About the Presentations

- The presentations cover the objectives found in the opening of each chapter
- All chapter objectives are listed in the beginning of each presentation
- You may customize the presentations to fit your class needs
- Some figures from the chapters are included. A complete set of images from the book can be found on the Instructor Resources CD-ROM and companion Website at login.cengage.com

Introduction to Programming in C++
Seventh Edition

Chapter 1:
An Introduction to Programming

Chapter Objectives

- Define the terminology used in programming
- Explain the tasks performed by a programmer
- Understand the employment opportunities for programmers and software engineers
- Explain the history of programming languages
- Explain the sequence, selection, and repetition structures
- Write simple algorithms using the sequence, selection, and repetition structures

Programming a Computer

It is important to understand the relationship between the terms programs, programmers, and programming languages.

Programs - The directions that humans give to computers

Programmers - The people who create these directions

Programming Languages - Special languages used by programmers to communicate directions to a computer

The Programmer's Job

- Programmers help solve computer problems
- Employee or freelance
- Typical steps involved
  - Meet with user to determine problem
  - Convert the problem into a program
  - Test the program
  - Provide user manual

What Traits Should a Software Developer Possess?

- Analytical skills
- Communication skills
- Creativity
- Customer-service skills
- Detail oriented
- Problem-solving skills
- Teamwork
- Technical skills

Employment Opportunities

- Computer software engineer: designs an appropriate solution to a user's problem
- Computer programmer: codes a computer solution
- **Coding** is the process of translating a computer solution into a language a computer can understand
- Some positions call for both engineering and programming

A Brief History of Programming Languages

There are many different types of programming languages. This chapter will discuss:

- Machine languages
- Assembly languages
- High-level procedure-oriented languages
- High-level object-oriented languages

Machine Languages

- The first programmers had to write the program instructions using only combinations of 0s and 1s
  - Example: 0000 0101 1100 0000
- Instructions written in 0s and 1s are called **machine language** or **machine code**
- Each type of machine has its own language
- Machine languages are the only way to communicate directly with the computer
- Programming in machine language: tedious and error-prone; requires highly trained programmers

Assembly Languages

- **Assembly languages** made writing code simpler than using only 0s and 1s
- **Mnemonics** – symbols used to represent the actual machine language instructions
  Example: 00000101 vs. BALR
- Assembly programs require an **assembler** to convert instructions into machine code
- Easier to write programs in assembly language
  - But still tedious and requires highly trained programmers

High-Level Languages

- **High-level languages** allow programmers to use English-like instructions
  Example: `taxAmount = total * taxRate`
- Each high-level language instruction is equivalent to more than one machine language instruction
- **Compilers** translate high-level instructions into 0s and 1s (machine language)
- **Interpreters** translate the program line by line as the program is running

High-Level Languages (cont.)

- When writing a **procedure-oriented program**, the programmer concentrates on the major tasks that the program needs to perform
  - Examples: COBOL, BASIC, C
- An **object-oriented program** requires the programmer to focus on the objects that the program can use to accomplish its goal
  - Examples: C++, Visual Basic, Java, C#
- Object-oriented programs allow for an object to be created that can be reused in more than one program

Control Structures

All computer programs are written using one or more of three basic **control structures**: **sequence**, **repetition**, and **selection**. Another term used for control structures are **logic structures**, because they control the logic flow of the program.

While in every program that is written the sequence structure will be used, in most all programs all three control structures will be used.

The Sequence Structure

- The **sequence structure** directs the computer to process the program instructions, one after another, in the order in which they are listed in the program
- An **algorithm** is a finite number of step-by-step instructions that accomplish a task
- Example: steps to pump gas at a self-service pump

The Sequence Structure (cont.)

Harold is standing outside facing a park bench. The bench is two steps away from Harold. Write the instructions that direct Harold to sit down on the bench.

1. walk forward
2. walk forward
3. turn left 90 degrees
4. turn left 90 degrees
5. sit down on the bench

algorithm

Figure 1-1 An example of the sequence structure

The Selection Structure

- The **selection structure** directs the computer to make a decision (evaluate a condition), and then take an appropriate action based upon that decision
- The selection structure allows the programmer to evaluate data, therefore properly controlling the logic flow of the program
- Another name for the selection structure is the **decision structure**
- Example: stopping or going at a signal light

The Selection Structure (cont.)

