

# PROGRAM DEVELOPMENT AND PROGRAMMING LANGUAGES

Introduction to Information and Communication  
Technologies

Dr. Muhammad Abdullah

---



Department of Data Science  
Faculty of Computing and Information Technology (FCIT)  
University of the Punjab, Lahore, Pakistan.

## Learning Objectives

1. Understand the differences between structured programming, object-oriented programming (OOP), aspect-oriented programming (AOP), and adaptive software development.
2. Identify and describe the activities involved in the program development life cycle (PDLC).
3. Understand what constitutes good program design and list several tools that can be used by computer professionals when designing a program.

## Learning Objectives

4. Explain the three basic control structures and how they can be used to control program flow during execution.
5. Discuss some of the activities involved with debugging a program and otherwise ensuring it is designed and written properly.
6. List some tools that can be used to speed up or otherwise facilitate the program development process.
7. Describe several programming languages in use today and explain their key features.

# Overview

- This chapter covers:
  - The most common approaches to program design and development
  - The phases of the program development life cycle (PDLC)
  - Tools that can be used to design and develop a program
  - Good program design techniques and types of program errors
  - Common programming languages

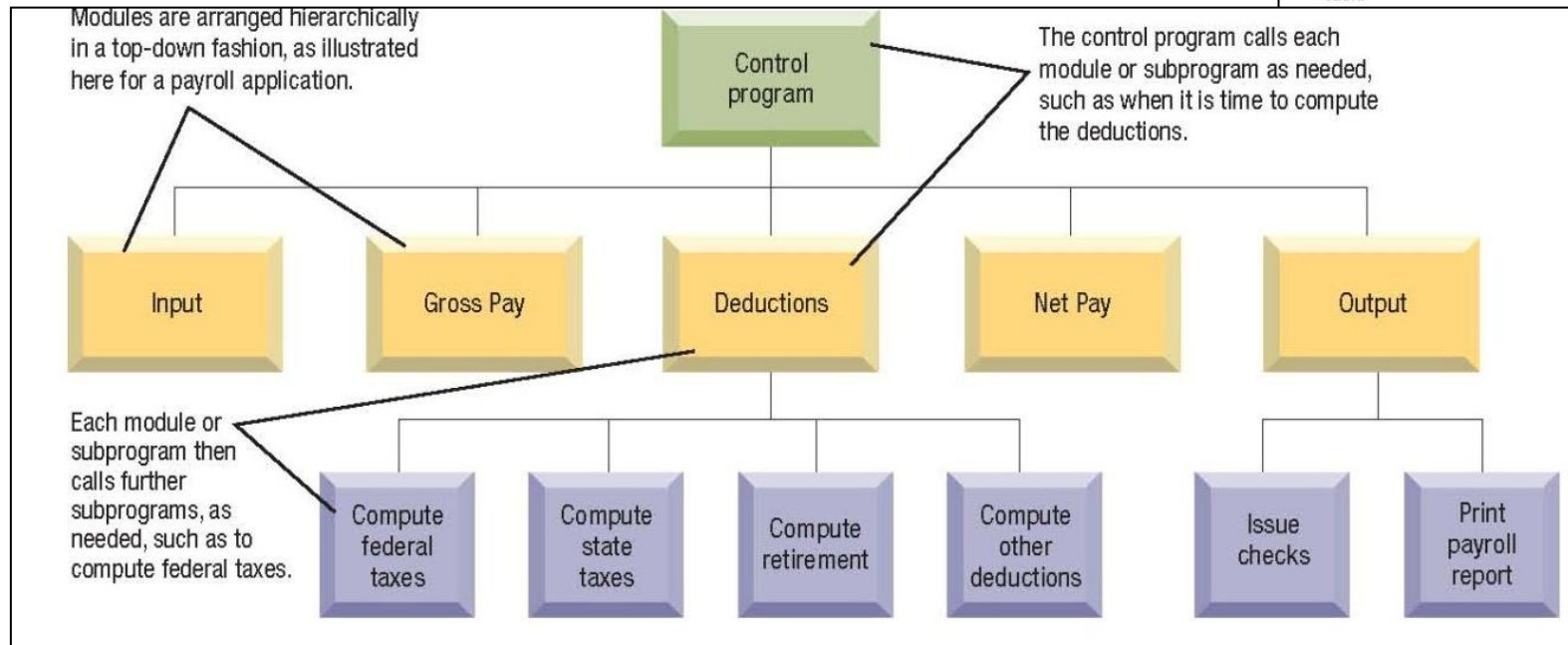
# Approaches to Program Design and Development

- **Procedural programming:** An approach to program design in which a program is separated into small modules that are called by the main program or another module when needed
  - Uses procedures (modules, subprograms): Smaller sections of code that perform specific tasks
  - Allows each procedure to be performed as many times as needed; multiple copies of code not needed
  - Prior to procedural programming, programs were one large set of instructions (used GOTO statements)
  - **Structured programming:** Goes even further, breaking the program into small modules (Top-down design)

# Approaches to Program Design and Development

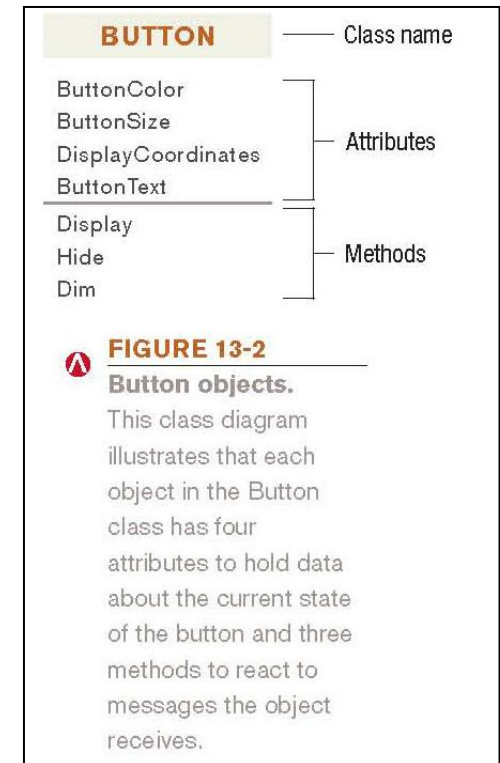
- Variables: Named memory locations that are defined for a program
  - Used to store the current value of data items used in the program

**FIGURE 13-1**  
**Structured programming.**  
A structured program is divided into individual modules; each module represents a very specific processing task.



# Approaches to Program Design and Development

- **Object-oriented programming (OOP):** Programs consist of a collection of objects that contain data and methods to be used with that data
  - Class: Group of objects that share some common properties
  - Instance: An individual object in a class
  - Attributes: Data about the state of an object
  - Methods: Perform actions on an object
  - Objects can perform nontraditional actions and be easily used by more than one program



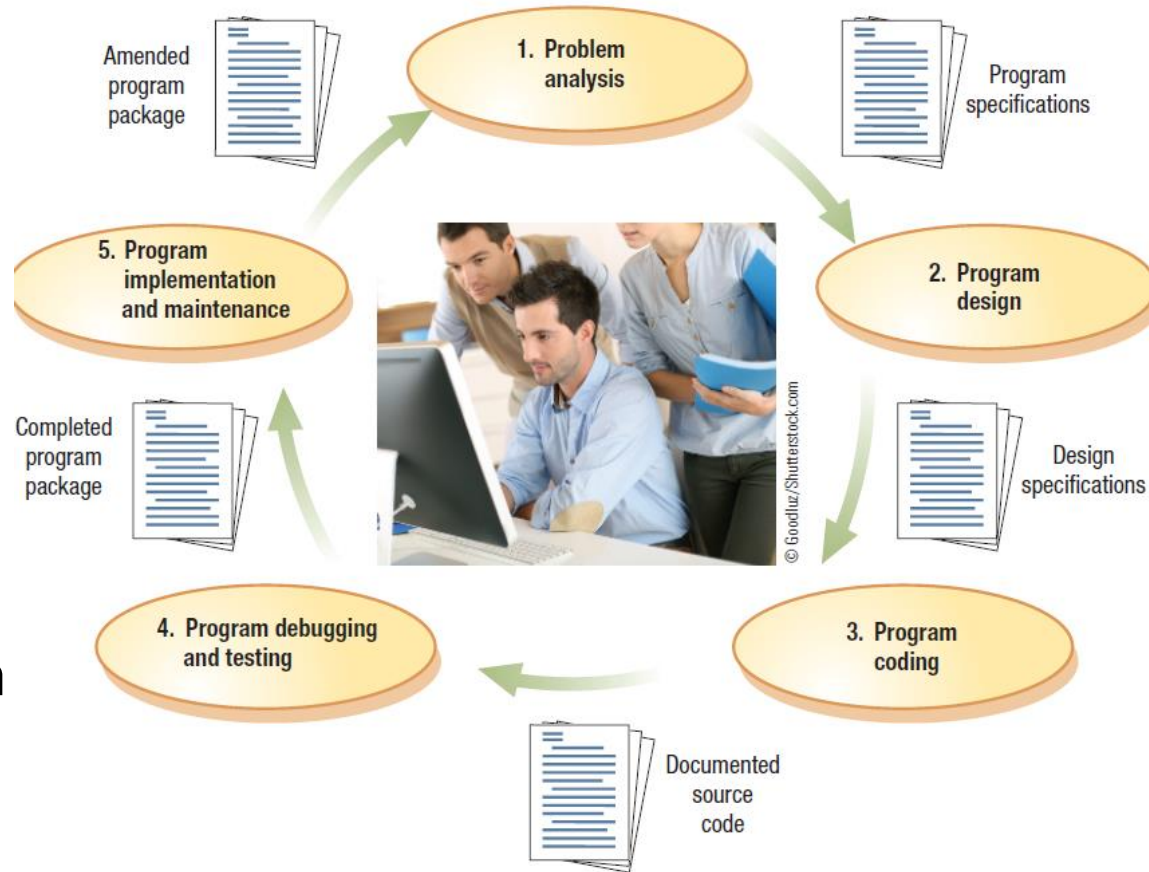
# Approaches to Program Design and Development

