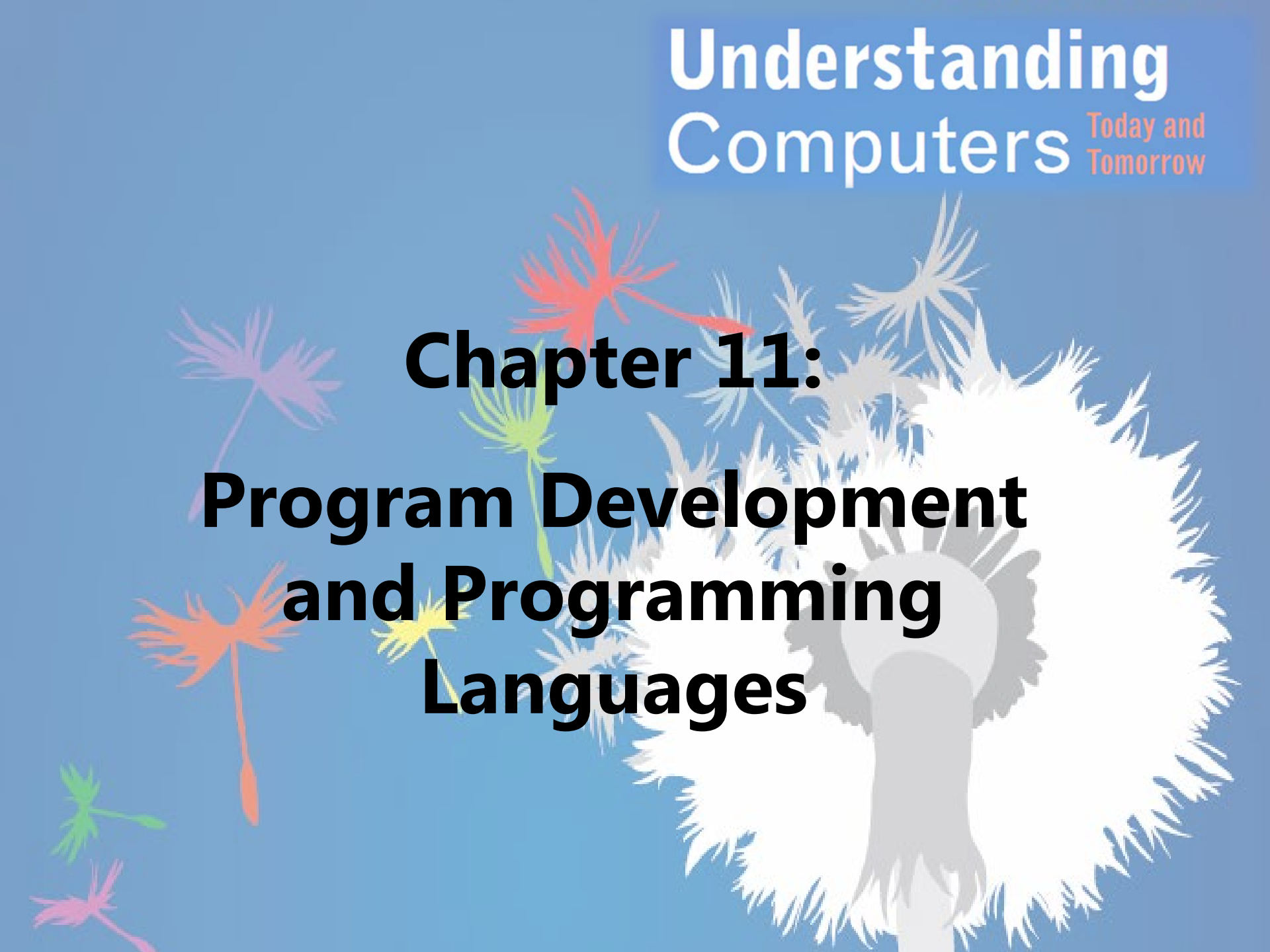


Understanding Computers Today and Tomorrow

Chapter 11: Program Development and Programming Languages





Programming

VonNeumann Architecture

- CPU, Memory, Storage
- Input, processing, output (IPO) cycle (text adds storage => IPOS)
- Stored program => becomes dynamic process in execution
 - Process has resources allocated by OS (CPU, Files, Internet Connections, Printers, etc.)
 - Process memory has Code, Data, Stack segments

Syntax & Semantics

- Syntax - words, symbols, and codes used to write computer programs
- Semantics – rules & logic

Variables => named symbols in memory

recall we must abstract and represent our real world in a computer

Boolean values => truth values (often determined in test)

