

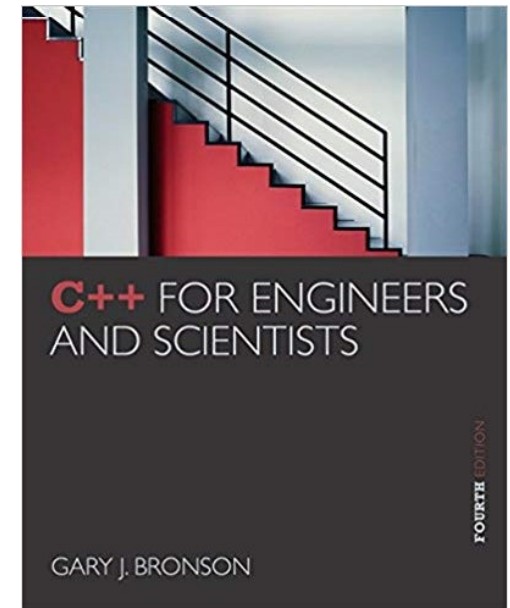
# ELEG 1043

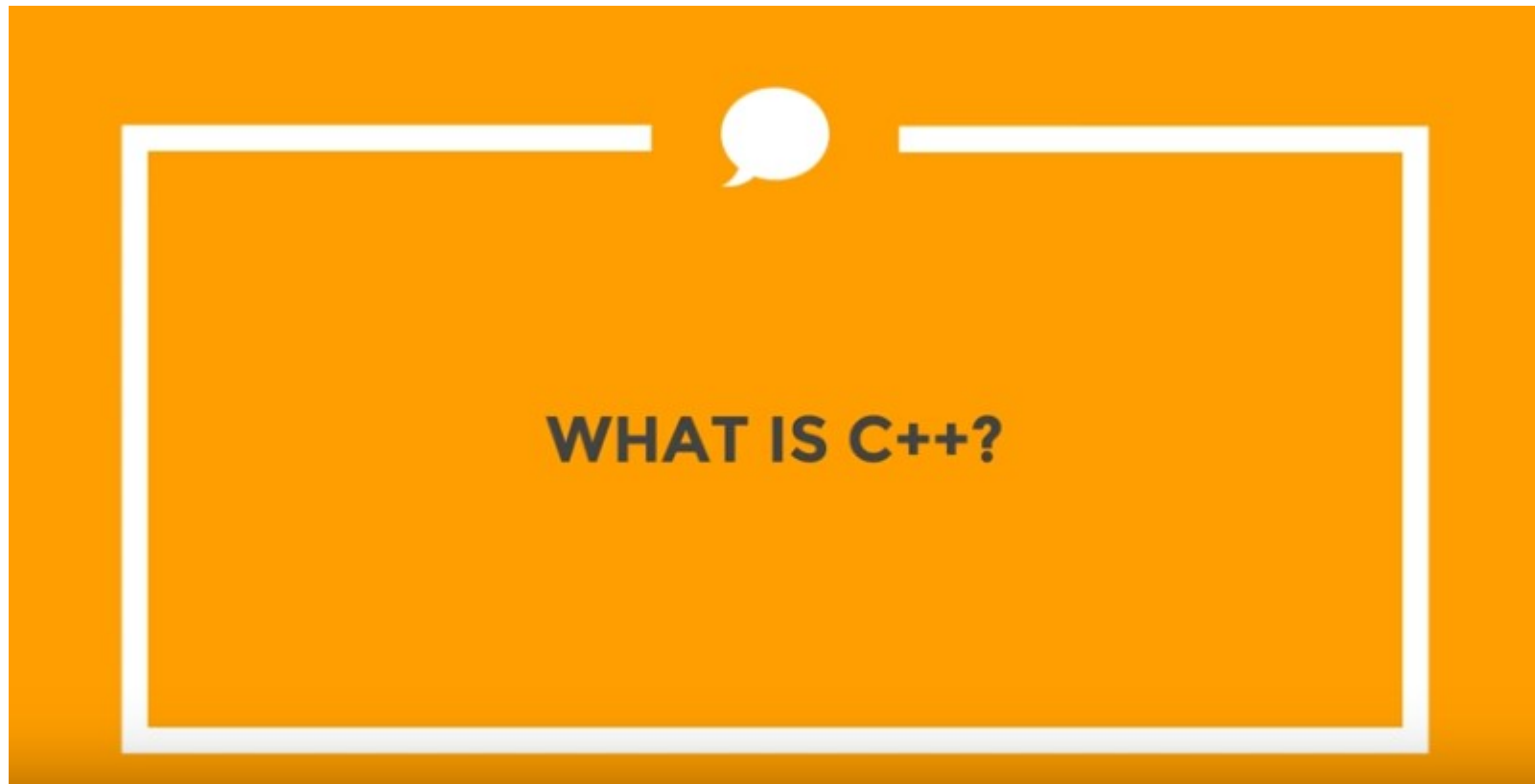
## Computer Applications in Engineering



# Source Materials

- Textbooks
  - Required
    - C++ for Engineers and Scientists *4th Edition* , Gary J. Bronson, Thompson Learning, ISBN-13: 978-1133187844 , ISBN-10: 1133187846
  - Recommended
    - Programming and Problem Solving with C++ by Nell Dale 6<sup>th</sup> Edition





[https://www.youtube.com/watch?v=kJkB\\_Tggk8U](https://www.youtube.com/watch?v=kJkB_Tggk8U)



# 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

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

# Example of Unit Analysis

## Task: Converting Days to Seconds

1st conversion:  
days to hours  
(cross out the days)

2nd conversion:  
hours to minutes  
(cross out the hours)

3rd conversion:  
minutes to seconds  
(cross out the minutes)

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



# 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

# Preliminary Two: Exponential and Scientific Notations (continued)

- Examples of exponential and scientific notation:

Decimal Notation	Exponential Notation	Scientific Notation
1625.	1.625e3	$1.625 \times 10^3$
63421.	6.3421e4	$6.3421 \times 10^4$
.00731	7.31e-3	$7.31 \times 10^{-3}$
.000625	6.25e-4	$6.25 \times 10^{-4}$

# Using Scientific Notation

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

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

# Using Scientific Notation (continued)

- If exponent is positive, it represents the actual number of zeros that follow the 1
- If exponent is negative, it represents one less than the number of zeros after the decimal point and before the 1
- Scientific notation can be used with any decimal number
  - Not just powers of 10

# Using Scientific Notation (continued)

- Common scientific notations have their own symbols

Value	Scientific Notation	Symbol	Name
0.000,000,000,001	$10^{-12}$	p	pico
0.000,000,001	$10^{-9}$	n	nano
0.000,001	$10^{-6}$	$\mu$	micro
0.001	$10^{-3}$	m	milli
1000	$10^3$	k	kilo
1,000,000	$10^6$	M	mega
1,000,000,000	$10^9$	G	giga
1,000,000,000,000	$10^{12}$	T	tera

