Computer Engineering Department
Islamic University of Gaza

Dr. Ayman Maliha
amaliha@iugaza.edu.ps
Homepage: http://site.iugaza.edu.ps/amaliha/

ECOM 2403
Computer Programming
Fall 2021-2022



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

1

Chapter 1
Introduction to Computers, Programs, and C++



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

2

What is programming?

- There are many definitions of what computer programming is.
- One that we can use is:

  Programming is how *you* get computers to *solve problems*.

- Without the programmer (**you**), the computer is useless. It does what you tell it to do.
- **Solve problems**: computers are tools.
  - They are complex tools
  - But they are not mysterious or magical
    - they exist to make tasks easier.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Computer Programs

- (or software) are what make computers work.
- Without software
  - Modern computers are just complicated machines for turning electricity into heat.
- Software
  - Runs your operating system, browser, email, games, movie player – just about everything.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Programming

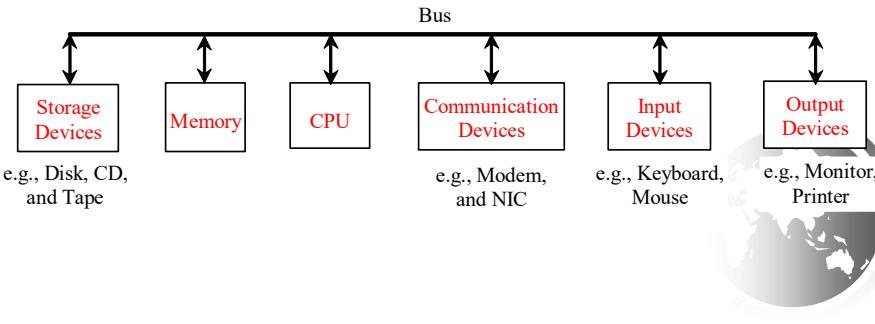
- **Is a creative task**
- There is no right or wrong way to solve a problem
  - There are choices to be made, and one way may seem better than another
    - But that doesn't mean the other is wrong!
- With the right skills and experience, a programmer can create software to solve an unlimited number of problems.
  - from telling you when your next train will arrive to playing your favourite music.
- The possibilities are constrained only by your imagination.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

5

What is a Computer?

A computer consists of a CPU, memory, hard disk, monitor, printer, and communication devices.

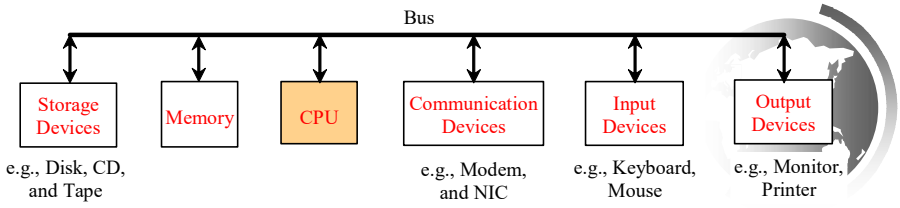


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

6

CPU

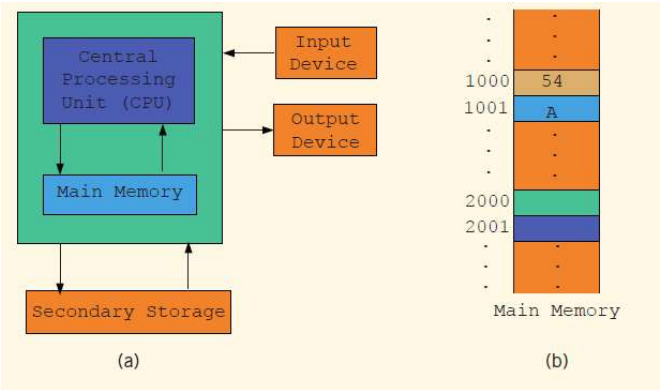
- The central processing unit (CPU) is the brain of a computer.
- It retrieves instructions from memory and executes them.
- The CPU speed is measured in megahertz (MHz), with 1 megahertz equaling 1 million pulses per second.
- The speed of the CPU has been improved continuously.
  - If you buy a PC now, you can get an Intel Pentium 4 Processor at 3 gigahertz (1 gigahertz is 1000 megahertz).



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

7

Central Processing Unit and Main Memory

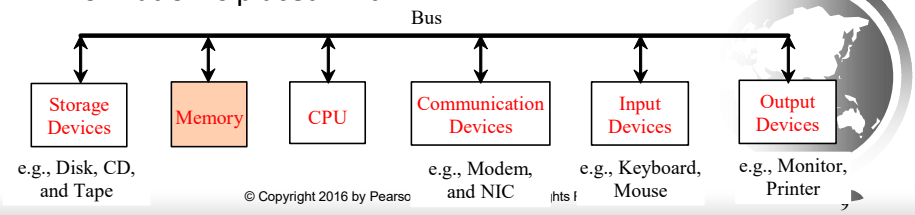


D.S. MALIK Book Figure 1-1

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Memory

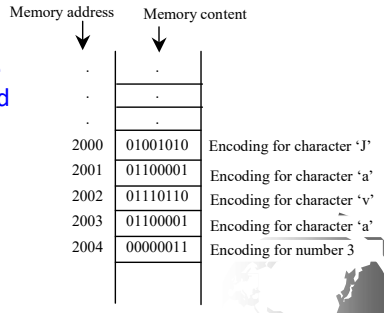
- *Memory* is to store data and program instructions for CPU to execute.
- A memory unit is an ordered sequence of bytes (cells), each holds eight bits.
- A program and its data must be brought to memory before they can be executed.
- A memory byte is never empty, but its initial content may be meaningless to your program.
- The current content of a memory byte is lost whenever new information is placed in it.



© Copyright 2016 by Pearso

How Data is Stored?

- Data can be numbers, characters, and strings
  - All are encoded as a series of bits (zeros and ones).
- Computers use zeros and ones
  - Because digital devices have two stable states, which are referred to as *zero* and *one* by convention.
- The programmers need not to be concerned about the encoding and decoding of data
  - It is performed automatically by the system based on the encoding scheme.

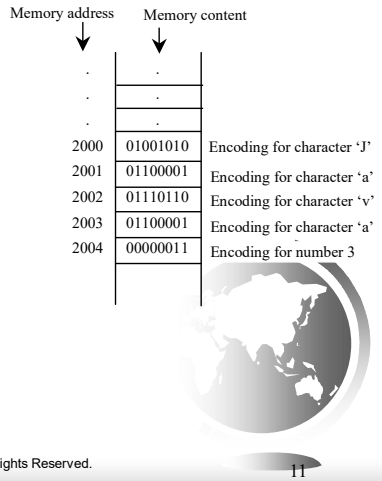


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

10

How Data is Stored?

- The encoding scheme varies.
  - character 'J' is represented by 01001010 in one byte.
- A small number such as three can be stored in a single byte.
  - If computer needs to store a large number that cannot fit into a single byte, it uses a number of adjacent bytes.
- No two data can share or split a same byte.
- A byte is the minimum storage unit in the computer.

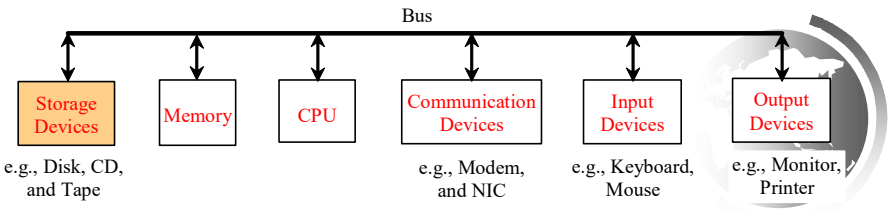


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

11

Storage Devices

- Memory is volatile
  - Because information is lost when the power is off.
- Programs and data are permanently stored on storage devices
  - They are moved to memory when the computer actually uses them.
- There are three main types of storage devices:
  - Magnetic disk drives (hard disks).
  - Optical disc drives (CD-R and CD-RW).
  - USB flash drives.

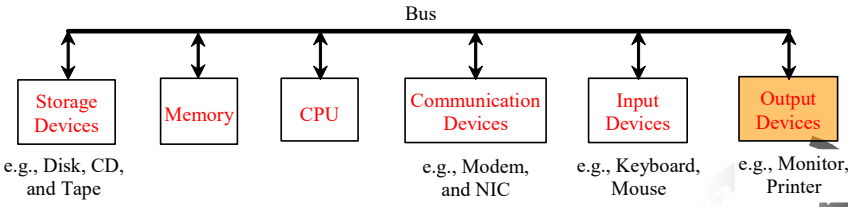


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

12

Output Devices: Monitor

- The monitor displays information (text and graphics).
- The resolution and dot pitch determine the quality of the display.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

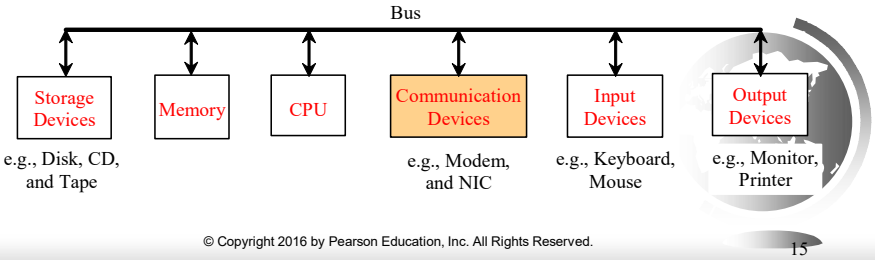
Monitor Resolution and Dot Pitch

- *Resolution*
  - Specifies the number of pixels per square inch.
  - Pixels (short for "picture elements") are tiny dots that form an image on the screen.
  - The higher the resolution, the sharper and clearer the image is.
  - PC monitors are usually 15-inch, 17-inch, 19-inch, or 21-inch.
  - For a 15-inch monitor, a comfortable resolution setting would be 640×480 (307,200 pixels).
- *Dot pitch*
  - The *dot pitch* is the amount of space between pixels.
  - The smaller the dot pitch, the better the display.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Communication Devices

- A ***regular modem*** uses a phone line and can transfer data in a speed up to 56,000 bps (bits per second).
- A ***DSL*** (digital subscriber line) also uses a phone line and can transfer data in a speed 20 times faster than a regular modem.
- A ***cable modem*** uses the TV cable line maintained by the cable company. A cable modem is as fast as a DSL.

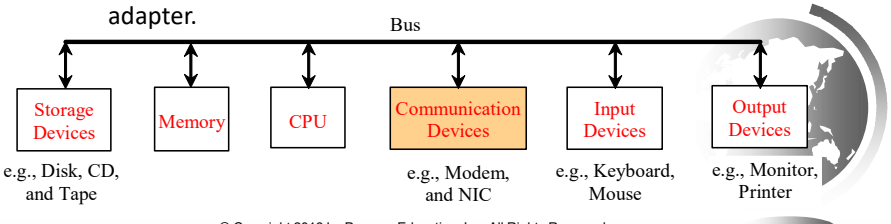


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

15

Communication Devices

- Network interface card (***NIC***) is a device to connect a computer to a local area network (**LAN**).
  - The LAN is commonly used in business, universities, and government organizations.
  - A typical type of NIC, called ***10BaseT***, can transfer data at 10 Mbps (million bits per second). (now 10/100 Mbps)
- ***Wireless networking*** is now extremely popular in homes, businesses, and schools.
  - Every laptop computer sold today is equipped with a wireless adapter.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

16

Software

- Software: programs that do specific tasks
- System programs control the computer
  - Operating system monitors the overall activity of the computer and provides services such as:
    - Memory management
    - Input/output activities
    - Storage management
- Application programs perform a specific task
  - Word processors
  - Spreadsheets
  - Games



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Programs

- Computer *programs*, known as *software*, are instructions to the computer.
- You tell a computer what to do through programs.
- Without programs, a computer is an empty machine.
- Computers do not understand human languages
  - So you need to use computer languages to communicate with them.

Programs are written using programming languages.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

18

Programming Languages

- Byte:
  - A sequence of eight bits
- Kilobyte (KB): 2^{10} bytes = 1024 bytes
- ASCII (American Standard Code for Information Interchange)
  - 128 characters
  - A is encoded as 1000001 (66th character)
  - 3 is encoded as 0110011



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Programming Languages

TABLE 1-1 Binary Units

Unit	Symbol	Bits/Bytes
Byte		8 bits
Kilobyte	KB	2^{10} bytes = 1024 bytes
Megabyte	MB	1024 KB = 2^{10} KB = 2^{20} bytes = 1,048,576 bytes
Gigabyte	GB	1024 MB = 2^{10} MB = 2^{30} bytes = 1,073,741,824 bytes
Terabyte	TB	1024 GB = 2^{10} GB = 2^{40} bytes = 1,099,511,627,776 bytes
Petabyte	PB	1024 TB = 2^{10} TB = 2^{50} bytes = 1,125,899,906,842,624 bytes
Exabyte	EB	1024 PB = 2^{10} PB = 2^{60} bytes = 1,152,921,504,606,846,976 bytes
Zettabyte	ZB	1024 EB = 2^{10} EB = 2^{70} bytes = 1,180,591,620,717,411,303,424 bytes

D.S. MALIK BookvTable 1-1



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Programming Languages

- EBCDIC
  - Used by IBM
  - 256 characters
- Unicode
  - 65536 characters
  - Two bytes are needed to store a character



