Chapter 13

Programming

Part 1: Program Development

Understanding Computers: Today and Tomorrow,
15th Edition

Hello :-) I am Omar!

GOTO Statement

```
10 Input A
15 Input B
20 B = A + 10
30 IF B > 12 GOTO 60
40 C = B / 3
50 IF C < 24 GOTO 10
60 Write C
70 IF Write Failed GOTO 15
80 Input D
```

http://www.yitsplace.com/Programming/images/SW_str2.gif

Long time ago.. Programs were written as one large set of instructions that sent control to different parts of the program as needed to perform actions in the proper order.

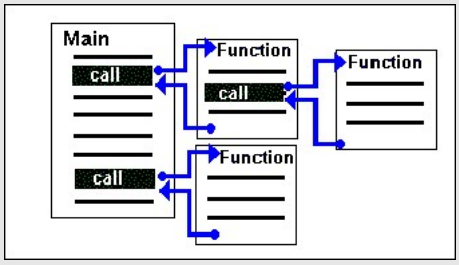
GOTO Statement

```
10 Input A
15 Input B
20 B = A + 10
30 IF B > 12 GOTO 60
40 C = B / 3
50 IF C < 24 GOTO 10
60 Write C
70 IF Write Failed GOTO 15
80 Input D
```

http://www.yitsplace.com/Programming/images/SW_str2.gif

This results in Spaghetti code.

How to solve the spaghetti mess?



http://coronet.iicm.edu/sail/scripts/lesson01.htm

Procedural programming

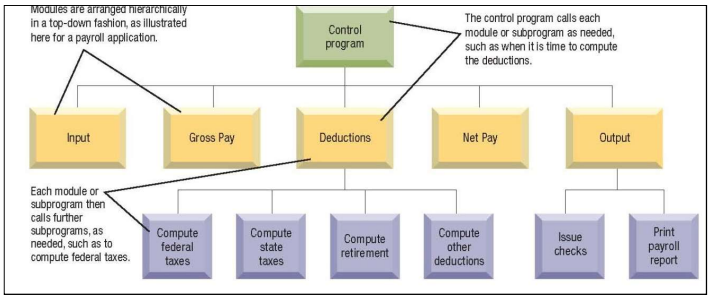
Approaches to Program Design and Development

- **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)

Structured Programming

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

FIGURE 12-1 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:** A template which define a collection of data and actions on that data
  - **Instance:** An object created from a class
  - **Attributes:** Data about the state of an object
  - **Methods:** Perform actions on an object
  - Possible to have zero of more objects created from a class (each with their own individualized data values)

The Program Development Life Cycle (PDLC)

- **Program development:** the process of creating programs
- **Program development life cycle (PDLC):** the steps involved with creating programs

FIGURE 13-3
The program development life cycle (PDLC).

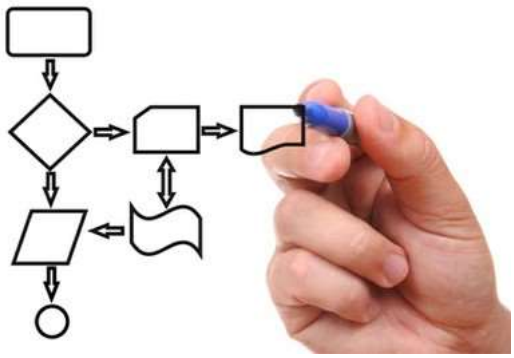
1. Problem Analysis

https://assets.entrepreneur.com/content/16x9/822/20150512050128-shutterstock-124904123.jpeg

1. Problem Analysis

- **Problem analysis: t**he step in the program development life cycle in which the *problem* is considered and the program specifications are developed
  - Specifications developed during the PDLC are reviewed by the systems analyst and the **programmer**
  - Goal: to understand the functions the software must perform
- ***Documentation (outcome)****:* Program specifications (what it does, timetable, programming language to be used, etc.)