**Table 1.2** Scientific Notational Symbols

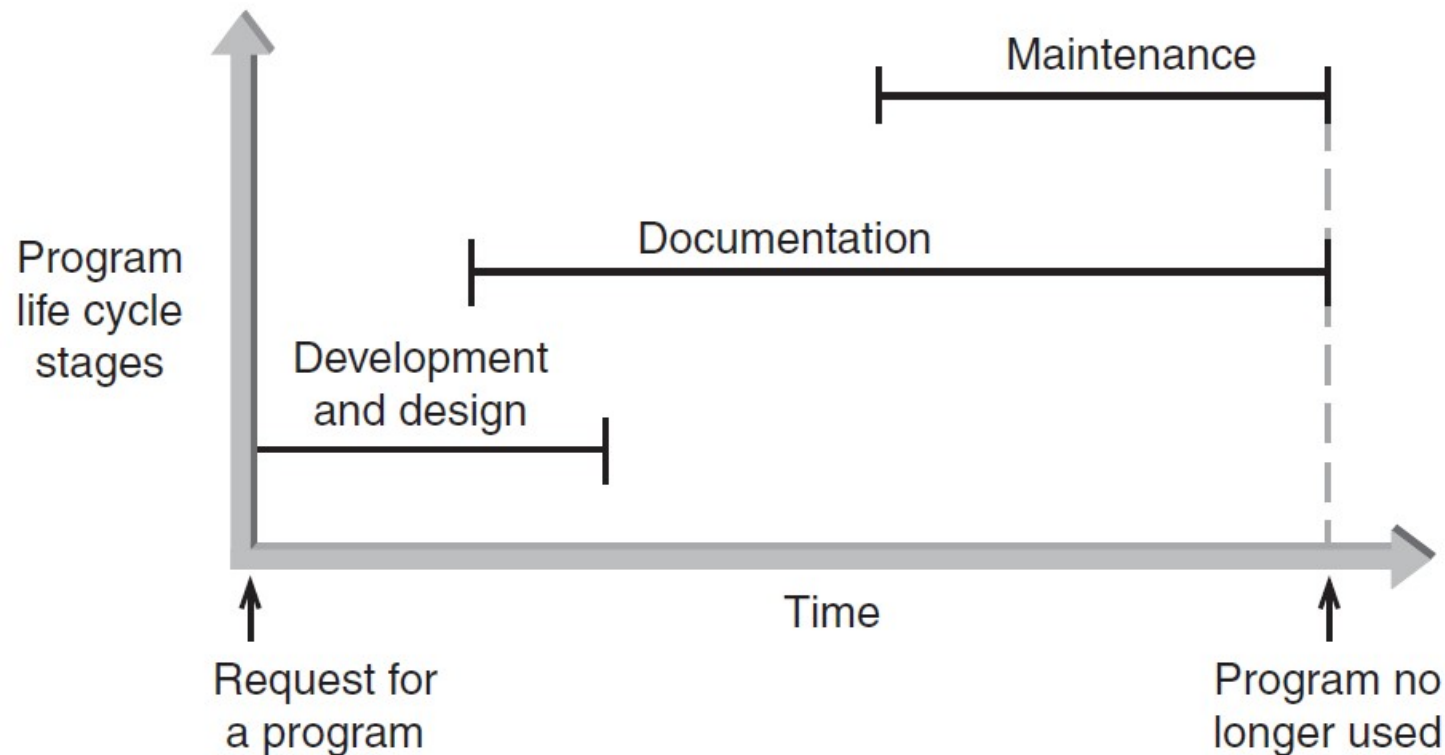
# Preliminary Three: Software Development

- **Computer program:** Self-contained set of instructions used to operate a computer to produce a specific result
  - Also called **software**
  - Solution developed to solve a particular problem, written in a form that can be executed on a computer

# Preliminary Three: Software Development (continued)

- **Software development procedure:** Helps developers understand the problem to be solved and create an effective, appropriate software solution
- **Software engineering:**
  - Concerned with creating readable, efficient, reliable, and maintainable programs and systems
  - Uses software development procedure to achieve this goal