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Programming Languages

Machine Language Assembly Language High-Level Language

- Machine language is a set of primitive instructions built into every computer.
- The instructions are in the form of binary code.
- Early computers were programmed in machine language
- To calculate wages = rate * hours in machine language:

```
100100 010001    //Load
100110 010010    //Multiply
100010 010011    //Store
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

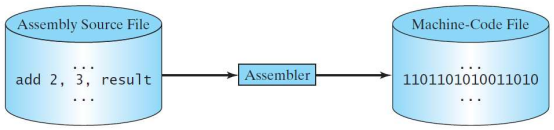
22

Programming Languages

Machine Language Assembly Language High-Level Language

- Assembly languages were developed to make programming easy.
- Assembly language instructions are mnemonic
- Assembler: translates a program written in assembly language into machine language
- Using assembly language instructions, adding two numbers may be written as an instruction in assembly code like this:

ADDF3 R1, R2, R3



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

23

Programming Languages

Machine Language Assembly Language High-Level Language

- The high-level languages are English-like and easy to learn and program.
- For example, the following is a high-level language statement that computes the area of a circle with radius 5:
- area = 5 * 5 * 3.1415;



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

24

Popular High-Level Languages

- COBOL (COmmon Business Oriented Language)
- FORTRAN (FORmula TRANslation)
- BASIC (Beginner All-purpose Symbolic Instructional Code)
- Pascal (named for Blaise Pascal)
- Ada (named for Ada Lovelace)
- C (whose developer designed B first)
- Visual Basic (Basic-like visual language developed by Microsoft)
- Delphi (Pascal-like visual language developed by Borland)
- C++ (an object-oriented language, based on C)
- Java (a popular object-oriented language, similar to C++)
- C# (a Java-like developed my Microsoft)



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Interpreting/Compiling Source Code

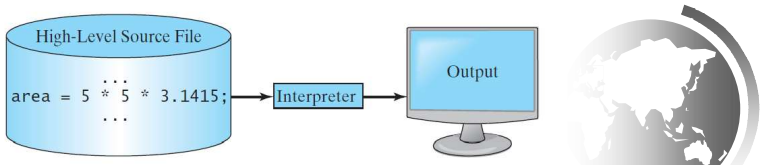
- A program written in a high-level language is called a ***source program*** or ***source code***.
- A computer cannot understand a source program
  - A source program must be translated into machine code for execution.
- The translation can be done using another programming tool called an ***interpreter*** or a ***compiler***.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Interpreting Source Code

- An interpreter reads one statement from the source code (execute one-by-one)
  - Translates it to the machine code or virtual machine code
  - Then executes it right away
- Note that a statement from the source code may be translated into several machine instructions.

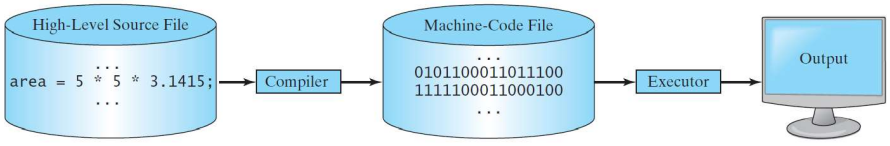


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

27

Compiling Source Code

- A compiler translates the entire source code into a machine-code file, and the machine-code file is then executed.

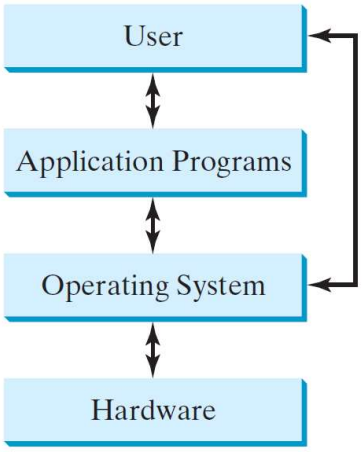


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

28

Operating Systems

- *Operating system* (OS) is a program that manages and controls a computer's activities.
  - The popular operating systems for general-purpose computers are Microsoft Windows, Mac OS, and Linux.
- Application programs, such as a Web browser or a word processor, cannot run unless an operating system is installed and running on the computer.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

History of C++

- C, C++, Java, and C# are very similar. C++ evolved from C. Java was modeled after C++. C# is a subset of C++ with some features similar to Java. If you know one of these languages, it is easy to learn the others.
- C evolved from the B language and the B language evolved from the BCPL language. BCPL was developed by Martin Richards in the mid-1960s for writing operating systems and compilers.
- C++ is an extension of C, developed by Bjarne Stroustrup at Bell Labs during 1983-1985. C++ added a number of features that improved the C language. Most importantly, it added the object-oriented programming.
- An international standard for C++ was created by American National Standards Institute (ANSI) in 1998 (C++98). C++11 was adopted just recently.

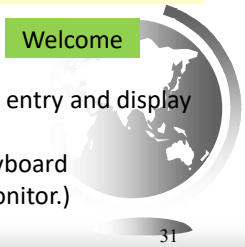
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

A Simple C++ Program

Let us begin with a simple C++ program that displays the message "Welcome to C++!" on the console.

```
#include <iostream>
using namespace std;
int main()
{
  // Display Welcome to C++ to the console
  cout << "Welcome to C++!" << endl;
  return 0;
}
```

```
Welcome to C++!
```



- Console is an old computer term that refers to the text entry and display device of a computer.
  - Console input means to receive input from the keyboard
  - console output means to display output on the monitor.)

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Special Characters

TABLE 1.2 Special Characters

Character	*Name*	*Description*
#	Pound sign	Used in `#include` to denote a preprocessor directive.
<>	Opening and closing angle brackets	Encloses a library name when used with `#include`.
()	Opening and closing parentheses	Used with functions such as `main()`.
{}	Opening and closing braces	Denotes a block to enclose statements.
//	Double slashes	Precedes a comment line.
<<	Stream insertion operator	Outputs to the console.
" "	Opening and closing quotation marks	Wraps a string (i.e., sequence of characters).
;	Semicolon	Marks the end of a statement.

```
#include <iostream>
using namespace std;
int main()
{
  // Display Welcome to C++ to the console
  cout << "Welcome to C++!" << endl;
  return 0;
}
```



Multiple Line Comments

```
// This application program prints Welcome to C++!
/* This application program prints Welcome to C++! */
/* This application program
   prints Welcome to C++! */
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Extending the Simple C++ Program

- Once you understand the program, it is easy to extend it to display more messages.
- For example, you can rewrite the program to display three messages, as shown.

```
#include <iostream>
using namespace std;
int main()
{
  cout << "Programming is fun!" << endl;
  cout << "Fundamentals First" << endl;
  cout << "Problem Driven" << endl;
  return 0;
}
```

```
Programming is fun!
Fundamentals First
Problem Driven
```

WelcomeWithThreeMessages



nts Reserved.

Computing with Numbers

- Further, you can perform mathematical computations and displays the result to the console.

```
#include <iostream>
using namespace std;
int main()
{
   cout << "(10.5 + 2 * 3) / (45 - 3.5) = ";
   cout << (10.5 + 2 * 3) / (45 - 3.5) << endl;

   return 0;
}
```

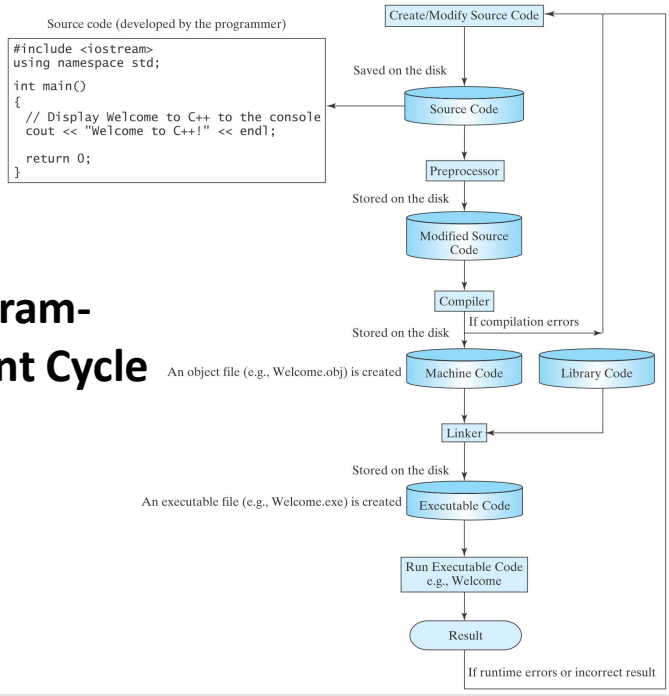
ComputeExpression

```
(10.5 + 2 * 3) / (45 – 3.5) = 0.39759036144578314
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

35

C++ Program-Development Cycle



Execute a C++ Program

- To execute a C++ program:
  - Use an editor to create a source program in C++
  - Preprocessor directives begin with # and are processed by the preprocessor
  - Use the compiler to:
    - Check that the program obeys the language rules
    - Translate into machine language (object program)



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Execute a C++ Program

- To execute a C++ program (cont'd.):
  - Linker:
    - Combines object program with other programs provided by the SDK to create executable code
    - Library: contains prewritten code you can use
  - Loader:
    - Loads executable program into main memory
  - The last step is to execute the program
- Some IDEs do all this with a Build or Rebuild command



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

C++ IDE Tutorial

- You can develop a C++ program from a command window or from an IDE.
- An IDE is software that provides an *integrated development environment* (*IDE*) for rapidly developing C++ programs.
- Editing, compiling, building, debugging, and online help are integrated in one graphical user interface.
- Just enter source code or open an existing file in a window, then click a button, menu item, or function key to compile and run the program.
- Examples of popular IDEs are Microsoft Visual C++, Dev-C++, Eclipse, and NetBeans.
- All these IDEs can be downloaded free.

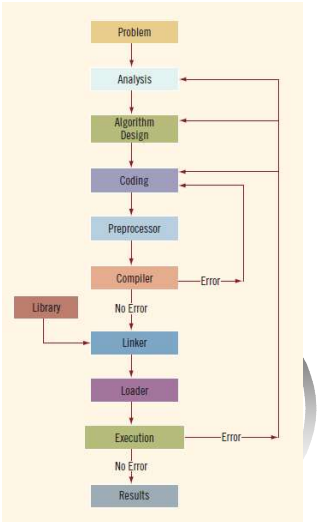


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

39

Programming with the Problem Analysis–Coding–Execution Cycle

- Algorithm:
  - Step-by-step problem-solving process
  - Solution achieved in finite amount of time
- Programming is a process of problem solving



D.S. MALIK Book Figure 1-3

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

The Problem Analysis–Coding–Execution Cycle (cont'd.)

- Step 1: Analyze the problem
  - Outline the problem and its requirements
  - Design steps (algorithm) to solve the problem
- Step 2: Implement the algorithm
  - Implement the algorithm in code
  - Verify that the algorithm works
- Step 3: Maintain
  - Use and modify the program if the problem domain changes



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

The Problem Analysis–Coding–Execution Cycle (cont'd.)

- Thoroughly understand the problem and all requirements
  - Does program require user interaction?
  - Does program manipulate data?
  - What is the output?
- If the problem is complex, divide it into subproblems
  - Analyze and design algorithms for each subproblem
- Check the correctness of algorithm
  - Can test using sample data
  - Some mathematical analysis might be required

Analyzing the problem is the first, and most important, step in the process.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

The Problem Analysis–Coding–Execution Cycle (cont'd.)

- Once the algorithm is designed and correctness verified
  - Write the equivalent code in high-level language
- Enter the program using text editor



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

The Problem Analysis–Coding–Execution Cycle (cont'd.)

- Run code through compiler
- If compiler generates errors
  - Look at code and remove errors
  - Run code again through compiler
- If there are no syntax errors
  - Compiler generates equivalent machine code
- Linker links machine code with system resources



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

The Problem Analysis–Coding–Execution Cycle (cont'd.)

- Once compiled and linked, loader can place program into main memory for execution
- The final step is to execute the program
- Compiler guarantees that the program follows the rules of the language
  - Does not guarantee that the program will run correctly



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example 1-1

- Design an algorithm to find the perimeter and area of a rectangle
- The perimeter and area of the rectangle are given by the following formulas:

```
perimeter = 2 * (length + width)
area = length * width
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example 1-1 (cont'd.)

- Algorithm:
  - Get length of the rectangle
  - Get width of the rectangle
  - Find the perimeter using the following equation:

```
perimeter = 2 * (length + width)
```

  - Find the area using the following equation:

```
area = length * width
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example 1-2

- In this example, we design an algorithm that calculates the sales tax and the price of an item sold in a particular state.
- The sales tax is calculated as follows:
  - The state's portion of the sales tax is 4%, and the city's portion of the sales tax is 1.5%.
  - If the item is a luxury item, such as a car more than $50,000, then there is a 10% luxury tax.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example 1-2 (cont'd.)

- The `stateSalesTax` and `citySalesTax` can be calculated using the following formulas:

```
stateSalesTax = salePrice · 0.04
citySalesTax = salePrice · 0.015
```

- Next, you can determine `luxuryTax` as follows:
- if (item is a luxury item)

```
luxuryTax = salePrice · 0.1
```

otherwise

```
luxuryTax = 0
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example 1-2 (cont'd.)

- Next, you can determine `salesTax` as follows:

```
salesTax = stateSalesTax + citySalesTax + luxuryTax
```

- Finally, you can calculate `amountDue` as follows:

```
amountDue = salePrice + salesTax
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example 1-2 (cont'd.)

- Algorithm:
  1. Get the selling price of the item.
  2. Determine whether the item is a luxury item.
  3. Find the state's portion of the sales tax using the formula:

     `stateSalesTax = salePrice · 0.04`

  4. Find the city's portion of the sales tax using the formula:

     `citySalesTax = salePrice · 0.015`



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example 1-2 (cont'd.)

- Algorithm:
  5. Find the luxury tax using the following formula:

     if (item is a luxury item)

     `luxuryTax = salePrice · 0.1`

     otherwise

     `luxuryTax = 0`

  6. Find `salesTax` using the formula:

     `salesTax = stateSalesTax + citySalesTax + luxuryTax`

  7. Find `amountDue` using the formula:

     `amountDue = salePrice + salesTax`



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example 1-3

- Calculate each student's grade
  - 10 students in a class; each student has taken five tests; each test is worth 100 points
- Design algorithms to:
  - Calculate the grade for each student and class average
  - Find the average test score
  - Determine the grade
- Data: students' names; test scores



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example 1-3 (cont'd.)

- Algorithm to determine the average test score:
  - Get the five test scores
  - Add the five test scores
    - Suppose sum stands for the sum of the test scores
  - Suppose average stands for the average test score:
    - `average = sum / 5;`



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example 1-3 (cont'd.)

- Algorithm to determine the grade:

```
 if average is greater than or equal to 90
    grade = A
otherwise
  if average is greater than or equal to 80 and less than 90
       grade = B
otherwise
  if average is greater than or equal to 70 and less than 80
       grade = C
otherwise
  if average is greater than or equal to 60 and less than 70
       grade = D
otherwise
       grade = F
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example 1-3 (cont'd.)