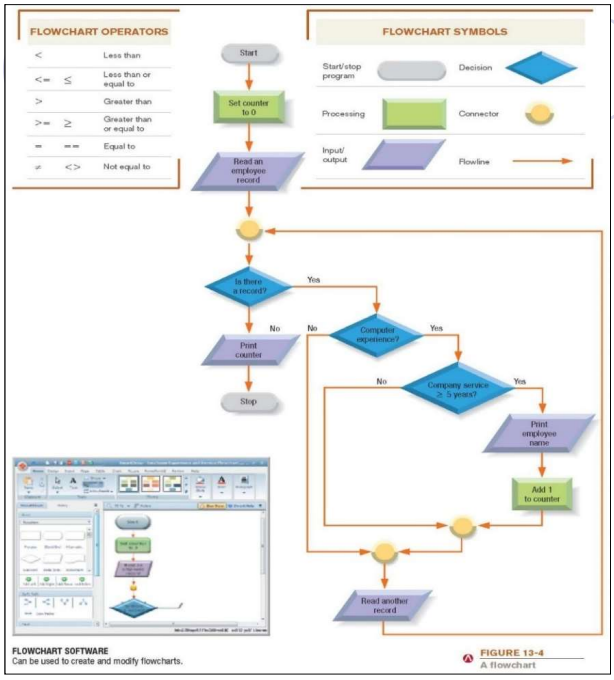
2. Program Design



2. Program Design

- **Program design:** the step in the program development life cycle in which the program specifications are expanded into a complete design of the *new program*
- Program design tools
  - **Structure charts** (depict the overall organization of a program)
  - **Program flowcharts** (show graphically step-by-step how a computer program will process data

Flowcharts



FLOWCHART SOFTWARE
Can be used to create and modify flowcharts.

FIGURE 13-4
A flowchart

Program Design, *Cont'd*

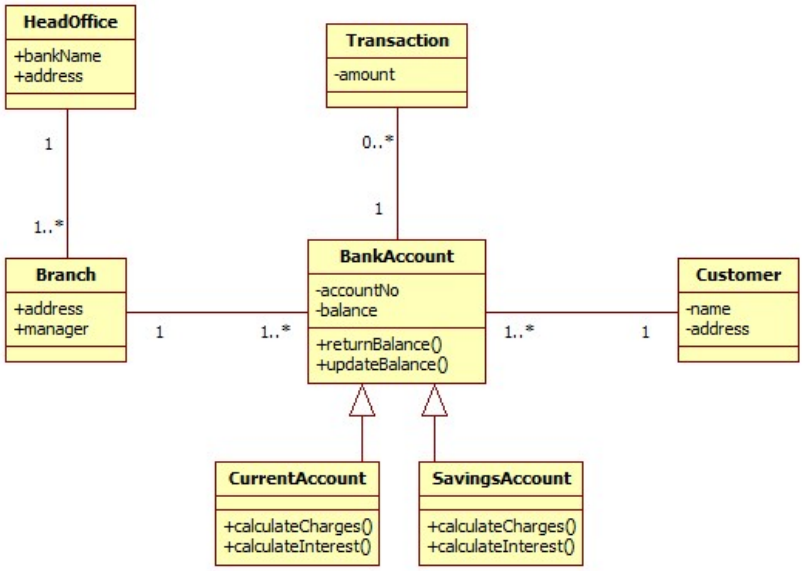
- Program design tools, *cont'd*
  - **Pseudocode:** uses English-like statements to outline the logic of a program

```
GET burger order
IF (want fries = yes) THEN
  Order fries
ENDIF
IF (want drink = yes) THEN
  Order drink
ENDIF
PAY cashier
GET order
```


Program Design, *Cont'd*

- Program design tools, *cont'd*
  - **Unified Modeling Language (UML) Models**
    - Often used when designing an object-oriented program
    - Each object is identified along with its corresponding class, class properties, and variables

Unified Modeling Language (UML)



https://www.technologyuk.net/computing/uml/class-diagrams.shtml

Program Design, *Cont'd*

- **Control structure:** a pattern for controlling the flow of logic in a computer program
  - *Sequence control structure* (series of procedures that follow one another)
  - *Selection control structure*
    - *If-then-else*
    - *Case*
  - *Repetition control structure*
    - *Do-while*
    - *Do-until*