- **Aspect-oriented programming (AOP):** Separates functions so program components can be developed and modified individually from one another
  - The components can be easily reused with separate nonrelated objects
- **Adaptive software development:** Designed to make program development faster and more efficient and focus on adapting the program as it is being written
  - Iterative and/or incremental
  - Includes RAD (rapid application development) and extreme programming (XP)
  - Agile software development: Focuses on building small functional program pieces during the project



# The Program Development Life Cycle (PDLC)

- **Program development:** The process of creating application programs
- **Program development life cycle (PDLC):** The process containing the five phases of program development



**FIGURE 13-3**  
The program development life cycle (PDLC). Each phase of the program development life cycle produces some type of documentation to pass on to the next phase.

# The Program Development Life Cycle (PDLC)

- **Problem analysis:** The problem is considered and the program specifications are developed
  - Specifications developed during the PDLC are reviewed by the systems analyst and the programmer (the person who will code the program)
  - Goal: To understand the functions the software must perform
  - Documentation: Includes program specifications (what it does, timetable, programming language to be used, etc)

# The Program Development Life Cycle (PDLC)

- **Program design:** The program specifications are expanded into a complete design of the new program
  - Good program design is extremely important
  - Program design tools
    - Structure charts: Depict the overall organization of a program
    - **Flowcharts:** Show graphically step-by-step how a computer program will process data
      - Use special symbols and relational operators
      - Can be drawn by hand or with flowcharting software

# Flowcharts

## FLOWCHART SYMBOLS

Start/stop  
program



Decision



Processing



Connector



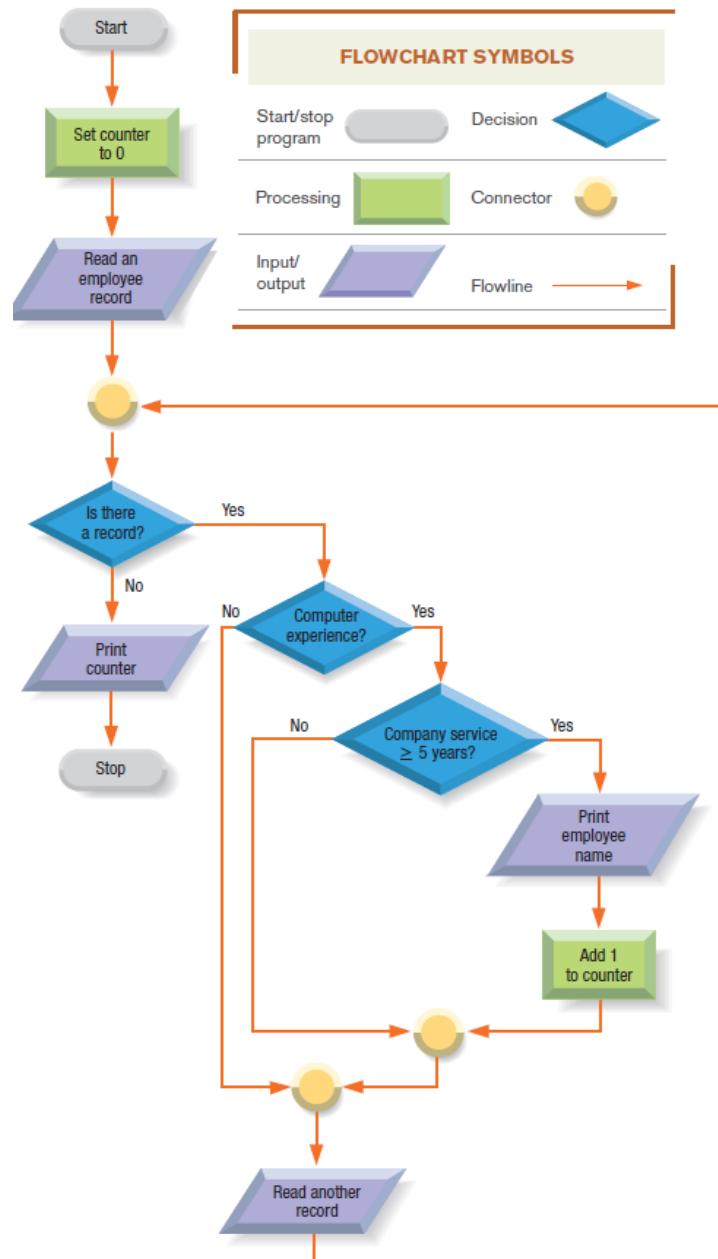
Input/  
output



Flowline



# Flowcharts



# The Program Development Life Cycle (PDLC)

- **Pseudocode:** Uses English-like statements to outline the logic of a program
- **Unified Modeling Language (UML) Models:** Set of standard notations for creating business models
  - Widely used in object-oriented programs
  - Includes class diagrams, use case diagrams, etc.



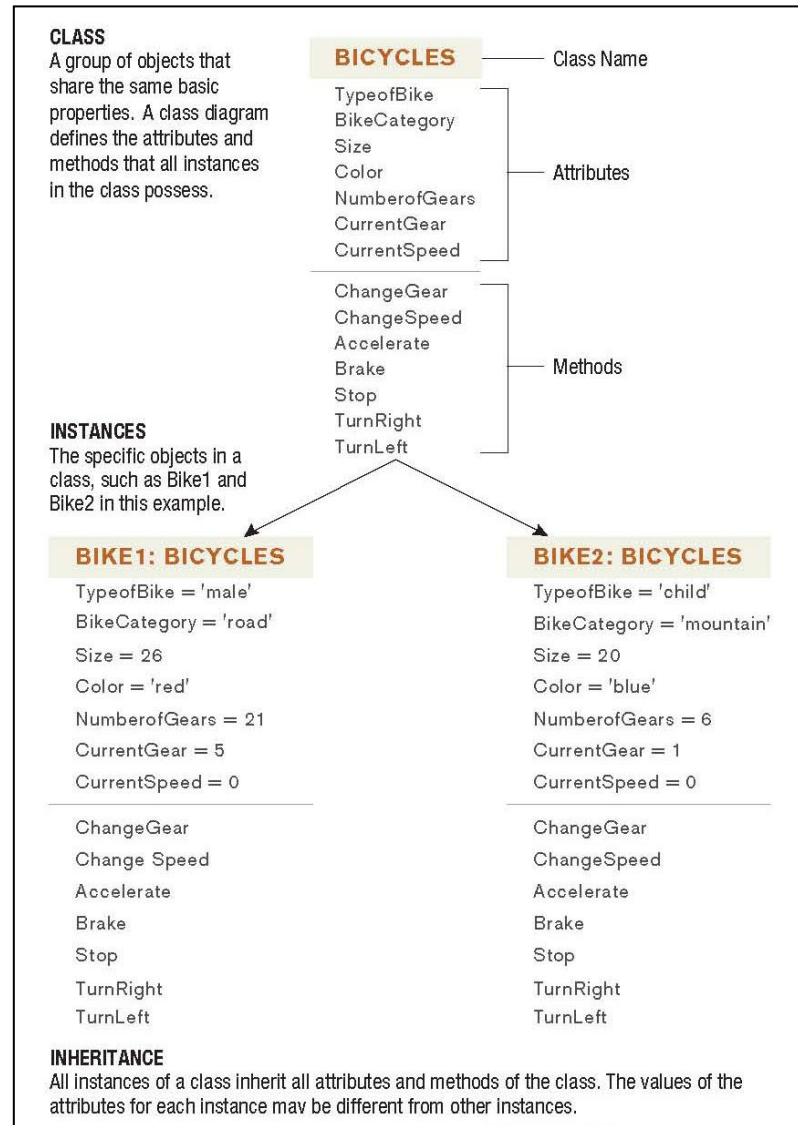
**FIGURE 13-5**

**Pseudocode.**

The problem is the same as illustrated in the flowchart in Figure 13-4.

```
Start
counter = 0
Read a record
DO WHILE there are records to process
  IF computer_experience
    IF company_service ≥ 5 years
      Print employee_name
      Increment counter
    ELSE
      Next statement
    END IF
  ELSE
    Next statement
  END IF
  Read another record
END DO
Print counter
Stop
```