- $3 > 2$ is true... if $x = 3$... $x > 2$ is true



Programming Language Evolution

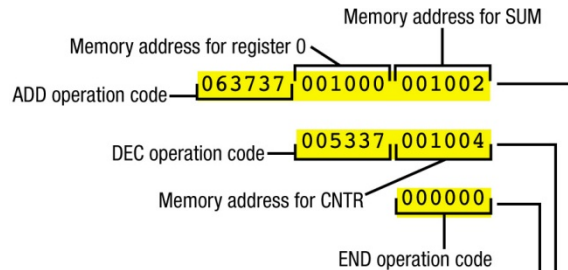
- Generations
 - 1st Gen – Machine Language
 - 2nd Gen – Assembly Language
 - 3rd Gen – High Level
 - 4th Gen – Declarative/SQL*
 - * Note this is speculation



Low Level Languages (1st/2nd Gen)

- Low-Level Languages (earliest programming languages)
 - 1st Gen - Machine language
 - Written at a very low level, just using 1s and 0s
 - 2nd Gen - Assembly language
 - Includes some names and other symbols to replace some of the 1s and 0s in machine language
 - 1 to 1 mapping of assembly instructions to machine instructions
 - Machine dependent
 - » Written for one specific type of computer

Assembly Language



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 the highlighted assembly language statements below.

FIGURE 13-17
Assembly and machine language.

ASSEMBLY LANGUAGE

Assembly language instructions 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).

Labels	Operation codes	Operands	Comments
	.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
	MOV	#0, SUM	; Initialize sum to 0
LOOP:	JSR	PC, RNUM	; Jump to subroutine RNUM to input number
	ADD	%0, 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	SUM, %0	; Move sum to register 0
	JSR	PC, PNUM	; Jump to subroutine PNUM to print sum
	.EXIT		(in 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



High Level Languages

- High-Level/3rd Generation Languages
 - Closer to natural languages
 - Machine independent (portable)
 - * Also visual or graphical languages
 - Use graphical interface to create programs any paradigm
 - Designed for educational/project mgmt. purposes



High Level Programming Paradigms

A paradigm is a model

To write a program, appropriate software for the application/programming language to be used is needed

4 Programming Paradigms =>

- Procedural/Structured (we'll look at these first)
 - Cobol, C, Fortran
- Object Oriented
 - Java, Python, C++/C#
 - * Note OOP has procedural constructs
- Logical/Declarative => Prolog
- Functional => Lisp



Procedural/Structured Control Structures

- Control Structure (Structured Programming)
 - A pattern for controlling the flow of logic in a computer program, module, or method
- 3 Control Structures
 - Sequential execution (default)
 - Selection execution (Boolean decision structure)
 - Repetition/Iteration execution (has Boolean decision structure)
 - repeated actions based on decision structure
- * 4th - Sub-programs (e.g. functions/procedures/methods/etc.)
- See Truth Tables in ciss100.com => Programming (both handouts and video)

Sequential Execution

- The Sequence Control Structure
 - Series of statements that follow one another
 - Default execution path
- Ex. Statements:
 $x = x + 2;$
 $y = x;$

*Note diagram blocks have single entry and exit

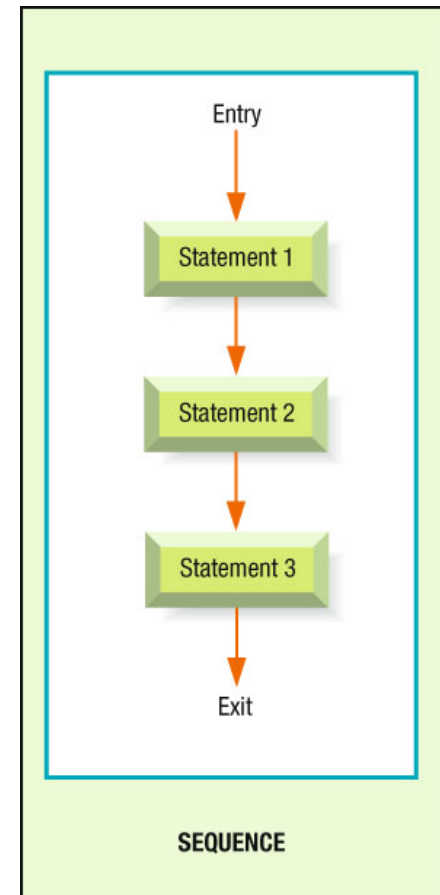


FIGURE 13-7



Selection

- The Selection Control Structure
 - Multiple paths, exit direction depends on result of a certain condition (logical determination of truth value)
 - » If-then-else (Can be just if-then)
 - if ($x > 2$) *($x > 2$ is Boolean and has a truth value)*
 - then perform some action
 - else perform a different action
 - Case control structure (switch)
 - » Allows for as many possible results of the specified condition as needed