Control Structures

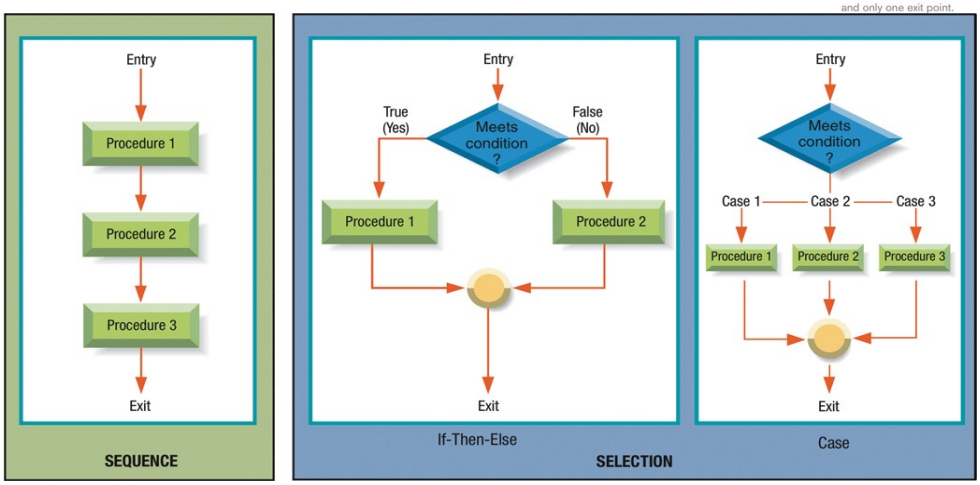


FIGURE 13-8 **Examples of the three fundamental control structures of structured programming.** Note that each structure has only one entry point and only one exit point.

Control Structures, *Cont'd*

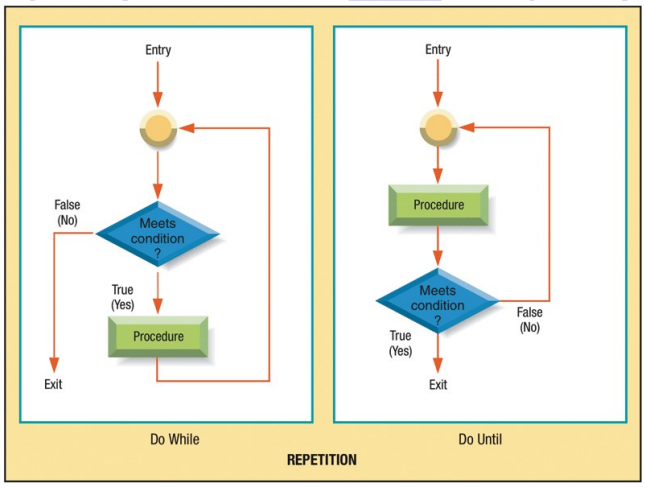


FIGURE 13-8 **Examples of the three fundamental control structures of structured programming.** Note that each structure has only one entry point and only one exit point.

Good Program Design

DIRECTIONS FOR HUMAN	DIRECTIONS FOR COMPUTER
1. Please make me a piece of toast with margarine.	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 onto a plate. 6. Open the silverware drawer, take out a knife, then close the silverware drawer. 7. Open the refrigerator, remove the margarine, and 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, and shut the refrigerator.

FIGURE 13-8 **Writing instructions for a computer versus a person.** A computer requires step-by-step instructions.

Program Design, *Cont'd*

- Good program design = essential
  - Saves time
  - Creates better programs
- Good program design principles:
  - Be specific (all things the program must do need to be specified
  - No logic errors (should trace flowchart to check logic)

- ***Documentation (outcome)***: Design specifications (expressed using flowcharts, pseudocode, structure charts, UML models)

3. Program Coding

- **Coding:** the process of writing the programming language statements to create a computer program
  - Coding creates source code
- 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

3. Program Coding

http://media02.hongkiat.com/programming-myth/programmer.jpg

Program Coding, *Cont'd*

- *Coding standards:* a list of rules designed to standardize programming styles
  - Make programs more universally 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
- ***Documentation (outcome)****:* Documented source code (including comments)

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.