- Main algorithm is as follows:
  - `totalAverage` = 0;
  - Repeat the following for each student:
    - Get student's name
    - Use the algorithm to find the average test score
    - Use the algorithm to find the grade
    - Update `totalAverage` by adding current student's average test score
  - Determine the class average as follows:
    - `classAverage = totalAverage / 10`



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Programming Methodologies

- Two popular approaches to programming design
  - Structured
  - Object-oriented



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Structured Programming

- Structured design:
  - Dividing a problem into smaller subproblems
- Structured programming:
  - Implementing a structured design
- The structured design approach is also called:
  - Top-down (or bottom-up) design
  - Stepwise refinement
  - Modular programming



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Object-Oriented Programming

- Object-oriented design (OOD)
  - Identify components called objects
  - Determine how objects interact with each other
- Specify relevant data and possible operations to be performed on that data
- Each object consists of data and operations on that data



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Programming Style and Documentation

- Appropriate Comments
- Proper Indentation and Spacing Lines

```
cout << 3+4*4;          <-- Bad style
cout << 3 + 4 * 4;      <-- Good style
```

- Block Styles

```
#include <iostream>
using namespace std;
int main()
{
  // Display Welcome to C++ to the console
  cout << "Welcome to C++!" << endl;
  return 0;
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Programming Errors

- Syntax Errors
  - Detected by the compiler
- Runtime Errors
  - Causes the program to abort
- Logic Errors
  - Produces incorrect result



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Syntax Errors

```
#include <iostream>
using namespace std
int main()
{
   cout << "Programming is fun << endl;
   return 0;
}
```

```
1>Test.cpp(4): error C2144: syntax error : 'int' should be preceded by ';'
1>Test.cpp(6): error C2001: newline in constant
1>Test.cpp(8): error C2143: syntax error : missing ';' before 'return'
```

ShowSyntaxErrors



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Runtime Errors

```
#include <iostream>
using namespace std;
int main()
{
   int i = 4;
   int j = 0;
   cout << i / j << endl;
   return 0;
}
```

ShowRuntimeErrors



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Logic Errors

ShowLogicErrors

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Celsius 35 is Fahrenheit degree " << endl;
    cout << (9 / 5) * 35 + 32 << endl;
    return 0;
}
```

```
Celsius 35 is Fahrenheit degree
67
```

It should be 95.



you need to use 9.0 / 5, which results in 1.8.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Errors

- Common Error 1: Missing Braces

```
int main()
{

}
```

← Type this closing brace right away to match the opening brace

- Common Error 2: Missing Semicolons

```
int main()
{
      cout << "Programming is fun!" << endl;
      cout << "Fundamentals First" << endl;
      cout << "Problem Driven" << endl
}
```

Missing a semicolon



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Errors

- Common Error 3: Missing Quotation Marks

```
cout << "Problem Driven;
```

Missing a quotation mark

- Common Error 4: Misspelling Names

```
1  int Main()
2  {
3    cout << (10.5 + 2 * 3) / (45 - 3.5) << endl;
4    return 0;
5  }
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

References

- Introduction to Programming with C++, Daniel Liang, Pearson Education Limited 2014
- C++ Programming- From Problem Analysis to Program Design - D.S. MALIK - 2018



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Computer Engineering Department
Islamic University of Gaza

Dr. Ayman Maliha
amaliha@iugaza.edu.ps
Homepage: http://site.iugaza.edu.ps/amaliha/

ECOM 2403
Computer Programming
Fall 2021-2022



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Chapter 2
Elementary Programming



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Introduction

- Computer program
  - Sequence of statements whose objective is to accomplish a task
- Programming
  - Process of planning and creating a program
- Real-world analogy: a recipe for cooking



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

*Writing a program involves **designing a strategy** for solving the problem then using a programming language to implement that strategy.*



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

4

Introducing Programming with an Example

Write a program to compute the area of a circle with radius 20.

```
The area is 1256.64
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

5

1. Read in the circle's radius.
2. Compute the area using the following formula:
   area = *radius* * *radius* * ∏
3. Display the result.

Algorithms can be described in natural languages or in *pseudocode*

```
int main()
{
  // Step 1: Read in radius
  // Step 2: Compute area
  // Step 3: Display the area
}
```

```
int main()
{
  double radius;
  double area;
  // Step 1: Read in radius
  // Step 2: Compute area
  // Step 3: Display the area
}
```



6

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    double radius;
7    double area;
8
9    // Step 1: Read in radius
10   radius = 20;
11
12   // Step 2: Compute area
13   area = radius * radius * 3.14159;
14
15   // Step 3: Display the area
16   cout << "The area is " << area << endl;
17
18   return 0;
19 }
```

```
The area is 1256.64
```

ComputeArea

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

7

animation

Trace a Program Execution

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

int main() {
  double radius;
  double area;

  // Step 1: Read in radius
  radius = 20;

  // Step 2: Compute area
  area = radius * radius * 3.14159;

  // Step 3: Display the area
  cout << "The area is ";
  cout << area << endl;
}
```

allocate memory for radius

radius | no value

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

8

animation

Trace a Program Execution

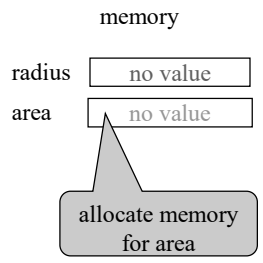
```
#include <iostream>
using namespace std;

int main() {
  double radius;
  double area;

  // Step 1: Read in radius
  radius = 20;

  // Step 2: Compute area
  area = radius * radius * 3.14159;

  // Step 3: Display the area
  cout << "The area is ";
  cout << area << std::endl;
}
```





© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace a Program Execution

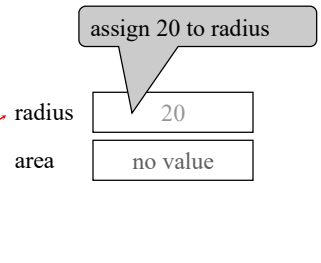
```
#include <iostream>
using namespace std;

int main() {
  double radius;
  double area;

  // Step 1: Read in radius
  radius = 20;

  // Step 2: Compute area
  area = radius * radius * 3.14159;

  // Step 3: Display the area
  cout << "The area is ";
  cout << area << std::endl;
}
```





© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace a Program Execution

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

int main() {
  double radius;
  double area;

  // Step 1: Read in radius
  radius = 20;

  // Step 2: Compute area
  area = radius * radius * 3.14159;

  // Step 3: Display the area
  cout << "The area is ";
  cout << area << std::endl;
}
```

memory

radius	20
area	1256.636

compute area and assign it to variable area



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace a Program Execution

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

int main() {
  double radius;
  double area;

  // Step 1: Read in radius
  radius = 20;

  // Step 2: Compute area
  area = radius * radius * 3.14159;

  // Step 3: Display the area
  cout << "The area is ";
  cout << area << std::endl;
}
```

memory

radius	20
area	1256.636

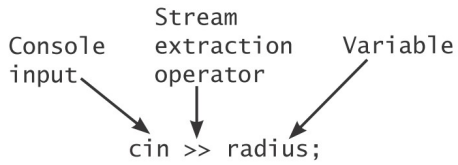
print a message to the console



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Reading Input from the Keyboard

- You can use the `cin` object to read input from the keyboard.



data flows from `cin` to a variable

```
cin >> variable; // cin → variable;
```

data flows from a variable or a string to `cout`

```
cout << "Welcome "; // cout ← "Welcome";
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6     // Step 1: Read in radius
7     double radius;
8     cout << "Enter a radius: ";
9     cin >> radius;
10
11    // Step 2: Compute area
12    double area = radius * radius * 3.14159;
13
14    // Step 3: Display the area
15    cout << "The area is " << area << endl;
16
17    return 0;
18 }
```

```
Enter a radius: 2.5
The area is 19.6349
```

```
Enter a radius: 23
The area is 1661.9
```

ComputeAreaWithConsoleInput

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Reading Multiple Input in One Statement

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6   // Prompt the user to enter three numbers
7   double number1, number2, number3;
8   cout << "Enter three numbers: ";
9   cin >> number1 >> number2 >> number3;
10
11   // Compute average
12   double average = (number1 + number2 + number3) / 3;
13
14   // Display result
15   cout << "The average of " << number1 << " " << number2
16   << " " << number3 << " is " << average << endl;
17
18   return 0;
19 }
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Reading Multiple Input in One Statement

```
Enter three numbers: 1 2 3
The average of 1 2 3 is 2
```

```
Enter three numbers: 10.5
11
11.5
The average of 10.5 11 11.5 is 11
```

ComputeAverage



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

The Basics of a C++ Program

- Function (or subprogram): collection of statements; when executed, accomplishes something
  - May be predefined or standard
- Syntax rules: rules that specify which statements (instructions) are legal or valid
- Semantic rules: determine the meaning of the instructions
- Programming language: a set of rules, symbols, and special words



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Comments

- Comments are for the reader, not the compiler
- Two types:
  - Single line: begin with //

```
// This is a C++ program.
// Welcome to C++ Programming.
```

  - Multiple line: enclosed between /* and */

```
/*
   You can include comments that can
   occupy several lines.
*/
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Reserved Words (Keywords)

- Reserved word symbols (or keywords):
  - Cannot be redefined within program
  - Cannot be used for anything other than their intended use

  Examples:
  - `int`
  - `float`
  - `double`
  - `char`
  - `const`
  - `void`
  - `return`



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Identifiers

- Identifier: the name of something that appears in a program
  - Consists of letters, digits, and the underscore character (_)
  - Must begin with a letter or underscore
- C++ is case sensitive
  - `NUMBER` is not the same as `number`
- An identifier cannot be a reserved word.
- Two predefined identifiers are `cout` and `cin`
  - Unlike reserved words, predefined identifiers may be redefined, but it is not a good idea



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Identifiers (cont'd.)

- Legal identifiers in C++:
  - first
  - conversion
  - payRate

TABLE 2-1 Examples of Illegal Identifiers

Illegal Identifier	Description
`employee Salary`	There can be no space between `employee` and `Salary`.
`Hello!`	The exclamation mark cannot be used in an identifier.
`one+two`	The symbol + cannot be used in an identifier.
`2nd`	An identifier cannot begin with a digit.

D.S. MALIK Book Table 2-1

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Reserved Words

`asm`	`do`	`inline`	`short`	`typeid`
`auto`	`double`	`int`	`signed`	`typename`
`bool`	`dynamic_cast`	`long`	`sizeof`	`union`
`break`	`else`	`mutable`	`static`	`unsigned`
`case`	`enum`	`namespace`	`static_cast`	`using`
`catch`	`explicit`	`new`	`struct`	`virtual`
`char`	`extern`	`operator`	`switch`	`void`
`class`	`false`	`private`	`template`	`volatile`
`const`	`float`	`protected`	`this`	`wchar_t`
`const_cast`	`for`	`public`	`throw`	`while`
`continue`	`friend`	`register`	`true`	
`default`	`goto`	`reinterpret_cast`	`try`	
`delete`	`if`	`return`	`typedef`	

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Whitespaces

- Every C++ program contains whitespaces
  - Include blanks, tabs, and newline characters
- Used to separate special symbols, reserved words, and identifiers
- Proper utilization of whitespaces is important
  - Can be used to make the program more readable



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Variables

Variable is a memory location whose content can be changed

```
1  // Compute the first area
2  radius = 1.0;                                          radius:  1.0
3  area = radius * radius * 3.14159;                        area:  3.14159
4  cout << "The area is " << area << " for radius " << radius;
5
6  // Compute the second area
7  radius = 2.0;                                          radius:  2.0
8  area = radius * radius * 3.14159;                        area:  12.56636
9  cout << "The area is " << area << " for radius " << radius;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Variables

- Data must be loaded into main memory before it can be manipulated
- Storing data in memory is a two-step process:
  - Instruct computer to allocate memory
  - Include statements to put data into memory



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Declaring Variables

The syntax for declaring a variable is:

```
datatype variableName;

datatype variable1, variable2,..., variablen;
```

```
int x;           // Declare x to be an integer variable;
double radius;   // Declare radius to be a double variable;
char a;          // Declare a to be a character variable;
int i, j, k;     // Declare i, j, and k as int variables
```



- All variables must be initialized before they are used
  - But not necessarily during declaration

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Declaring and Initializing in One Step

```
int x = 1;
```
equivalent to
```
int x;
x = 1;
```

```
double d = 1.4;
```
equivalent to
```
double d;
d = 1.4;
```

```
int i = 1, j = 2;
```
equivalent to
```
int i, j;
i = 1;
j = 2;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Assignment Statement

The assignment statement takes the form:

```
variable = expression;
```

- An expression represents a computation involving values, variables, and operators that, taking them together, evaluates to a value
- A variable is said to be initialized the first time a value is placed into it
- In C++, = is called the assignment operator



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Assignment Statements

```
x = 1;          // Assign 1 to x;
radius = 1.0;   // Assign 1.0 to radius;
a = 'A';        // Assign 'A' to a;
```

```
int y = 1;                // Assign 1 to variable y
double radius = 1.0;      // Assign 1.0 to variable radius
int x = 5 * (3 / 2);      // Assign the value of the
                          //expression to x
x = y + 1;                // Assign the addition of y and 1 to x
area = radius * radius * 3.14159; // Compute area
```

```
i = j = k = 1;
```

which is equivalent to

```
k = 1;
j = k;
i = j;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Allocating Memory with Constants

A named constant is an identifier that represents a permanent value.

- Syntax to declare a named constant:

```
const datatype CONSTANTNAME = value;
```

- CONSTANTNAME: memory location whose content can't change during execution
- A constant **must be declared and initialized** in the same statement.
- In C++, `const` is a reserved word

```
const double PI = 3.14159;
const int SIZE = 3;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Named Constants

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    const double PI = 3.14159;
7
8    // Step 1: Read in radius
9    double radius;
10   cout << "Enter a radius: ";
11   cin >> radius;
12
13   // Step 2: Compute area
14   double area = radius * radius * PI;
15
16   // Step 3: Display the area
17   cout << "The area is ";
18   cout << area << endl;
19
20   return 0;
21 }
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Data Types

- Data type: set of values together with a set of operations
- C++ data types fall into three categories:
  - Simple data type
  - Structured data type
  - Pointers



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Simple Data Types

- Three categories of simple data
  - Integral: integers (numbers without a decimal)
    - Can be further categorized:
      - `char, short, int, long, bool, unsigned char, unsigned short, unsigned int, unsigned long`
  - Floating-point: decimal numbers
  - Enumeration type: user-defined data type



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

`int` Data Type

- Examples:

```
-6728
0
78
+763
```

- Cannot use a comma within an integer
  - Commas are only used for separating items in a list



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

bool Data Type

- bool type
  - Two values: true and false
  - Manipulate logical (Boolean) expressions
- true and false
  - Logical values
- bool, true, and false
  - Reserved words



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

char Data Type

- The smallest integral data type
- Used for single characters: letters, digits, and special symbols
- Each character is enclosed in single quotes
  - 'A', 'a', '0', '*', '+', '$', '&'
- A blank space is a character
  - Written ' ', with a space left between the single quotes



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Floating-Point Data Types

- `float`: represents any real number
  - Range: -3.4E+38 to 3.4E+38 (four bytes)
- `double`: represents any real number
  - Range: -1.7E+308 to 1.7E+308 (eight bytes)
- Minimum and maximum values of data types are system dependent



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Floating-Point Data Types (cont'd.)

- Maximum number of significant digits (decimal places) for `float` values: 6 or 7
- Maximum number of significant digits for `double`: 15
- Precision: maximum number of significant digits
  - Float values are called single precision
  - Double values are called double precision



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Numerical Data Types

TABLE 2.1 Numeric Data Types

Name	*Synonymy*	*Range*	*Storage Size*
`short`	`short int`	-2^{15} to $2^{15}-1$ (−32, 768 to 32,767)	16-bit signed
`unsigned short`	`unsigned short int`	0 to $2^{16}-1$ (65535)	16-bit unsigned
`int signed`		-2^{31} to $2^{31}-1$ (−2147483648 to 2147483647)	32-bit
`unsigned`	`unsigned int`	0 to $2^{32}-1$ (4294967295)	32-bit unsigned
`long`	`long int`	-2^{31} (−2147483648) to $2^{31}-1$ (2147483647)	32-bit signed
`unsigned long`	`unsigned long int`	0 to $2^{32}-1$ (4294967295)	32-bit unsigned
`float`		Negative range:	32-bit IEEE 754
		−3.4028235E+38 to −1.4E−45	
		Positive range:	
		1.4E−45 to 3.4028235E+38	
`double`		Negative range:	64-bit IEEE 754
		−1.7976931348623157E+308 to −4.9E−324	
		Positive range:	
		4.9E−324 to 1.7976931348623157E+308	
`long double`		Negative range:	80-bit
		−1.18E+4932 to −3.37E−4932	
		Positive range:	
		3.37E−4932 to 1.18E+4932	
		Significant decimal digits: 19	

C++ standard
Storage size is at least for integral types

Numerical Data Types

TABLE 2.1 Numeric Data Types

Name	*Synonymy*	*Range*	*Storage Size*
`short`	`short int`	-2^{15} to $2^{15}-1$ (−32, 768 to 32,767)	16-bit signed
`unsigned short`	`unsigned short int`	0 to $2^{16}-1$ (65535)	16-bit unsigned
`int signed`		-2^{31} to $2^{31}-1$ (−2147483648 to 2147483647)	32-bit
`unsigned`	`unsigned int`	0 to $2^{32}-1$ (4294967295)	32-bit unsigned
`long`	`long int`	-2^{31} (−2147483648) to $2^{31}-1$ (2147483647)	32-bit signed
`unsigned long`	`unsigned long int`	0 to $2^{32}-1$ (4294967295)	32-bit unsigned
`float`		Negative range:	32-bit IEEE 754
		−3.4028235E+38 to −1.4E−45	
		Positive range:	
		1.4E−45 to 3.4028235E+38	
`double`		Negative range:	64-bit IEEE 754
		−1.7976931348623157E+308 to −4.9E−324	
		Positive range:	
		4.9E−324 to 1.7976931348623157E+308	
`long double`		Negative range:	80-bit
		−1.18E+4932 to −3.37E−4932	
		Positive range:	
		3.37E−4932 to 1.18E+4932	
		Significant decimal digits: 19	

C++11: long long is defined in C++11

