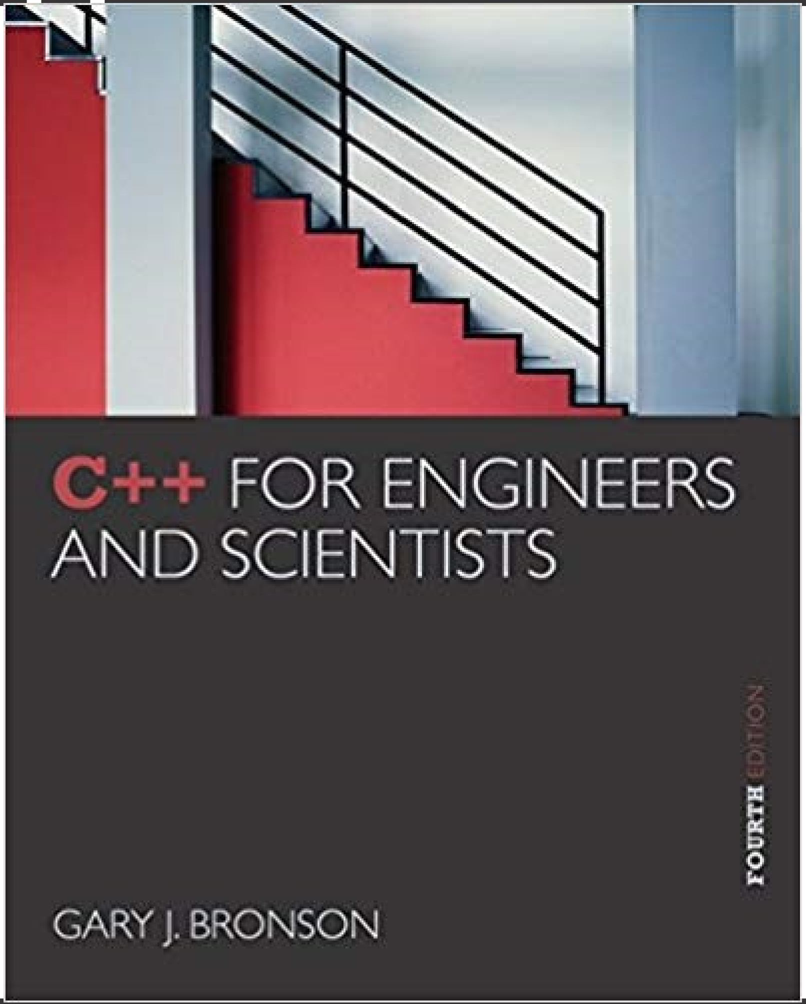
# ELEG 1043
# Computer Applications in Engineering



