CSC511Sem
Introduction to C++

Week 1

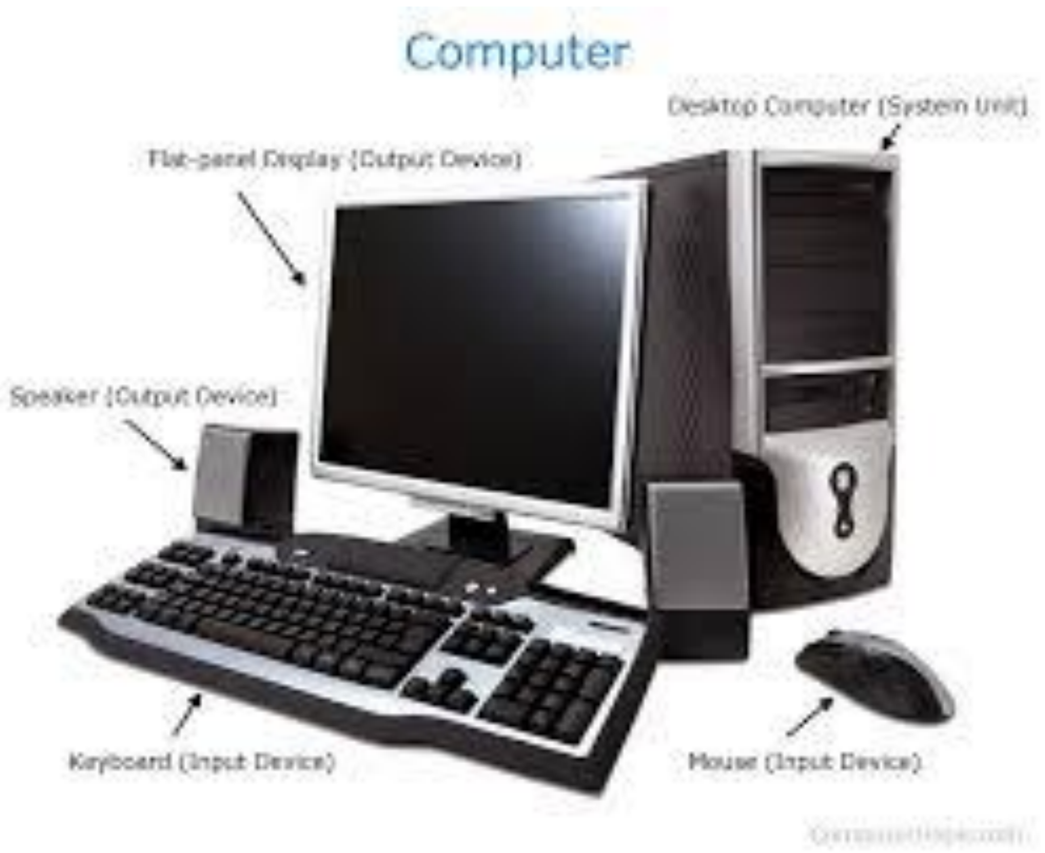
Introduction to Computer Systems

OBJECTIVES

- explain computer systems software, hardware and the Internet
- explain the definition of a program
- explain the program design process
- explain problem solving
- use various design tools to help breakdown problems into manageable programmable bits
- discuss different ways of writing algorithms
- discuss software development phases and steps (Analysis, Design, Coding & testing)

What is a Computer system?