```
1  #include <iostream>
2  #include <climits>
3  #include <cfloat>
4  using namespace std;
5
6  int main()
7  {
8    cout << "INT_MIN is " << INT_MIN << endl;
9    cout << "INT_MAX is " << INT_MAX << endl;
10   cout << "LONG_MIN is " << LONG_MIN << endl;
11   cout << "LONG_MAX is " << LONG_MAX << endl;
12   cout << "FLT_MIN is " << FLT_MIN << endl;
13   cout << "FLT_MAX is " << FLT_MAX << endl;
14   cout << "DBL_MIN is " << DBL_MIN << endl;
15   cout << "DBL_MAX is " << DBL_MAX << endl;
16
17   return 0;
18 }
```

```
INT_MIN is -2147483648
INT_MAX is 2147483647
LONG_MIN is -2147483648
LONG_MAX is 2147483647
FLT_MIN is 1.17549e-038
FLT_MAX is 3.40282e+038
DBL_MIN is 2.22507e-308
DBL_MAX is 1.79769e+308
```



, Inc. All Rights Reserved.

`sizeof` Function

- You can use the `sizeof` function to find the size of a type.
- For example, the following statement displays the size of `int`, `long`, and `double` on your machine.

```
cout << sizeof(int) << " " <<
sizeof(long) << " " << sizeof(double);
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7    cout << "The size of int: " << sizeof(int) << " bytes" << endl;
8    cout << "The size of long: " << sizeof(long) << " bytes" << endl;
9    cout << "The size of double: " << sizeof(double)
10        << " bytes" << endl;
11
12   double area = 5.4;
13   cout << "The size of variable area: " << sizeof(area)
14        << " bytes" << endl;
15
16   int age = 31;
17   cout << "The size of variable age: " << sizeof(age)
18        << " bytes" << endl;
19
20   return 0;
21 }
```

```
The size of int: 4 bytes
The size of long: 4 bytes
The size of double: 8 bytes
The size of variable area: 8 bytes
The size of variable age: 4 bytes
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Synonymous Types

- `short int` is synonymous to `short`.
- `unsigned short int` is synonymous to `unsigned short`.
- `unsigned int` is synonymous to `unsigned`.
- `long int` is synonymous to `long`.
- `unsigned long int` is synonymous to `unsigned long`.
- For example,

```
short int i = 2;
```

is same as

```
short i = 2;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Numeric Literals

A *literal* is a constant value that appears directly in a program

- For example, 34, 1000000, and 5.0 are literals in the following statements:

```
int i = 34;
long k = 1000000;
double d = 5.0;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

octal and hex Literals

- By default, an integer literal is a decimal integer number.
- To denote an octal integer literal, use a leading *0* (zero),
- To denote a hexadecimal integer literal, use a leading *0x* or *0X* (zero x).
- For example, the following code displays the decimal value 65535 for hexadecimal number FFFF and decimal value 8 for octal number 10.

```
cout << 0xFFFF << " " << 010;
```

```
Displays 65535 8
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Scientific Notation

- Floating-point literals can also be specified in scientific notation
- A special syntax is used to write scientific notation numbers.
- For example,
  - 1.23456×10^2 is written as 1.23456E2 or 1.23456E+2
  - 1.23456×10^{-2} as 1.23456E-2.
- E (or e) represents an exponent and can be in either lowercase or uppercase.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

why called floating-point?

- The `float` and `double` types are used to represent numbers with a decimal point.
- Why are they called *floating-point numbers*?
- **These numbers are stored into scientific notation.**
- When a number such as `50.534` is converted into scientific notation such as `5.0534e+1`, its decimal point is moved (i.e., floated) to a new position.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Numeric Operators

- Arithmetic expressions: contain values and arithmetic operators
- Operands: the number of values on which the operators will work
- Operators can be unary (one operand) or binary (two operands)



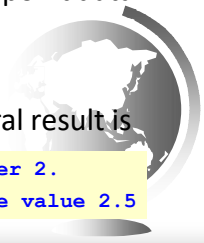
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Arithmetic Operators

- C++ arithmetic operators:

Name	Meaning	Example	Result
+	Addition	34 + 1	35
-	Subtraction	34.0 – 0.1	33.9
*	Multiplication	300 * 30	9000
/	Division	1.0 / 2.0	0.5
%	Remainder	20 % 3	2

- +, -, *, and / can be used with integral and floating-point data types
- Use % only with integral data types
- When you use / with integral data types, the integral result is **truncated** (no rounding)

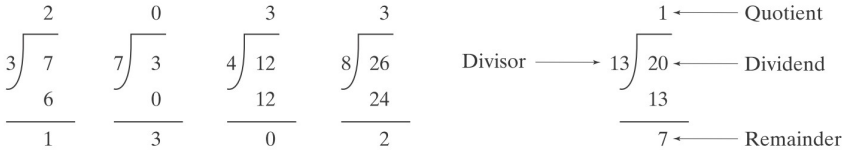


```
5 / 2    yields an integer 2.
5.0 / 2   yields a double value 2.5
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Remainder Operator

- Remainder is very useful in programming.



- `5 % 2` yields 1 (the remainder of the division)

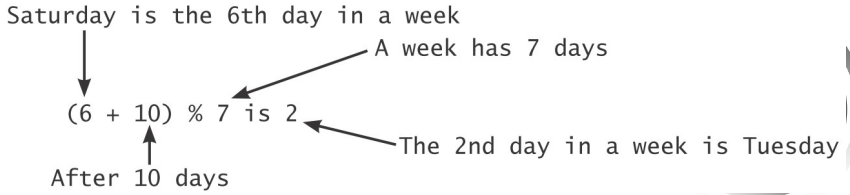
```
34 % 5 = 4
-34 % 5 = -4
34 % -5 = 4
-34 % -5 = -4
```

The remainder is negative only if the dividend is negative

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Remainder Examples

- an even number % 2 is always 0 and an odd number % 2 is always 1.
  - So you can use this property to determine whether a number is even or odd.
- Suppose today is Saturday and you and your friends are going to meet in 10 days. What day is in 10 days?
- You can find that day is Tuesday using the following expression:



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example: Displaying Time

Write a program that obtains minutes and the remaining seconds from an amount of time in seconds.

```
Enter an integer for seconds: 500
500 seconds is 8 minutes and 20 seconds
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    // Prompt the user for input
7    int seconds;
8    cout << "Enter an integer for seconds: ";
9    cin >> seconds;
10   int minutes = seconds / 60;
11   int remainingSeconds = seconds % 60;
12   cout << seconds << " seconds is " << minutes <<
13   " minutes and " << remainingSeconds << " seconds " << endl;
14
15   return 0;
16 }
```

Line#	seconds	minutes	remainingSeconds
9	500		
10		8	
11			20



DisplayTime

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Exponent Operations

```
cout << pow(2.0, 3) << endl; // Display 8.0
cout << pow(4.0, 0.5) << endl; // Display 2.0
cout << pow(2.5, 2) << endl; // Display 6.25
cout << pow(2.5, -2) << endl; // Display 0.16
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Arithmetic Expressions

$$\frac{3+4x}{5} - \frac{10(y-5)(a+b+c)}{x} + 9\left(\frac{4}{x} + \frac{9+x}{y}\right)$$

is translated to

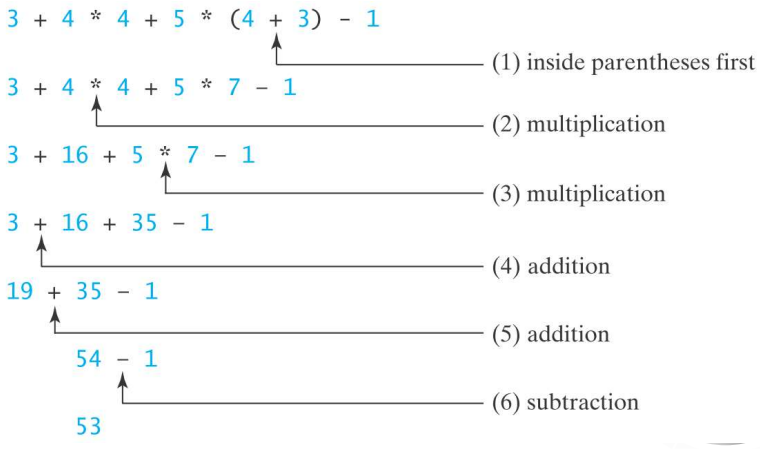
```
(3+4*x)/5 - 10*(y-5)*(a+b+c)/x + 9*(4/x + (9+x)/y)
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

How to Evaluate an Expression

C++ uses arithmetic precedence rule for evaluating expression



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

How to Evaluate an Expression

What is the order of evaluation in the following expressions?

```
a + b + c + d + e
  1   2   3   4
```

```
a + b * c - d / e
  3   1   4   2
```

```
a / (b + c) - d % e
  2    1      4   3
```

```
a / (b * (c + (d - e)))
  4    3     2     1
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example: Converting Temperatures

Write a program that converts a Fahrenheit degree to Celsius using the formula:

$$celsius = (\frac{5}{9})(fahrenheit - 32)$$

```
Enter a degree in Fahrenheit: 100
Fahrenheit 100 is 37.7778 in Celsius
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    // Enter a degree in Fahrenheit
7    double fahrenheit;
8    cout << "Enter a degree in Fahrenheit: ";
9    cin >> fahrenheit;
10
11   // Obtain a celsius degree
12   double celsius = (5.0 / 9) * (fahrenheit - 32);
13
14   // Display result
15   cout << "Fahrenheit " << fahrenheit << " is " <<
16   celsius << " in Celsius" << endl;
17
18   return 0;
19 }
```

Line#	fahrenheit	celsius
7	undefined	
9	100	
12		37.7778

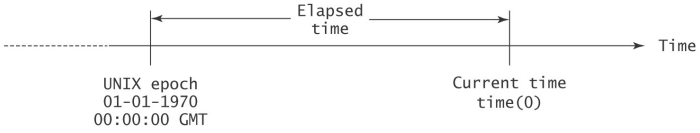
FahrenheitToCelsius



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study: Displaying the Current Time

- Write a program that displays current time in GMT in the format hour:minute:second such as 1:45:19.
- The `time(0)` function returns the current time in seconds elapsed since the time 00:00:00 on January 1, 1970 GMT (`ctime`)
- This time is known as the Unix epoch
  - 1970 was the year when the Unix operating system was formally introduced.





© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

61

```
1  #include <iostream>
2  #include <ctime>
3  using namespace std;
4
5  int main()
6  {
7    // Obtain the total seconds since the midnight, Jan 1, 1970
8    int totalSeconds = time(0);
9
10   // Compute the current second in the minute in the hour
11   int currentSecond = totalSeconds % 60;
12
13   // Obtain the total minutes
14   int totalMinutes = totalSeconds / 60;
15
16   // Compute the current minute in the hour
17   int currentMinute = totalMinutes % 60;
18
19   // Obtain the total hours
20   int totalHours = totalMinutes / 60;
21
22   // Compute the current hour
23   int currentHour = totalHours % 24;
24
25   // Display results
26   cout << "Current time is " << currentHour << ":"
27        << currentMinute << ":" << currentSecond << " GMT" << endl;
28
29   return 0;
30 }
```

ShowCurrentTime

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

62

ShowCurrentTime Output

Variables \ Line#	8	11	14	17	20	23
totalSeconds	1203183086					
currentSecond		26				
totalMinutes			20053051			
currentMinute				31		
totalHours					334217	
currentHour						17

```
Current time is 17:31:26 GMT
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Expressions

- Integral expression: all operands are integers
  - Yields an integral result
  - Example: `2 + 3 * 5`
- Floating-point expression: all operands are floating-point
  - Yields a floating-point result
  - Example: `12.8 * 17.5 - 34.50`



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Mixed Expressions

- Mixed expression:
  - Has operands of different data types
  - Contains integers and floating-point
- Examples of mixed expressions:

```
2 + 3.5
6  /  4 + 3.9
5.4  *  2 – 13.6 + 18  /  2
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Mixed Expressions (cont'd.)

- Evaluation rules:
  - If operator has same types of operands
    - Evaluated according to the type of the operands
  - If operator has both types of operands
    - Integer is changed to floating-point
    - Operator is evaluated
    - Result is floating-point
  - Entire expression is evaluated according to precedence rules



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Expressions Evaluation

```
// This program illustrates how mixed expressions are evaluated.
#include <iostream>
using namespace std;
int main()
{
   cout << "3 / 2 + 5.5 = " << 3 / 2 + 5.5 << endl;
   cout << "15.6 / 2 + 5 = " << 15.6 / 2 + 5 << endl;
   cout << "4 + 5 / 2.0 = " << 4 + 5 / 2.0 << endl;
   cout << "4 * 3 + 7 / 5 - 25.5 = "
        << 4 * 3 + 7 / 5 - 25.5
        << endl;
   return 0;
}
```

```
3 / 2 + 5.5 = 6.5
15.6 / 2 + 5 = 12.8
4 + 5 / 2.0 = 6.5
4 * 3 + 7 / 5 - 25.5 = -12.5
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Augmented Assignment Operators

TABLE 2.3 Augmented Assignment Operators

Operator	*Name*	*Example*	*Equivalent*
`+=`	Addition assignment	`i += 8`	`i = i + 8`
`-=`	Subtraction assignment	`i -= 8`	`i = i - 8`
`*=`	Multiplication assignment	`i *= 8`	`i = i * 8`
`/=`	Division assignment	`i /= 8`	`i = i / 8`
`%=`	Modulus assignment	`i %= 8`	`i = i % 8`

```
x /= 4 + 5.5 * 1.5;
```

is same as

```
x = x / (4 + 5.5 * 1.5);
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Increment and Decrement Operators

TABLE 2.4 Increment and Decrement Operators

Operator	*Name*	*Description*	*Example (assume i = 1)*
`++var`	preincrement	Increment `var` by `1` and use the new `var` value in the statement	`int j = ++i;` `// j is 2, i is 2`
`var++`	postincrement	Increment `var` by `1`, but use the original `var` value in the statement	`int j = i++;` `// j is 1, i is 2`
`--var`	predecrement	Decrement `var` by `1` and use the new `var` value in the statement	`int j = --i;` `// j is 0, i is 0`
`var--`	postdecrement	Decrement `var` by `1` and use the original `var` value in the statement	`int j = i--;` `// j is 1, i is 0`

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

69

Increment and Decrement Operators, cont.

