Chapter 8: Introduction to High-level Language Programming

Invitation to Computer Science, C++ Version, Third Edition

Objectives

In this chapter, you will learn about:

- High-level languages
- Introduction to C++
- Virtual data storage
- Statement types
- Putting the pieces together

Where Do We Stand?

- Early days of computing
  - Programmers were satisfied with assembly language
    - Programs written by technically oriented people
  - Later decades
    - Programmers demanded a more comfortable programming environment
    - Programs could be written by “nontechie” people

High-level Languages

- High-level programming languages
  - Called third-generation languages
  - Overcame deficiencies of assembly language
  - Programmer didn’t need to manage details of data storage or movement

High-level Languages (continued)

- Expectations of a high-level language program (continued)
  - Programmer can take a macroscopic view of tasks; “primitive operations” can be larger
  - Programs will be portable
  - Code will be closer to standard English and use standard mathematical notation
**Introduction to C++**

- Some components of program in Figure 8.2
  - Comments
    - Give information to human readers of code
  - Include directive
    - The linker includes object code from a library
  - Using directive
    - Tells compiler to look in a namespace for definitions not mentioned in the program

---

**Virtual Data Storage**

- **Identifiers**: names in a programming language
- **Keywords**: have special meanings in C++
- **C++**: case-sensitive, free-format language
- **Data items can be constants or variables**

**Virtual Data Storage (continued)**

- A declaration of a data item tells
  - Whether the item is a constant or a variable
  - The identifier used to name the item
  - The data type of the item
### Statement Types

- Input/output statement
  - Input statement
    - Collects a specific value from the user for a variable within the program
  - Output statement
    - Writes a message or the value of a program variable to the user’s screen or to a file

### Statement Types (continued)

- Assignment statement
  - Assigns a value to a program variable
- Control statement
  - Directs the flow of control
    - Can cause it to deviate from usual sequential flow

### Input/Output Statements

- Example
  - Pseudocode
    - Get value for Radius
  - C++
    - cin >> Radius;
  - cin: input stream
  - Code for extraction operator (>>) and the definition of the cin stream come from the iostream library and std namespace

### Input/Output Statements (continued)

- Example
  - Pseudocode
    - Print the value of Circumference
  - C++
    - cout << Circumference;
  - cout: output stream
  - Code for the insertion operator (<<) and the definition of the cout stream come from the iostream library and std namespace

---

**Figure 8.5**

Some of the C++ Standard Data Types

**Figure 8.6**

A 12-Element Array Hits
### The Assignment Statement

- **General form**
  - **Pseudocode**
    - Set the value of “variable” to “arithmetic expression”
  - **C++**
    - variable = expression:
      1. Expression on the right is evaluated
      2. The result is written into the memory location named on the left

### Control Statements

- **Types of control mechanisms**
  - **Sequential**
    - Instructions are executed in order
  - **Conditional**
    - Choice of which instructions to execute next depends on some condition
  - **Looping**
    - Group of instructions may be executed many times

### Control Statements (continued)

- **Default mode of execution: sequential**
- **Conditional flow of control**
  - Evaluation of a Boolean condition (also called a Boolean expression)
  - Which programming statement to execute next is decided based on the value of the Boolean condition (true or false)

### Control Statements (continued)

- **Conditional flow of control (continued)**
  - if-else statement
    - if (Boolean condition)
      - S1;
      - else
        - S2;
  - if variation of the if-else statement
    - if (Boolean condition)
      - S1;

---

![Figure 8.12](image1.png)

**Figure 8.12**
Conditional Flow of Control (If-Else)

![Figure 8.13](image2.png)

**Figure 8.13**
If-Else with Empty Else
Control Statements (continued)

- Looping (iteration)
  - The loop body may be executed repeatedly based on the value of the Boolean condition
  - while statement
    - while (Boolean condition)
      - S1;

Putting the Pieces Together

- At this point, we can:
  - Perform input and output
  - Assign values to variables
  - Direct the flow of control using conditional statements or looping
- For a complete program, we need to:
  - Assemble the statements in the correct order
  - Fill in the missing pieces

Meeting Expectations

- C++ meets the four expectations for a high-level programming language

- Expectations
  - Programmer need not manage details of the movement of data items within memory, nor pay any attention to where they are stored