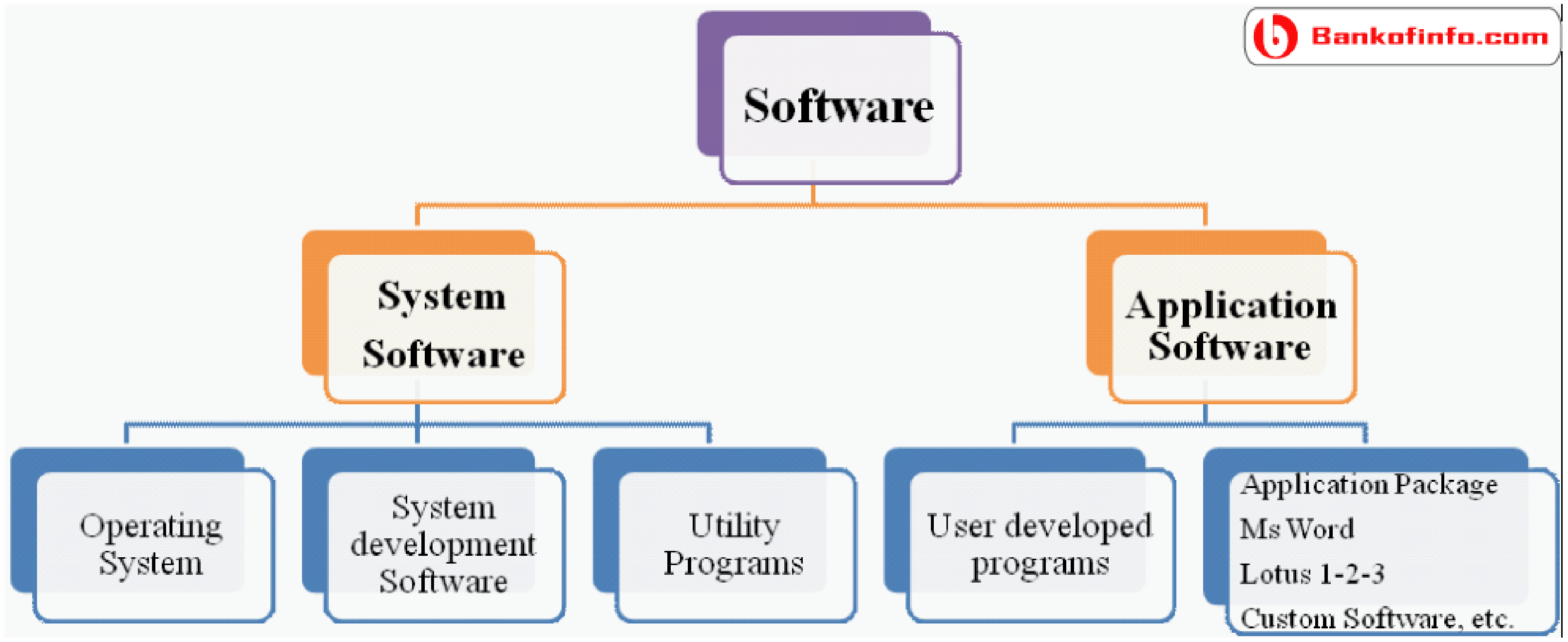
- A complete, working computer. Computer systems will include the computer along with any software and peripheral devices that are necessary to make the computer function. Every computer system, for example, requires an operating system.


Computer
Desktop Computer (System Unit)
Flat-panel Display (Output Device)
Speaker (Output Device)
Keyboard (Input Device)
Mouse (Input Device)

What is a Software program?

- **Program:** self-contained set of instructions used to operate a computer to produce a specific result
  - Also called **software**
- **Programming:** the process of writing a program, or software

SOFTWARE


Bankofinfo.com
Software
System Software
Application Software
Operating System
System development Software
Utility Programs
User developed programs
Application Package
Ms Word
Lotus 1-2-3
Custom Software, etc.

Systems Software

- System software is a type of computer program that is designed to run a computer's hardware and application programs.
- If we think of the computer system as a layered model, the system software is the interface between the hardware and user applications.

- The operating system (OS) is the best-known example of system software.

System Software (continued)

- **Operating system:** the set of system programs used to operate and control a computer; also called OS
- Tasks performed by the OS include
  - Memory management
  - Allocation of CPU time
  - Control of input and output
  - Management of secondary storage devices

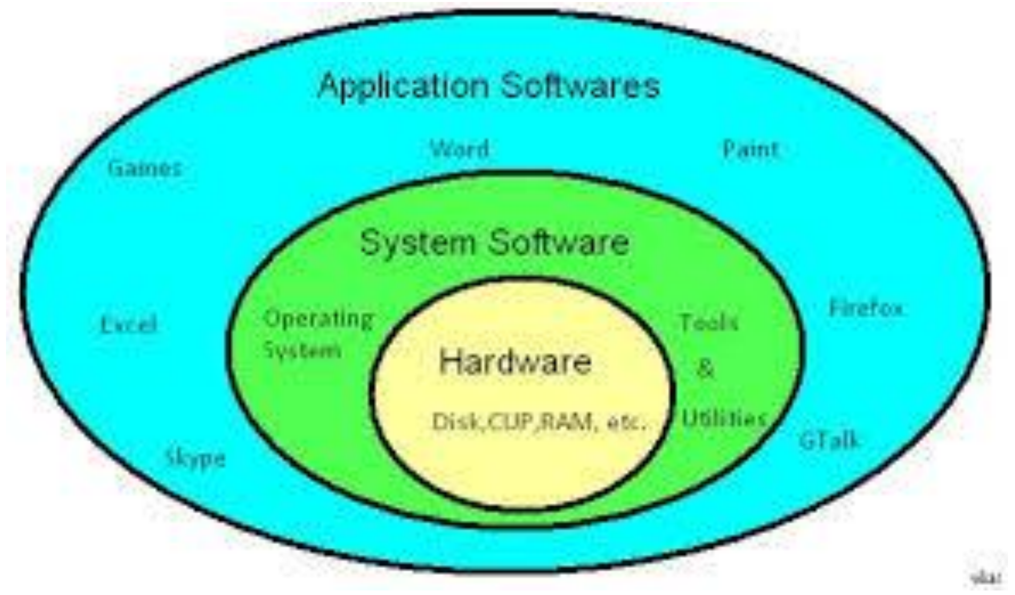
System Software (continued)