# 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 Three: Software Development (continued)

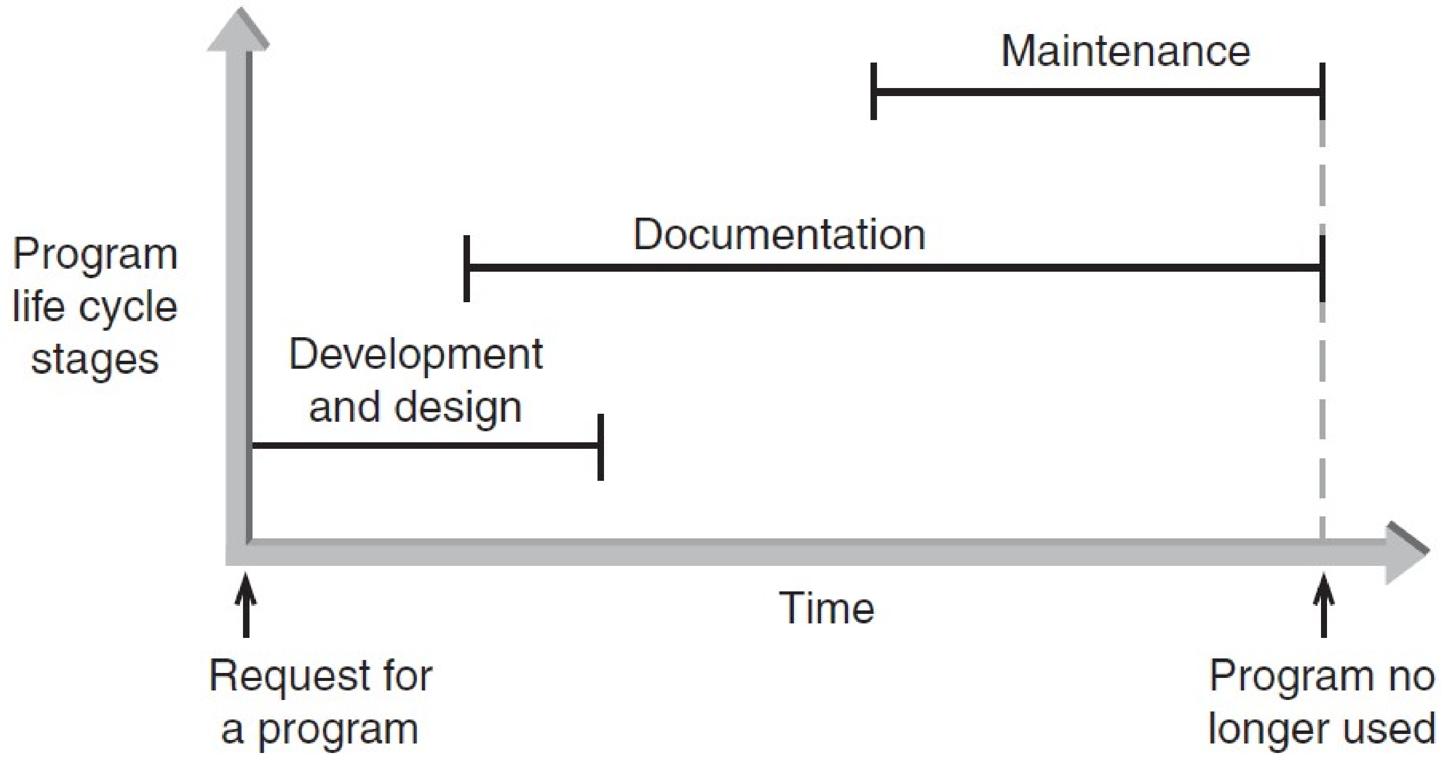


**Figure 1.2** The three phases of program development

## 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

## A Case Study: Radar Speed Trap

- Step 1: Analyze the Problem
  - Understand the desired outputs
  - Determine the required inputs
- Step 2: Develop a Solution
  - Determine the algorithms to be used
  - Use top-down approach to design
- Step 3: Code the Solution
- Step 4: Test and Correct the Program

## A Case Study: Radar Speed Trap (continued)

- Analyze the Problem
  - Output: Speed of the car
  - Inputs: Emitted frequency and received frequency
- Develop a Solution
  - Algorithm:
    - Assign values to f0 and f1
    - Calculate and display speed

## 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

# Chapter 2: Problem Solving Using C++

C++ FOR ENGINEERS AND SCIENTISTS

10

## Objectives

In this chapter, you will learn about:

- Modular programs
- Programming style
- Data types
- Arithmetic operations
- Variables and declaration statements
- Common programming errors

## Introduction to C++

- **Modular program:** A program consisting of interrelated segments (or **modules)** arranged in a logical and understandable form
  - Easy to develop, correct, and modify
- Modules in C++ can be classes or functions

## Introduction to C++ (continued)

- **Function:** Accepts an input, processes the input, and produces an output
  - A function's processing is encapsulated and hidden within the function

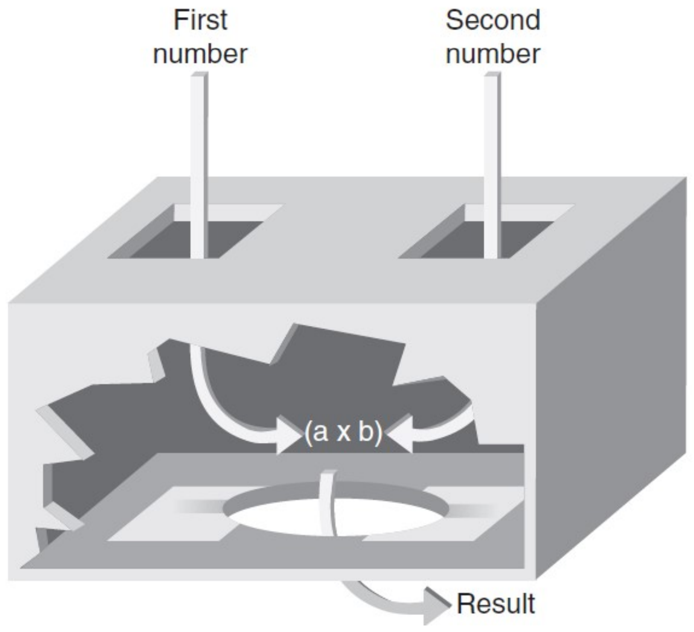


**Figure 2.2** A multiplying function

## Introduction to C++ (continued)

- **Class:** Contains both data and functions used to manipulate the data
- **Identifier:** A name given to an element of the language, such as a class or function
  - Rules for forming identifier names:
    - First character must be a letter or underscore
    - Only letters, digits, or underscores may follow the initial letter (no blanks allowed)
    - Keywords cannot be used as identifiers
    - Maximum length of an identifier = 1024 characters