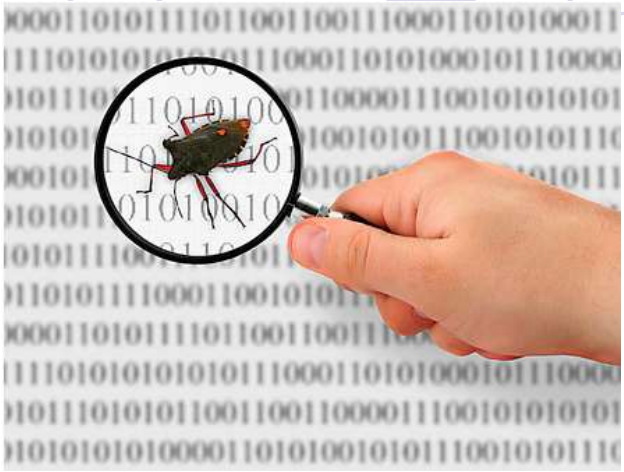
Comments at the top of a program should identify the name and author of the program, date written and last modified, purpose of the program, and variables used in the program.

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.

4. Program Debugging and Testing



http://www.tonex.com/wp-content/uploads/software-testing-company-28.png

4. Program Debugging and Testing

- Debugging: the process of ensuring a program is free of errors (*bugs*)
- Before they can be debugged, coded programs need to be translated into executable code
  - **Source code:** coded program before it is compiled
    - What the programmer sees
  - **Object code:** machine language version of a program
    - What is executed on a computer

Program Debugging and Testing, *Cont'd*

- **Language translator:** program that converts source code to machine language
- 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. Assembly language is used by professional programmers and are associated with a specific computer architecture, such as specific Windows or Mac computers.

Program Debugging and Testing, *Cont'd*

- Preliminary debugging: Finds initial errors
- **Compiler errors**: Program does not run
  - Typically **Syntax errors**: When the programmer has not followed the rules of the programming language
  - Much easier to find and fix
- **Run-time error:** Error that occurs while the program is running (or has run)
  - **Logic (or Semantic) errors**: Program will often run but produces incorrect results
    - Use a '+' sign instead of a '-' sign in a calculation
  - Some run-time errors are so grievous that program crashes

Preliminary Debugging

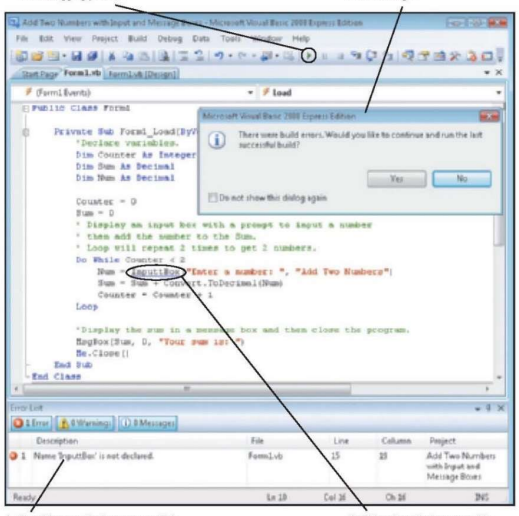


FIGURE 13-12
Syntax errors. Syntax errors occur when the syntax (grammar rules) for a program is not followed precisely; they become obvious when compiling a program.

Program Debugging and Testing, *Cont'd*

- 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, test document description of software commands, troubleshooting guide to help with difficulties, etc.)