Repetition

- Repetition Control Structure (Iteration or Loops)
 - Repeat series of steps based on decision structure
 - » While/For (Boolean test at entry)
 - » While $x > 0$
 - » Loop may never be entered
 - » Do-while/Repeat-Until (test at exit)
 - » While/Until statement is last statement in loop
 - » Loop always executed once
 - » 3 parts of loop
 - » Initialization (outside of loop or in For statement)
 - » Test
 - » Body
 - » Update

Structured Programming Control Structures

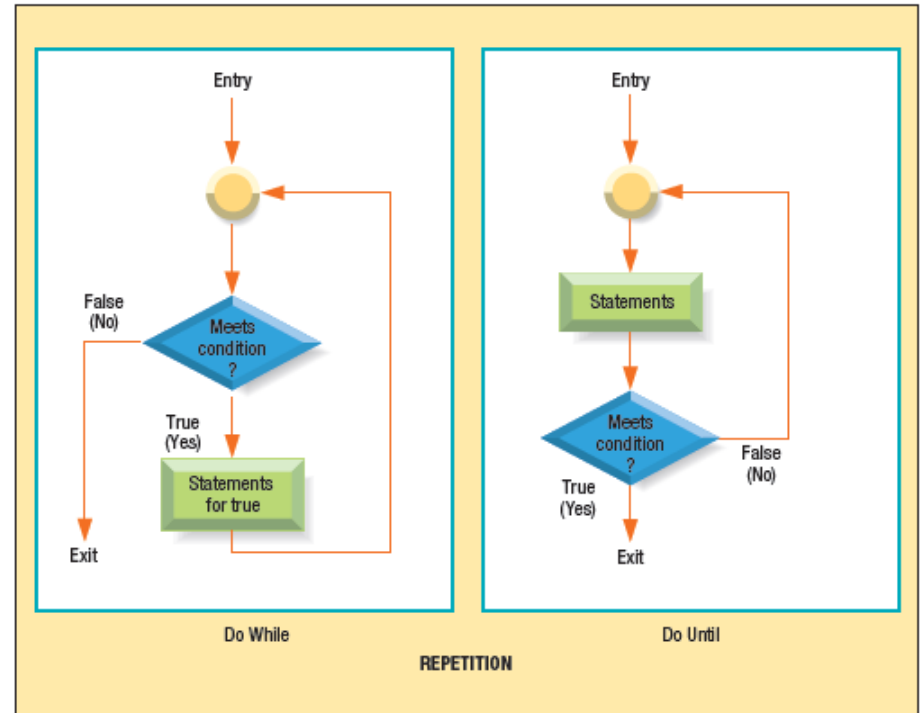
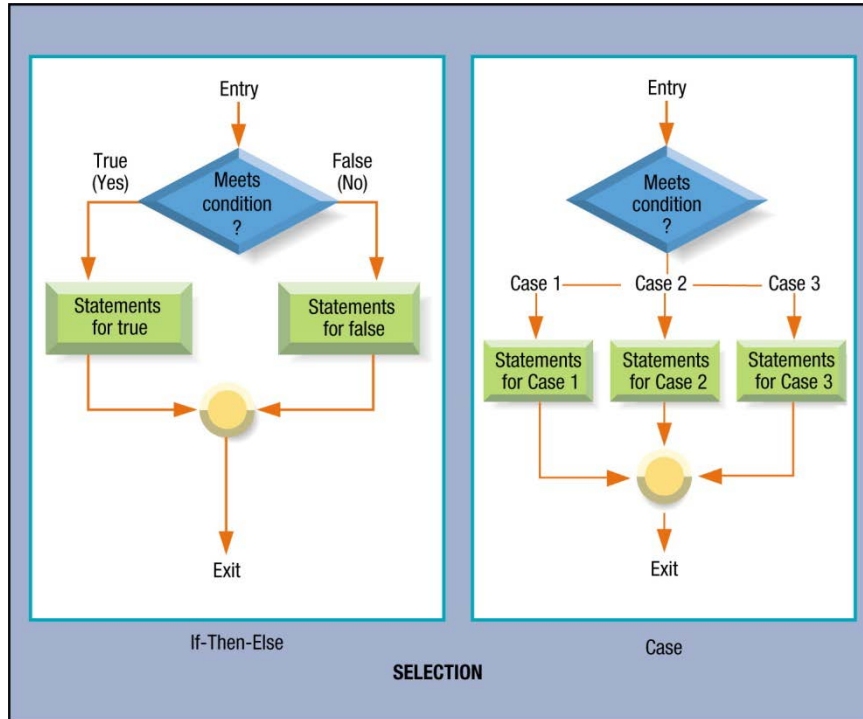


FIGURE 13-7

The three fundamental control structures. Note that each structure has only one entry point and only one exit point.