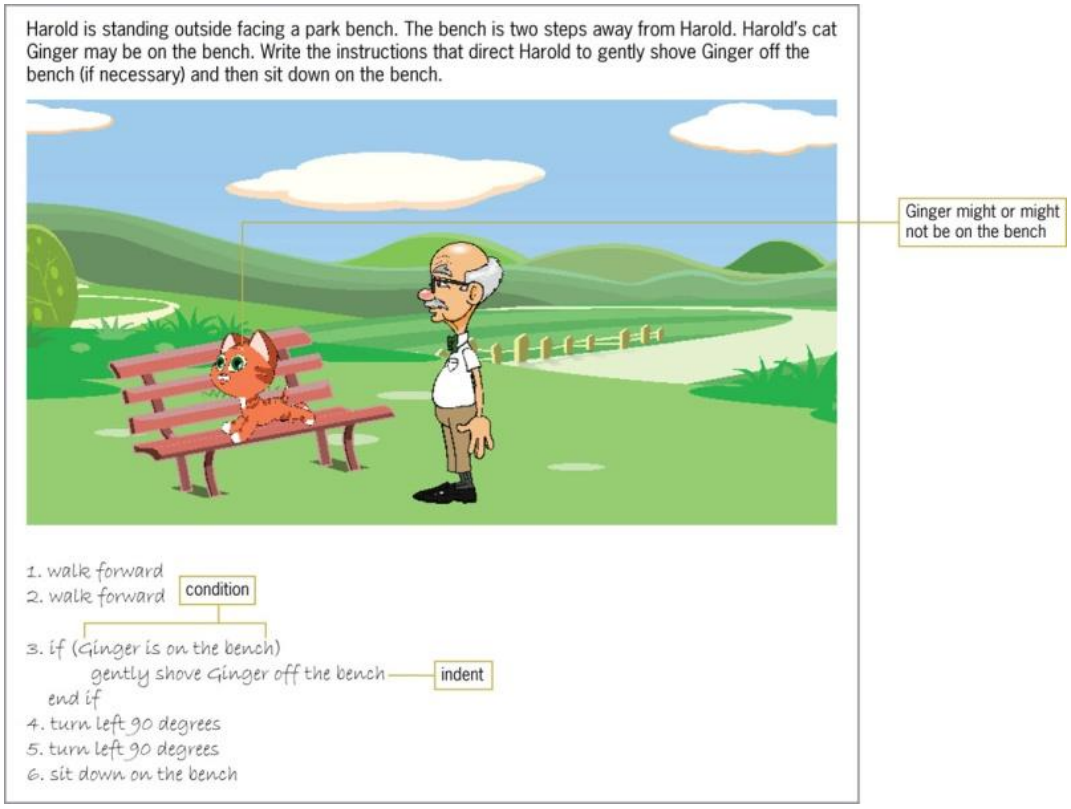

Figure 1-2 An example of the selection structure

The Selection Structure (cont.)

