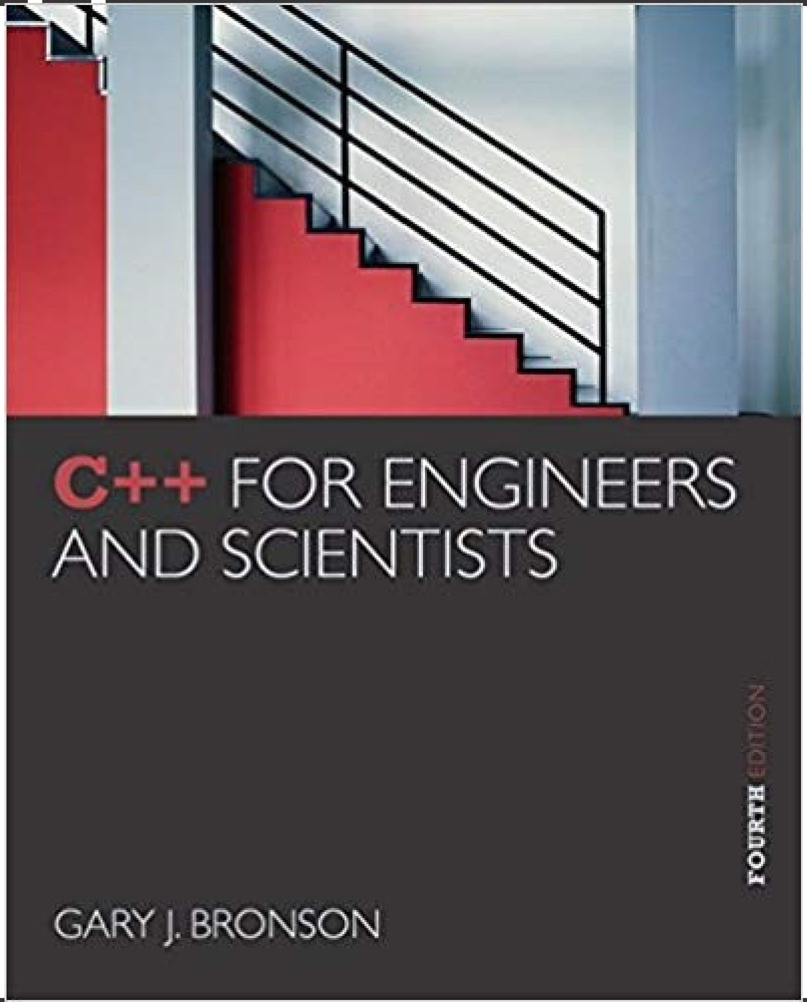
ELEG 1043
Computer Applications in Engineering



Chapter 1: Preliminaries

C++ FOR ENGINEERS AND SCIENTISTS

Acknowledgement

- Some of the slides or images are from various sources. The copyright of those materials belongs to their original owners.

Objectives

In this chapter, you will learn about:

- Unit analysis
- Exponential and scientific notations
- Software development
- Algorithms
- Software, hardware, and computer storage
- Common programming errors

Preliminary One: Unit Analysis

- Using consistent and correct units when making computations is crucial
- Performing a **unit analysis:**
  - Include only the units and conversion factors in an equation
  - Cancel out corresponding units in the numerator and denominator

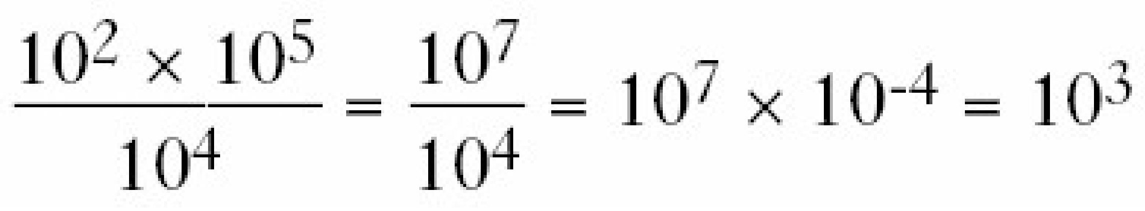
$$\cancel{days} \times \frac{24\ \cancel{hr}}{\cancel{day}} \times \frac{60\ min}{\cancel{hr}}$$

Preliminary Two: Exponential and Scientific Notations

- Many engineering and scientific applications deal with extremely large and extremely small numbers
  – Written in **exponential notation** to make entering the numbers in a computer program easier
  – Written in **scientific notation** to performing hand calculations for verification purposes

Using Scientific Notation

- Convenient for evaluating formulas that use very large or very small numbers
- Two basic exponential rules
  - Rule 1: 10^n x 10^m = 10^{n+m} for any values, positive or negative, of *n* and *m*
  - Rule 2: $1/10^{-n}$ = 10^n for any positive or negative value of *n*



$$\frac{10^2 \times 10^5}{10^4} = \frac{10^7}{10^4} = 10^7 \times 10^{-4} = 10^3$$

