolution of programming techniques

Teaching Tips

Understanding Computer Components and Operations

1. Introduce the fundamental concepts of **hardware** and **software.** Explain the purpose of a **program** and that writing instructions is called **programming.** Make a distinction between **application software** and **system software.** Explain the different operations of **input**, **processing**, and **output,** and explain how **data** is turned into **information**. Note that the processing that goes on is handled by the **central processing unit**.

2. Discuss the role of a **programming language** for writing computer instructions. Note that the process is called coding the program and the end result is called source code. Explain that programmers must follow **syntax** rules of the language, just like we follow rules of English when we speak or write.

3. Explain that a **compiler** or **interpreter** turns the source code into object code (also called **machine language**). Explain the origin of using the **binary form** representation of data. Note that when a program **runs** or **executes**, the logical steps are carried out in the order the programmer intended them to be.

Teaching Tip	A good way to handle an introduction to programming languages is to have the students list as many languages as they have heard of.

4. Distinguish logical from **semantic errors**.

5. Describe the role of **main memory**, or **random access memory (RAM)** and how that memory is **volatile**.

Quick Quiz 1

1. Name the three major operations that computer hardware and software accomplish.
   Answer: input, processing, and output

2. What type of software translates a programmer's statements to binary form?
   Answer: a compiler

3. Where does a program have to be loaded before it can be used?
   Answer: into main memory, also called random access memory (RAM)

Understanding the Programming Process

1. Emphasize the seven development steps used by professional programmers.

Teaching Tip	Note that a professional programmer usually does not just sit down at a computer keyboard and start typing.

Understanding the Problem

1. Step through the example in this section and discuss the types of questions that programmers would ask to define the problem. Note that there is no "right" answer to the questions posed on page 7 and that programmers must understand the rules of a business before writing programs.

Planning the Logic

1. Ensure that students understand that the heart of the programming process lies in planning the program's logic. Note that **algorithms** are better designed on paper rather than on the computer, just like a blueprint for a house appears on paper before the house is built.

2. Explain that programmers **desk-check** their code before the computer runs it.

Coding the Program

1. Discuss the similarities between programming languages.

Teaching Tip	A good way to illustrate differences is to take the same IF statement and code it in several different languages. This shows the student that the concept of an IF statement is the same no matter what language it's coded in.

2. Explore the basis for choosing between programming languages based on built-in capabilities.

3. Reinforce that the planning step is the most important and discuss why.

Using Software to Translate the Program into Machine Language

1. Discuss the role that the compiler plays in translating between a **high-level programming language** and the **low-level machine language** native to the computer.

2. Ask how many programs would be written if we had to code in binary?

Testing the Program

1. Draw attention to the difference between syntax and logic errors.

2. Explain what a structured walk-through is and why finding errors before the program is coded saves time and money.

3. Note that most languages allow the programmer to step through the code line by line.

Putting the Program into Production

1. Explain that a program can stand alone or be part of a large system of programs all designed to accomplish specific tasks.

2. Explore the non-programming issues related to putting a program into production, such as training and **conversion**.

Maintaining the Program

1. Describe some of the reasons why **maintenance** is important.

2. Explore some of the implications of maintenance that have an impact on initial programming, such as good clarity in naming variables.

Quick Quiz 2

1. What is the process of walking through a program's logic on paper before actually writing the program called?
   Answer: desk-checking

2. What kind of language does a computer understand?
   Answer: low-level machine language

3. What is program maintenance?
   Answer: required changes made after a program is put into production

Using Pseudocode Statements and Flowchart Symbols

1. Highlight the fact that **pseudocode** and **flowcharting** are tools to represent the logical steps required to solve a problem.

Teaching Tip	When programmers plan the logic for a solution to a programming problem, they often use one of two tools: pseudocode or flowcharts.

2. Review with your students the basic symbols associated with flowcharting.

3. Explain that the **input symbol** and **output symbol** are identical – only the words inside the symbol change to designate input or output – therefore it is called the **input/output symbol** or **I/O symbol**.

4. Highlight the fact that the **processing symbol** is a catch-all symbol and that a lot of different operations can occur there.

5. Note that some programmers put arrows on the **flowlines** in a flowchart and that **terminal symbols** start and stop the flowchart.

The Advantages of Repetition

1. Highlight the value that repetition represents in a computing context.

2. Explain how businesses may use the same program every day for years.

Quick Quiz 3

1. List some of the main symbols used for flowcharting.
   Answer: input, processing, output, and terminal symbols; flowlines or arrows

2. Describe the purpose of flowcharting and pseudocode.
   Answer: to represent the logical steps it takes to solve a problem

Using and Naming Variables and Constants

1. Emphasize the fundamental importance of **variables**, also called **identifiers**, as named memory locations.

2. Describe the use of camel casing as an aid to clarify variable names.

3. Highlight that the syntax of this text requires that variable names must be one word (most programming languages allow more than one word, but no spaces between words) and should have an appropriate meaning.