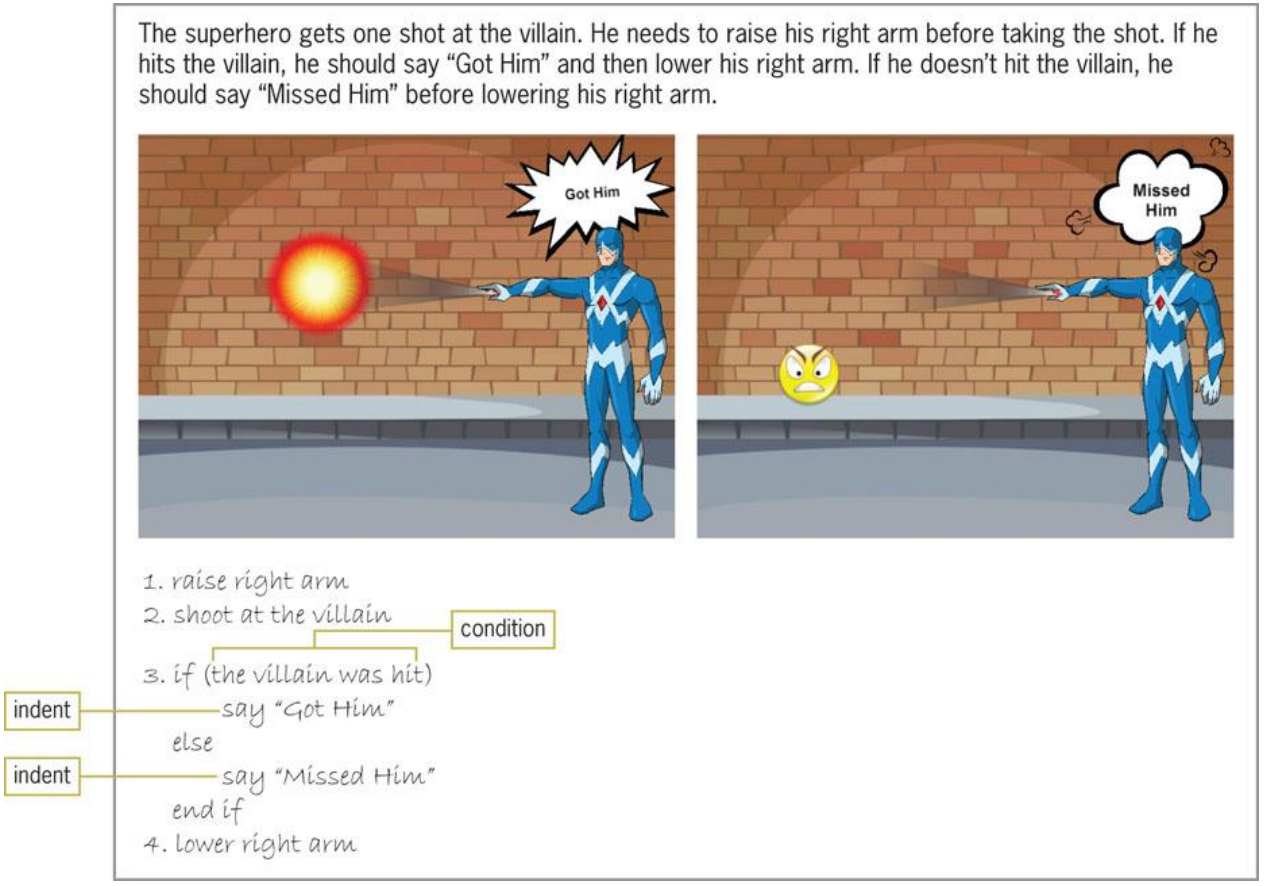

Figure 1-3 Another example of the selection structure

The Repetition Structure

- The **repetition structure**, commonly called **iteration** or **looping**, directs the computer to repeat one or more program instructions until some condition is met
- This condition may be checked at the beginning or end of the set of instructions to be processed dependent upon the language being used
- The repetition structure allows the programmer to repeatedly process a set of instructions, while only typing them in once

The Repetition Structure (cont.)