- **Multi-user** system: a system that allows more than one user to run programs on the computer simultaneously
- **Multitasking** system: a system that allows each user to run multiple programs simultaneously; also called **multiprogrammed** system

Application Software

- **Application software:** programs written to perform particular tasks for users.

- Layered model

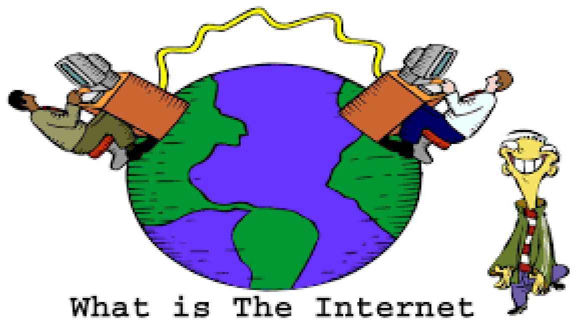


HARDWARE

- Computer hardware are the physical parts or components of a computer, such as the
  - monitor, keyboard, computer data storage, graphic card, sound card and motherboard.
- By contrast, software is instructions that can be stored and ran by hardware.

INTERNET

- The Internet, sometimes called simply "the Net," is a worldwide system of computer networks - a network of networks in which users at any one computer can, if they have permission, get information from any other computer (and sometimes talk directly to users at other computers).



Software Development/Programming

- **Software development procedure:** method for solving problems and creating software solutions
  - Consists of three phases:
    - Phase I: Development and Design
    - Phase II: Documentation
    - Phase III: Maintenance
- **Software engineering:** discipline concerned with creating efficient, reliable, maintainable software

Software Development (continued)