# Introduction to C++ (continued)

- Examples of valid C++ identifiers:

```
degToRad  intersect   addNums
slope     bessell     multTwo
findMax   density
```

- Examples of invalid C++ identifiers:

| | |
|---|---|
| **`1AB3`** | (begins with a number) |
| **`E*6`** | (contains a special character) |
| **`while`** | (this is a keyword) |

## Comments

- **Comments:** Explanatory remarks in the source code added by the programmer
- **Line comment:** Begins with // and continues to the end of the line
  - Example:

```
// this program displays a message
#include <iostream>
using namespace std;

 int main ()
 {
      cout << "Hello there world!"; //displays text
     return 0;
 }
```

## Comments (continued)

- **Block comments:** comments that span across two or more lines
  - Begin with `/*` and end with `*/`
  - Example:

```
/* This is a block comment that
spans
across three lines */
```

## Data Types

- **Data type:** A set of values and the operations that can be applied to these values
- Two fundamental C++ data groupings:
  - **Class data type** (a class): Created by the programmer
  - **Built-in data type** (primitive type): Part of the C++ compiler

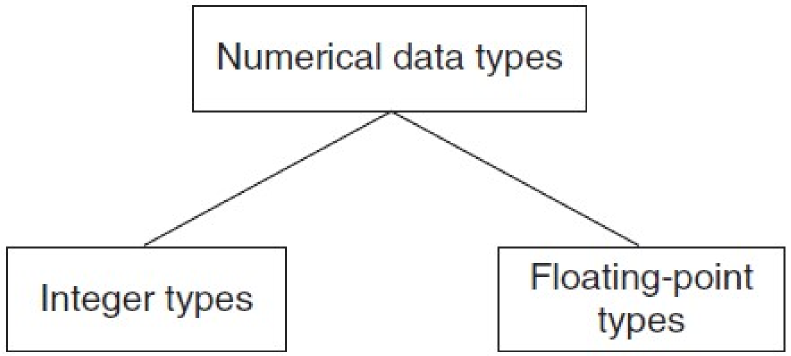


**Figure 2.5** Built-in data types

## Integer Data Types

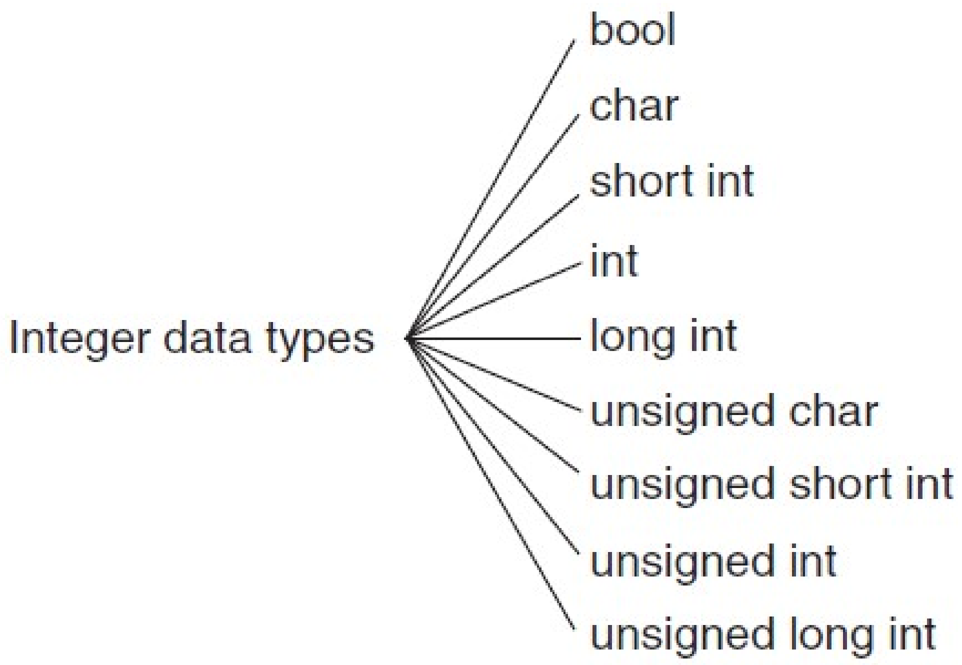


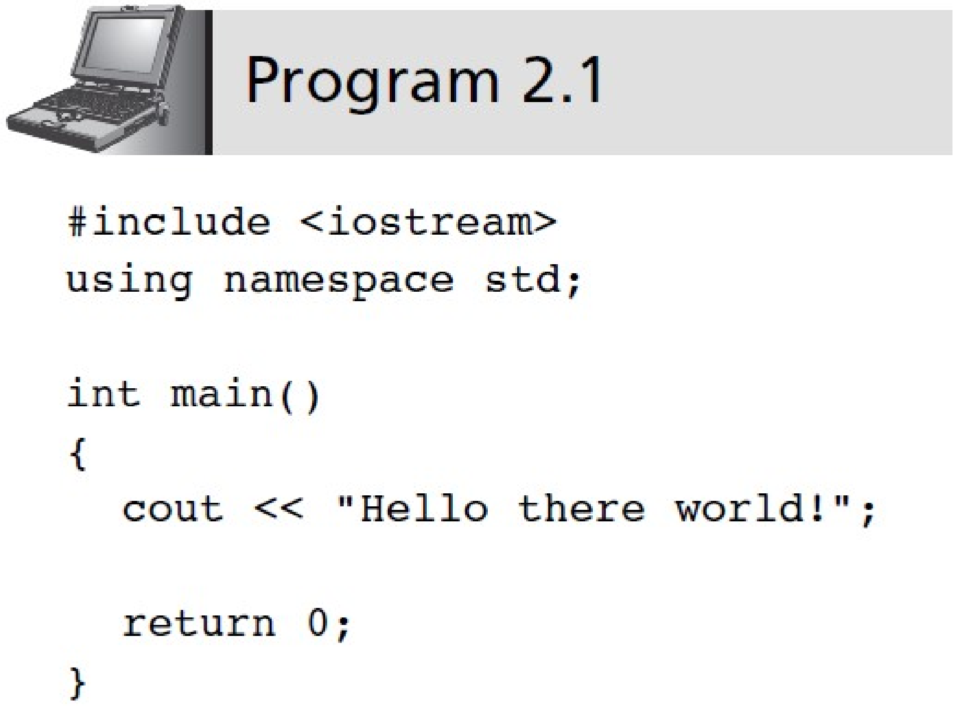
**Figure 2.6** C++ integer data types

## Signed and Unsigned Data Types

- **Signed data type:** One that permits negative, positive, and zero values
- **Unsigned data type:** Permits only positive and zero values
  - An unsigned data type provides essentially double the range of its signed counterpart

## The `cout` Object

- **`cout`** object: An output object that sends data to a standard output display device



Program 2.1

```
#include <iostream>
using namespace std;

int main()
{
  cout << "Hello there world!";

  return 0;
}
```

# Chapter 3: Assignment, Formatting, and Interactive Input

C++ FOR ENGINEERS AND SCIENTISTS

## Objectives

In this chapter, you will learn about:

- Assignment operations
- Formatting numbers for program output
- Using mathematical library functions
- Program input using the **`cin`** object
- Symbolic constants
- A case study involving acid rain
- Common programming errors