Meeting Expectations (continued)

- Expectations (continued)
  - Programmer can take a macroscopic view of tasks, thinking at a higher level of problem-solving
  - Programs written in high-level languages will be portable rather than machine-specific
  - Programming statements in a high-level language
    - Will be closer to standard English
    - Will use standard mathematical notation

Managing Complexity: Divide and Conquer

- Divide and conquer
  - To solve a problem, divide it into smaller pieces
- In a computer program
  - Divide the code into modules (subprograms), each doing a part of the overall task
  - Empower these modules to work together to solve the original problem
Using Functions

- Function
  - A module of code in C++
  - Named using ordinary C++ identifiers
- Subtask functions: optional
- The main function: mandatory

Using Functions (continued)

- To invoke a subtask function, the main function gives
  - Name of the function
  - Argument list for the function
- Argument list: list of identifiers for variables that concern that function
- Any function can have its own constant and variable declarations

Writing Functions

- A function header consists of:
  - Return indicator: classifies a function as a void or a nonvoid function
  - Function identifier
  - Parameter list
- By default, arguments in C++ are passed by value

Some C++ Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
<th>Term</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local variable</td>
<td>Declared and known only within a function</td>
<td>Global constant</td>
<td>Declared outside any function and known everywhere</td>
</tr>
<tr>
<td>Argument passed by value</td>
<td>Function receives a copy of the value and can make no permanent changes to the value</td>
<td>Argument passed by reference</td>
<td>Function gets access to memory location where the value is stored; changes it makes value to the value persist after control returns to main function</td>
</tr>
<tr>
<td>Void function</td>
<td>Performs a task, function invocation is a complete C++ statement</td>
<td>Nonvoid function</td>
<td>Requires a value; must include a return statement; function invocation is used within another C++ statement</td>
</tr>
</tbody>
</table>
Object-Oriented Programming

- Object-oriented programming (OOP)
  - A program is a simulation of some part of the world that is the domain of interest
  - Each object is an example drawn from a class of similar objects
- Key elements of OOP
  - Encapsulation
    - A class consists of its subtask modules and its properties
    - Both are “encapsulated” in the class

Key elements of OOP (continued)

- Inheritance
  - Once a class A of objects is defined, a class B of objects can be defined as a “subclass” of A
- Polymorphism
  - One name, the name of the service to be performed, has several meanings, depending on the class of the object providing the service

What Have We Gained?

- Two major advantages of OOP
  - Software reuse
  - A more natural “world view”

Graphical Programming: Graphics Primitives

- Bitmapped display
  - The screen is made up of thousands of pixels, laid out in a two-dimensional grid
- Frame buffer
  - Memory that stores the actual screen image
- The terminal hardware displays on the screen the frame buffer value of every individual pixel

- Graphics library
  - Software containing a collection of functions that control the setting and clearing of pixels
  - Virtually all modern programming languages come with a graphics library

Figure 8.34
Pixel Numbering System in a Bitmapped Display
The Big Picture: Software Engineering

- **Software life cycle**
  - Overall sequence of steps needed to complete a large-scale software project
  - Implementation represents a relatively small part of the cycle

---

Scaling Up

- Programs written by students
  - No longer than a few hundred lines
- Real-world programs
  - 2, 3, or 4 orders of magnitude larger
- Large-scale software development
  - Extensive planning and design needed
  - A team of programmers needed
  - “Software engineering”

---

The Software Life Cycle

- Each step in the software development life cycle
  - Has a specific purpose and activities
  - Should result in a written document
  - The feasibility study
  - Problem specification
  - Program design

---

The Software Life Cycle (continued)

- Algorithm selection or development, and analysis
- Coding
- Debugging
- Testing, verification, and benchmarking
- Documentation
- Maintenance

---

Modern Environments

- Integrated Development Environment (IDE) speeds development by providing
  - A text editor
  - A file manager
  - A compiler
  - A linker and loader
  - Tools for debugging
In a high-level language, the programmer:

- Need not manage storage
- Can think about the problem at a higher level
- Can use more powerful and more natural-language-like program instructions
- Can write a much more portable program

C++ is an object-oriented, high-level programming language

- if-else statement creates a conditional flow of control
- while loop can be used for iteration
- Software life cycle: overall sequence of steps needed to complete a large-scale software project