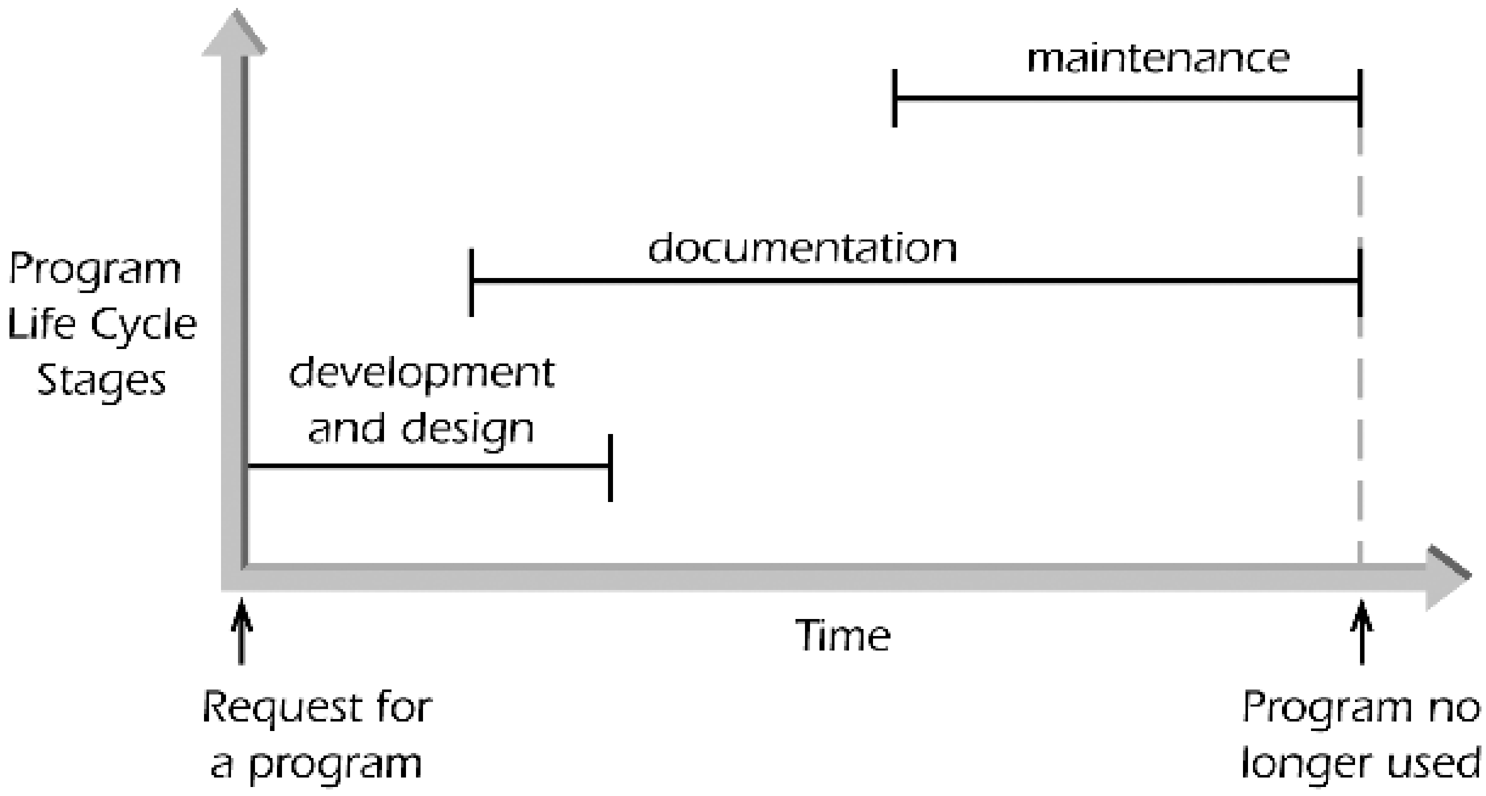


Figure 1.6 The three phases of software development.

Software Development (continued) Phase I. Development and Design

- **Program requirement:** request for a program or a statement of a problem
- After a program requirement is received, Phase I begins
- Phase I consists of four steps:
  - Analysis
  - Design
  - Coding
  - Testing

Software Development (continued) : Phase I (continued)