Preliminary Three: Software Development

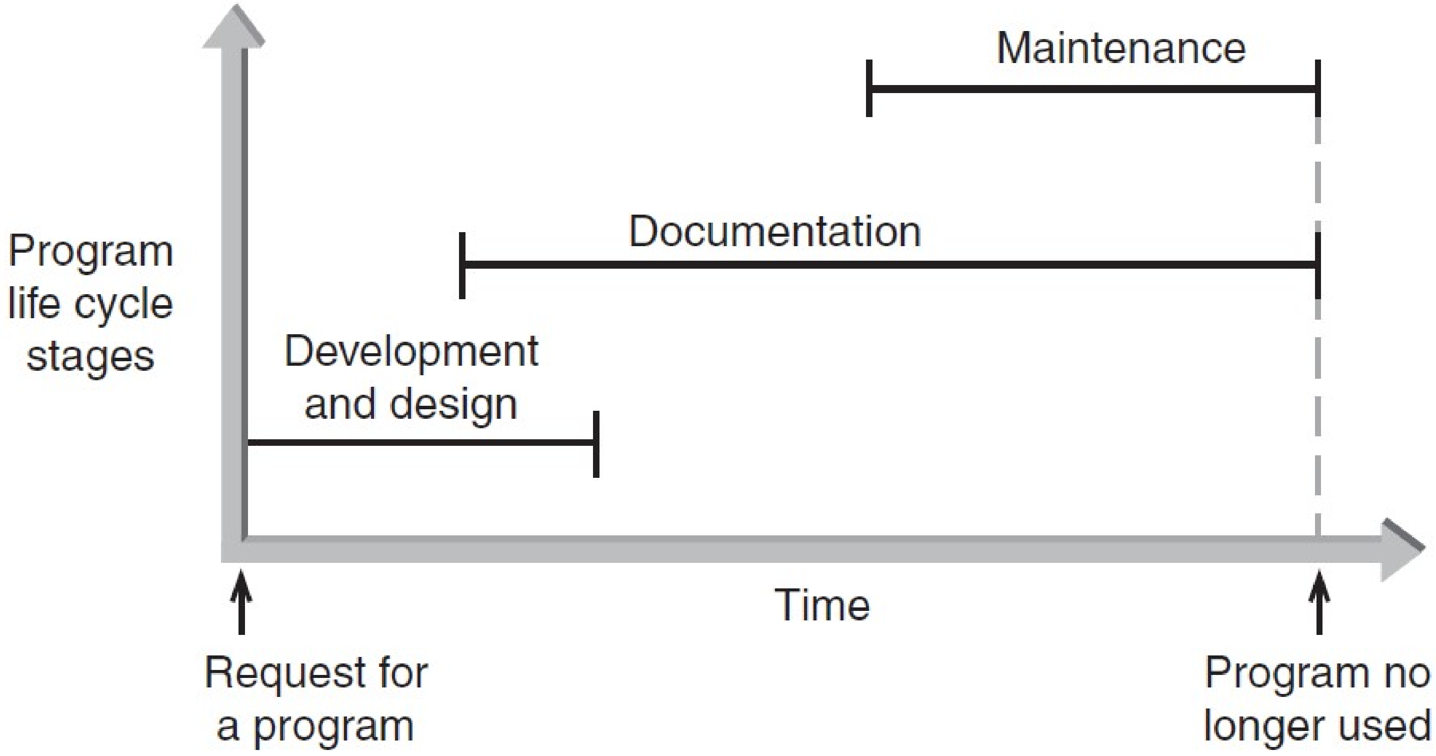


Figure 1.2 The three phases of program development

Phase I: Development and Design (continued)

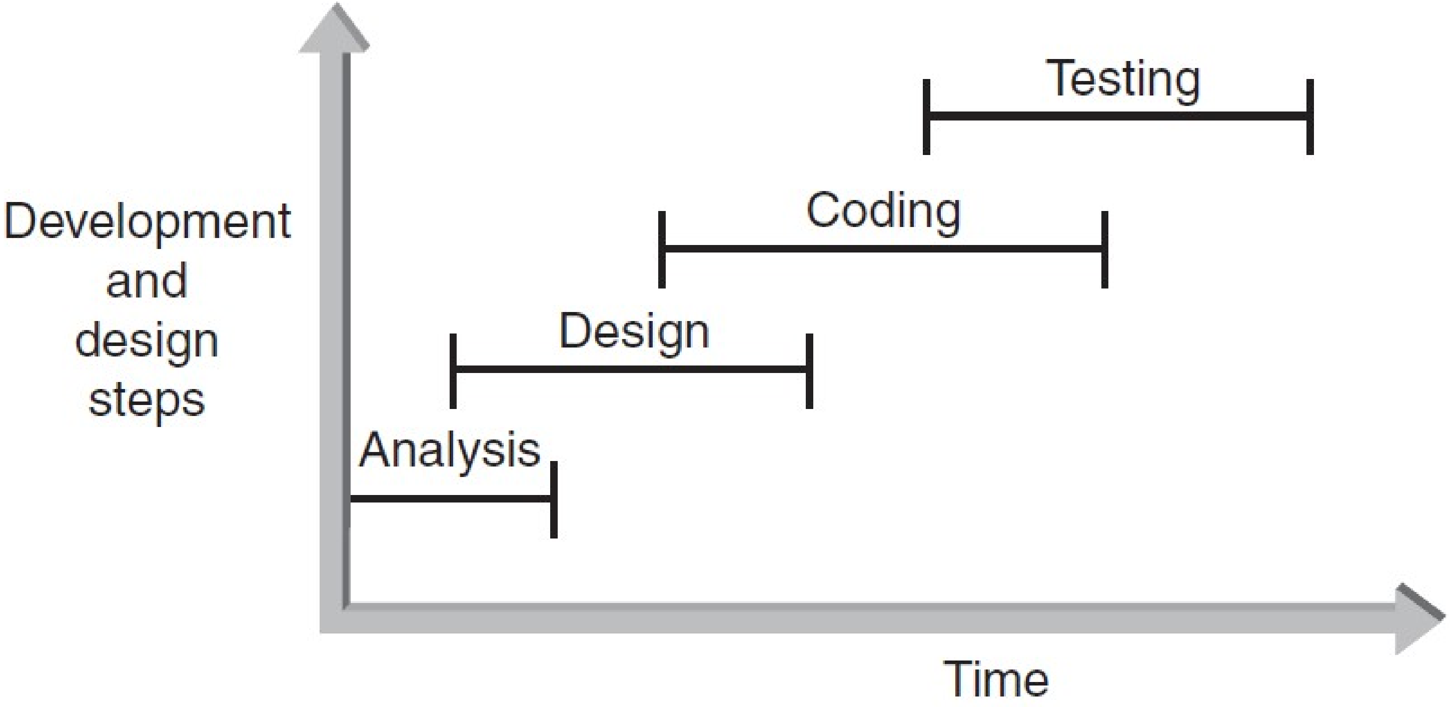
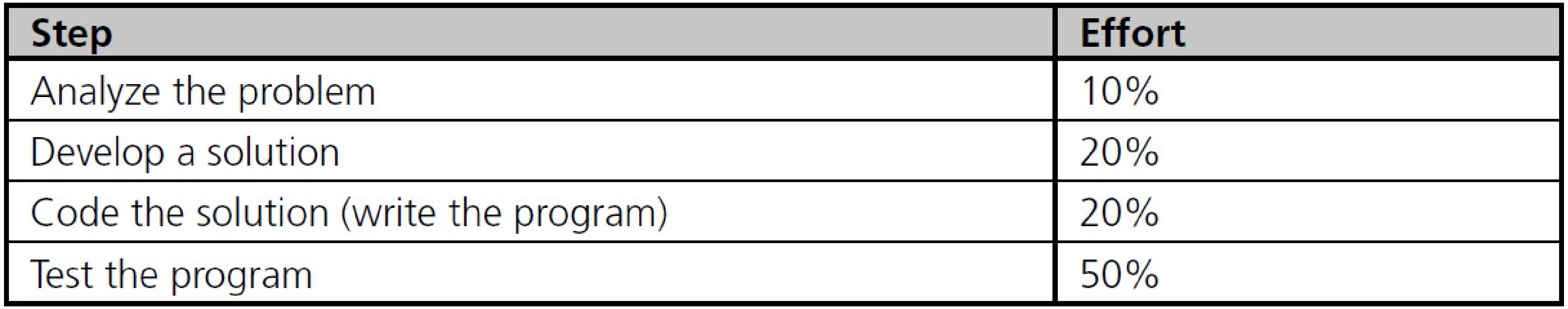