# Preliminary Three: Software Development (continued)



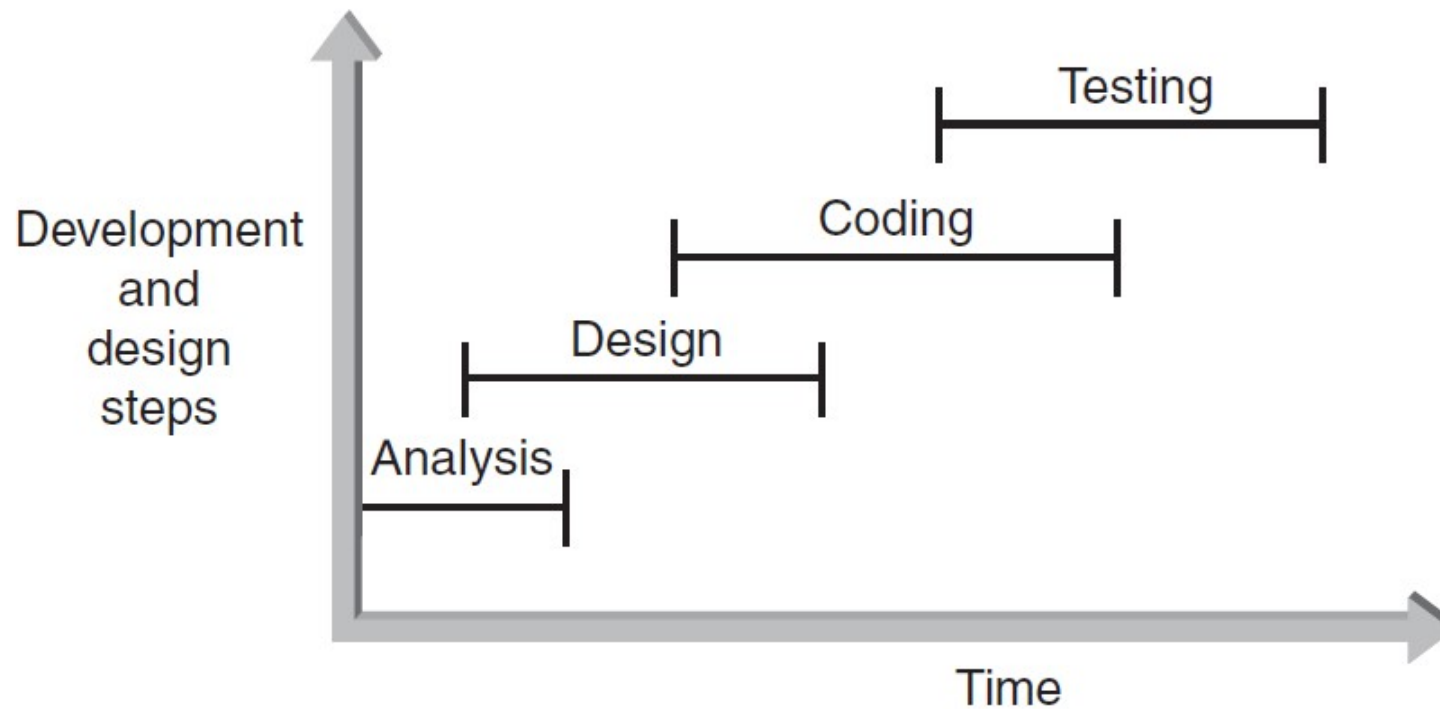
**Figure 1.2** The three phases of program development



# 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

# Phase I: Development and Design (continued)



**Figure 1.3** The development and design steps

# Phase I: Development and Design (continued)

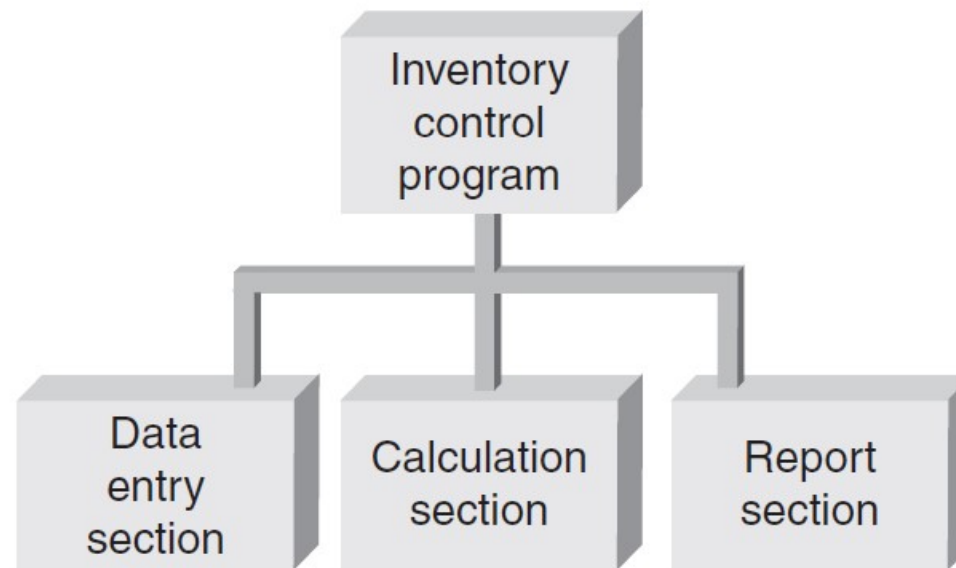
- Step 1: Analyze the Problem
  - Determine and understand the **output** items the program must produce
  - Determine the **input** items
  - Both items referred to as the problem's input/output (I/O)