```
int i = 10;
int newNum = 10 * i++;
cout << "i is " << i
  << ", newNum is " << newNum;
```

Same effect as →

```
int newNum = 10 * i;
i = i + 1;
```

```
i is 11, newNum is 100
```

```
int i = 10;
int newNum = 10 * (++i);
cout << "i is " << i
  << ", newNum is " << newNum;
```

Same effect as →

```
i = i + 1;
int newNum = 10 * i;
```

```
i is 11, newNum is 110
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

70

Caution

- For most binary operators, C++ does not specify the operand evaluation order.
  - **Normally, you assume that the left operand is evaluated before the right operand.**
  - **<u>This is not guaranteed in C++</u>.**
- For example, suppose i is 1; then the expression `++i + i`
  - Evaluates to 4 (2 + 2) if the left operand (++i) is evaluated first
  - Evaluates to 3 (2 + 1) if the right operand (i) is evaluated first.

Since C++ cannot guarantee the operand evaluation order, you should not write code that depends on the operand evaluation order



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Type Casting

```
int i = 34.7;      // i becomes 34
double f = i;      // f is now 34
double g = 34.3;   // g becomes 34.3
int j = g;         // j is now 34
```

- Implicit casting
  `double d = 3;` (type widening)
- Explicit casting
  `int i = static_cast<int>(3.0);` (type narrowing)
  `int i = (int)3.9;` (Fraction part is truncated)



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

NOTE

- Casting does not change the variable being cast.
- For example, `d` is not changed after casting in the following code:

```
double d = 4.5;
// d is not changed
int i = static_cast<int>(d);
```

The GNU and Visual C++ compilers will give a warning when you narrow a type unless you use `static_cast` to make the conversion explicit.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example: Keeping Two Digits After Decimal Points

Write a program that displays the sales tax with two digits after the decimal point.

```
Enter purchase amount: 197.55
Sales tax is 11.85
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    // Enter purchase amount
7    double purchaseAmount;
8    cout << "Enter purchase amount: ";
9    cin >> purchaseAmount;
10
11   double tax = purchaseAmount * 0.06;
12   cout << "Sales tax is " << static_cast<int>(tax * 100) / 100.0;
13
14   return 0;
15 }
```

Line#	purchaseAmount	tax	Output
7	Undefined		
9	197.55		
11		11.853	
12			Sales tax is 11.85

SalesTax

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

75

Creating a C++ Program

- A C++ program is a collection of functions, one of which is the function main
- The first line of the function main is called the heading of the function:
  - `int main()`
- The statements enclosed between the curly braces ({ and }) form the body of the function

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Creating a C++ Program (cont'd.)

- A C++ program contains two types of statements:
  - Declaration statements: declare things, such as variables
  - Executable statements: perform calculations, manipulate data, create output, accept input, etc.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Creating a C++ Program (cont'd.)

- C++ program has two parts:
  - Preprocessor directives
  - The program
- Preprocessor directives and program statements constitute C++ source code (.cpp)
- Compiler generates object code (.obj)
- Executable code is produced and saved in a file with the file extension .exe



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Preprocessor Directives

- C++ has a small number of operations
- Many functions and symbols needed to run a C++ program are provided as collection of libraries
- Every library has a name and is referred to by a header file
- Preprocessor directives are commands supplied to the preprocessor program
- All preprocessor commands begin with #
- No semicolon at the end of these commands



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Preprocessor Directives (cont'd.)

- Syntax to include a header file:

```
#include <headerFileName>
```

- For example:

```
#include <iostream>
```

  - Causes the preprocessor to include the header file `iostream` in the program
- Preprocessor commands are processed before the program goes through the compiler



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

`namespace` and Using `cin` and `cout` in a Program

- `cin` and `cout` are declared in the header file `iostream`, but within `std` namespace
- To use `cin` and `cout` in a program, use the following two statements:

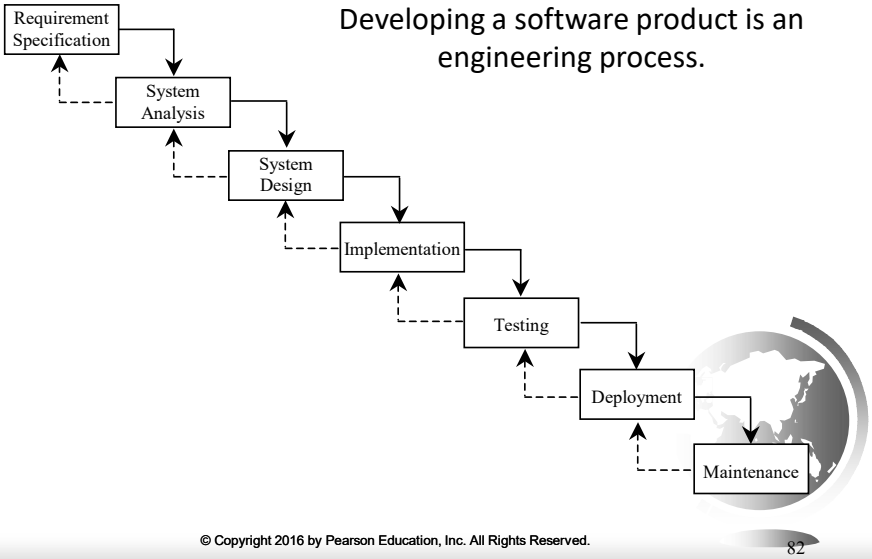
```
#include <iostream>
using namespace std;
```

- Or `std::cin` and `std::cout`



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

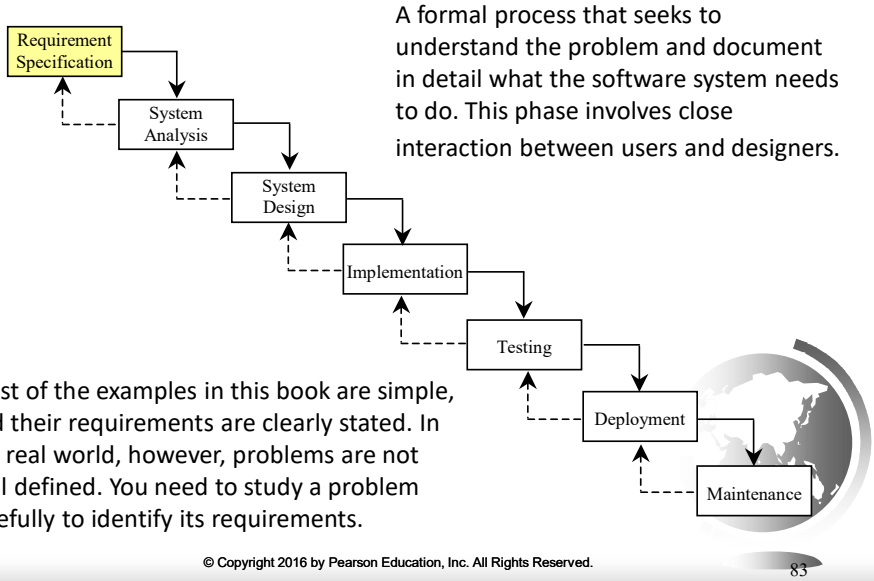
Software Development Process



Developing a software product is an engineering process.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Requirement Specification



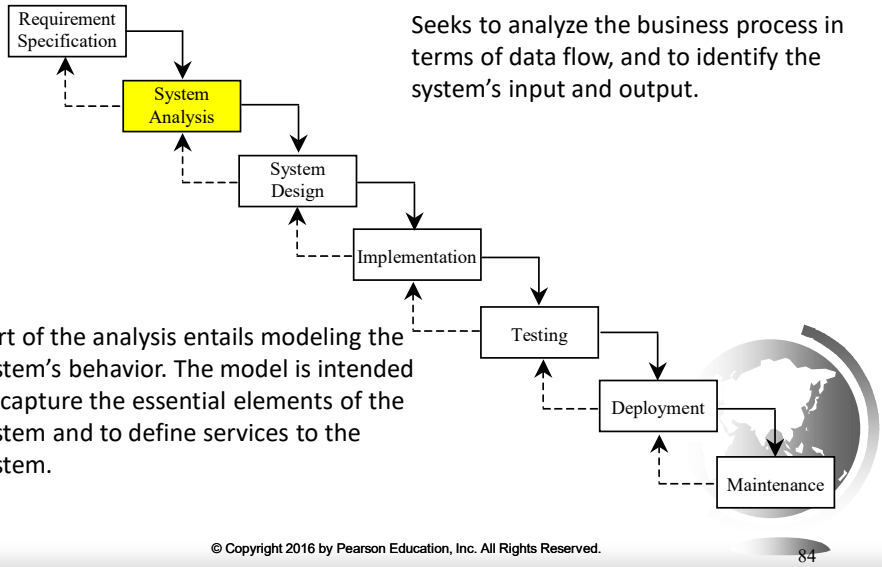
A formal process that seeks to understand the problem and document in detail what the software system needs to do. This phase involves close interaction between users and designers.

Most of the examples in this book are simple, and their requirements are clearly stated. In the real world, however, problems are not well defined. You need to study a problem carefully to identify its requirements.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

83

System Analysis



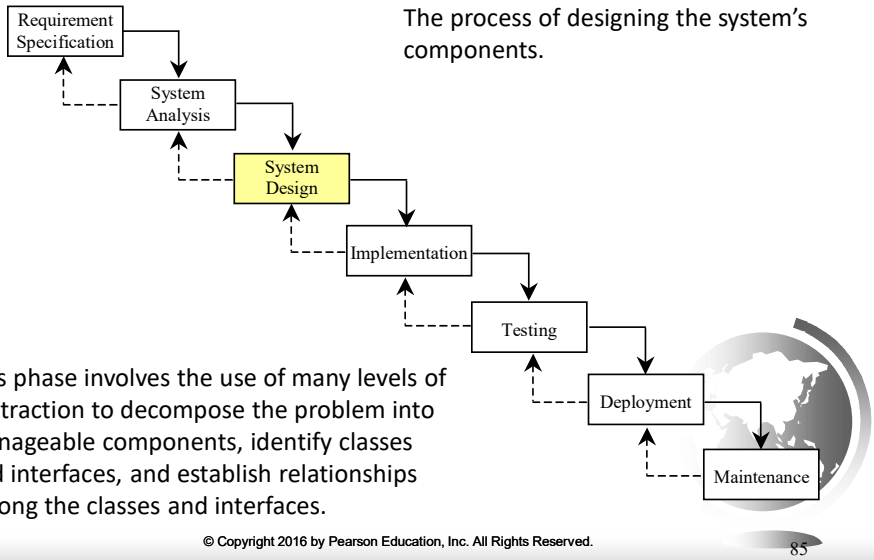
Seeks to analyze the business process in terms of data flow, and to identify the system's input and output.

Part of the analysis entails modeling the system's behavior. The model is intended to capture the essential elements of the system and to define services to the system.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

84

System Design



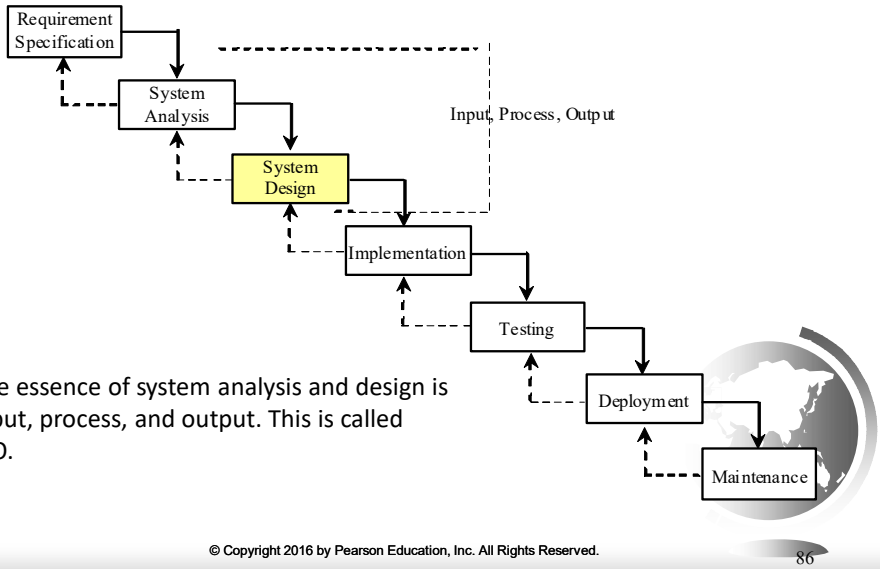
The process of designing the system's components.