Figure 1.3 The development and design steps

Phase I: Development and Design (continued)

- Step 4: Test and Correct the Program (continued)
  - Table 1.3 lists the comparative amount of effort typically expended on each development and design step in large commercial programming projects



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

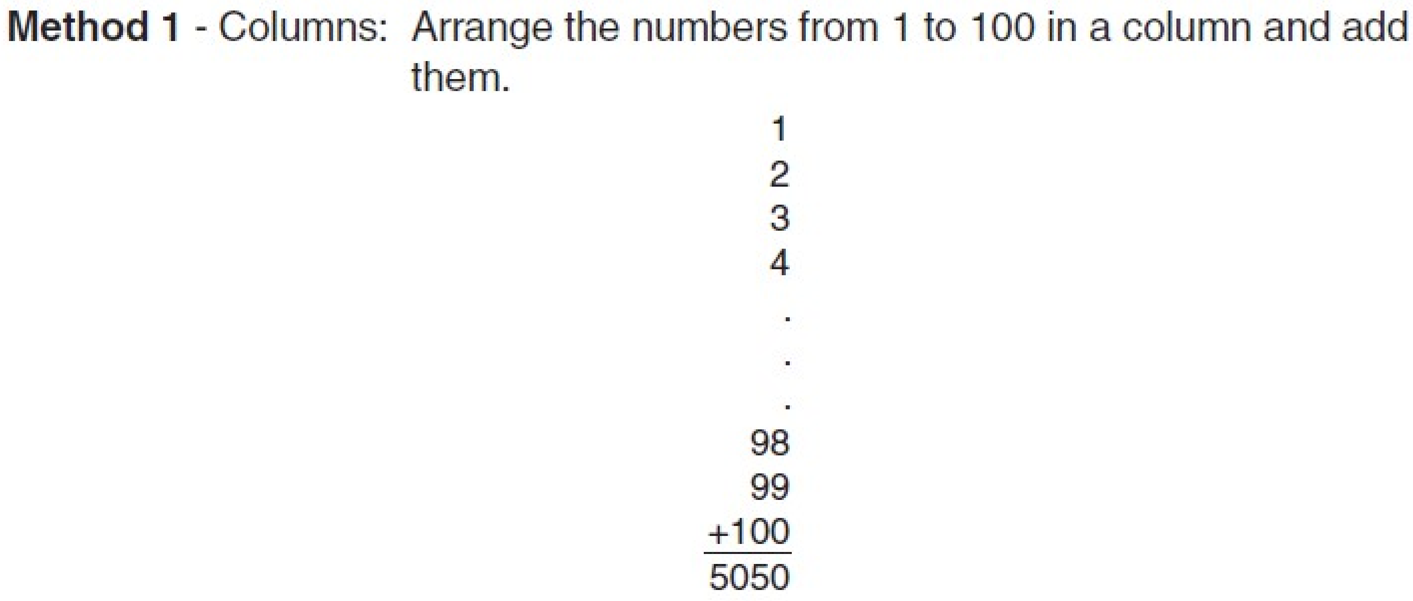
Table 1.3 Effort Expended in Phase I

Preliminary Four: Algorithms

- **Algorithm:** Step-by-step sequence of instructions
  - Must terminate
  - Describes how the data is to be processed to produce the desired output
- **Pseudocode:** English-like phrases used to describe steps in an algorithm
- **Formula:** Mathematical equations
- **Flowchart:** Diagrams with symbols