# Unified Modeling Language (UML) Models



**FIGURE 13-6**

## Class diagrams.

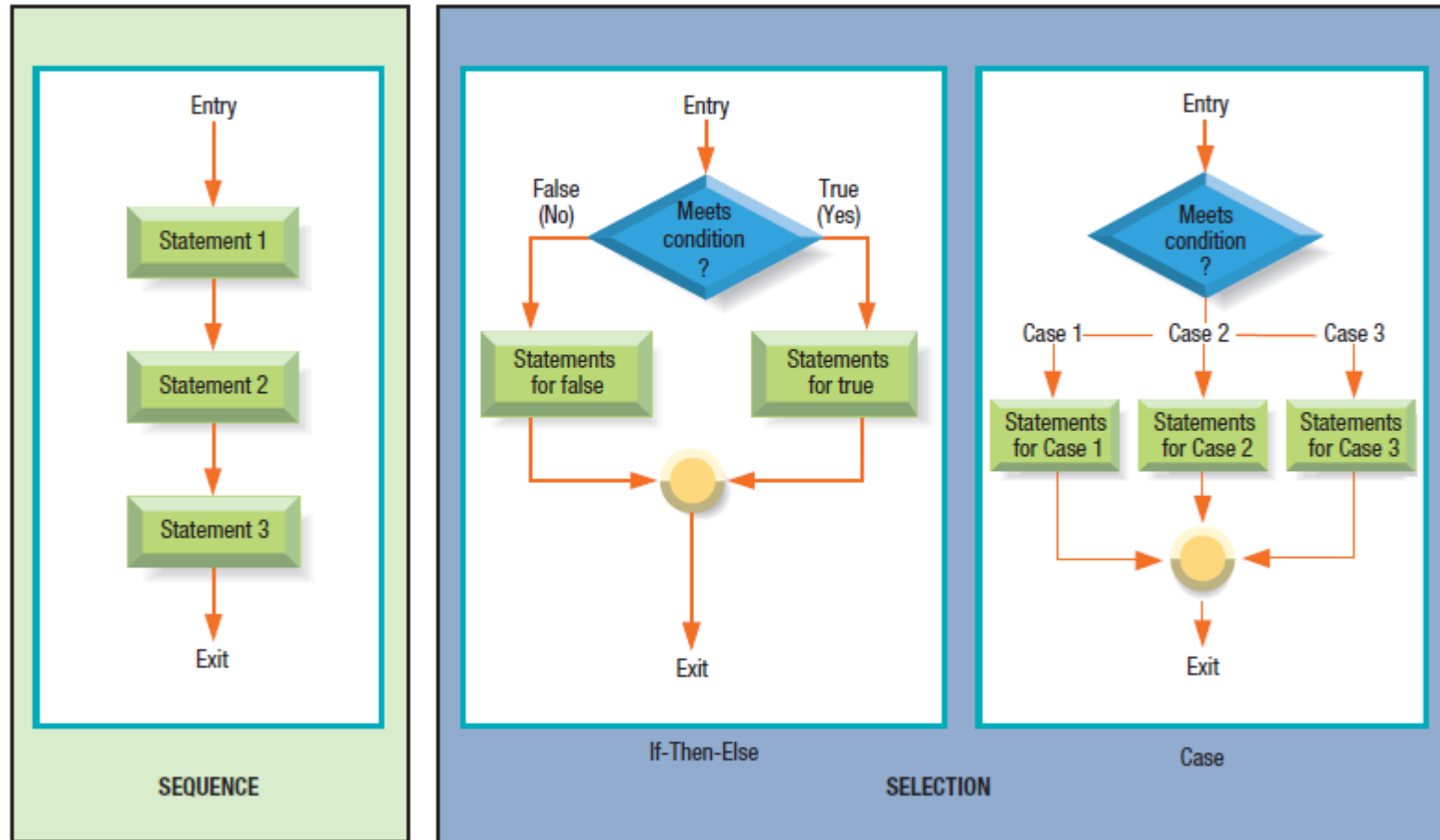
This example shows one class and two instances of that class.

# The Program Development Life Cycle (PDLC)

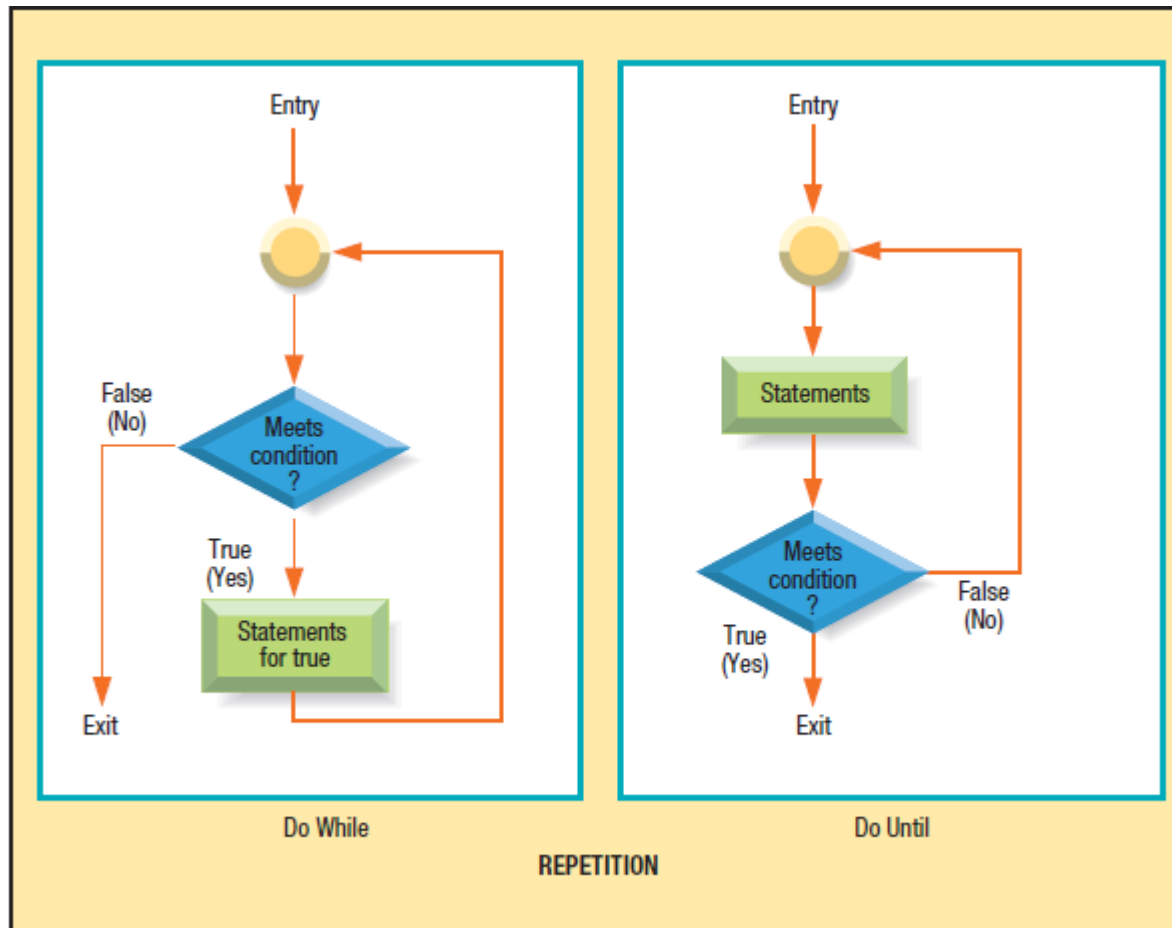
- **Control structure:** A pattern for controlling the flow of logic in a computer program, module, or method
  - Sequence control structure: Series of statements that follow one another
  - Selection control structure: Multiple paths, direction depends on result of test
    - If-then-else
    - Case (avoids nested if-then-else statements)
  - Repetition control structure: Repeat series of steps
    - Do-while
    - Do-until



# Control Structures



# Control Structures



# The Program Development Life Cycle (PDLC)

- Good program design:
  - Is essential
  - Saves time
- Good program design principles:
  - Be specific
    - All things the program must do need to be specified
  - One-entry-point/one-exit-point rule
  - No infinite loops or other logic errors
    - Infinite loop: Series of steps that repeat forever
- Design should be tested to ensure logic is correct
  - Desk check; tracing tables
- Documentation: Includes design specifications

# Good Program Design

## DIRECTIONS FOR PERSON

1. Please make a piece of toast with margarine for me.

© Les Scholtz/Shutterstock.com;  
Copyright © 2015 Cengage Learning®



## DIRECTIONS FOR COMPUTER

1. Take one slice of bread out of the bag of bread.
2. Put the bread into the slot on the toaster, narrow edge first with the widest part of the bread fitting into the widest part of the slot.
3. Push the start lever on the toaster down.
4. When the toast pops up, remove the toast from the toaster.
5. Place the toast on a plate.
6. Open the silverware drawer, take out a knife, then close the silverware drawer.
7. Open the refrigerator, remove the margarine, then shut the refrigerator.
8. Take the lid off the margarine.
9. Scoop out one teaspoon of margarine with the knife.
10. Spread the margarine on the top side of the toast, evenly covering that surface of the toast.
11. Place the lid back on the margarine.
12. Open the refrigerator, replace the margarine, then shut the refrigerator.



### FIGURE 13-8

Writing instructions for a computer versus a person. A computer requires step-by-step instructions.