# Phase I: Development and Design (continued)

- Step 2: Develop a Solution
  - Select the exact set of steps, called an “algorithm,” to solve the problem
  - Refine the algorithm
    - Start with initial solution in the analysis step until you have an acceptable and complete solution
  - Check solution

# Phase I: Development and Design (continued)

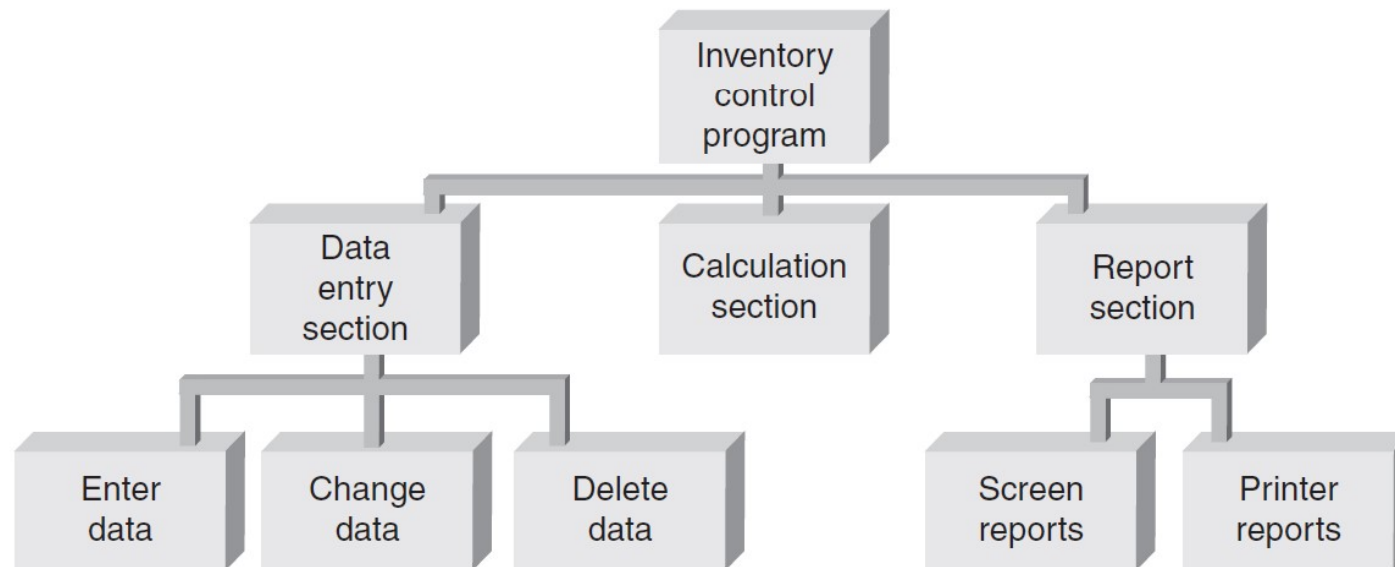
- Step 2: Develop a Solution (continued)
  - Example: a first-level structure diagram for an inventory tracking system



**Figure 1.4** A first-level structure diagram

# Phase I: Development and Design (continued)

- Step 2: Develop a Solution (continued)
  - Example: a second-level structure diagram for an inventory tracking system with further refinements



**Figure 1.5** A second-level structure diagram

# Phase I: Development and Design (continued)

- Step 3: Code the Solution
  - Consists of actually writing a C++ program that corresponds to the solution developed in Step 2
  - Program should contain well-defined patterns or structures of the following types:
    - Sequence
    - Selection
    - Iteration
    - Invocation

# Phase I: Development and Design (continued)

- Step 3: Code the Solution (continued)
  - **Sequence:** Defines the order in which instructions are executed
  - **Selection:** Allows a choice between different operations, based on some condition
  - **Iteration:** Allows the same operation to be repeated based on some condition
    - Also called looping or repetition
  - **Invocation:** Involves invoking a set of statements when needed



# Phase I: Development and Design (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

# 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


# Phase II: Documentation

- Five main documents for every problem solution:
  - Program description
  - Algorithm development and changes
  - Well-commented program listing
  - Sample test runs
  - Users' manual

# Phase III: Maintenance

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

# 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
- Software
  - [Github](#) 
  - [BitBucket](#) 