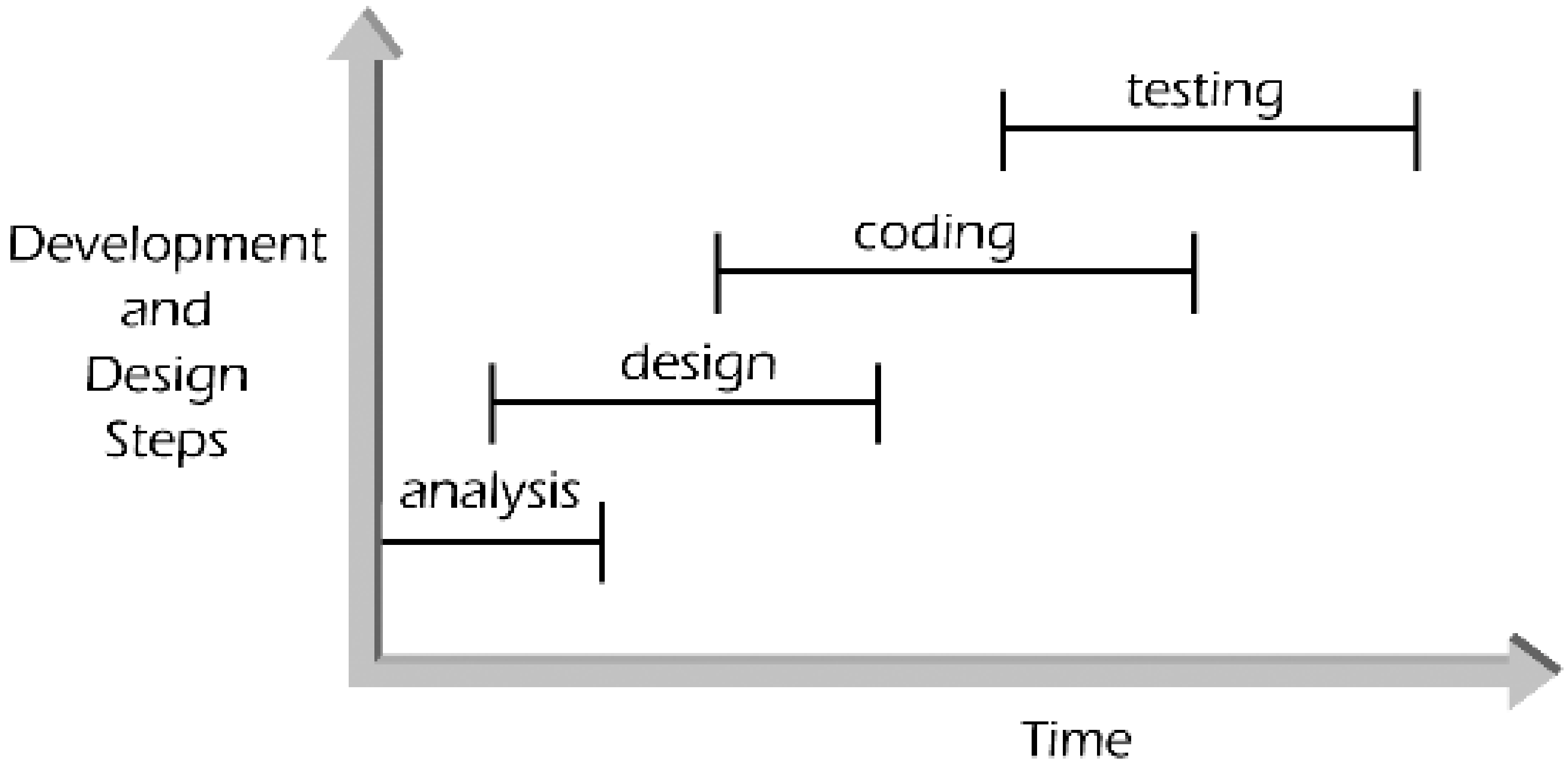


Figure 1.7 The development and design steps.

Software Development: Phase I (continued)

- **Step 1: Analyze the Problem**
  - Analyst must understand
    - What outputs are required
    - What inputs will be needed
    - How the inputs can be used to produce the desired outputs
  - Failure to analyze and understand the requirements leads to a failed solution!

Problem Solution and Software Development: Phase I (continued)

- **Step 2: Develop a Solution**
  - **Algorithm:** the exact steps used to solve a problem
  - Algorithm is usually defined at high level, then refined to detailed lower levels
  - Structure level diagrams may be used to represent the levels of analysis

Software Development: Phase I (continued)

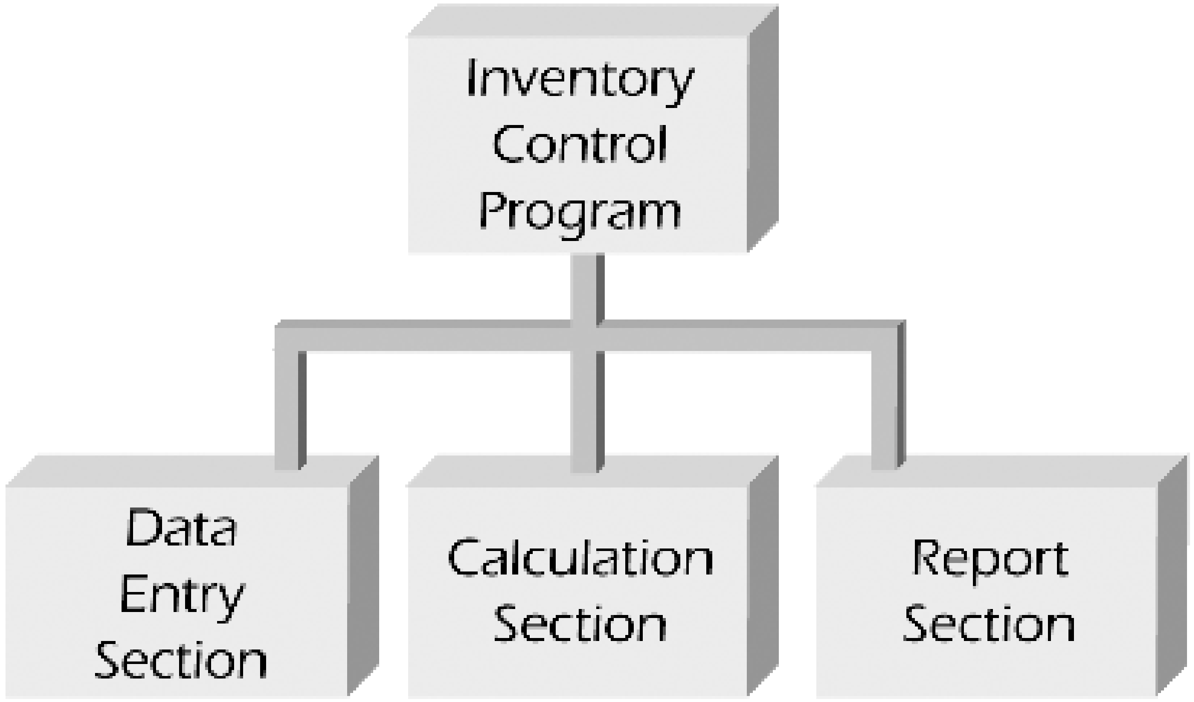


Figure 1.8 First-level structure diagram.