Preliminary Four: Algorithms (continued)

- Problem: Calculate the sum of all whole numbers from 1 through 100

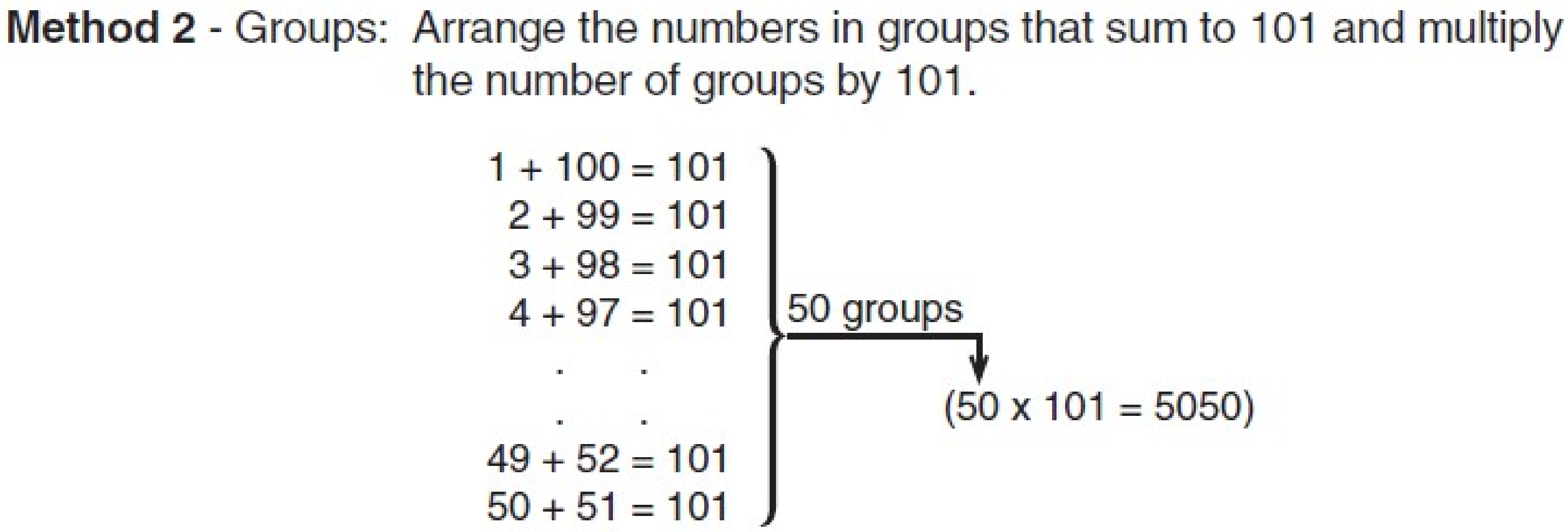


Method 1 - Columns: Arrange the numbers from 1 to 100 in a column and add them.

```
   1
   2
   3
   4
   .
   .
   .
  98
  99
+100
----
5050
```

Figure 1.6 Summing the numbers 1 to 100

Preliminary Four: Algorithms (continued)



Method 2 - Groups: Arrange the numbers in groups that sum to 101 and multiply the number of groups by 101.

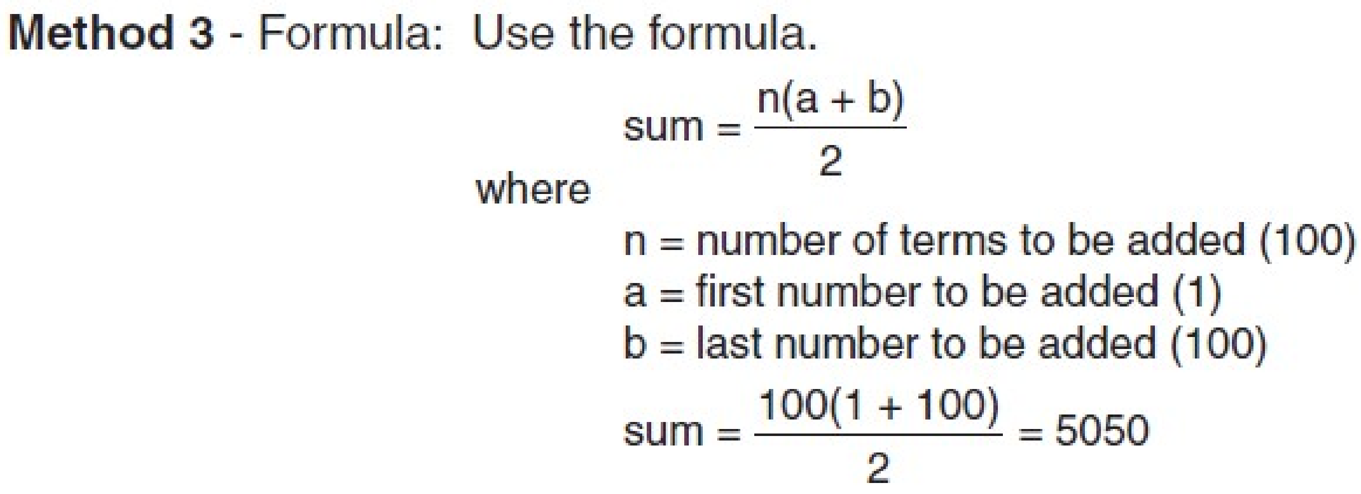
$1 + 100 = 101$
$2 + 99 = 101$
$3 + 98 = 101$
$4 + 97 = 101$
. .
. .
$49 + 52 = 101$
$50 + 51 = 101$

50 groups

($50 \times 101 = 5050$)

Figure 1.6 Summing the numbers 1 to 100 (continued)

Preliminary Four: Algorithms (continued)



Method 3 - Formula: Use the formula.