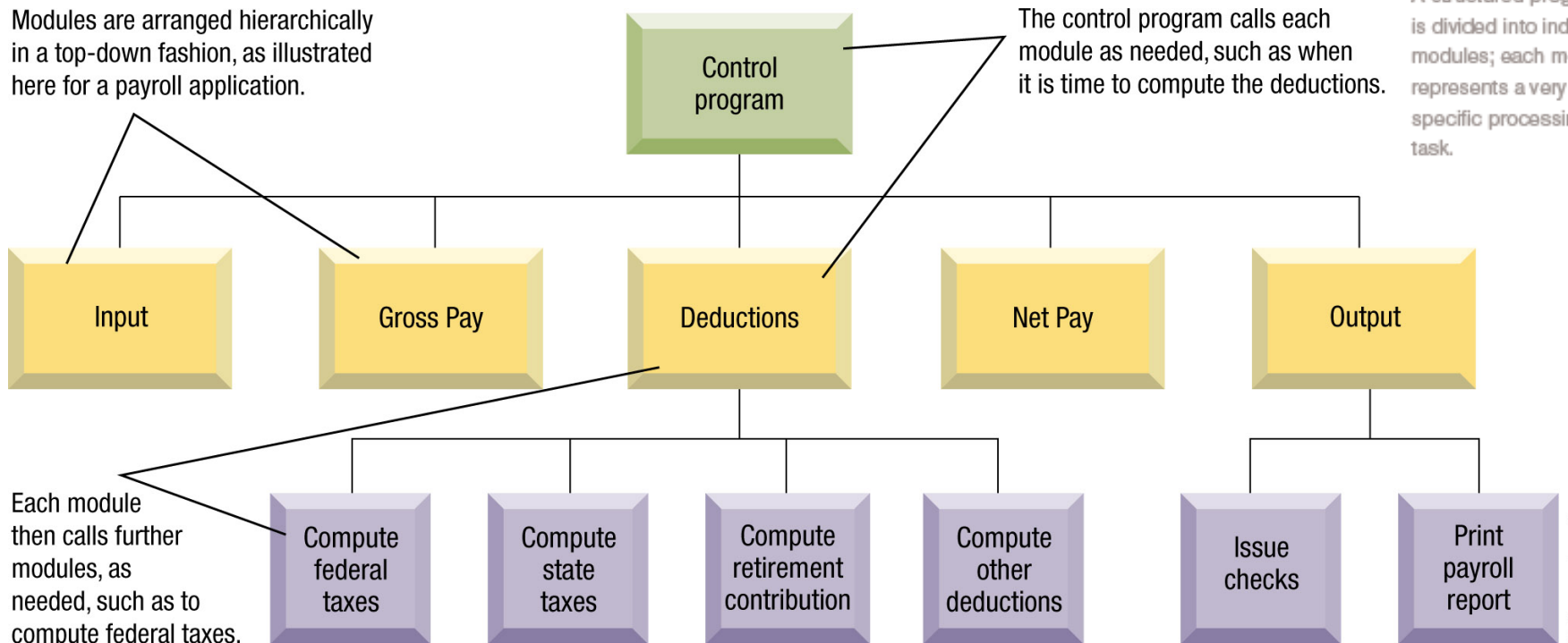


Procedural Programming

- Procedural Programming
 - Design where program separated into small modules/procedures
 - Procedure called by main program or another
 - Allows each procedure to be performed as many times as needed; multiple copies of code not needed
- Top Down
 - Decomposition
 - Stepwise refinement to solve problem
- Bottom Up
 - Library Development
 - Tool kits

Top Down Decomposition

Modules are arranged hierarchically in a top-down fashion, as illustrated here for a payroll application.



The control program calls each module as needed, such as when it is time to compute the deductions.

Each module then calls further modules, as needed, such as to compute federal taxes.