# Program Design Testing

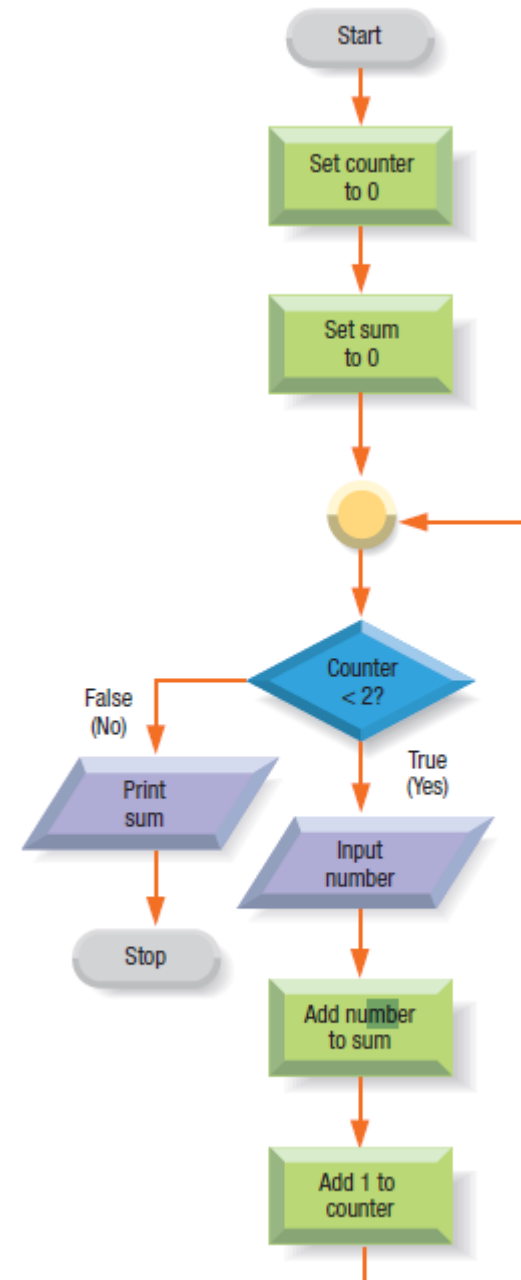
## DESK CHECK RESULTS FOR CORRECT FLOWCHART

Flowchart Stage	Counter	Decision Test Results (Counter < 2)	Number	Sum
Initialization	0	–	–	0
First decision test	0	T (enters loop)	–	0
After first loop	1	–	6	6
Second decision test	1	T (enters loop)	6	6
After second loop	2	–	3	9
Third decision test	2	F (exits loop)	3	9

Test data: 6, 3; Expected results: Sum = 9; Actual results: Sum = 9



**FIGURE 13-9**  
Desk checking a  
flowchart.



# Program Design Testing

## DESK CHECK RESULTS FOR INCORRECT FLOWCHART

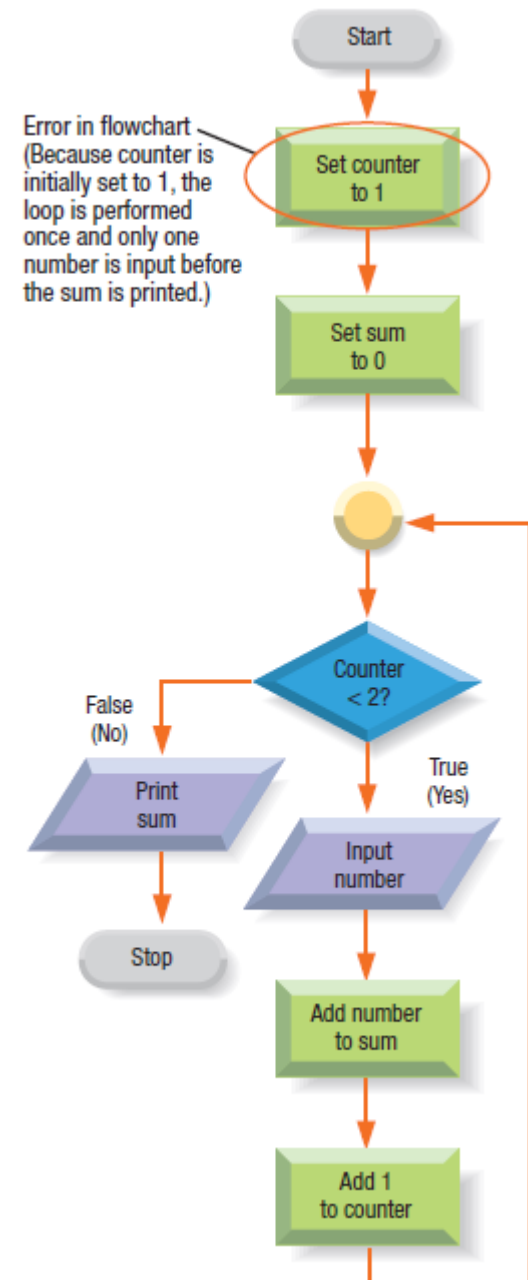
Flowchart Stage	Counter	Decision Test Results (Counter < 2)	Number	Sum
Initialization	1	—	—	0
First decision test	1	T (enters loop)	—	0
After first loop	2	—	6	6
Second decision test	2	F (exits loop)	6	6

Test data: 6, 3; Expected results: Sum = 9; Actual results: Sum = 6



**FIGURE 13-9**

Desk checking a flowchart.



# The Program Development Life Cycle (PDLC)

- **Program coding:** The program code is written using a programming language.
  - When choosing a programming language, consider:
    - Suitability to the application
    - Integration with other programs
    - Standards for the company
    - Programmer availability
    - Portability if being run on multiple platforms
    - Development speed
  - Coding creates source code

# Coding Standards

- Coding standards: Rules designed to standardize programming
  - Makes programs more readable and easier to maintain
  - Includes the proper use of comments to:
    - Identify the programmer and last modification date
    - Explain variables used in the program
    - Identify the main parts of the program
- Reusable code: Pretested, error-free code segments that can be used over and over again with minor modifications
  - Can greatly reduce development time
- Documentation: Includes documented source code



# Comments

## COMMENTS

Comments are usually preceded by a specific symbol (such as \*, C, ', #, or //); the symbol used depends on the programming language being used. Anything else in a comment line is ignored by the computer.

Comments at the top of a program should identify the name and author of the program, date written and last modified, purpose of the program, and variables used in the program.

Comments in the main part of a program should indicate what each section of the program is doing. Blank comment lines can also be used to space out the lines of code, as needed for readability.

```
*****
* This program inputs two numbers, computes their sum,      *
* and displays the sum.                                     *
*                                                           *
* Written by: Deborah Morley  3/12/10                       *
*****
* Variable list                                             *
* SUM: Running sum                                         *
* CNTR: Counter                                            *
* NUM: Number inputted                                     *
*
    REAL SUM, CNTR, NUM
*****
*
* INITIALIZE VARIABLES
    SUM = 0
    CNTR= 0
*
* INPUT NUMBER, ADD IT TO THE SUM, INCREMENT COUNTER, AND THEN
* REPEAT UNTIL TWO NUMBERS HAVE BEEN ENTERED
    DO 10 CNTR = 1, 2
```



**FIGURE 13-10**

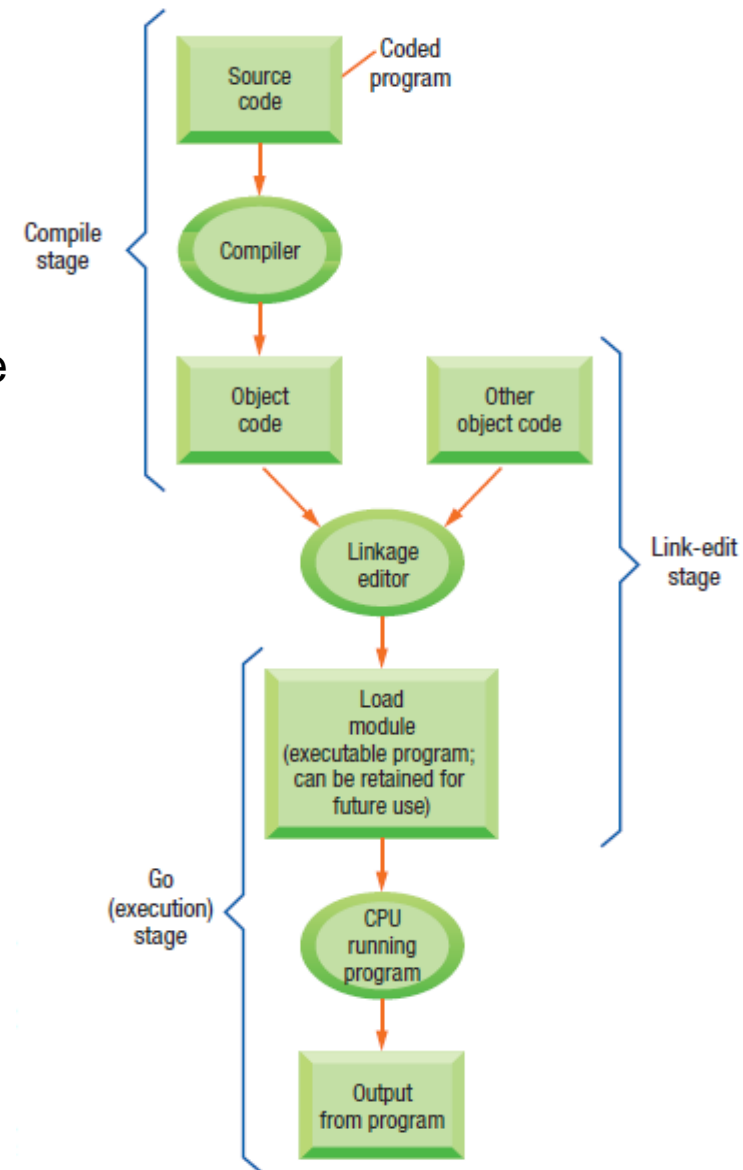
Program comments.