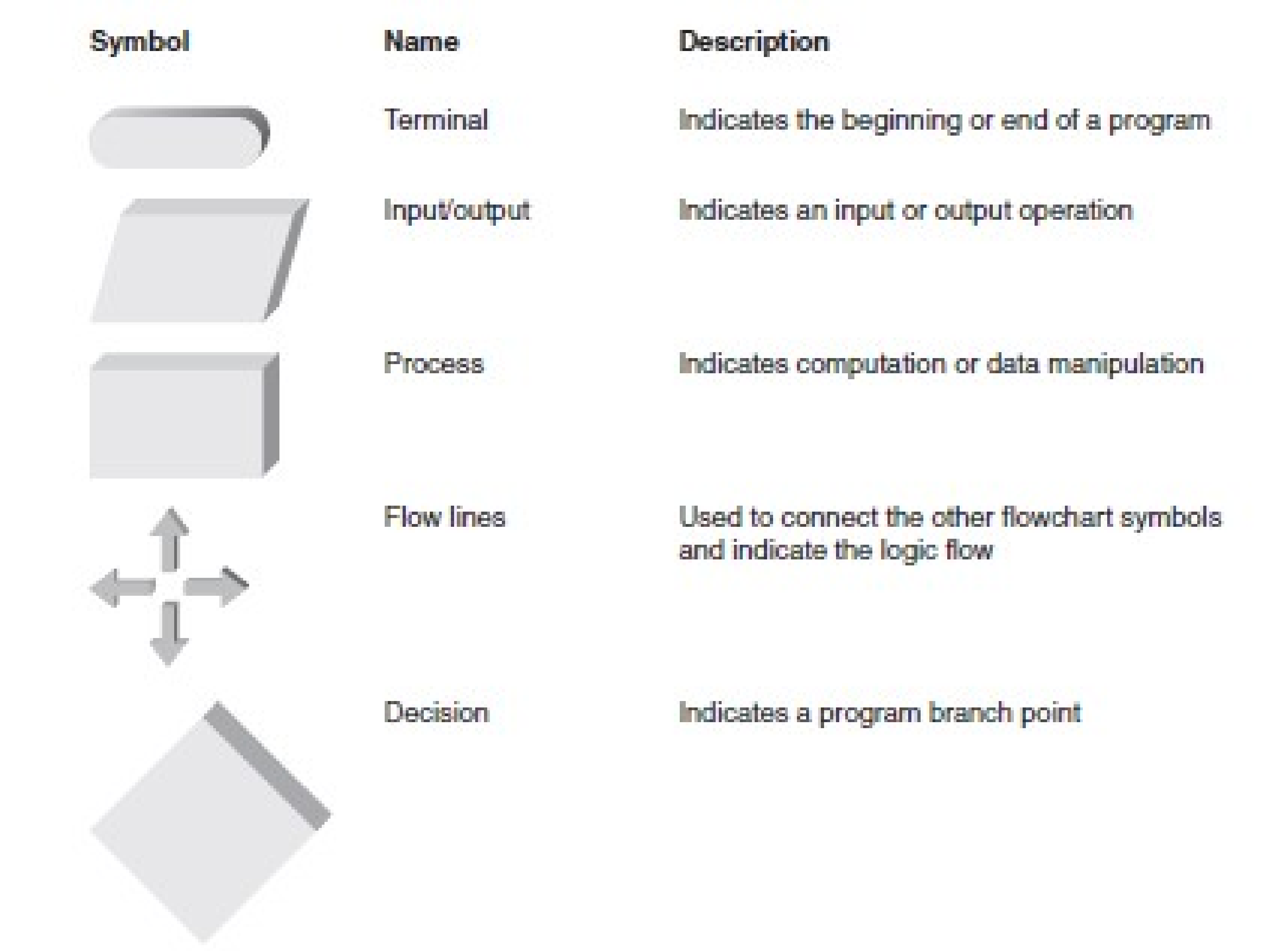
$$\text{sum} = \frac{n(a + b)}{2}$$

where

n = number of terms to be added (100)
a = first number to be added (1)
b = last number to be added (100)