This phase involves the use of many levels of abstraction to decompose the problem into manageable components, identify classes and interfaces, and establish relationships among the classes and interfaces.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

85

IPO

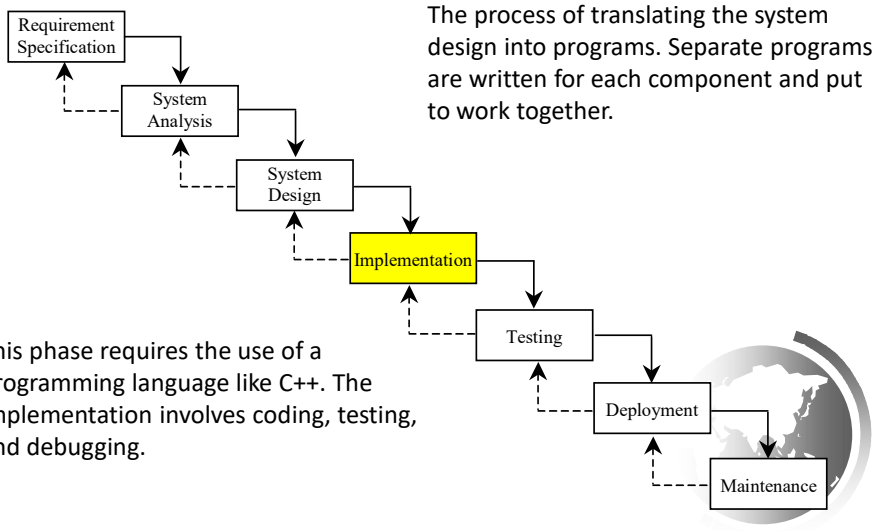


The essence of system analysis and design is input, process, and output. This is called IPO.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

86

Implementation



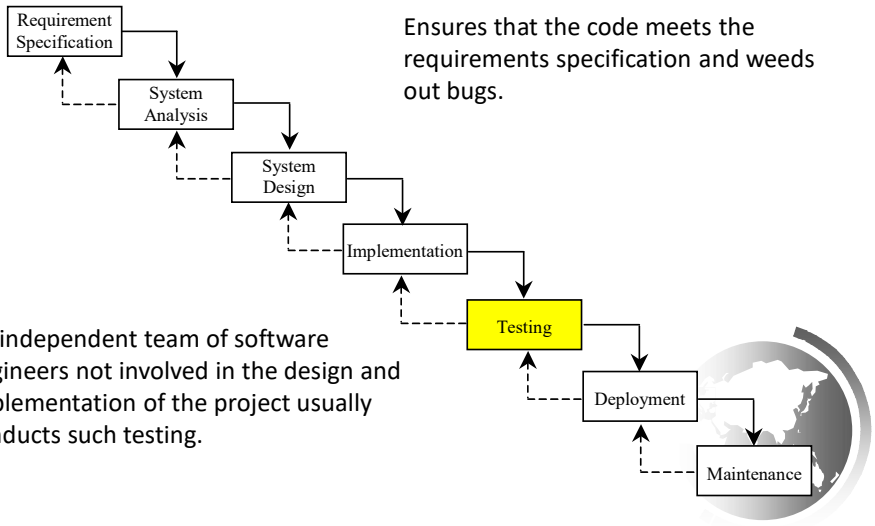
The process of translating the system design into programs. Separate programs are written for each component and put to work together.

This phase requires the use of a programming language like C++. The implementation involves coding, testing, and debugging.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

87

Testing



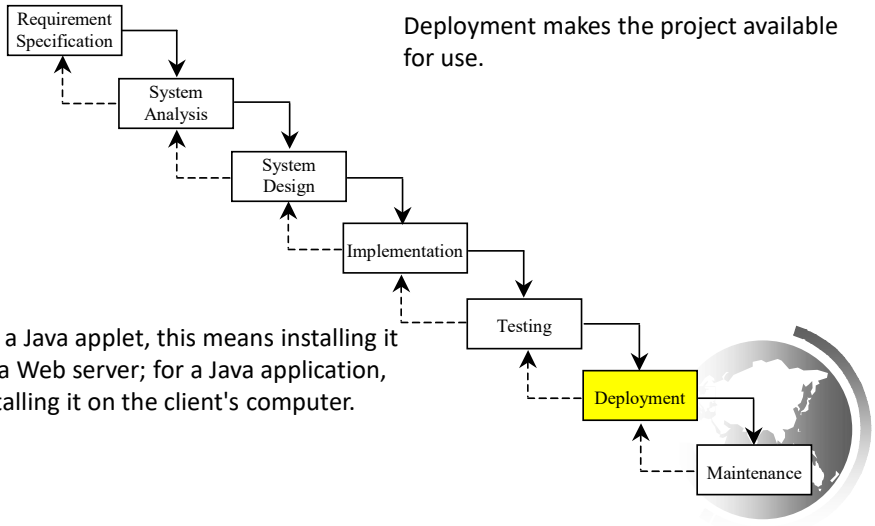
Ensures that the code meets the requirements specification and weeds out bugs.

An independent team of software engineers not involved in the design and implementation of the project usually conducts such testing.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

88

Deployment



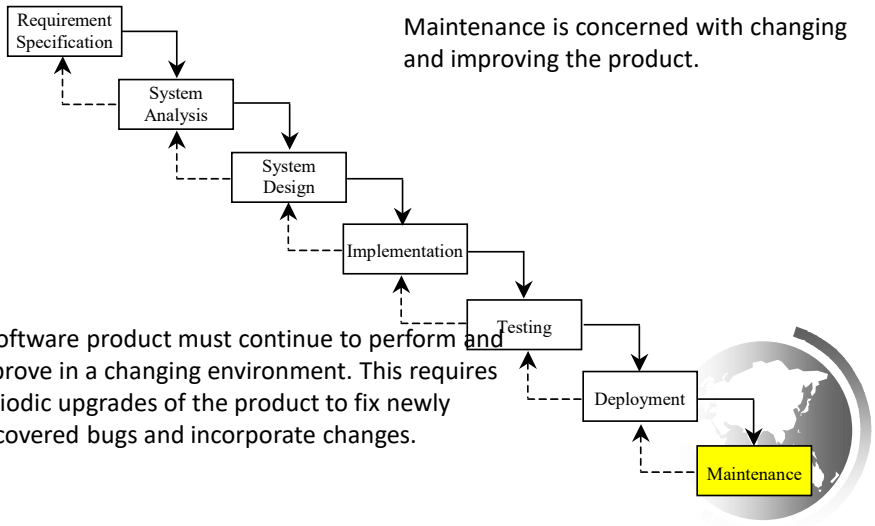
Deployment makes the project available for use.

For a Java applet, this means installing it on a Web server; for a Java application, installing it on the client's computer.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

89

Maintenance



Maintenance is concerned with changing and improving the product.

A software product must continue to perform and improve in a changing environment. This requires periodic upgrades of the product to fix newly discovered bugs and incorporate changes.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

90

Problem: Computing Loan Payments

- **Write a program that computes loan payments.**
  - The loan can be a car loan, a student loan, or a home mortgage loan.
- For an **introductory programming course**, we focus on
  - Requirements specification
  - Analysis
  - Design
  - Implementation
  - Testing



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Stage 1: Requirements Specification

- This program lets the user enter the interest rate, number of years, and loan amount
- computes monthly payment and total payment.

Stage 2: System Analysis

$$monthlyPayment = \frac{loanAmount \times monthlyInterestRate}{1 - \frac{1}{(1 + monthlyInterestRate)^{numberOfYears \times 12}}}$$

$$totalPayment = monthlyPayment \times numberOfYears \times 12$$



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Stage 3: System Design

Step 1. Prompt the user to enter the annual interest rate, the loan amount, and the number of years. (The interest rate is commonly expressed as a percentage of the principal for a period of one year. This is known as the annual interest rate.)

Step 2. The input for the annual interest rate is a number in percent format, such as 4.5%. The program needs to convert it into a decimal by dividing it by **100**. To obtain the monthly interest rate from the annual interest rate, divide it by **12**, since a year has 12 months. To obtain the monthly interest rate in decimal format, you must divide the annual interest rate in percentage by **1200**. For example, if the annual interest rate is 4.5%, then the monthly interest rate is 4.5/1200 = 0.00375.

Step 3. Compute the monthly payment using the preceding formula.

Step 4. Compute the total payment, which is the monthly payment multiplied by **12** and multiplied by the number of years.

Step 5. Display the monthly payment and total payment.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

93

Stage 4: Implementation

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int main()
6  {
7    // Enter yearly interest rate
8    cout << "Enter yearly interest rate, for example 8.25: ";
9    double annualInterestRate;
10   cin >> annualInterestRate;
11
12   // Obtain monthly interest rate
13   double monthlyInterestRate = annualInterestRate / 1200;
14
15   // Enter number of years
16   cout << "Enter number of years as an integer, for example 5: ";
17   int numberOfYears;
18   cin >> numberOfYears;
19
20   // Enter loan amount
21   cout << "Enter loan amount, for example 120000.95: ";
22   double loanAmount;
23   cin >> loanAmount;
```

94

Stage 4: Implementation

```
24
25   // Calculate payment
26   double monthlyPayment = loanAmount * monthlyInterestRate / (1
27   - 1 / pow(1 + monthlyInterestRate, numberOfYears * 12));
28   double totalPayment = monthlyPayment * numberOfYears * 12;
29
30   monthlyPayment = static_cast<int>(monthlyPayment * 100) / 100.0;
31   totalPayment = static_cast<int>(totalPayment * 100) / 100.0;
32
33   // Display results
34   cout << "The monthly payment is " << monthlyPayment << endl <<
35   "The total payment is " << totalPayment << endl;
36
37   return 0;
38 }
```

Stage 5: Testing

ComputeLoan

```
Enter annual interest rate, for example 7.25: 3
Enter number of years as an integer, for example 5: 5
Enter loan amount, for example 120000.95: 1000
The monthly payment is 17.96
The total payment is 1078.12
```


Problem: Monetary Units

- This program lets the user enter the amount in decimal representing dollars and cents and output a report listing the monetary equivalent in single **dollars**, **quarters (25)**, **dimes (10)**, **nickels (5)** , and **pennies**.
- Your program should report maximum number of dollars, then the maximum number of quarters, and so on, in this order.

```
Enter an amount in double, for example 11.56: 11.56
Your amount 11.56 consists of
11 dollars
2 quarters
0 dimes
1 nickels
1 pennies
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    // Receive the amount
7    cout << "Enter an amount in double, for example 11.56: ";
8    double amount;
9    cin >> amount;
10
11   int remainingAmount = static_cast<int>(amount * 100);
12
13   // Find the number of one dollars
14   int numberOfOneDollars = remainingAmount / 100;
15   remainingAmount = remainingAmount % 100;
16
17   // Find the number of quarters in the remaining amount
18   int numberOfQuarters = remainingAmount / 25;
19   remainingAmount = remainingAmount % 25;
20
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
21   // Find the number of dimes in the remaining amount
22   int numberOfDimes = remainingAmount / 10;
23   remainingAmount = remainingAmount % 10;
24
25   // Find the number of nickels in the remaining amount
26   int numberOfNickels = remainingAmount / 5;
27   remainingAmount = remainingAmount % 5;
28
29   // Find the number of pennies in the remaining amount
30   int numberOfPennies = remainingAmount;
31
32   // Display results
33   cout << "Your amount " << amount << " consists of" << endl <<
34   " " << numberOfOneDollars << " dollars" << endl <<
35   " " << numberOfQuarters << " quarters" << endl <<
36   " " << numberOfDimes << " dimes" << endl <<
37   " " << numberOfNickels << " nickels" << endl <<
38   " " << numberOfPennies << " pennies";
39
40   return 0;
41 }
```

ComputeChange

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Output Tracing

Variables \ Line#	9	11	14	15	18	19	22	23	26	27	30
amount	11.56										
remainingAmount		1156		56		6		6		1	
numberOfOneDollars			11								
numberOfQuarters					2						
numberOfDimes							0				
numberOfNickels									1		
numberOfPennies											1

```
Enter an amount in double, for example 11.56: 11.56
Your amount 11.56 consists of
11 dollars
2 quarters
0 dimes
1 nickels
1 pennies
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Errors

- **Common Error 1:** Undeclared/Uninitialized Variables and Unused Variables

```
double interestRate = 0.05;
double interest = interestrate * 45;
```

- **Common Error 2:** Integer Overflow

```
// value will actually become -32768
short value = 32767 + 1;


// value will actually become 32767
short value = -32768 - 1;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Errors

- **Common Error 3:** Round-off Errors

```
float a = 1000.43;
float b = 1000.0;
cout << a - b << endl;
```

displays `0.429993`, **not** `0.43`

- **Common Error 4:** Unintended Integer Division

```
int number1 = 1;
int number2 = 2;
double average = (number1 + number2) / 2;
cout << average << endl;
```

```
int number1 = 1;
int number2 = 2;
double average = (number1 + number2) / 2.0;
cout << average << endl;
```

- **Common Error 5:** Forgetting Header Files

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

101

References

- Introduction to Programming with C++, Daniel Liang, Pearson Education Limited 2014
- C++ Programming- From Problem Analysis to Program Design - D.S. MALIK - 2018

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

102

Computer Engineering Department
Islamic University of Gaza

Dr. Ayman Maliha
amaliha@iugaza.edu.ps
Homepage: http://site.iugaza.edu.ps/amaliha/

ECOM 2403
Computer Programming
Fall 2021-2022



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.



Chapter 3

Selections

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Control Structures

- All computer programs, simple or complex, are written using one or more of three basic structures:
  - Sequence
  - Selection
  - repetition
- These structures are called ***control structures*** or ***logic structures*** because they control the flow of a program's logic.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

3

Control Structures

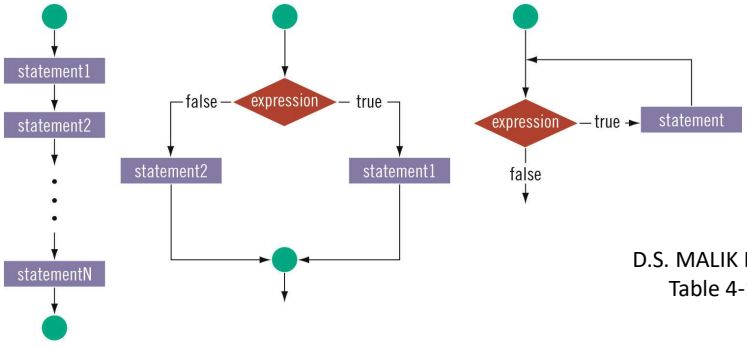
- Sequence Structure
  - Directs the computer to process the program instructions, one after another.
- Selection Structure
  - Indicates that a decision (based on some condition) needs to be made
  - The decision is followed by an appropriate action.
- Repetition Structure (iteration)
  - Directs the computer to repeat one or more instructions until some condition is met (stop repeating)



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

4

Control Structures



a. Sequence b. Selection c. Repetition

FIGURE 4-1 Flow of execution

D.S. MALIK Book
Table 4-1

The decision statements are based on *boolean expressions* (or *conditions*) that evaluate to true or false

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

5

The `bool` Type and Operators

- Often in a program you need to compare two values, such as whether i is greater than j.
- C++ provides six comparison operators (also known as relational operators) that can be used to compare two values.
- You can declare a variable of type `bool` as follows:

```
bool lightsOn = true;
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

6

The `bool` Type and Operators

- Internally, C++ uses 1 to represent true and 0 for false.
- If you display a bool value to the console, 1 is displayed if the value is true and 0 if it is false.
- For example,

```
cout << (4 < 5);
```

- displays 1, because 4 < 5 is true.

```
cout << (4 > 5);
```

- displays 0, because 4 > 5 is false.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Relational Operators

TABLE 3.1 Relational Operators

Operator	*Mathematics Symbol*	*Name*	*Example (radius is 5)*	*Result*
`<`	<	less than	`radius < 0`	`false`
`<=`	≤	less than or equal to	`radius <= 0`	`false`
`>`	>	greater than	`radius > 0`	`true`
`>=`	≥	greater than or equal to	`radius >= 0`	`true`
`==`	=	equal to	`radius == 0`	`false`
`!=`	≠	not equal to	`radius != 0`	`true`

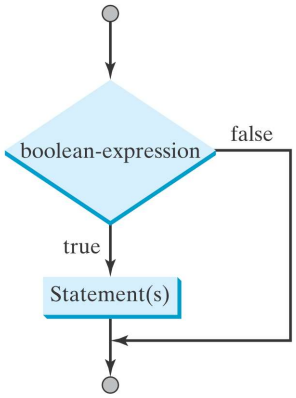


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

One-way `if` Statements

An **if** *statement is a construct that enables a program to specify alternative paths of execution.*