SOFTWARE DEVELOPMENT: PHASE I (CONTINUED)

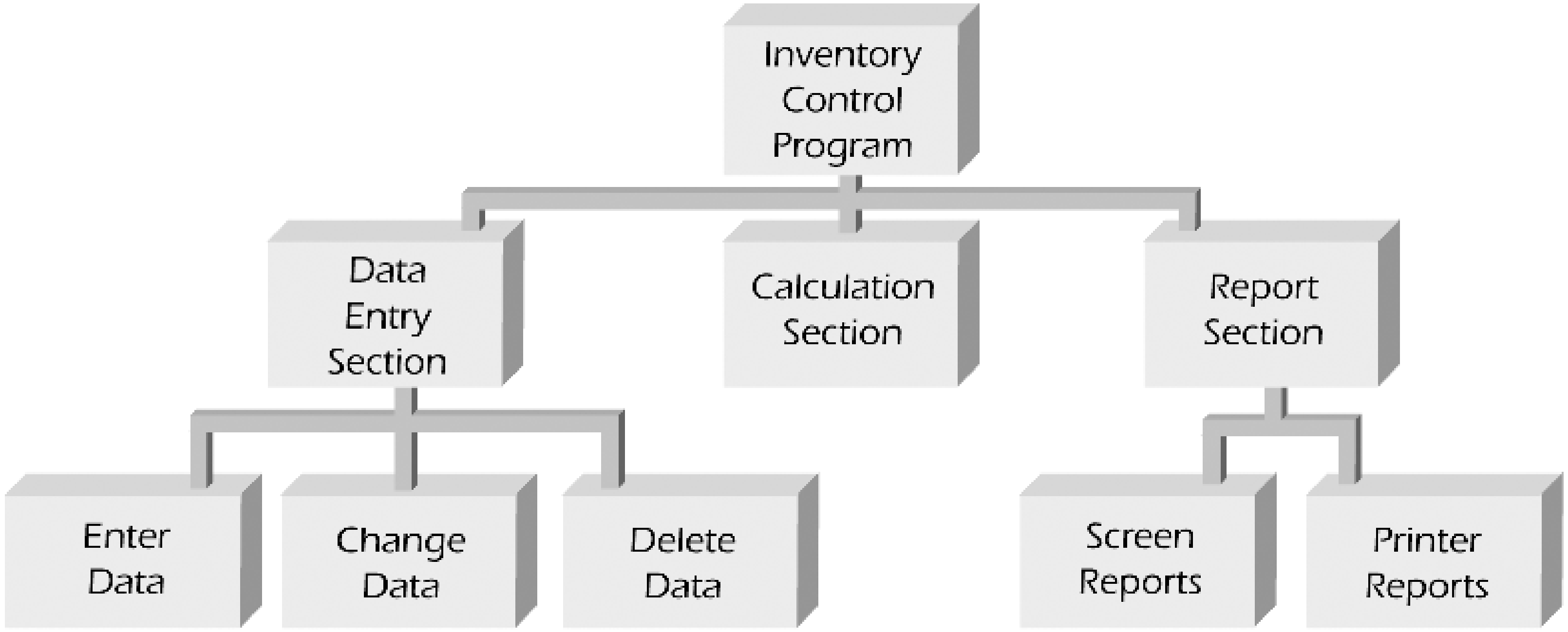


Figure 1.9 Second-level refinement structure diagram.