# The Program Development Life Cycle (PDLC)

- Program debugging and testing: The process of ensuring a program is free of errors (bugs) and works as it is supposed to
  - Before they can be debugged, coded programs need to be translated into executable code
    - Source code: Coded program before it is compiled
    - Object code: Machine language version of a program
    - Language translator: Program that converts source code to machine language

# The Program Development Life Cycle (PDLC)

- Types of language translators:
  - Compilers: Language translator that converts an entire program into machine language before executing it
  - Interpreters: Translates one line of code at one time
  - Assemblers: Convert assembly language programs into machine language



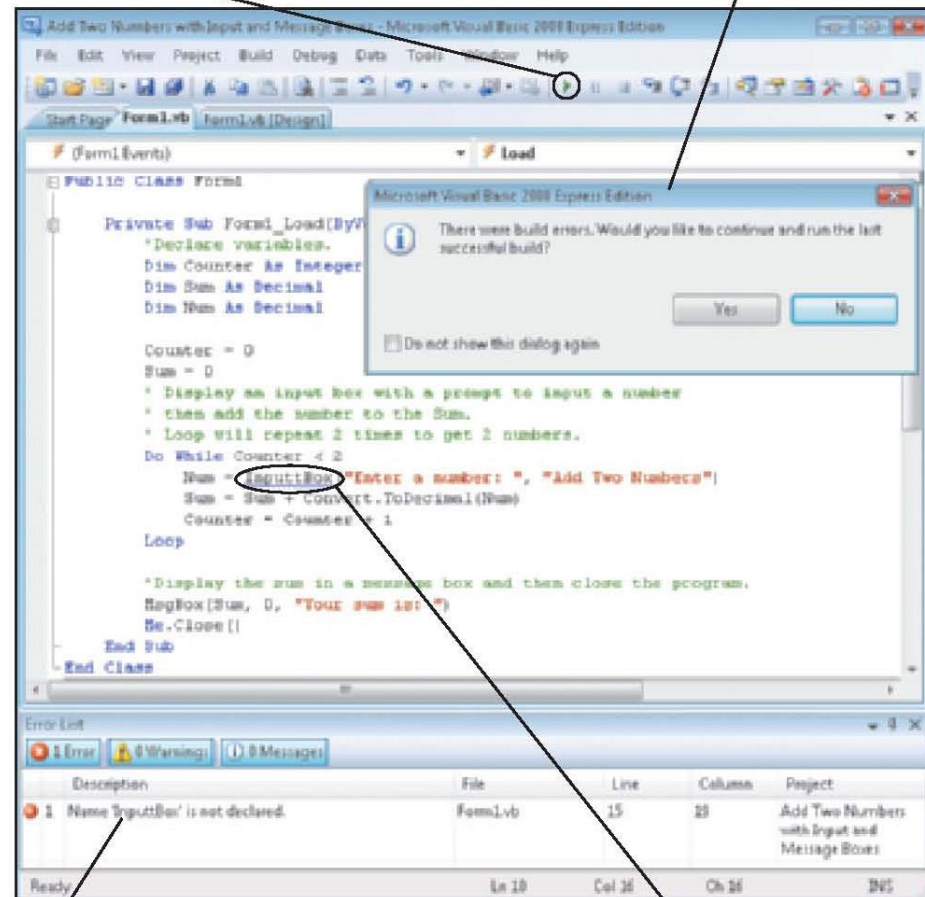
# The Program Development Life Cycle (PDLC)

- Preliminary debugging: Finds initial errors
  - Compiler errors: Program doesn't run
    - Typically syntax errors: When the programmer has not followed the rules of the programming language
  - Run time error: Error that occurs when the program is running
    - Logic errors: Program will run but produces incorrect results
    - Dummy print statements can help locate logic errors and other run time errors

# Preliminary Debugging

1. Clicking the Debug button starts the compilation and debugging process.

2. If a compiler error is encountered, the application typically displays an error message.



4. The debugger displays an error list containing all compiler errors.

3. This misspelled command is marked by a blue wavy underline.

**FIGURE 13-12**  
**Syntax errors.** Syntax errors occur when the syntax (grammar rules) for a program is not followed precisely; they become obvious when compiling a program.

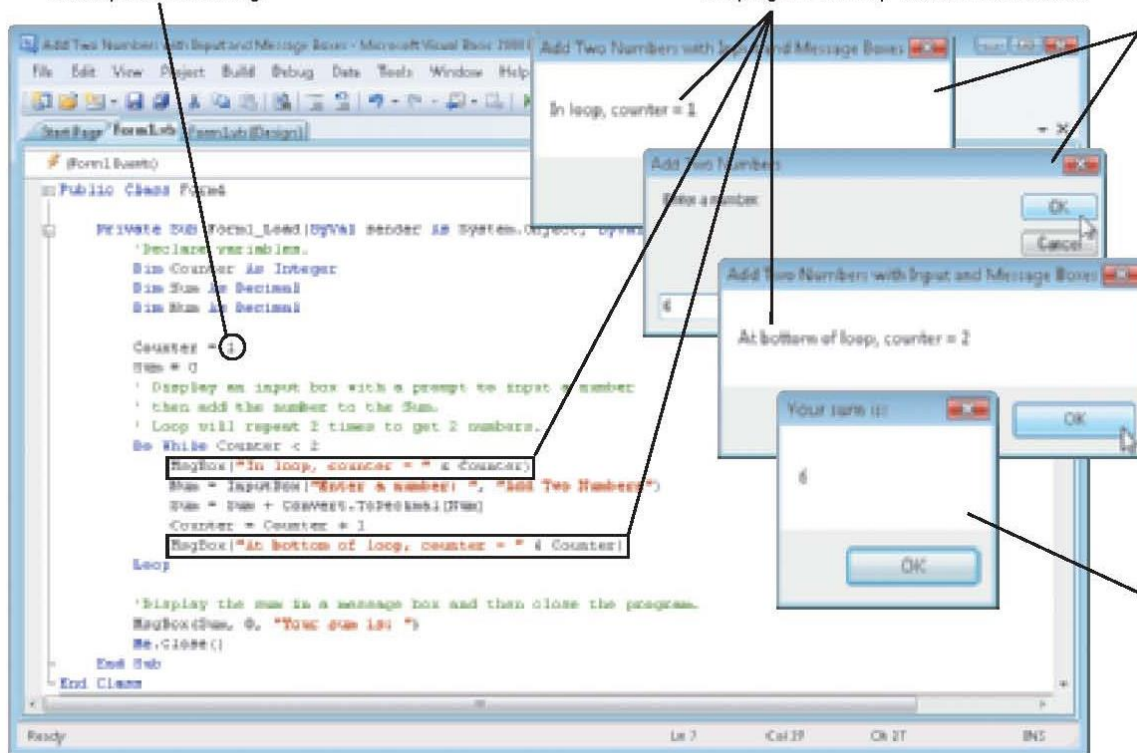
# Preliminary Debugging

1. With logic errors, such as initializing a counter to the wrong number as shown here, the program will run but the output will be wrong.

2. Adding dummy print statements to display the values of key variables and key locations in the program can help to determine the error.

3. The dummy print statements, as well as the regular input and output messages belonging to the program, are displayed at the appropriate times when the program is executed.

4. The dummy print statements reveal that the loop is performed only once before the sum is displayed and help the programmer locate the counter initialization error.



**FIGURE 13-13**  
**Logic errors.** Logic errors are more difficult to identify; dummy print statements can help determine the error.

# The Program Development Life Cycle (PDLC)

- Testing: Occurs after the program appears to be correct to find any additional errors
  - Should use good test data
  - Tests conditions that will occur when the program is implemented
  - Should check for coding omissions (product quantity allowed to be  $< 0$ , etc.)
  - Alpha test (inside organization)
  - Beta test (outside testers)
- Documentation: Completed program package (user's manual, description of software commands, troubleshooting guide to help with difficulties, etc.)

# The Program Development Life Cycle (PDLC)

- Program implementation and maintenance: Installing and maintaining the program
  - Once the system containing the program is up and running, the implementation process is complete
  - Program maintenance: Process of updating software so it continues to be useful
    - Very costly
  - Documentation: Amended program package



## Quick Quiz

1. Which approach to programming uses the concept of inheritance?
  - a. Procedural
  - b. Object-oriented
  - c. Aspect-oriented
2. True or False: An infinite loop is an example of a logic error.
3. A(n)\_\_\_\_\_ is a program design tool that shows graphically step-by-step the actions a computer program will take.