5. Program Implementation and Maintenance

- 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 for years to come (as hardware changes)
    - Very costly
  - *Documentation*: Amended program package

5. Program Implementation and Maintenance

http://bookkeepersme.com/wp-content/uploads/2014/05/software-maintenance-services-dubai.jpg

Chapter 13

Programming

Part 2: Programming Languages

Programming Languages

Programming Languages

- **Programming language:** a set of rules used to write computer programs
  - Programs are written using a *programming software package* for the selected programming language
- Categories of programming languages
  - **Low-level languages** (difficult to code in; *machine dependent)*
    - **Machine language:** computer instructions consisting of 1s and 0s
    - **Assembly language:** Includes some names and other symbols to replace the 1s and 0s in machine language

Assembly Language

Memory address for SUM

Memory address for R0

ADD operation code — `063737 001000 001002`

DEC operation code — `005337 001004`

Memory address for CNTR — `000000`

END operation code

MACHINE LANGUAGE
Machine language instructions are typically in binary form, and the memory address locations, as well as the instructions themselves, need to be specified. The highlighted machine language instructions shown to the left correspond to some of the assembly language statements listed below.

ABOUT ASSEMBLY AND MACHINE LANGUAGE
Although all programs are converted to machine language before they are executed, virtually no one programs using machine language anymore. Occasionally, a program may be written in assembly language to make it extra efficient.

Labels | Operation codes | Operands | Comments

ASSEMBLY LANGUAGE
Assembly language instructions typically use mnemonic operating codes to make the instructions much easier to understand. Note that data must still be moved in and out through the registers (register 0 in this example).

```
        .TITLE   SUM TWO NUMBERS
        .ENABL   AMA                        ; Enable absolute memory addressing
        .GLOBL   RNUM, PNUM                 ; Subroutines to be used
        .MCALL   .TTYIN, .TTYOUT, .EXIT     ; System library macros to be used

START:  MOV      #2,CNTR                    ; Initialize counter to 2

LOOP:   JSR      PC,RNUM                    ; Jump to subroutine RNUM to input number
        ADD      R0,SUM                     ; Add inputted number (in register 0) to sum
        DEC      CNTR                       ; Decrement counter
        BNE      LOOP                       ; Repeat loop if counter is not equal to 0

        MOV      R0,SUM                     ; Move sum to register 0
        JSR      PC,PNUM                    ; Jump to subroutine PNUM to print sum (in
        .EXIT                                   register 0)

CNTR:   .BLKW    1                          ; Reserve 1 word of memory space for CNTR
SUM:    .BLKW    1                          ; Reserve 1 word of memory space for SUM
        .END     START                      ; End of program
```

FIGURE 13-17
Assembly language 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
  - Y2K scare

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.
```

Understanding Computers: Today and Tomorrow, 15th Edition

41

Common Programming Languages

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

Comments are enclosed in { } braces.

The symbol := is used instead of the equal sign.

Semicolons mark the end of command statements.

```
program sum_numbers;

        var
          Num, Sum : real;
          Cntr : integer;

        begin

{ Initialize variables }
          Sum := 0;

{ Input a number, add it to the sum, and repeat }
{ until two numbers have been entered           }
          for Cntr := 1 to 2 do
            begin
              write('Enter number: ');
              readln(Num);
              Sum:= Sum + Num;
            end;

{ Print the sum }
          writeln('The sum of the numbers you entered is ',Sum);
        end.
```

Understanding Computers: Today and Tomorrow, 15th Edition

42

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

Comments are preceded by a single quotation mark.

Programs typically include input statements that pause the program until the user supplies the appropriate data.

```
'Clear the screen
CLS
'
'Initialize variables
SUM = 0
CNTR = 0
'
'Input number and add it to sum until two numbers have been
'entered.
DO
    INPUT "Enter number: ", NUM
    SUM = SUM + NUM
    CNTR = CNTR + 1
LOOP UNTIL CNTR = 2
'
'When done looping, display Sum on screen
PRINT "The sum of the numbers you entered is "; SUM
END
```


Common Programming Languages

- **C:** Designed for system programming
- **C++:** Object-oriented versions of C
- **C#:** Used for Web applications

Comments are preceded by two slashes //.

The instructions in a function or loop are enclosed in { } 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;
}
```

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

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
  - Easier to learn
  - Gaming, scientific, database, and Web applications
  - Used by organizations such as Google, Yahoo, and NASA

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.