FIGURE 13-1

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



Structured High Level Language Ex => Cobol

– COBOL

- Designed for business transaction processing
- Makes extensive use of modules
- Strength lies in batch processing and its stability
- Programs are lengthy and take a long time to write
- Considered to be outdated by some
- New versions are evolving
 - COBOL.NET

Cobol(3rd Gen)

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.
```

FIGURE 13-20
The adding-two-numbers program written in COBOL



Structured High Level Language Ex => C (C, C++)

- C, C++, and C#
 - C
 - Much closer to assembly language than other high-level languages
 - Designed for system programming
 - C++
 - Object-oriented versions of C
 - Very popular for graphical applications
 - C# (C sharp)
 - Used to create Web applications and XML-based Web services
 - Objective-C:
 - For iPhone and other Apple applications (dated)
 - => Now Swift which can be integrated w/Obj-C

C++

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 iCtr = 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;
        iCtr = iCtr + 1;
    }
    while(iCtr < 2);

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

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



Object Oriented Design

- Object-Oriented Programming (OOP & OOD)
 - Programs consist of a collection of objects that contain data and methods to be used with that data (uses procedural constructs within OOP)
 - Class
 - Group of objects that share some common properties
 - Instance/Object
 - An individual object in a class



Approaches to Design and Development

- Attributes/Data => state of an object
- Behaviors/Methods => Perform actions on an object
- UML below introduced in Sys Analysis chapter

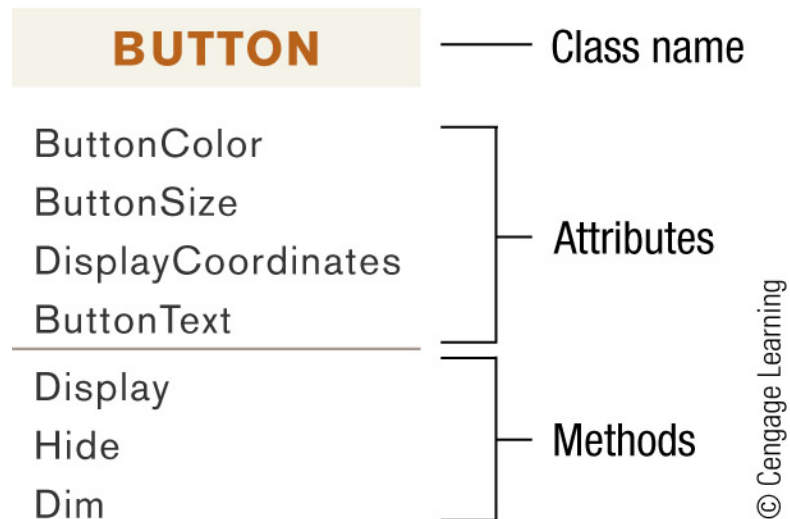


FIGURE 13-2

Button objects. This class diagram illustrates that each object in the Button class has four attributes to hold data about the current state of the button and three methods to react to messages the object receives.



Object Oriented Design

- OOD & OOP
- Encapsulation
 - Bundles data/methods to work on data within class
 - Hides the internal representation/state=> Security
 - Checking Acct Example
- Inheritance
 - Derives & extends parent class to child class
 - Supports code reuse
- Polymorphism
 - Many forms => class can override parents methods



OOD Example => Java

- Java
 - High-level, object-oriented programming language frequently used for Web-based applications
 - Contain Structured components
 - Java programs are compiled into bytecode
 - Can run on any computer that includes Java Virtual Machine (Java VM)
 - Is one of the most popular programming languages today

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 out attribute and println method in the System class of the java.io package are 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);
    }
}
```

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



Programming Languages

- 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

The Program Development Life Cycle (PDLC)

- Program Development (application software development)
 - The process of creating application programs
- Program Development Life Cycle (PDLC)
 - The five phases of program development