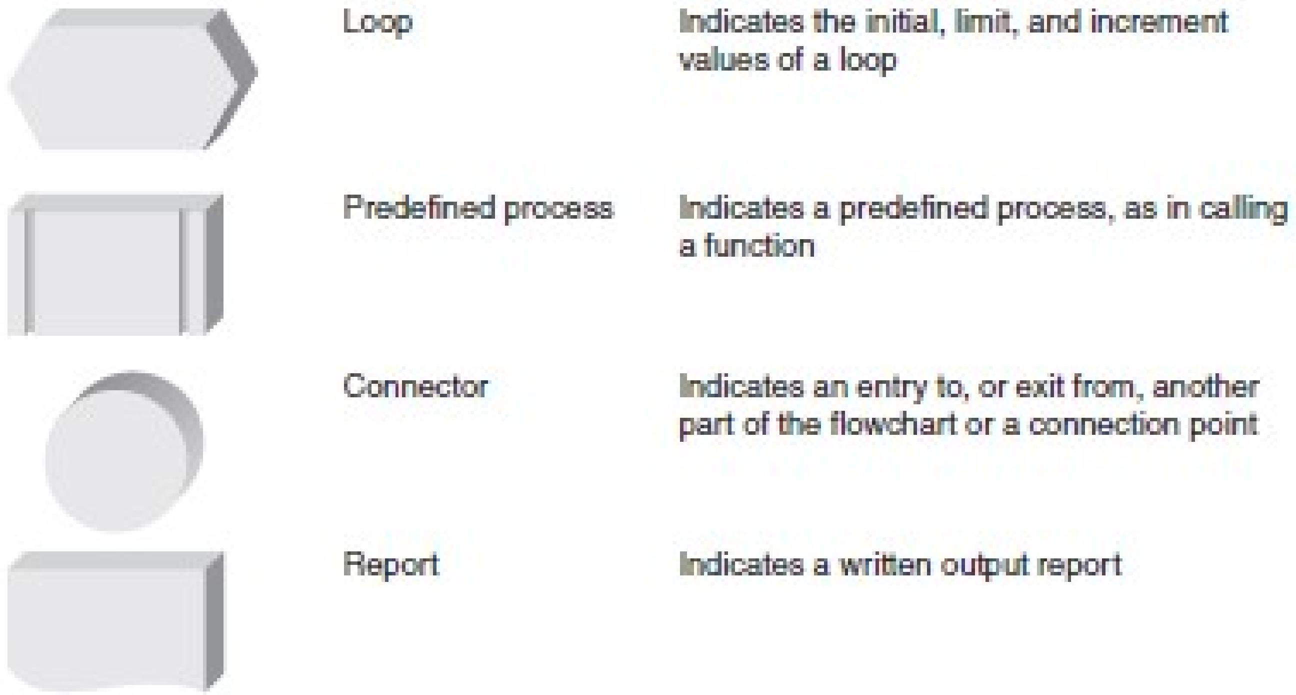
$$\text{sum} = \frac{100(1 + 100)}{2} = 5050$$

Figure 1.6 Summing the numbers 1 to 100 (continued)



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

Figure 1.7 Flowchart symbols



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

Figure 1.7 Flowchart symbols (continued)

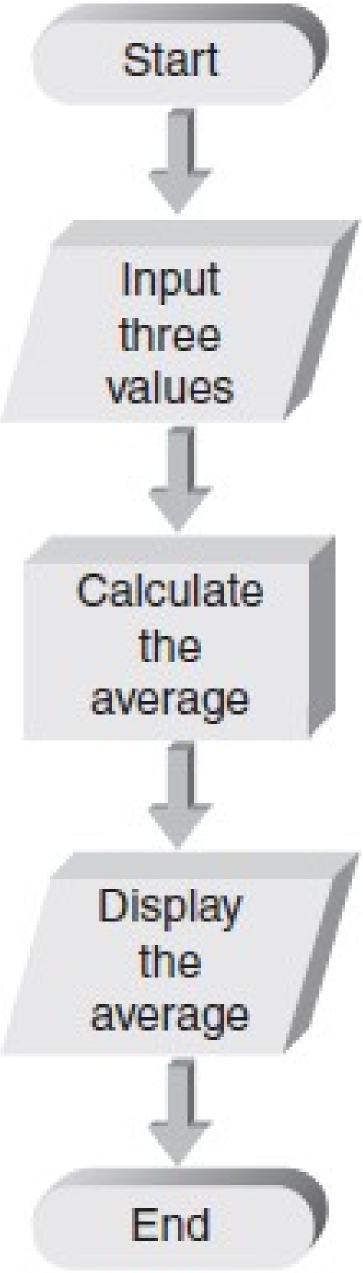


Figure 1.8 Flowchart for calculating the average of three numbers

Preliminary Four: Algorithms (continued)

- **Select an algorithm and understand the required steps**
- **Coding the algorithm**

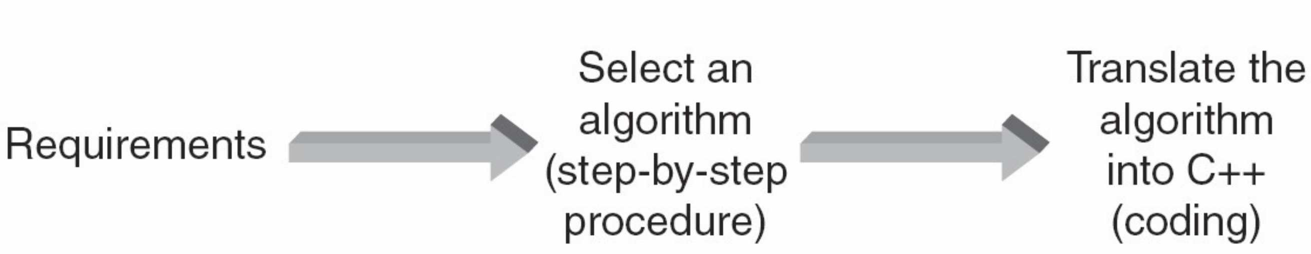

Requirements
Select an algorithm (step-by-step procedure)
Translate the algorithm into C++ (coding)

Figure 1.9 Coding an algorithm

Software, Hardware, and Computer Storage

- **Programming:** Process of writing a program, or software
- **Programming language:**
  - Set of instructions used to construct a program
  - Comes in a variety of forms and types

Machine Language

- **Machine language programs**: only programs that can actually be used to operate a computer
  - Also referred to as executable programs (executables)
  - Consists of a sequence of instructions composed of binary numbers
  - Contains two parts: an instruction and an address

Assembly Language

- **Assembly language programs**: Substitute word-like symbols, such as ADD, SUB, and MUL, for binary opcodes
  - Use decimal numbers and labels for memory addresses
    - Example: `ADD 1, 2`
- **Assemblers:** Translate programs into machine language