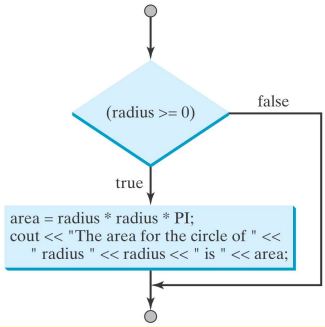
```
if (booleanExpression)
{
  statement(s);
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

9

One-way `if` Statements




```
if (radius >= 0)
{
  area = radius * radius * PI;
  cout << "The area for the circle of " <<
    " radius " << radius << " is " << area;
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

10

Note

```
if i > 0
{
  cout << "i is positive" << endl;
}
```

(a) Wrong

```
if (i > 0)
{
  cout << "i is positive" << endl;
}
```

(b) Correct

```
if (i > 0)
{
  cout << "i is positive" << endl;
}
```

(a)

Equivalent

```
if (i > 0)
  cout << "i is positive" << endl;
```

(b)



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Examples

Write a program that prompts the user to enter an integer. If the number is a multiple of 5, print **`HiFive`**. If the number is divisible by 2, print **`HiEven`**.

```
Enter an integer: 4
HiEven
```

```
Enter an integer: 30
HiFive
HiEven
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    // Prompt the user to enter an integer
7    int number;
8    cout << "Enter an integer: ";
9    cin >> number;
10
11   if (number % 5 == 0)
12     cout << "HiFive" << endl;
13
4    if (number % 2 == 0 )
15     cout << "HiEven" << endl;
16
17   return 0;
18 }
```

```
Enter an integer: 4
HiEven
```

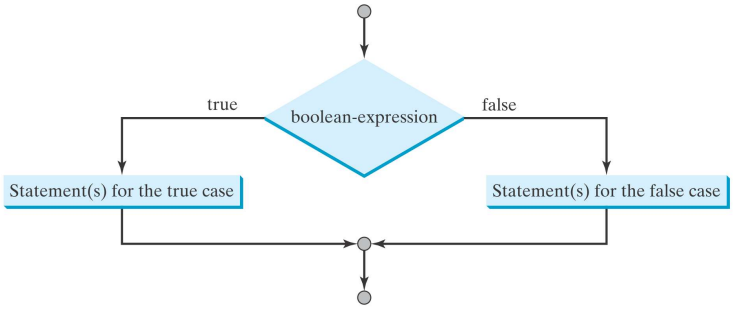
```
Enter an integer: 30
HiFive
HiEven
```

SimpleIfDemo

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

13

The `if...else` Statement




```
if (booleanExpression)
{
  statement(s)-for-the-true-case;
}
else
{
  statement(s)-for-the-false-case;
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

14

`if-else` Example

```
if (radius >= 0)
{
  area = radius * radius * PI;
  cout << "The area for the circle of radius " <<
  radius << " is " << area;
}
else
{
  cout << "Negative radius";
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Nested `if` Statements

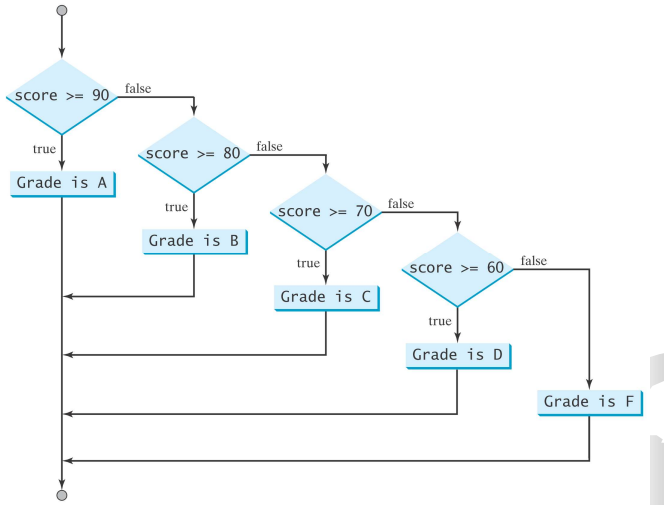
An `if` statement can be inside another `if` statement to form a nested `if` statement.

```
if (i > k)
{
  if (j > k)
    cout << "i and j are greater than k";
}
else
  cout << "i is less than or equal to k";
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Multiple Alternative if Statements



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Multiple Alternative if Statements

```
if (score >= 90.0)
  cout << "Grade is A";
else
  if (score >= 80.0)
    cout << "Grade is B";
  else
    if (score >= 70.0)
      cout << "Grade is C";
    else
      if (score >= 60.0)
        cout << "Grade is D";
      else
        cout << "Grade is F";
```

(a)

Equivalent

This is better

```
if (score >= 90.0)
  cout << "Grade is A";
else if (score >= 80.0)
  cout << "Grade is B";
else if (score >= 70.0)
  cout << "Grade is C";
else if (score >= 60.0)
  cout << "Grade is D";
else
  cout << "Grade is F";
```

(b)

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace if-else statement

Suppose score is 70.0

The condition is false

```
if (score >= 90.0)
  cout << "Grade is A";
else if (score >= 80.0)
  cout << "Grade is B";
else if (score >= 70.0)
  cout << "Grade is C";
else if (score >= 60.0)
  cout << "Grade is D";
else
  cout << "Grade is F";
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace if-else statement

Suppose score is 70.0

The condition is false

```
if (score >= 90.0)
  cout << "Grade is A";
else if (score >= 80.0)
  cout << "Grade is B";
else if (score >= 70.0)
  cout << "Grade is C";
else if (score >= 60.0)
  cout << "Grade is D";
else
  cout << "Grade is F";
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace if-else statement

Suppose score is 70.0

The condition is true

```
if (score >= 90.0)
  cout << "Grade is A";
else if (score >= 80.0)
  cout << "Grade is B";
else if (score >= 70.0)
  cout << "Grade is C";
else if (score >= 60.0)
  cout << "Grade is D";
else
  cout << "Grade is F";
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace if-else statement

Suppose score is 70.0

grade is C

```
if (score >= 90.0)
  cout << "Grade is A";
else if (score >= 80.0)
  cout << "Grade is B";
else if (score >= 70.0)
  cout << "Grade is C";
else if (score >= 60.0)
  cout << "Grade is D";
else
  cout << "Grade is F";
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace if-else statement

Suppose score is 70.0

Exit the if statement

```
if (score >= 90.0)
  cout << "Grade is A";
else if (score >= 80.0)
  cout << "Grade is B";
else if (score >= 70.0)
  cout << "Grade is C";
else if (score >= 60.0)
  cout << "Grade is D";
else
  cout << "Grade is F";
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Error 1: Forgetting Necessary Braces

```
if (radius >= 0)
  area = radius * radius * PI;
  cout << "The area "
    << " is " << area;
```

(a) Wrong

```
if (radius >= 0)
{
  area = radius * radius * PI;
  cout << "The area "
    <<  " is " <<  area;
}
```

(b) Correct

Equivalent

```
if (radius >= 0)
  area = radius * radius * PI;

cout << "The area "
    << " is " << area;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Error 2:
Wrong Semicolon at the if Line

Logic Error

```
if (radius >= 0);
{
  area = radius * radius * PI;
  cout << "The area "
    << " is " << area;
}
```

(a)

Equivalent

Empty Body

```
if (radius >= 0) { };
{
  area = radius * radius * PI;
  cout << "The area "
    <<  " is " <<  area;
}
```

(b)

- This mistake is hard to find, because it is not a compilation error or a runtime error, it is a **logic error**.
- This error often occurs when you use the next-line block style.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Error 3:
Mistakenly Using = for ==

```
if (count = 1)
  cout << "count is zero" << endl;
else
  cout << "count is not zero" << endl;
```

Always displays

```
count is zero
```

Recall that any nonzero value evaluates to **`true`** and zero value evaluates to **`false`**.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Error 4:
Redundant Testing of Boolean Values

To test whether a **bool** variable is **true** or **false** in a test condition, it is redundant to use the equality testing operator like the code in

```
if (even == true)
  cout <<"It is even.";
```

(a)

Equivalent

```
if (even)
  cout << "It is even.";
```

(b)

This is better

Another good reason is to avoid errors that are difficult to detect. Using the = operator instead of the == operator such as error 3.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Error 5:
Dangling `else` Ambiguity

```
int i = 1;
int j = 2;
int k = 3;

if (i > j)
  if (i > k)
    cout << "A";
else
    cout << "B";
```

(a)

Equivalent

This is better with correct indentation

```
int i = 1;
int j = 2;
int k = 3;

if (i > j)
  if (i > k)
    cout << "A";
  else
    cout << "B";
```

(b)

The `else` clause matches the most recent `if` clause in the same block.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Error 5: Dangling `else` Ambiguity

- To force the **`else`** clause to match the first **`if`** clause, you must add a pair of braces:

```
int i = 1; int j = 2; int k = 3;

if (i > j){
  if (i > k)
    cout << "A";
}
else
  cout << "B";
```

This statement prints B.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Error 6: Equality Test of Two Floating-Point Values

```
double x = 1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1;
if (x == 0.5)
   cout << "x is 0.5" << endl;
else
   cout << "x is not 0.5" << endl;
```

```
const double EPSILON = 1E-14;
double x = 1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1;
if (abs(x - 0.5) < EPSILON)
   cout << "x is approximately 0.5" << endl;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Pitfall 1: Simplifying Boolean Variable Assignment

```
if (number % 2 == 0)
  even = true;
else
  even = false;
```

(a)

Equivalent

This is better

```
bool even
  = number % 2 == 0;
```

(b)



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Pitfall 2: Avoiding Duplicate Code in Different Cases

```
if (inState)
{
  tuition = 5000;
  cout << "The tuition is " << tuition << endl;
}
else
{
  tuition = 15000;
  cout << "The tuition is " << tuition << endl;
}
```

```
if (inState)
{
  tuition = 5000;
}
else
{
  tuition = 15000;
}
cout << "The tuition is " << tuition << endl;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Common Pitfall 3: Integer Values Can Be Used as Boolean Values

- In C++, a Boolean `true` is treated as `1` and `false` as `0`.
- A numeric value can be used as a Boolean value.

```
if (!amount <= 50)
  cout << "Amount is more than 50";
```

(a)

```
if (!(amount <= 50))
  cout << "Amount is more than 50";
```

(b)

Always true

Why?



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Problem: Body Mass Index

- Body Mass Index (BMI) is a measure of health on weight.
- It can be calculated by taking your weight in kilograms and dividing by the square of your height in meters.
- The interpretation of BMI for people 16 years or older is as follows:

BMI	Interpretation
BMI < 18.5	Underweight
18.5 = BMI < 25.0	Normal
25.0 = BMI < 30.0	Overweight
30.0 = BMI	Obese

```
Enter weight in pounds: 146
Enter height in inches: 70
BMI is 20.9486
Normal
```



```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    // Prompt the user to enter weight in pounds
7    cout << "Enter weight in pounds: ";
8    double weight;
9    cin >> weight;
10
11   // Prompt the user to enter height in inches
12   cout << "Enter height in inches: ";
13   double height;
14   cin >> height;
15
16   const double KILOGRAMS_PER_POUND = 0.45359237; // Constant
17   const double METERS_PER_INCH = 0.0254; // Constant
18
19   // Compute BMI
20   double weightInKilograms = weight * KILOGRAMS_PER_POUND;
21   double heightInMeters = height * METERS_PER_INCH;
22   double bmi = weightInKilograms /
23   (heightInMeters * heightInMeters);
24
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
25   // Display result
26   cout << "BMI is " << bmi << endl;
27   if (bmi < 18.5)
28      cout << "Underweight" << endl;
29   else if (bmi < 25)
30      cout << "Normal" << endl;
31   else if (bmi < 30)
32      cout << "Overweight" << endl;
33   else
34      cout << "Obese" << endl;
35
36   return 0;
37 }
```

ComputeBMI

```
Enter weight in pounds: 146
Enter height in inches: 70
BMI is 20.9486
Normal
```

Line#	weight	height	weightInKilograms	heightInMeters	bmi	Output
9	146					
14		70				
20			66.22448602			
21				1.778		
22					20.9486	
26						BMI is 20.9486
32						Normal

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example: Computing Taxes

- The US federal personal income tax is calculated based on the filing status and taxable income.
- There are four filing statuses: single filers, married filing jointly, married filing separately, and head of household.

(single, 10000); tax = 8350*0.1 + (10000-8350)*0.15
(single, 35000); tax = 8350*0.1+(33950-8350)*0.15+(35000-33950)*0.25

TABLE 3.2 2009 U.S. Federal Personal Tax Rates

Marginal Tax Rate	*Single*	*Married Filing Jointly or Qualifying Widow(er)*	*Married Filing Separately*	*Head of Household*
10%	$0 – $8,350	$0 – $16,700	$0 – $8,350	$0 – $11,950
15%	$8,351 – $33,950	$16,701 – $67,900	$8,351 – $33,950	$11,951 – $45,500
25%	$33,951 – $82,250	$67,901 – $137,050	$33,951 – $68,525	$45,501 – $117,450
28%	$82,251 – $171,550	$137,051 – $208,850	$68,526 – $104,425	$117,451 – $190,200
33%	$171,551 – $372,950	$208,851 – $372,950	$104,426 – $186,475	$190,201 – $372,950
35%	$372,951+	$372,951+	$186,476+	$372,951+

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

37

Example: Computing Taxes

```
(0-single filer, 1-married jointly or qualifying widow(er),
2-married separately, 3-head of household)
Enter the filing status: 0
Enter the taxable income: 400000
Tax is 117684
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

38

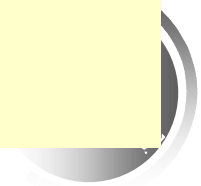
Example: Computing Taxes, cont.

```
if (status == 0)
{
  // Compute tax for single filers
}
else if (status == 1)
{
  // Compute tax for married file jointly
}
else if (status == 2)
{
  // Compute tax for married file separately
}
else if (status == 3)
{
  // Compute tax for head of household
}
else
{
  // Display wrong status
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    // Prompt the user to enter filing status
7    cout << "(0-single filer, 1-married jointly, "
8         << "or qualifying widow(er)," << endl
9         << "2-married separately, 3-head of household)" << endl
10        << "Enter the filing status: ";
11
12   int status;
13   cin >> status;
14
15   // Prompt the user to enter taxable income
16   cout << "Enter the taxable income: ";
17   double income;
18   cin >> income;
19
20   // Compute tax
21   double tax = 0;
22
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
23   if (status == 0) // Compute tax for single filers
24   {
25      if (income <= 8350)
26         tax = income * 0.10;
27      else if (income <= 33950)
28         tax = 8350 * 0.10 + (income - 8350) * 0.15;
29      else if (income <= 82250)
30         tax = 8350 * 0.10 + (33950 - 8350) * 0.15 +
31               (income - 33950) * 0.25;
32      else if (income <= 171550)
33         tax = 8350 * 0.10 + (33950 - 8350) * 0.15 +
34              (82250 - 33950) * 0.25 + (income - 82250) * 0.28;
35      else if (income <= 372950)
36         tax = 8350 * 0.10 + (33950 - 8350) * 0.15 +
37            (82250 - 33950) * 0.25 + (171550 - 82250) * 0.28 +
38            (income - 171550) * 0.33;
39      else
40         tax = 8350 * 0.10 + (33950 - 8350) * 0.15 +
41              (82250 - 33950) * 0.25 + (171550 - 82250) * 0.28 +
42            (372950 - 171550) * 0.33 + (income - 372950) * 0.35;
43   }
44   else if (status == 1) // Compute tax for married file jointly
45   {
46      // Left as exercise
47   }
```

41

```
48   else if (status == 2) // Compute tax for married separately
49   {
50      // Left as exercise
51   }
52   else if (status == 3) // Compute tax for head of household
53   {
54      // Left as exercise
55   }
56   else
57   {
58     cout << "Error: invalid status";
59     return 0;
60   }
61
62   // Display the result
63   cout << "Tax is " << static_cast<int>(tax * 100) / 100.0 << endl;
64
65   return 0;
66 }
```

ComputeAndInterprtTax

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

42

Generating Random Numbers

- `rand()` function (`cstdlib`)
  - Generates a random number
  - Returns a random integer between `0` and `RAND_MAX`.
    - `RAND_MAX` is platform-dependent constant.
  - The numbers produced are pseudorandom.
    - Every time it is executed on the same system, `rand()` produces the same sequence of numbers.