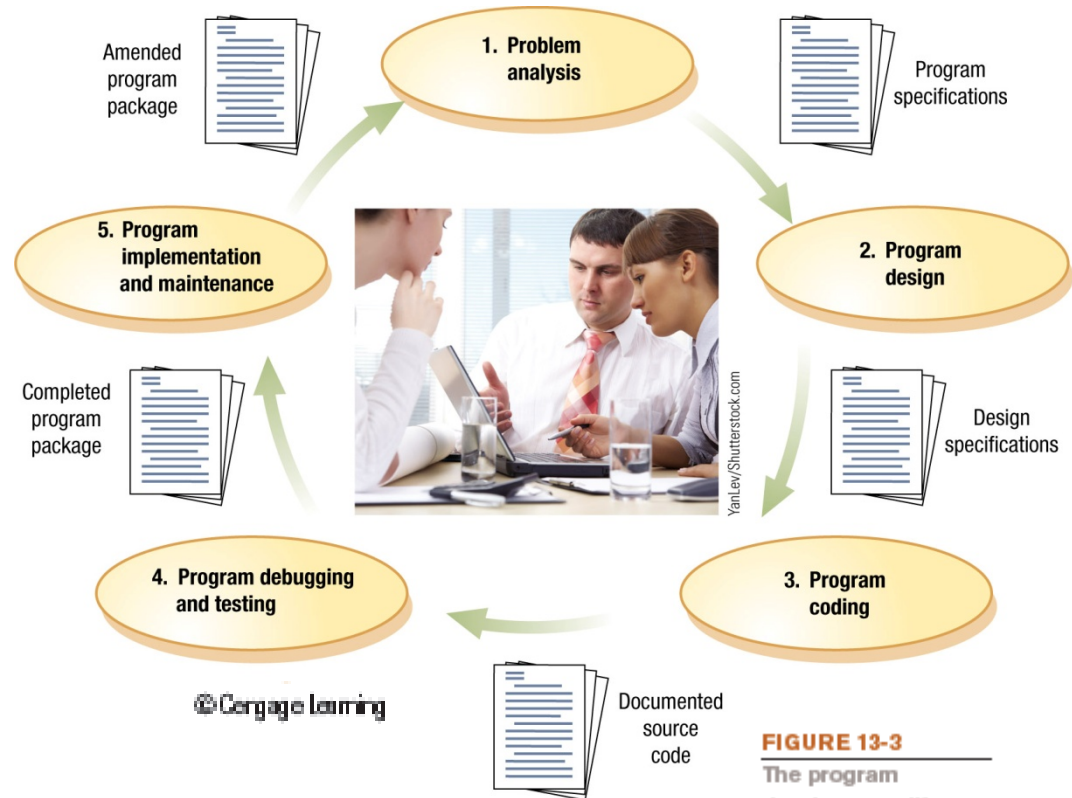


FIGURE 13-3
The program development life cycle (PDLC). Each phase of the program development life cycle produces some type of documentation to pass on to the next phase.



PDLC - Analysis

- 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 is to understand the functions the software must perform
 - Documentation: Program Specifications
 - Result of the first phase of the PDLC outlining what the program must do



PDLC - Design

- Program Design
 - The program specifications are expanded into a complete design of the new program
 - Algorithm for the program is developed
 - Careful planning and design of a computer program are extremely important



PDLC – Design Tools

Program Design Tools

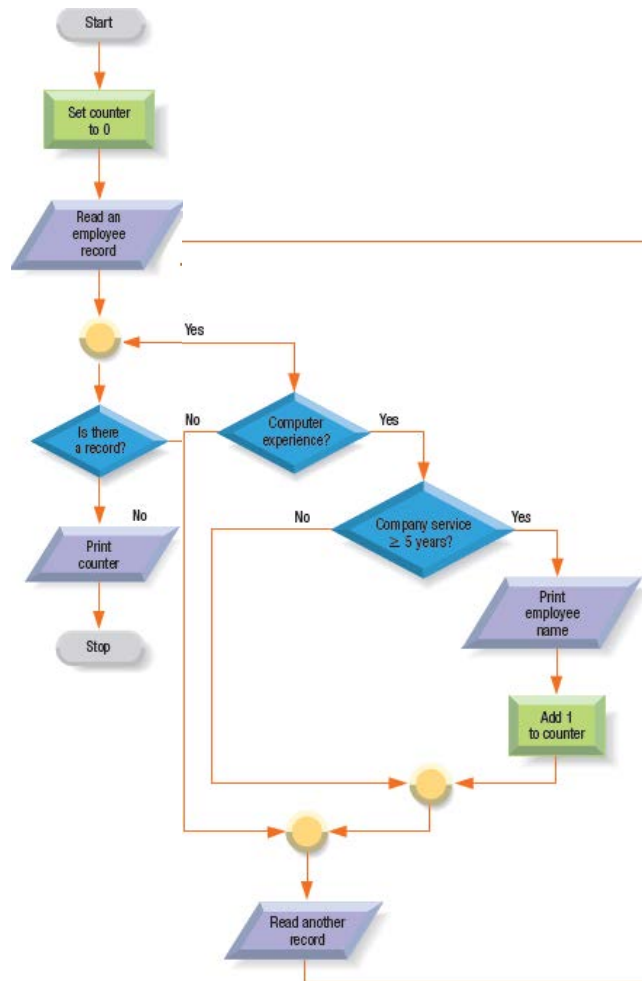
=> diagrams, tables, models and structure/hierarchy charts

- hierarchy chart depict organization of a program
 - => top down design
- 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
- Pseudocode
 - Uses English-like statements to outline the logic of a program rather than the flowchart's graphical symbols
- Object Oriented Design – Unified Modeling Language