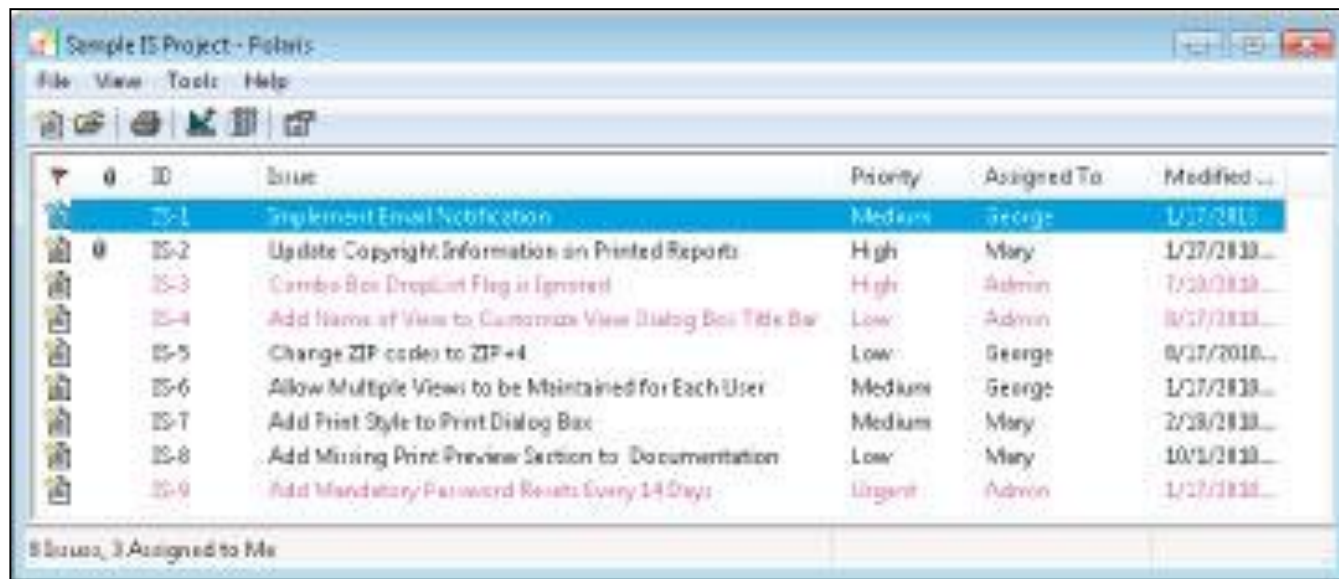
Answers:

1) b; 2) True; 3) flowchart

# Tools for Facilitating Program Development

- Application Lifecycle Management (ALM): Creating and managing an application during its entire lifecycle, from design through retirement
  - Tools include:
    - Requirements management: Keeping track of and managing the program requirements as they are defined and then modified
    - Configuration management: Keeping track of the progress of a program development project
    - Issue tracking: Recording issues such as bugs or other problems that arise during development or after the system is in place

# Tools for Facilitating Program Development



ID	Issue	Priority	Assigned To	Modified
IS-1	Implement Email Notification	Medium	George	1/17/2010
IS-2	Update Copyright Information on Printed Reports	High	Mary	1/17/2010
IS-3	Combo Box DropDownList Flag is Ignored	High	Admin	7/19/2010
IS-4	Add Name of View to Customize View Dialog Box Title Bar	Low	Admin	8/17/2010
IS-5	Change ZIP codes to ZIP+4	Low	George	8/17/2010
IS-6	Allow Multiple Views to be Maintained for Each User	Medium	George	1/17/2010
IS-7	Add Print Style to Print Dialog Box	Medium	Mary	2/19/2010
IS-8	Add Missing Print Preview Section to Documentation	Low	Mary	10/15/2010
IS-9	Add Mandatory Password Resets Every 14 Days	Urgent	Admin	1/17/2010

9 Issues, 3 Assigned to Me

**FIGURE 13-14**

**Issue tracking**

**software.** Allows you to track issues during the development and life of an application.

# Tools for Facilitating Program Development

- Application generator: Software program that helps programmers develop software
  - Macro recorders: Record and play back a series of keystrokes
  - Report and form generators: Tools that enable individuals to prepare reports and forms quickly

**FIGURE 13-15**  
**Form generators.**  
The database form generator shown here is used to create input screens for a database application.

The screenshot displays a software interface for creating a database form. The title bar at the top reads "Form Builder" and the window title is "= Nz([Contact, Name]) 'Untitled'". Below the title bar, there's a menu bar with options like "File", "Edit", "Format", "Tools", "Window", and "Help". The main workspace is divided into two panes. The left pane, titled "General", contains a list of fields with their corresponding data types: "Company" (Text), "First Name" (Text), "Last Name" (Text), "DOB" (Date), "Phone Number" (Text), "Business Phone" (Text), "Home Phone" (Text), "Fax Number" (Text), "Address" (Text), "City" (Text), and "State/Province" (Text). The right pane, titled "Fields", shows a list of fields with their corresponding data types: "Company" (Text), "First Name" (Text), "Last Name" (Text), "DOB" (Date), "Phone Number" (Text), "Business Phone" (Text), "Home Phone" (Text), "Fax Number" (Text), "Address" (Text), "City" (Text), and "State/Province" (Text). The interface is designed to allow users to drag and drop fields from the left pane to the right pane to build a form.

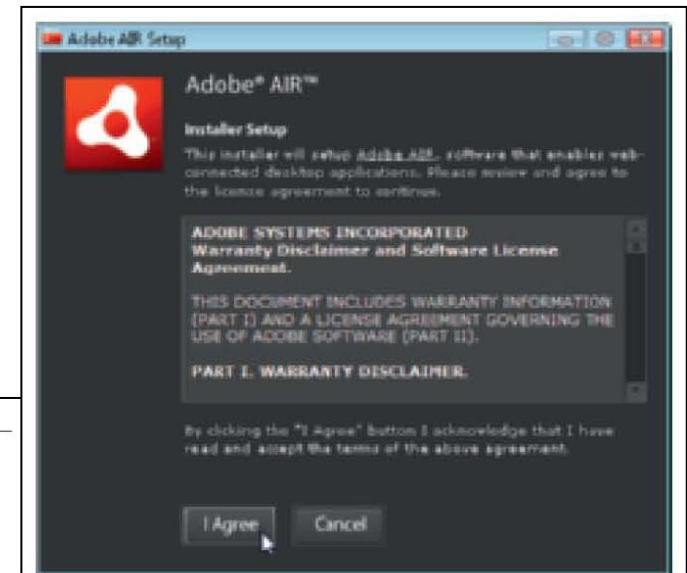
# Tools for Facilitating Program Development

- Device software development tools: Assist with developing embedded software to be used on devices, such as cars, ATM machines, consumer devices, etc
- Software development kits (SDKs): Designed for a particular platform; enables programmers to develop applications more quickly and easily
  - Released by hardware or software companies
  - e.g. iPhone SDK
- Application Program Interfaces (APIs): Help applications interface with a particular operating system
  - Often used in conjunction with Web sites

# Tools for Facilitating Program Development

- Rich Internet Application (RIA): Web-based applications that work like installed software programs
  - Desktop RIA can access local files and used without an Internet connection
  - Web-based RIAs are common
  - Tools to develop RIAs
    - Adobe AIR

**FIGURE 13-16**  
Adobe AIR must be installed on a computer in order to run AIR applications.



## Quick Quiz

1. Which of the following is not an Application Lifecycle Management (ALM) tool?
  - a. Requirements definition software
  - b. Code generator
  - c. Application program interface (API)
2. True or False: A software development kit (SDK) is designed for a particular platform and allows programmers to develop applications quickly for that platform.
3. A(n) \_\_\_\_\_ is often used to create the forms or input screens used to input data into a program or database.

Answers:

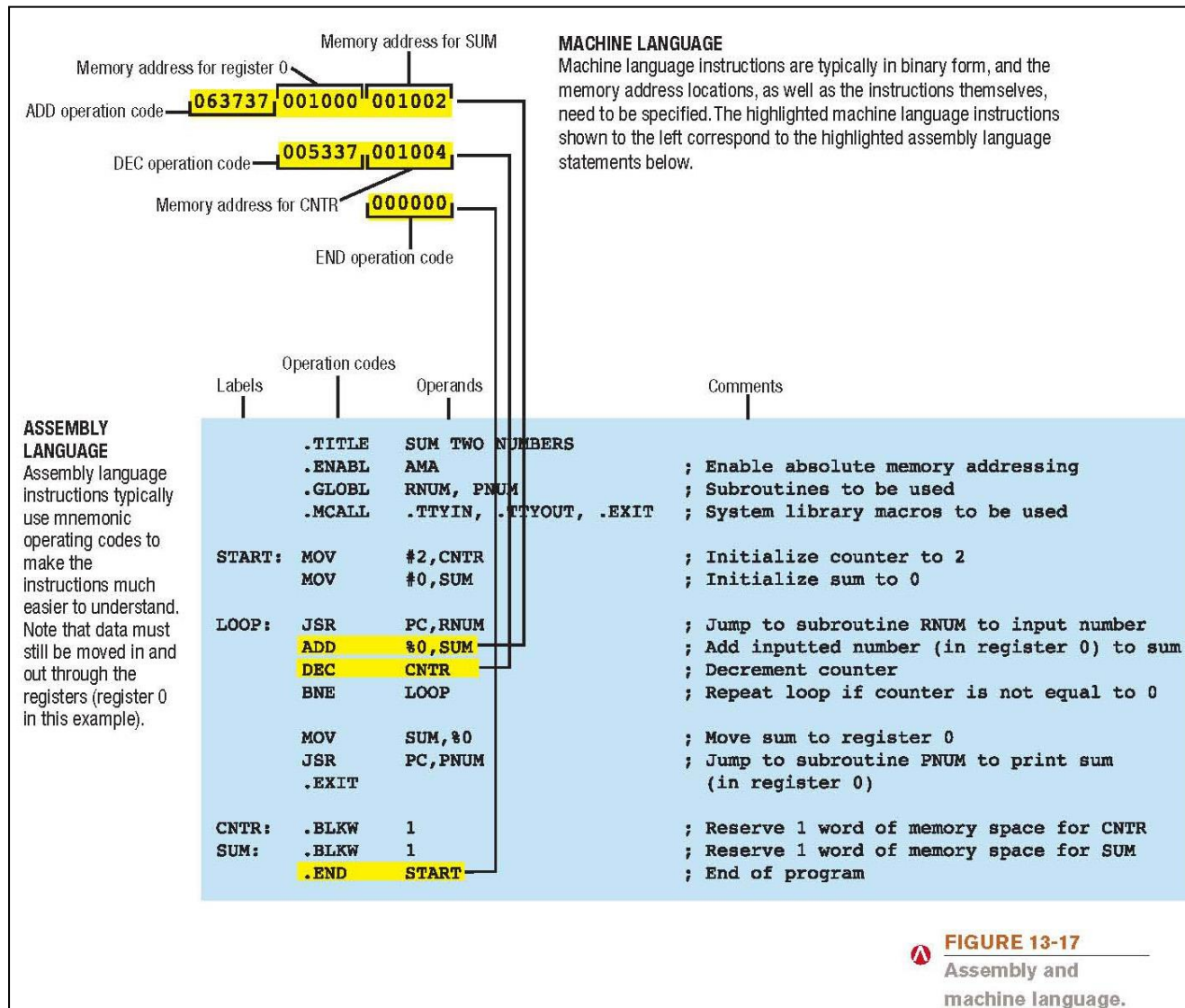
- 1) c; 2) True; 3) form generator

# Programming Languages

- Programming language: A set of rules, words, symbols, and codes used to write computer programs
  - To write a program, you need appropriate software for the programming language you will be using
- Categories of programming languages
  - Low-level languages: Difficult to code in; machine dependent
    - Machine language: 1s and 0s
    - Assembly language: Includes some names and other symbols to replace some of the 1s and 0s in machine language



# Programming Languages



**FIGURE 13-17**  
Assembly and machine language.

# Programming Languages

- High-level languages: Closer to natural languages
  - Machine independent
  - Includes 3GLs (FORTRAN, BASIC, COBOL, C, etc.) and object-oriented languages (Visual Basic, C#, Python, Java, etc.)
  - Visual or graphical languages: Use graphical interface to create programs
- Fourth-generation languages (4GLs): Even closer to natural languages and easier to work with than high-level
  - Declarative rather than procedural
  - Includes structured query language (SQL) used with databases

# Common Programming Languages

- **FORTRAN:** High-level programming language used for mathematical, scientific, and engineering applications
  - Efficient for math, engineering and scientific applications
  - Still used today for high-performance computing tasks (weather forecast)

**FIGURE 13-19**  
The adding-two-numbers program written in FORTRAN.

Comments are preceded by an asterisk or a C.

```
REAL SUM, CNTR, NUM
*
* INITIALIZE VARIABLES
  SUM = 0
*
* INPUT NUMBER, ADD IT TO THE SUM, AND THEN
* REPEAT UNTIL TWO NUMBERS HAVE BEEN ENTERED
  DO 10 CNTR = 1, 2
    WRITE(*,*) 'Enter number'
    READ(*,*) NUM
    SUM = SUM + NUM
  10 CONTINUE
*
* PRINT THE SUM
  WRITE(*,*) 'SUM IS ', SUM
*
  END
```

Program statements can be numbered in order to control loops and other types of branching.

# Common Programming Languages

- COBOL: Designed for business transaction processing
  - Makes extensive use of modules and submodules
  - Being phased out in many organizations
  - Evolving (COBOL.NET)



**FIGURE 13-20**

The adding-two-numbers program written in COBOL.

Comments are preceded by an asterisk.

Most COBOL programs use a number of modules to break the program into manageable pieces. These submodules are called from the main control module using these statements.

Three submodules are used in this program.

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  RESULT      PIC 9(3) VALUE ZERO.
01  CNTR        PIC 9(1) VALUE ZERO.
01  NUM         PIC 9(2) VALUE ZERO.

*****
PROCEDURE DIVISION.
*****
    PERFORM InitVariables
    PERFORM GetNumber UNTIL CNTR = 2
    PERFORM PrintSum
    STOP RUN.

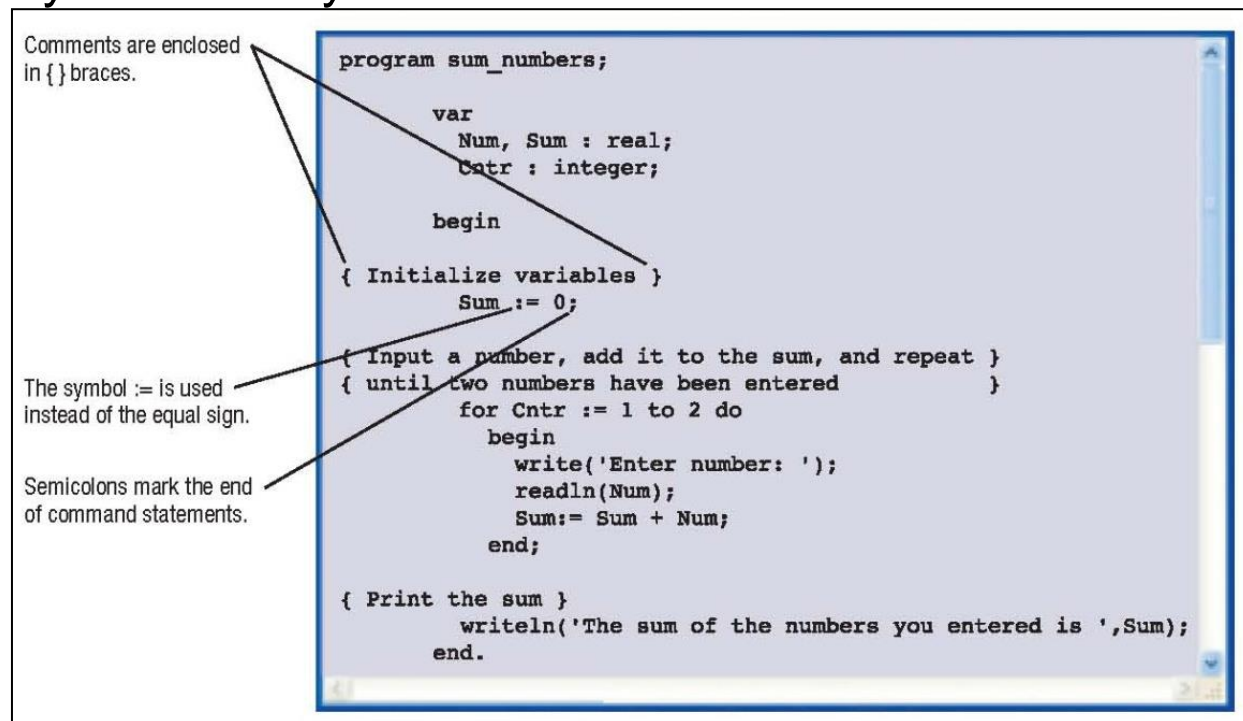
*****
InitVariables.
*****
* This module initializes the RESULT and CNTR variables to 0.
  MOVE 0 TO RESULT
  MOVE 0 TO CNTR.
*End of InitVariables

*****
GetNumber.
*****
* This module inputs a number, adds it to the result, and
* increments the counter.
  DISPLAY "Enter Number: " WITH NO ADVANCING
  ACCEPT NUM
  COMPUTE RESULT = RESULT + NUM
  COMPUTE CNTR = CNTR + 1.
*End of GetNumber module.