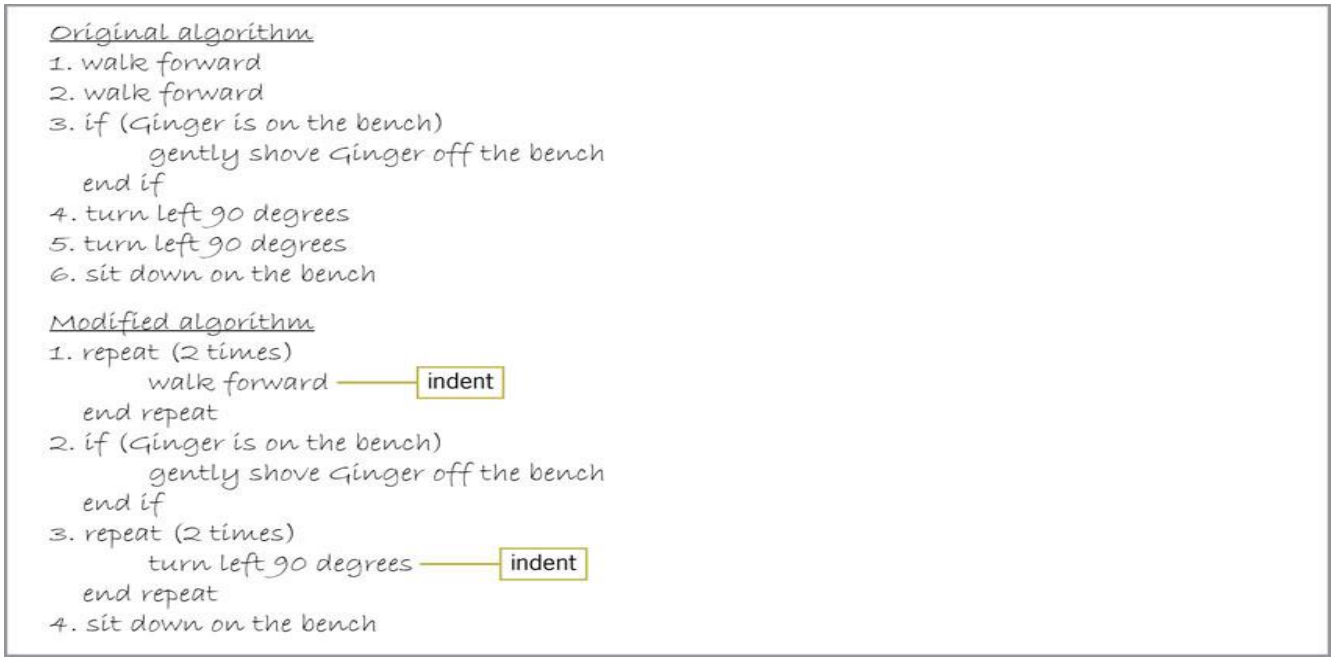

Figure 1-4 Original algorithm and modified algorithm showing the repetition structure

The Repetition Structure (cont.)

- What could you do if you do not know precisely how many steps separate Harold from the boxes as described on the bottom of page 9?

The Repetition Structure (cont.)

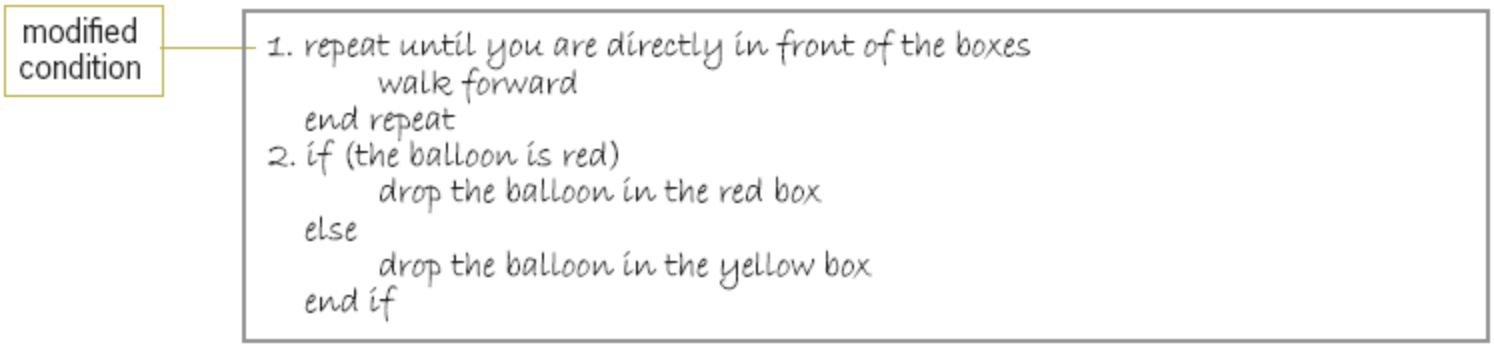

Figure 1-5 Algorithm showing the modified condition in the repetition structure

Summary

- Programs are step-by-step instructions that tell a computer how to perform a task
- Programmers use programming languages to communicate with the computer
- First programming languages were machine language using 0s and 1s
- Assembly languages followed, using mnemonics
- High-level languages can be used to created procedure-oriented or object-oriented programs

Summary (cont.)

- An algorithm is a finite number of step-by-step instructions that accomplish a task
- Algorithms utilize three basic control structures: sequence, selection, and repetition
- The sequence structure directs the computer to process the program instructions, one after another, in the order in which they are listed
- The selection structure directs the computer to make a decision (evaluate a condition), and then take an appropriate action based upon that decision

Summary (cont.)

- The repetition structure, commonly called iteration or looping, directs the computer to repeat one or more program instructions until some condition is met
- The sequence structure is used in all programs
- Most programs also contain both the selection and repetition structures

Lab 1-1: Stop and Analyze

- A local business employs five salespeople and pays a 3% bonus on a salesperson's sales
- Your task is to create a program that calculates the amount of each salesperson's bonus
- The program should print each salesperson's name and bonus amount

```
repeat 5 times
        enter the salesperson's name and sales amount
        calculate the bonus amount by multiplying the sales amount by 3%
        print the salesperson's name and bonus amount
end repeat
```

Lab 1-2: Plan and Create

- Using only the instructions shown below, create an algorithm that shows the steps an instructor takes when grading a test that contains 25 questions

```
end if
end repeat
If (the student's answer is not the same as the correct answer)
mark the student's answer incorrect
read the student's answer and the correct answer
repeat 25 times
```

Lab 1-3: Modify

- Modify the algorithm shown in Lab 1-1 so that it gives a 3.5% bonus to salespeople selling more than $2,000
- All other salespeople should receive a 3% bonus