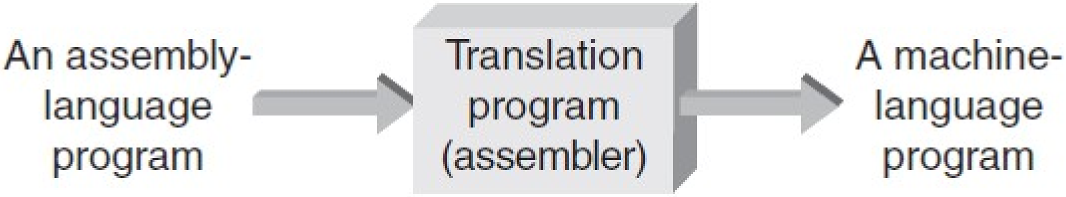

An assembly-language program
Translation program (assembler)
A machine-language program

Figure 1.10 Assembly-language programs must be translated

Low- and High-Level Languages

- **Low-level languages:** Languages that use instructions tied directly to one type of computer
  - Examples: machine language, assembly language
- **High-level languages:** Instructions resemble written languages, such as English
  - Can be run on a variety of computer types
  - Examples: Visual Basic, C, C++, Java, Python

Low- and High-Level Languages (continued)

- **Source code:** The programs written in a high- or low-level language
  - Source code must be translated to machine instructions in one of two ways:
    - **Interpreter:** Each statement is translated individually and executed immediately after translation (Python)
    - **Compiler:** All statements are translated and stored as an executable program, or object program; execution occurs later
      - C++ is predominantly a compiled language

Low- and High-Level Languages (continued)

- Large C++ programs may be stored in two or more separate program files due to:
  - Use of previously written code
  - Use of code provided by the compiler
  - Modular design of the program (for reusability of components)
- **Linker:** Combines all of the compiled code required for the program

Procedural and Object Orientations

- Programs can also be classified by their orientation:
  - **Procedural:** Available instructions are used to create self-contained units called procedures (C)
  - **Object-oriented:** Reusable objects, containing code and data, are manipulated
    - Object-oriented languages support reusing existing code more easily
- C++ contains features of both

Application and System Software

- **Application software:** Programs written to perform particular tasks for users
- **System software:** Collection of programs to operate the computer system
  - System software must be loaded first; called booting the system
  - **Bootstrap loader:** A permanent, automatically executed component to start the boot process

Application and 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

Application and 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 users to run multiple programs simultaneously
  - Also called multiprogrammed system

The Development of C++

- The purpose of most application programs is to process data to produce specific results

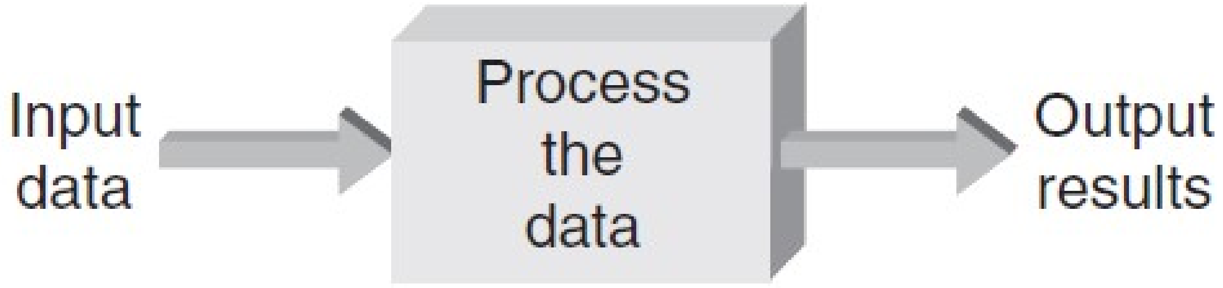


Figure 1.12 Basic procedural operations

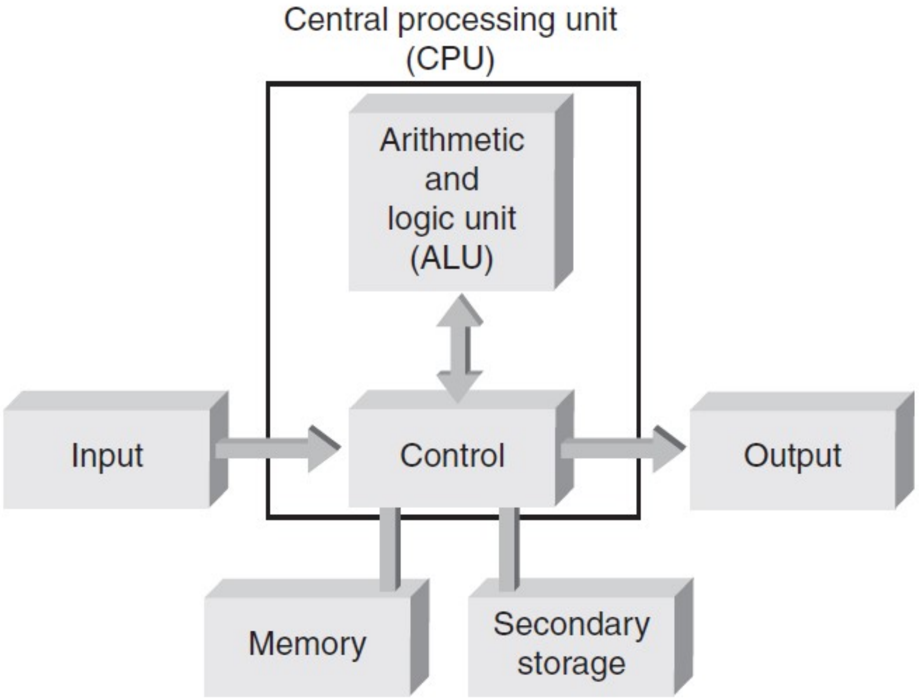
The Development of C++ (continued)