Teaching Tip	Note that when designing the logic of a computer program, students should not concern themselves with the specific syntax of any particular computer language.

Assigning Values to Variables

1. Review the **assignment statement** and the **assignment operator** with the students.

2. Note the difference between regular math (where the answer is to the right of the equal sign) and computer math (where the answer is to the left of the equal sign).

Teaching Tip	Programmers must distinguish between numeric and string variables because computers handle the two types of data differently.

3. Discuss named constants and magic numbers.

Performing Arithmetic Operations

1. Introduce the standard arithmetic operators. Explain why there is no x symbol for multiplication and no ÷ symbol for division

2. Review the importance of rules of precedence in understanding the order in which operations in the same statement are carried out. Students may be familiar with the mnemonic “Please Excuse My Dear Aunt Sally.”

Quick Quiz 4

1. What is another name for a variable?
   Answer: identifier

2. The equal sign in the following expression represents what type of operator?
   `calculatedAnswer = originalNumber * 2`
   Answer: the assignment operator

Understanding Data Types and Declaring Variables

1. Explain the concept of constants and the difference between a **numeric constant** and a **string constant**.

2. Explain the need for **data types** and the different types available.

3. Describe the general process of **declaring a variable**.

4. Explain how variables should be **initialized** because they contain **garbage values** when they are declared.

Quick Quiz 5

1. What are the two basic types of data?
   Answer: text (or string) and numeric

2. Describe the data that can be held in a string variable.
   Answer: letters of the alphabet and other special characters such as punctuation marks

Ending a Program by Using Sentinel Values

1. Draw the students' attention to the **infinite loop** problem.

2. Discuss the use of a **sentinel value** or **dummy value** to stop a program and that many programmers use the term **eof** (end of file) to refer to this marker.

Teaching Tip	A repeating flow of logic with no end is a major flaw called an infinite loop.

3. Explain that testing a value is **making a decision**.

Quick Quiz 6

1. What is the name for a value that means, "Stop the program?"
   Answer: sentinel value

2. What is the use for an **eof** sentinel value?
   Answer: to recognize the end of data in a file automatically

Understanding the Evolution of Programming Techniques

1. Explain the two major techniques (**procedural** and **object-oriented programming**) used to develop programs and discuss their similarities and differences.

2. Note that students have a lot of experience using object-oriented programming when they work on their own computer's operating system.

Quick Quiz 7

1. What major programming technique focuses on the procedures that programmers create?
   Answer: procedural programming

2. What is the focus of object-oriented programming?
   Answer: Objects, or "things," and their features and behaviors.

Class Discussion Topics

1. In a real world application, such as balancing a check book or handling online sales transactions, how can unit analysis help prevent errors?

2. What do you think are the advantages/disadvantages of using pseudocode versus flowcharts for describing algorithms? Should both be used as part of system documentation? Why or why not?

3. How can naming conventions, such as camel casing and relating variable names to the meaning of the value, simplify the work of programmers, both when creating a piece of software and when maintaining it?

4. Understanding the evolution of programming techniques can help to use those techniques more effectively. Explore the differences between procedural and object oriented programming techniques, and the relative strengths of each methodology.

Additional Projects

1. Have students create a preliminary analysis of a request for a program that will allow the instructor to maintain grades for the course. A written document of the required features and functionality should be created.

2. Have students do online research to learn about the history of programming languages and produce a brief report on the nature, advantages, and disadvantages of two popular programming languages.

Additional Resources

1. The link below provides a brief history of the major mainframe computer players in the 1960s and 1970s. Most notably, it describes Burroughs Corporation's B5000, a mainframe that had no assembly language; all of the operating system software was written in variants of the ALGOL language.

   The OS architecture used on this system was years ahead of its time. The current line of Unisys (formerly Burroughs) computers use the same basic OS architecture. Many of the capabilities of the B5000 and its successor systems have now also been adopted into PC operating systems.
   www.cs.uiowa.edu/~jones/assem/summer97/notes/28.html

2. A history of computer languages (an excellent PowerPoint presentation):
   www.cs.wpi.edu/~dfinkel/Courses/CS2136_D04/class02.ppt

3. The first bug:
   www.computer.org/history/development/1945.htm

Key Terms

- **Algorithm -** The sequence of steps necessary to solve any problem.
- **Assignment operator –** An equal sign.
- **Assignment statements –** A statement in which any operation performed to the right of the equal sign results in a value that is placed in the memory location to the left of the equal sign.
- **Application software -** Comprises all the programs you apply to a task—word-processing programs, spreadsheets, payroll and inventory programs, and even games.
- **Binary form –** The machine language, represented as a series of 0s and 1s.
- **Camel casing** – A variable naming convention in which the first letter of a variable name is lower case and any words that follow in the variable name begin with an upper case letter, making the name easier to read.
- **Central processing unit (CPU) -** The hardware component that performs processing tasks.
- **Coding the program -** When you write a computer program.