PDLC – Design Tools

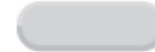
FLOWCHART OPERATORS

<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
=	Equal to
≠	Not equal to



FLOWCHART SYMBOLS

Start/stop program



Decision



Processing



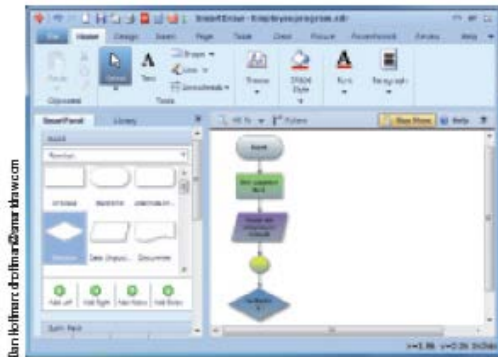
Connector



Input/output



Flowline



FLOWCHART SOFTWARE

Can be used to create and modify flowcharts.

FIGURE 13-4

A flowchart example.



PDLC – Design Tools

```
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
```

FIGURE 13-5

Pseudocode.

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

PDLC – Object Oriented Design (OOD)

- Unified Modeling Language (UML) Models
 - Set of standard notations for creating business models
 - Widely used in object-oriented programs
 - Includes class diagrams and use-case diagrams

CLASS

A group of objects that share the same basic properties. A class diagram defines the attributes and methods that all instances in the class possess.

BICYCLES

TypeofBike
BikeCategory
Size
Color
NumberofGears
CurrentGear
CurrentSpeed

Class Name

Attributes

ChangeGear
ChangeSpeed
Accelerate
Brake
Stop
TurnRight
TurnLeft

Methods

INSTANCES

The specific objects in a class, such as Bike1 and Bike2 in this example.

BIKE1: BICYCLES

TypeofBike = 'male'
BikeCategory = 'road'
Size = 26
Color = 'red'
NumberofGears = 21
CurrentGear = 5
CurrentSpeed = 0

ChangeGear
Change Speed
Accelerate
Brake
Stop
TurnRight
TurnLeft

BIKE2: BICYCLES

TypeofBike = 'child'
BikeCategory = 'mountain'
Size = 20
Color = 'blue'
NumberofGears = 6
CurrentGear = 1
CurrentSpeed = 0

ChangeGear
ChangeSpeed
Accelerate
Brake
Stop
TurnRight
TurnLeft

FIGURE 13-6

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

INHERITANCE

All instances of a class inherit all attributes and methods of the class. The values of the attributes for each instance may be different from other instances.



PDLC – Design Tenets

- Good Program Design
 - Essential
 - Saves time
 - Good Program Design Principles
 - Be Specific
 - » All things the program must do or consider must be specified
 - Follow the One-Entry-Point/One-Exit-Point Rule
 - No Infinite Loops or Logic Errors
 - » Infinite loop is a series of steps that repeat forever



PDLC – Design Testing

- Program Design Testing
 - Design should be tested to ensure logic is correct
 - Desk check
 - Trace tables
- Documentation: Design Specifications
 - Illustrates the program needed to fulfill the program requirements
 - Expressed using structure charts, flowcharts, pseudocode, and UML models
 - Include any test data and results from desk checking



PDLC - Coding

- Program Coding
 - The program code is written using a programming language
 - Choosing a Programming Language
 - Suitability to the application
 - Integration with other programs
 - Standards for the company
 - Programmer availability
 - Portability if being run on multiple platforms
 - Development speed



PDLC - Coding

- The Coding Process
 - 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



PDLC – Coding Standards

- Reusable code
 - Pretested, error-free code segments that can be used over and over again with minor modifications
 - Can greatly reduce development time
- Documentation: Documented Source Code
 - Program coding phase results in the program written in the desired programming language
 - Should include enough comments (internal documentation) so that the source code is easy to understand and update