- Early procedural languages included:
  - FORTRAN: Formula Translation
  - ALGOL: Algorithmic Language
  - COBOL: Common Business Oriented Language
  - BASIC: Beginners All-purpose Symbolic Instruction Code
  - Pascal
  - C
- Early object-oriented language:
  - C++

Computer Hardware

- **Computer hardware:** Components that support the capabilities of the computer

Figure 1.15 Basic hardware units of a computer



Computer Hardware (continued)

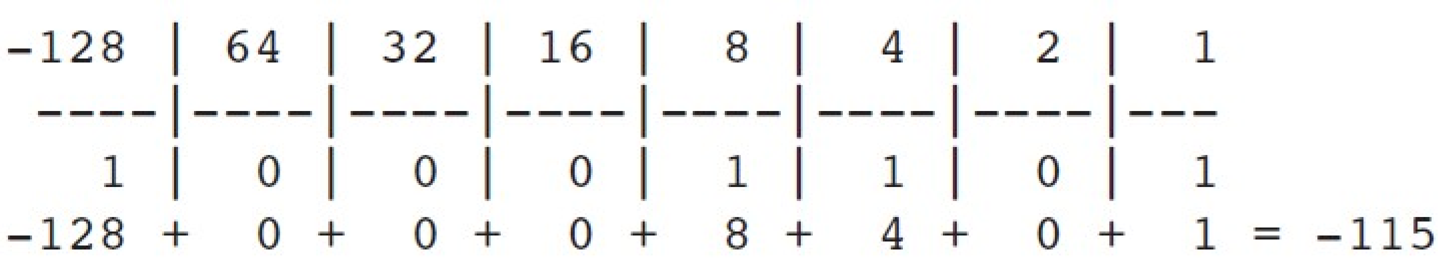
- Components include:
  - **Arithmetic and logic unit (ALU):** Performs arithmetic and logic functions
  - **Control unit:** Directs and monitors overall operations
  - **Memory unit:** Stores instructions and data
  - **Input and output (I/O) unit:** Interfaces to peripheral devices
  - **Secondary storage**: Nonvolatile permanent storage such as hard disks
  - **Central processing unit (CPU):** Also called microprocessor; combines the ALU and control unit on a single chip

Computer Storage

- **Bit:** Smallest unit of data; value of 0 or 1
- **Byte:** Grouping of 8 bits representing a single character
- **Character codes:** Collection of patterns of 0s and 1s representing characters
  - Examples: ASCII, EBCDIC

Computer Storage (continued)

- **Number codes:** Patterns used to store numbers
- **Two's complement** number code: Represents a decimal number as a binary number of 0s and 1s
  - – Determine with a value box

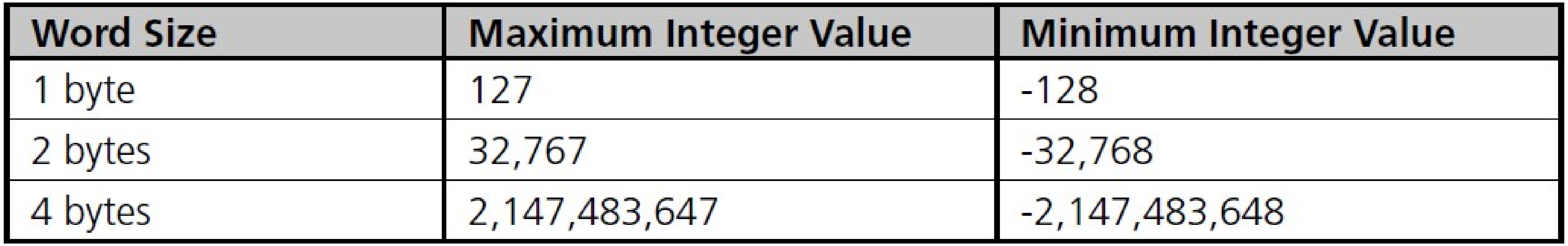


```
-128 | 64 | 32 | 16 |  8 |  4 |  2 |  1
 ----|----|----|----|----|----|----|---
   1 |  0 |  0 |  0 |  1 |  1 |  0 |  1
-128 +  0 +  0 +  0 +  8 +  4 +  0 +  1 = -115
```

Figure 1.18 Converting 10001101 to a base 10 number

Computer Storage (continued)

- **Word:** Grouping of one or more bytes
  - Facilitates faster and more extensive data access
- Number of bytes in a word determines the maximum and minimum values that can be stored:



Word Size	Maximum Integer Value	Minimum Integer Value
1 byte	127	-128
2 bytes	32,767	-32,768
4 bytes	2,147,483,647	-2,147,483,648

Table 1.4 Word size and Integer Values

Common Programming Errors

- Common errors include:
  - Failing to use consistent units
  - Using an incorrect form of a conversion factor
  - Rushing to write and run a program before fully understanding the requirements
  - Not backing up a program
  - Not appreciating that computers respond only to explicitly defined algorithms

Summary

- To determine correct forms of a conversion factor, perform a unit analysis
- Software: Programs used to operate a computer
- Programming language types:
  - Low-level languages
    - Machine language (executable) programs
    - Assembly languages
  - High-level languages
    - Compiler and interpreter languages

Summary (continued)

- Software engineering: discipline concerned with creating readable, efficient, reliable, and maintainable programs
- Three phases in software development:
  - Program development and design
  - Documentation
  - Maintenance

Summary (continued)

- Four steps in program development and design:
  - Analyze the problem
  - Develop a solution
  - Code the solution
  - Test and correct the solution
- Algorithm: Step-by-step procedure that describes how a task is performed
- Computer program: Self-contained unit of instructions and data used to operate a computer to produce a desired result