- **Compiler or interpreter -** Language translation software that converts a programmer's statements to binary form.
- **Conversion -** The entire set of actions an organization must take to switch over to using a new program or set of programs, can sometimes take months or years to accomplish.
- **Data -** Through hardware devices, data, or facts, enter the computer system.
- **Decision symbol -** The diamond symbol usually contains a question, the answer to which is one of two mutually exclusive options—yes or no.
- **Declare the variable -** Provide the program with a data type and an identifier.
- **Desk Checking -** The process of walking through a program's logic on paper before you actually write the program.
- **Dummy value -** A preselected value that stops the execution of a program.
- **Eof –** A marker that automatically acts as a sentinel.
- **Execute or run -** When a program's instructions are carried out.
- **Flowchart -** A pictorial representation of the logical steps it takes to solve a problem.
- **Flowlines –** Lines and arrows to connect the steps in a flowchart.
- **Hardware -** The equipment or devices associated with a computer.
- **High-level programming language -** The English-like statements a programmer has coded.
- **Identifier** - A variable name is also called an identifier.
- **Infinite loop –** A repeating flow of logic with no end.
- **Initialized -** Assigning initial values to variables.
- **Information -** After data items have been processed, they become information.
- **Input -** Hardware devices that perform input operations include keyboards and mice.
- **Input symbol –** A parallelogram symbol in a flowchart designating an input operation.
- **Input/Output symbol or I/O symbol –** A parallelogram symbol in a flowchart designating an input or an output operation.
- **Garbage value -** Uninitialized variables have an unknown value.
- **Logic –** Making sure you give the instructions to the computer in a specific sequence, not leaving any instructions out, and not adding extraneous instructions.
- **Logical errors –** Statements that have correct syntax but are out of sequence or ask the computer to perform operations in a wrong order.
- **Low-level machine language –** A language created by a compiler or interpreter that the computer understands.
- **Machine language** or **object code -** The computer's on/off circuitry language, represented as a series of 0s and 1s.
- **Magic number -** An unnamed constant, like 0.06, whose meaning is not immediately apparent.
- **Main memory** (also **Random access memory**) **–** location where a copy of the instructions must be placed in memory before the program can be run.
- **Maintenance –** The process of updating programs.
- **Memory -** The internal storage in a computer.
- **Modularity -** The ability to build programs from smaller segments.
- **Named constants** - Similar to a variable, except that, unlike a variable that can be assigned different values over time, a named constant can be assigned a value only once.
- **Numeric constant (Literal numeric constant, unnamed numeric constant**) - A specific numeric value.

- **Numeric variable -** Can have mathematical operations performed on it; it can hold digits, and can usually hold a decimal point and a sign indicating positive or negative if you want.
- **Object-oriented programming -** Focuses on objects, or "things," and describes their features (also called attributes) and behaviors.
- **Output –** Information often is sent to a printer, monitor, or some other output device so people can view, interpret, and use the results.
- **Output symbol -** A parallelogram symbol in a flowchart designating an input operation.
- **Pascal casing** – Each word in a variable name begins with an upper case letter.
- **Procedural programming -** Focuses on the procedures that programmers create.
- **Processing -** Processing data items may involve organizing them, checking them for accuracy, or performing mathematical operations on them.
- **Processing symbol –** A rectangle symbol in a flowchart that contains arithmetic operation statements.
- **Program code -** The instructions you write in a computer program.
- **Programming -** The process of writing software instructions.
- **Programming language -** Software instructions are written in a computer programming language, such as Visual Basic, C#, C++, or Java.
- **Pseudocode -** An English-like representation of the logical steps it takes to solve a problem.
- **Rules of precedence –** Rules that dictate the order in which operations in the same statement are carried out.
- **Semantic error -** Occur when the grammar is correct but the statement makes no sense in the current context.
- **Sentinel value -** Predetermined value that means "Stop."
- **Software -** Computer instructions; software tells the hardware what to do.
- **Source code -** Program code is also called source code.
- **String constant -** Using a specific text value, or string of characters, such as "Amanda", within quotation marks.
- **String variable** - A separate type of variable that can hold letters of the alphabet and other special characters such as punctuation marks.
- **Syntax –** The rules governing a computer language's word usage and punctuation.
- **System software -** Comprises the programs used to manage your computer, including operating systems such as Windows, Linux, or UNIX.
- **Terminal symbols –** Oval symbols used to start and stop a flowchart.
- **Variables –** Named memory locations whose contents can vary over time.
- **Volatile –** Memory contents are lost every time the computer loses power.