PDLC – Coding Example

COMMENTS

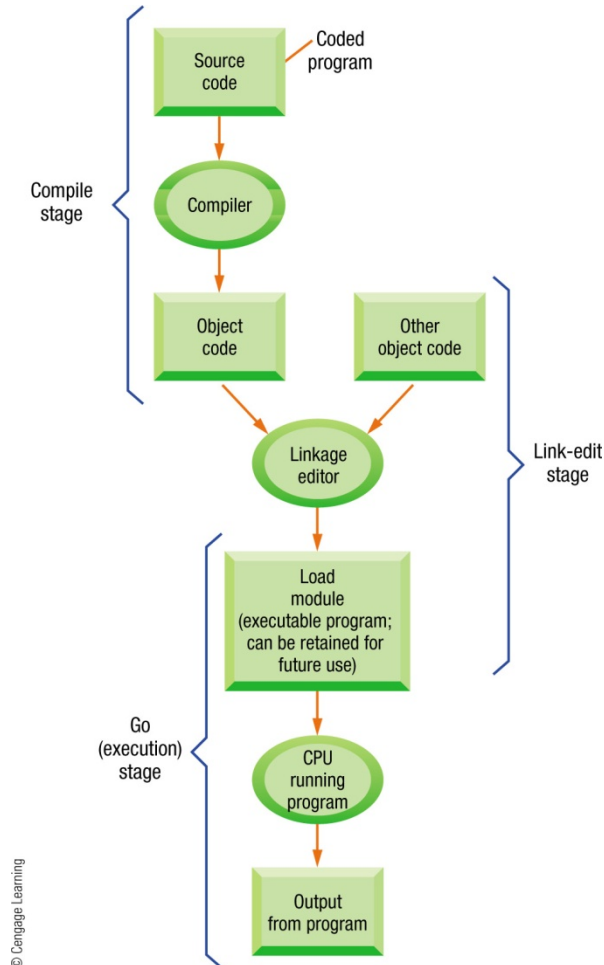
Comments are usually preceded by a specific symbol (such as *, C, ', #, or //); the symbol used depends on the programming language being used. Anything else in a comment line is ignored by the computer.

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  3/12/12                       *
*****
* 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
```

Executable Code Generation



© Cengage Learning

FIGURE 13-11
Compiler and linkage editor. A compiler and a linkage editor convert source code into executable code.



Executable Code Generation

Translate Coded Programs into Executable Code

- Source code => object code the computer can execute
- Assemblers convert assembly language
- Compilers - static
 - Language translator that converts an entire program into machine language before executing it (once)
 - Designed for specific programming languages such as Java or Python
 - Note simple single HL $y = x + 2 \Rightarrow 3$ machine instructions
 - `mov x, r2`
 - `add 2, r2`
 - `mov res, y`
 - Interpreters - dynamic
 - Translates one line of code at one time
 - Translated over and over each time it is run



PDLC Debugging & Testing

- Program Debugging and Testing
 - The process of ensuring a program is free of errors (bugs) and works as it is supposed to
- Preliminary Debugging
 - Compiler and Syntax Errors
 - As programs are compiled or interpreted, errors occur which prevent the program from running properly
 - Syntax errors occur when the programmer has not followed the rules of the programming language



PDLC Debugging & Testing

- Logic errors are errors in the logic of the program
 - Program will run but produce incorrect results
 - Dummy print statements can help locate logic errors and other run time errors
 - IDEs with step debugging useful
- Run Time and Logic Errors
 - Run time errors occur when the program is running



PDLC Debugging & Testing

– Testing

- Occurs after the program appears to be correct to find any additional errors
- Uses good test data—data that is very similar to the actual data that will be used in the program
- Tests conditions that will occur when the program is implemented
- Checks for coding omissions (i.e., product quantity allowed to be < 0)
- Edge testing of boundary conditions



PDLC – Debugging & Testing

- Testing => 2 stages
 - Alpha test—internal on-site test
 - Beta test—outside test
- Testing Documentation: Completed Program Package
 - Copy of the test data, test results, finished program code, and other documentation generated during the testing phase should be added to the program package
 - Developer documentation
 - User documentation



PDLC – Implementation & Maintenance

- Program Implementation and Maintenance
 - 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
 - Program package should be updated to reflect new problems or issues that occur and what changes to the program were necessary



PDLC Tools

- Application Lifecycle Management (ALM) Tools
 - 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



PDLC Tools

- Application Generators
 - Software program that helps programmers develop software
 - Macros
 - Record and play back a series of keystrokes
 - Programmers write them in a macro programming language such as Visual Basic for Applications
 - Report and Form Generators
 - Tools that enable individuals to prepare reports and forms quickly



PDLC Tools

- Device Software Development Tools
 - Assist with developing embedded software to be used on devices, such as cars, ATM machines, and consumer devices
- Software Development Kits (SDKs) and Application Program Interfaces (APIs)
 - Designed for a particular platform
 - Enables programmers to develop applications more quickly and easily
 - Often released by hardware or software companies
 - iOS SDK—allows third party developers to create new applications for iPhone, iPad, iPod Touch



PDLC Tools

- Application Program Interfaces (APIs)
 - Help applications interface with a particular operating system/environment
 - Often used in conjunction with Web sites
- Rich Internet Application (RIA) Tools
 - 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



Agile/Extreme Programming

- Agile software development
 - Goal is to create software rapidly
 - Embraces iterative approach
 - Focuses on building small functional program pieces during the project
 - Includes earlier adaptive software approaches such as RAD (rapid application development) and extreme programming (XP)



Summary

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