Software Development: Phase I (continued)

- **Step 3: Code the Solution**
  - Also called writing the program, and implementing the solution
  - Program should contain well-defined patterns or structures of the following types:
    - Sequence
    - Selection
    - Iteration
    - Invocation

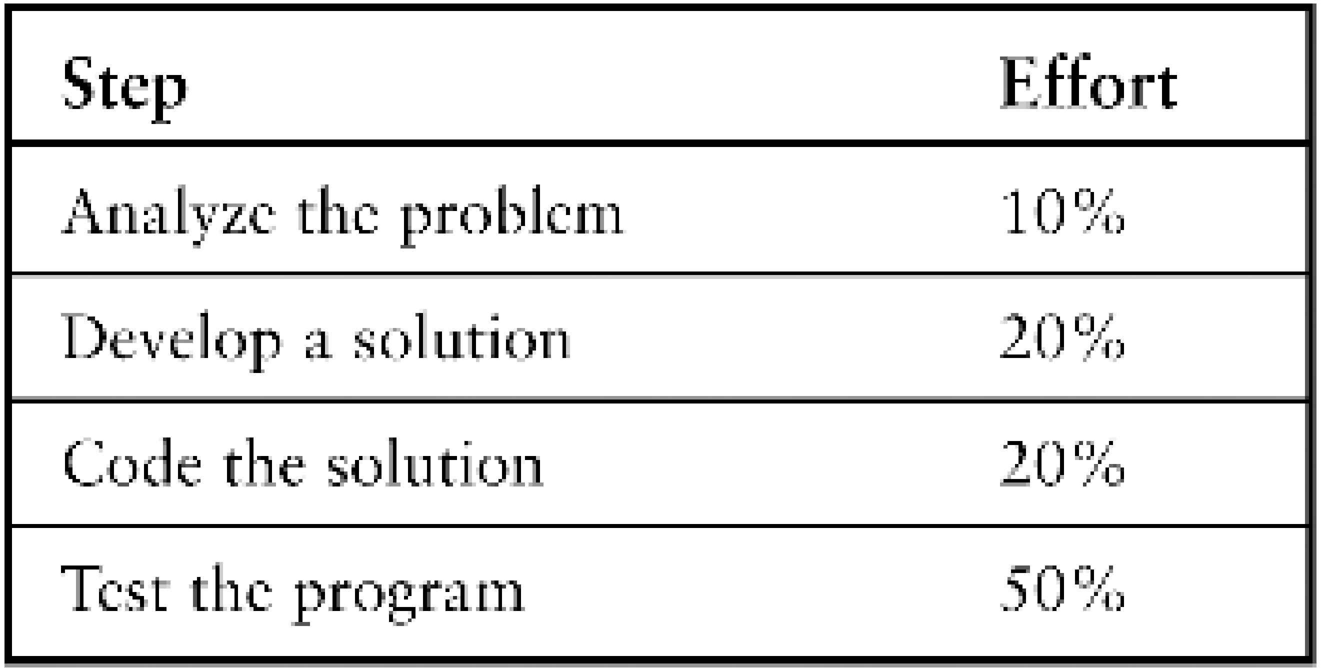
Software Development: Phase I (continued)

- **Step 4: Test and Correct the Program**
  - **Testing**: method to verify correctness and that requirements are met
  - **Bug**: a program error
  - **Debugging**: the process of locating an error, and correcting and verifying the correction
  - Testing may reveal errors, but does not guarantee the absence of errors

Software Development: Phase I (continued)

- Testing requires the use of meaningful, representative test data
- Impossible to test all possible combinations of data and operations in a complex program
- Testing plans must be developed to ensure good coverage in testing
- Testing usually requires more time than other steps in Phase I

Software Development: Phase I (continued)



Step	Effort
Analyze the problem	10%
Develop a solution	20%
Code the solution	20%
Test the program	50%

Relative effort for steps in Phase I

Software Development: Phase II. Documentation

- Many documents may be required, including
  - Program description
  - Algorithm development and changes
  - Well-commented program listing
  - Sample test runs
  - Users' manual

Software Development: Phase III. Maintenance

- **Maintenance** includes
  - Ongoing correction of newly discovered bugs
  - Revisions to meet changing user needs
  - Addition of new features
- Maintenance usually the longest phase, and may be the primary source of revenue
- Good documentation vital for effective maintenance

Software Development: Backup

- **Backup:** process of making copies of program code and documentation on a regular basis
- Backup copies = insurance against loss or damage
  - Consider using off-site storage for additional protection

WHAT IS PROBLEM SOLVING?

- Solving problems is the core of computer science.
- Programmers must first understand how a human solves a problem, then understand how to translate this "algorithm" into something a computer can do, and finally how to "write" the specific syntax (required by a computer) to get the job done.

Problem solving (Cont..)

- A computer is a tool that can be used to implement a plan for solving a problem.
- A computer program is a set of instructions for a computer. These instructions describe the steps that the computer must follow to implement a plan.
- An **algorithm** is a plan for solving a problem.
- A person must design an algorithm.
- A person must translate an algorithm into a computer program

Problem solving (Cont..)