# Objectives

In this chapter, you will learn about:

- Assignment operations
- Formatting numbers for program output
- Using mathematical library functions
- Program input using the `cin` object
- Symbolic constants
- A case study involving acid rain
- Common programming errors

# Assignment Operations

- **Assignment Statement:** Assigns the value of the expression on the right side of the = to the variable on the left side of the =
  - int  a  =  2;

    left   right
- Another assignment statement using the same variable will overwrite the previous value with the new value

  Examples:

```
slope = 3.7;
slope = 6.28;  (Overwrite)
```

## Assignment Operations (continued)

Program 3.1

```
// This program calculates the volume of a cylinder,
// given its radius and height
#include <iostream>
using namespace std;

int main()
{
  double radius, height, volume;
  radius = 2.5;
  height = 16.0;
  volume = 3.1416 * radius * radius * height;
  cout << "The volume of the cylinder is " << volume << endl;

  return 0;
}
```

## Assignment Operations (continued)

- Additional assignment operators provide short cuts: **`+=`**, **`-=`**, **`*=`**, **`/=`**, **`%=`**

  Example:

  ```
  sum = sum + 10;
  ```

  is equivalent to: **`sum += 10;`**

  ```
  price *= rate +1;
  ```

  is equivalent to:

  ```
  price = price * (rate + 1);
  ```

# Assignment Operations (continued)

- **Increment operator ++:** Unary operator for the special case when a variable is increased by 1
- **Prefix increment operator** appears before the variable
  - Example: `++i`
- **Postfix increment operator** appears after the variable
  - Example: `i++`

## Assignment Operations (continued)

- Example: `k = ++n;  //prefix increment`

  is equivalent to:

```
 n = n + 1;  //increment n first
 k = n;         //assign n’s value to k
```

- Example: `k = n++;  //postfix increment`

  is equivalent to

```
 k = n;         //assign n’s value to k
  n = n + 1;  //and then increment n
```

# Assignment Operations (continued)

- **Decrement operator `--`:** Unary operator for the special case when a variable is decreased by 1
- **Prefix decrement operator** appears before the variable
  - Example: `--i;`
- **Postfix decrement operator** appears after the variable
  - Example: `i--;`