*****
PrintSum.
*****
* This module prints the final RESULT.
  DISPLAY "The sum of the numbers you entered is " RESULT.
*End of PrintSum module.
```

# Common Programming Languages

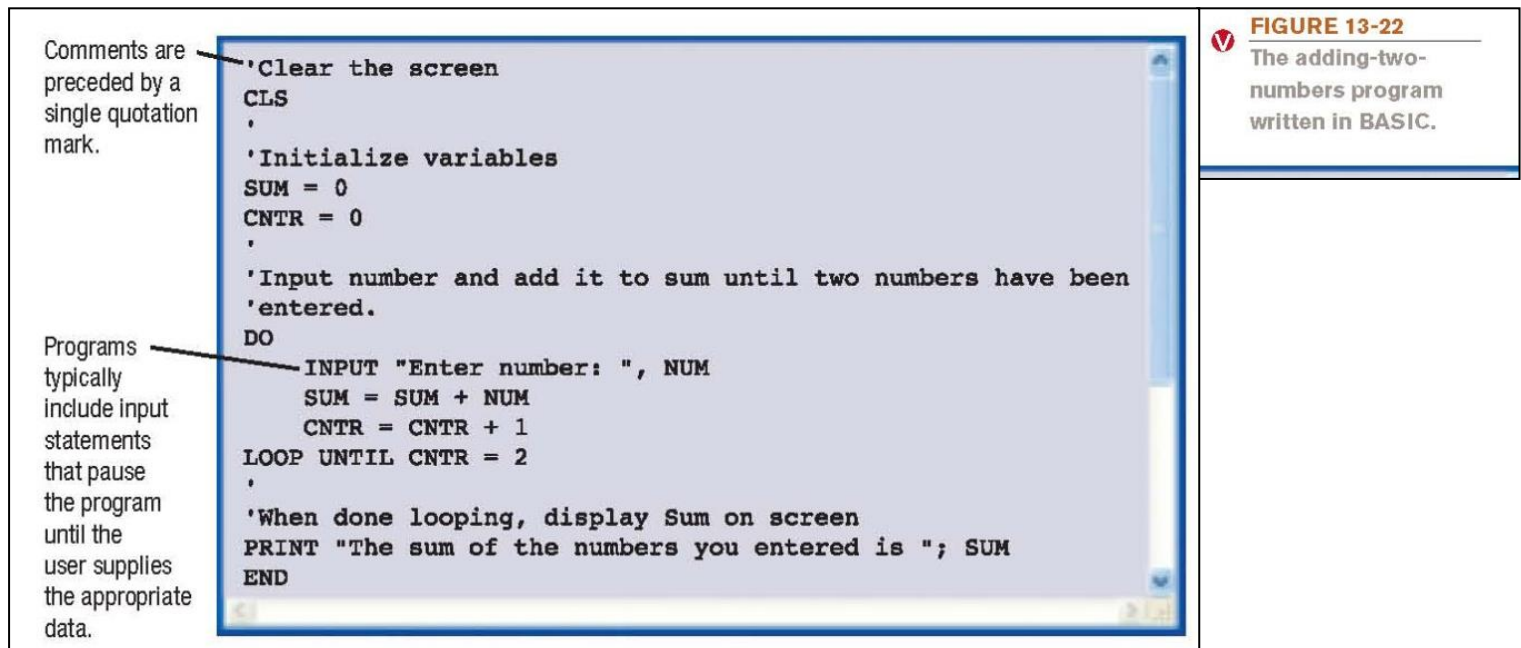
- Pascal: Created as a teaching tool to encourage structured programming
  - Contains a variety of control structures used to manipulate modules systematically



**FIGURE 13-21**  
The adding-two-numbers program written in Pascal.

# Common Programming Languages

- BASIC: Easy-to-learn, high-level programming language that was developed to be used by beginning programmers
  - Visual Basic: Object-oriented version of BASIC; uses a visual environment

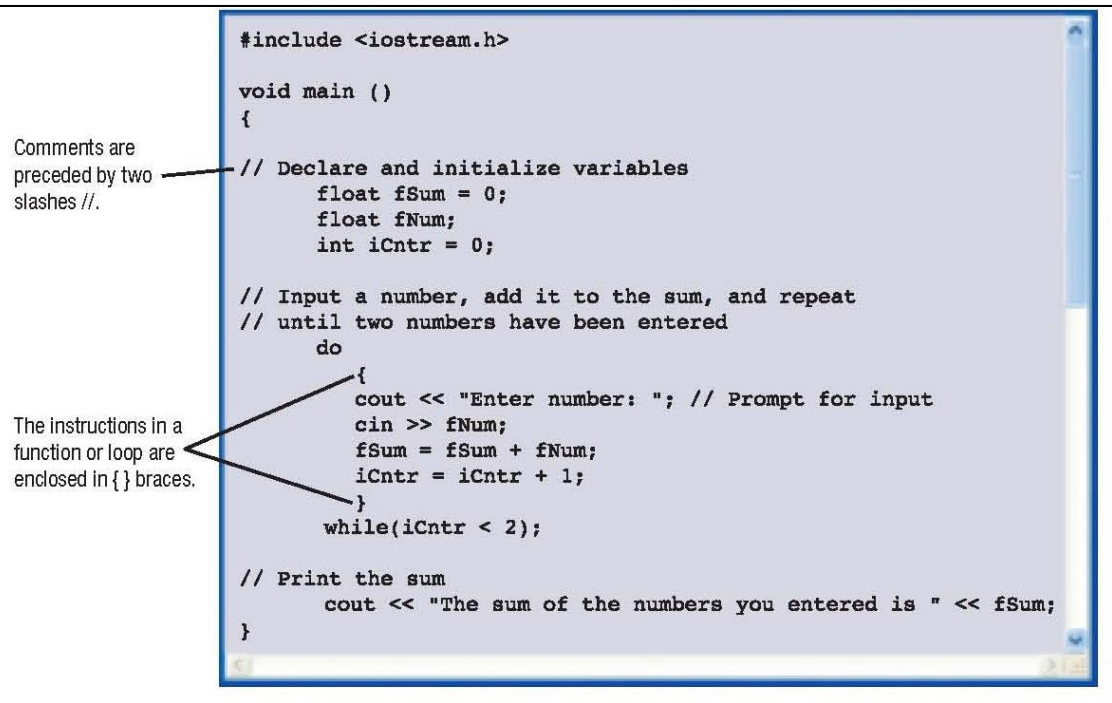




# Common Programming Languages

- C: Designed for system programming
- C++: Object-oriented versions of C
- C#: Used for Web applications
- Objective-C: For iPhone and other Apple applications

**FIGURE 13-23**  
The adding-two-numbers program written in C++.



The image shows a screenshot of a C++ program in a code editor. The code is for a program that adds two numbers. It includes a header file, a main function, variable declarations, input handling, a loop for adding numbers, and output. Two annotations with arrows point to specific parts of the code: one points to a comment line, and the other points to a block of code enclosed in curly braces.

```
#include <iostream.h>

void main ()
{
    // Declare and initialize variables
    float fSum = 0;
    float fNum;
    int iCntr = 0;

    // Input a number, add it to the sum, and repeat
    // until two numbers have been entered
    do
    {
        cout << "Enter number: "; // Prompt for input
        cin >> fNum;
        fSum = fSum + fNum;
        iCntr = iCntr + 1;
    }
    while(iCntr < 2);

    // Print the sum
    cout << "The sum of the numbers you entered is " << fSum;
}
```

Comments are preceded by two slashes //.

The instructions in a function or loop are enclosed in {} braces.

# Common Programming Languages

- Java: High-level, object-oriented programming language frequently used for Web-based applications
  - Java programs are compiled into bytecode
  - Can run on any computer that includes Java Virtual Machine (Java VM)
  - Can be used to write Java applets
    - Scroll text on Web page, games, calculators, etc
  - Is one of the most popular programming languages today



# Common Programming Languages

**FIGURE 13-24**  
The adding-two-numbers program written in Java.

The java.io package will handle the user input; \* indicates all classes will be available.

Comments within the code are preceded by two slashes //.

The attribute *out* and *println* method in the *System* class of the java.io package is used to output the results.

```
import java.io.*;
public class AddTwo {
    public static void main(String[] args) throws IOException {
        BufferedReader stdin =
            new BufferedReader ( new InputStreamReader( System.in ) );
        String inData;
        int iSum = 0;
        int iNum = 0;
        int iCntr = 0;

        // Input a number, add it to the sum, and repeat
        // until two numbers have been entered
        do
        {
            System.out.println("Enter number: ");
            inData = stdin.readLine();           // get number in character form
            iNum = Integer.parseInt( inData );    // convert inData to integer
            iSum = iSum + iNum;
            iCntr = iCntr + 1;
        }
        while (iCntr < 2);

        // Print the sum
        System.out.println("The sum of the numbers you entered is " + iSum);
    }
}
```

# Common Programming Languages

- Python: Open-source, dynamic, object-oriented language that can be used to develop a variety of applications
  - Gaming, scientific, database, and Web applications
  - Only recently gaining a following

Comments are preceded  
by a pound symbol #.

```
# Initialize variable
total = 0.0

# Input a number, add it to the total, and repeat
# until two numbers have been entered
for iteration in range(2):
    text = raw_input("Enter number: ")
    total = total + float(text)

# Print the sum
print "The sum of the numbers you entered is", total
```

The indented statements  
in this For statement will  
be executed two times.



**FIGURE 13-25**

The adding-two-  
numbers program  
written in Python.

# Common Programming Languages

- Ruby: Open-source, object-oriented language that can be used to create general-purpose or Web applications
  - Uses a syntax that is fairly easy to read and write, allowing programmers to create database-driven Web applications easily and quickly

## Quick Quiz

1. An example of a high-level programming language is \_\_\_\_\_.
  - a. Pascal
  - b. Assembly language
  - c. Machine language
2. True or False: Visual Basic is an object-oriented version of COBOL.
3. Java applets are small programs written in the \_\_\_\_\_ programming language.

Answers:

1) a; 2) False; 3) Java

# Summary

- Approaches to Program Design and Development
- The Program Development Life Cycle (PDLC)
- Tools for Facilitating Program Development
- Programming Languages