- Computer Programmers are problem solvers. In order to solve a problem on a computer you must:
  - Know how to represent the information (data) describing the problem.
  - Determine the steps to transform the information from one representation into another.

Problem solving (Cont..)

- What are algorithms?
- Algorithms essentially, are a set of instructions to tell the computer what to do.
- *Ways to represent algorithms:*
  - Flowchart
  - formula
  - Pseudo code

Example of an algorithm

- **Problem**: I need a send a birthday card to my brother, Mark.

- **Analysis**: I don't have a card. I prefer to buy a card rather than make one myself.

  - algorithm:

    - Go to a store that sells greeting cards
    - Select a card
    - Purchase a card
    - Mail the card

- Write an algorithm to add two numbers entered by user.

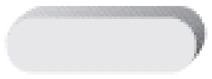
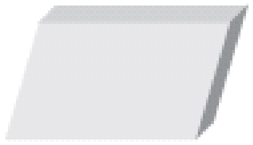
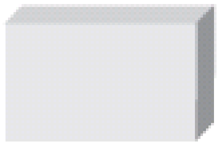
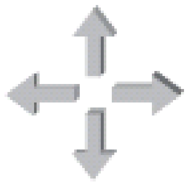
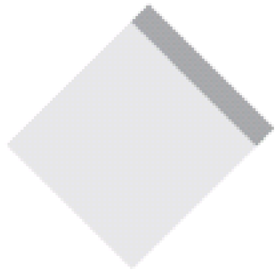
  - Step 1: Start
  - Step 2: Declare variables num1, num2 and sum.
  - Step 3: Read values num1 and num2.
  - Step 4: Add num1 and num2 and assign the result to sum.
  - $\text{sum} \leftarrow \text{num1} + \text{num2}$
  - Step 5: Display sum
  - Step 6: Stop

ALGORITHMS (CONTINUED)

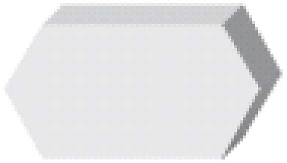
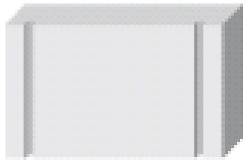
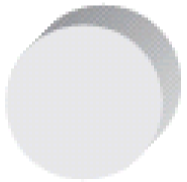
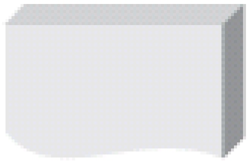
There are many ways to write an algorithm

- **Pseudocode:** English-like phrases used to describe the algorithm
- **Formula:** description of a mathematical equation
- **Flowchart:** diagram showing the flow of instructions in an algorithm
  - – Flowcharts use special symbols

ALGORITHMS: FLOWCHARTS

Symbol	Name	Description
	Terminal	Indicates the beginning or end of a program
	Input/Output	Indicates an input or output operation
	Process	Indicates computation or data manipulation
	Flow Lines	Used to connect the other flowchart symbols and indicate the logic flow
	Decision	Indicates a program branch point

Algorithms: Flowcharts (continued)

Symbol	Name	Description
	Loop	Indicates the initial, limit, and increment values of a loop
	Predefined Process	Indicates a predefined process, as in calling a function
	Connector	Indicates an entry to, or exit from, another part of the flowchart or a connection point
	Report	Indicates a written output report

Algorithms: Flowcharts (continued)

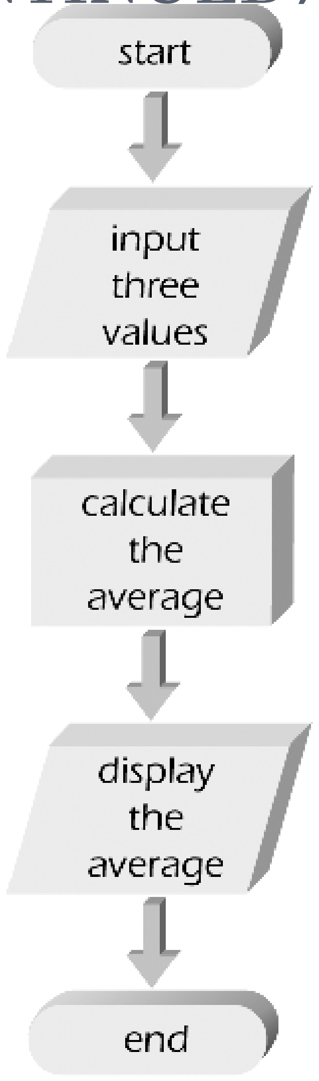


Figure 1.12 Flowchart for calculating the average of three numbers.