To obtain a random integer between 0 and 9, use
`rand() % 10`

How to generate numbers between 1 and 6?

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Generating Random Numbers

- The `rand()` function's algorithm uses the seed (default 1) to control numbers generating.
- Changing the seed results in different sequence of random numbers.
- Use the `srand(seed)` function to change the seed.
  - In the `cstdlib` header file.
- To ensure that the seed value is different each time you run the program, use `time(0)`.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example: A Simple Math Learning Tool

- This example creates a program for a first grader to practice subtractions.
- The program randomly generates two single-digit integers **number1** and **number2** with **number1 >= number2** and displays a question such as "What is 9 – 2?" to the student.
- After the student types the answer, the program displays a message to indicate whether the answer is correct.

```
What is 5 - 2? 3
You are correct!
```

```
What is 4 - 2? 1
Your answer is wrong.
4 - 2 should be 2
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example: A Simple Math Learning Tool

The program can work as follows:

1. Generate two single-digit integers into **number1** and **number2**.
2. If **number1 < number2**, swap **number1** with **number2**.
3. Prompt the student to answer, "**What is number1 – number2?**"
4. Check the student's answer and display whether the answer is correct.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

SubtractionQuiz

```
1  #include <iostream>
2  #include <ctime> // for time function
3  #include <cstdlib> // for rand and srand functions
4  using namespace std;
5
6  int main()
7  {
8    // 1. Generate two random single-digit integers
9    srand(time(0));
10   int number1 = rand() % 10;
11   int number2 = rand() % 10;
12
13   // 2. If number1 < number2, swap number1 with number2
14   if (number1 < number2)
15   {
16      int temp = number1;
17      number1 = number2;
18      number2 = temp;
19   }
20
21   // 3. Prompt the student to answer "what is number1 - number2?"
22   cout << "What is " << number1 << " - " << number2 << "? ";
23   int answer;
24   cin >> answer;
25
26   // 4. Grade the answer and display the result
27   if (number1 - number2 == answer)
28      cout << "You are correct!";
29   else
30     cout << "Your answer is wrong." << endl << number1 << " - "
31          << number2 << " should be " << (number1 - number2) << endl;
32
33   return 0;
34 }
```

Logical Operators

The logical operators !, &&, and || can be used to create a compound Boolean expression.

TABLE 3.3 Boolean Operators

Operator	*Name*	*Description*
!	not	logical negation
&&	and	logical conjunction
||	or	logical disjunction

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

48

Truth Table for Operator !

TABLE 3.4 Truth Table for Operator !

p	*!p*	*Example (assume age = 24, weight = 140)*
`true`	`false`	`!(age > 18)` is `false`, because `(age > 18)` is `true`.
`false`	`true`	`!(weight == 150)` is `true`, because `(weight == 150)` is `false`.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Truth Table for Operator &&

TABLE 3.5 Truth Table for Operator &&

p1	*p2*	*p1 && p2*	*Example (assume age = 24, weight = 140)*
`false`	`false`	`false`	`(age > 18) && (weight <= 140)` is `true`, because
`false`	`true`	`false`	`(age > 18)` and `(weight <= 140)` are both `true`.
`true`	`false`	`false`	`(age > 18) && (weight > 140)` is `false`, because
`true`	`true`	`true`	`(weight > 140)` is `false`.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Truth Table for Operator ||

TABLE 3.6 Truth Table for Operator ||

p1	*p2*	*p1* || *p2*	*Example (assume age = 24, weight = 140)*
`false`	`false`	`false`	`(age > 34) || (weight <= 140)` is `true`, because
`false`	`true`	`true`	`(weight <= 140)` is `true`.
`true`	`false`	`true`	`(age > 34) || (weight >= 150)` is `false`, because
`true`	`true`	`true`	`(age > 34)` and `(weight >= 150)` are both `false`.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Examples

- Write a program that checks
  - ✓ whether a number is divisible by 2 and 3
  - ✓ whether a number is divisible by 2 or 3
  - ✓ whether a number is divisible by 2 or 3 but not both.

```
Enter an integer: 4
4 is divisible by 2 or 3.
4 is divisible by 2 or 3, but not both.
```

```
Enter an integer: 18
18 is divisible by 2 and 3
18 is divisible by 2 or 3.
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    int number;
7    cout << "Enter an integer: ";
8    cin >> number;
9
10   if (number % 2 == 0 && number % 3 == 0)
11      cout << number << " is divisible by 2 and 3." << endl;
12
13   if (number % 2 == 0 || number % 3 == 0)
14      cout << number << " is divisible by 2 or 3." << endl;
15
16   if ((number % 2 == 0 || number % 3 == 0) &&
17       !(number % 2 == 0 && number % 3 == 0))
18      cout << number << " divisible by 2 or 3, but not both." << endl;
19
20   return 0;
21 }
```

TestBooleanOperators

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Short-Circuit Operator

- When evaluating p1 && p2,
  - ✓ C++ first evaluates p1 and then evaluates p2 if p1 is true;
  - ✓ if p1 is false, it does not evaluate p2.
- When evaluating p1 || p2,
  - ✓ C++ first evaluates p1 and then evaluates p2 if p1 is false;
  - ✓ if p1 is true, it does not evaluate p2.
- Therefore
  - ✓ && is referred to as the *conditional* or *short-circuit AND* operator
  - ✓ || is referred to as the *conditional* or *short-circuit OR* operator.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Examples

- Write a program that lets the user enter a year and checks whether it is a leap year.

```
// A leap year is divisible by 4
   boolean isLeapYear = (year % 4 == 0);
// A leap year is divisible by 4 but not by 100
   isLeapYear = isLeapYear && (year % 100 != 0);
// A leap year is divisible by 4 but not by 100 or divisible by 400
   isLeapYear = isLeapYear || (year % 400 == 0);
```

- So you can use the following Boolean expression to check whether a year is a leap year:

```
(year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

55

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    cout << "Enter a year: ";
7    int year;
8    cin >> year;
9
10   // Check if the year is a leap year
11   bool isLeapYear =
12           (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
13
14   // Display the result in a message dialog box
15   if (isLeapYear)
16      cout << year << " is a leap year" << endl;
17   else
18      cout << year << " is a not leap year" << endl;
19
20   return 0;
21 }
```

```
Enter a year: 1900
1900 is not a leap year
```

```
Enter a year: 2008
2008 is a leap year
```

LeapYear

n, Inc. All Rights Reserved.

56

Problem: Lottery

Randomly generates a lottery of a two-digit number, prompts the user to enter a two-digit number, and determines whether the user wins according to the following rule:

- ✓ If the user input matches the lottery in exact order, the award is $10,000.
- ✓ If the user input matches the lottery, the award is $3,000.
- ✓ If one digit in the user input matches a digit in the lottery, the award is $1,000.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Sample Output

```
Enter your lottery pick (two digits): 00
The lottery number is 0
Exact match: you win $10,000
```

```
Enter your lottery pick (two digits): 45
The lottery number is 54
Match all digits: you win $3,000
```

```
Enter your lottery pick: 23
The lottery number is 34
Match one digit: you win $1,000
```

```
Enter your lottery pick: 23
The lottery number is 14
Sorry, no match
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

58

```
1  #include <iostream>
2  #include <ctime> // for time function
3  #include <cstdlib> // for rand and srand functions
4  using namespace std;
5
6  int main()
7  {
8    // Generate a lottery
9    srand(time(0));
10   int lottery = rand() % 100;
11
12   // Prompt the user to enter a guess
13   cout << "Enter your lottery pick (two digits): ";
14   int guess;
15   cin >> guess;
16
17   // Get digits from lottery
18   int lotteryDigit1 = lottery / 10;
19   int lotteryDigit2 = lottery % 10;
20
21   // Get digits from guess
22   int guessDigit1 = guess / 10;
23   int guessDigit2 = guess % 10;
24
25   cout << "The lottery number is " << lottery << endl;
26
```

```
27   // Check the guess
28   if (guess == lottery)
29      cout << "Exact match: you win $10,000" << endl;
30   else if (guessDigit2 == lotteryDigit1
31            && guessDigit1 == lotteryDigit2)
32      cout << "Match all digits: you win $3,000" << endl;
33   else if (guessDigit1 == lotteryDigit1
34            || guessDigit1 == lotteryDigit2
35            || guessDigit2 == lotteryDigit1
36            || guessDigit2 == lotteryDigit2)
37      cout << "Match one digit: you win $1,000" << endl;
38   else
39      cout << "Sorry, no match" << endl;
40
41   return 0;
42 }
```

Variable \ Line#	10	15	18	19	22	23	37
lottery	34						
guess		23					
lotteryDigit1			3				
lotteryDigit2				4			
guessDigit1					2		
guessDigit2						3	
output							Match one digit: you win $1,000



60

`switch` Statements

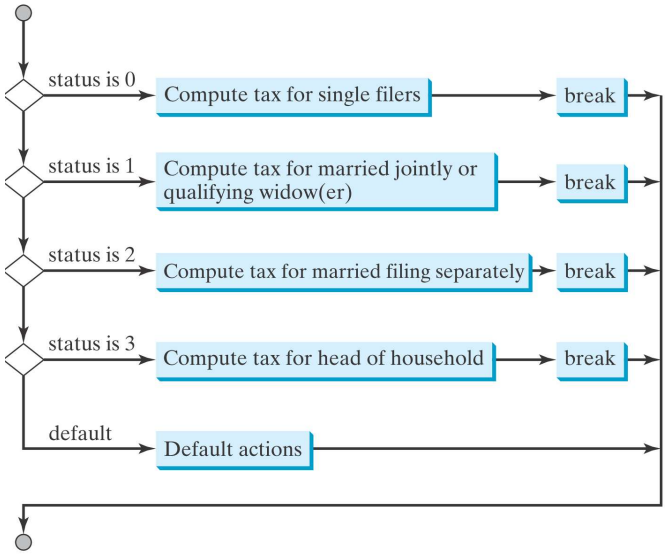
`switch` statement is used to simplify coding for multiple conditions.

```
switch (status)
{
  case 0:  compute taxes for single filers;
           break;
  case 1:  compute taxes for married file jointly;
           break;
  case 2:  compute taxes for married file separately;
           break;
  case 3:  compute taxes for head of household;
           break;
  default: cout << "Errors: invalid status‟ << endl;
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

61

`switch` Statement Flow Chart



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

62

`switch` Statement Rules

The switch-expression must yield a value of char, byte, short, or int type and must always be enclosed in parentheses.

The value1, ..., and valueN must have the same data type as the value of the switch-expression. The resulting statements in the case statement are executed when the value in the case statement matches the value of the switch-expression. Note that value1, ..., and valueN are constant expressions, meaning that they cannot contain variables in the expression, such as 1 + x.

```
switch (switch-expression) {
  case value1:  statement(s)1;
                break;
  case value2: statement(s)2;
                break;
  ...
  case valueN: statement(s)N;
                break;
  default: statement(s)-for-
default;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

`switch` Statement Rules

The keyword break is optional, but it should be used at the end of each case in order to terminate the remainder of the switch statement.
If the break statement is not present, the next case statement will be executed.

The default case, which is optional, can be used to perform actions when none of the specified cases matches the switch-expression.

```
switch (switch-expression) {
  case value1:  statement(s)1;
                break;
  case value2: statement(s)2;
                break;
  ...
  case valueN: statement(s)N;
                break;
  default: statement(s)-for-default;
}
```

When the value in a **case** statement matches the value of the **switch-expression**, the statements *starting from this case* are executed until either a **break** statement or the end of the **switch** statement is reached.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace switch statement

Suppose day is 2:

```
switch (day)
{
  case 1: // Fall to through to the next case
  case 2: // Fall to through to the next case
  case 3: // Fall to through to the next case
  case 4: // Fall to through to the next case
  case 5: cout << "Weekday"; break;
  case 0: // Fall to through to the next case
  case 6: cout << "Weekend";
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

65

animation

Trace switch statement

Suppose day is 2:

```
switch (day)
{
  case 1: // Fall to through to the next case
  case 2: // Fall to through to the next case
  case 3: // Fall to through to the next case
  case 4: // Fall to through to the next case
  case 5: cout << "Weekday"; break;
  case 0: // Fall to through to the next case
  case 6: cout << "Weekend";
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

66

animation

Trace switch statement

Suppose day is 2:

```
switch (da
{
  case 1: // Fall to through to the next case
  case 2: // Fall to through to the next case
  case 3: // Fall to through to the next case
  case 4: // Fall to through to the next case
  case 5: cout << "Weekday"; break;
  case 0: // Fall to through to the next case
  case 6: cout << "Weekend";
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace switch statement

Suppose day is 2:

```
switch (da
{
  case 1: // Fall to through to the next case
  case 2: // Fall to through to the next case
  case 3: // Fall to through to the next case
  case 4: // Fall to through to the next case
  case 5: cout << "Weekday"; break;
  case 0: // Fall to through to the next case
  case 6: cout << "Weekend";
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace switch statement

Suppose day is 2:

```
switch (da
{
  case 1:    Fall to through to the next case
  case 2: /  Fall to through to the next case
  case 3: /  Fall to through to the next case
  case 4: /  Fall to through to the next case
  case 5: cout << "Weekday"; break;
  case 0: // Fall to through to the next case
  case 6: cout << "Weekend";
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

69

animation

Trace switch statement

Suppose day is 2:

```
swi    (day)
{
  c   e 1: // Fall to through to the next case
  c   se 2: // Fall to through to the next case
  c   se 3: // Fall to through to the next case
  c   se 4: // Fall to through to the next case
  c   ase 5: cout << "Weekday"; break;
  c   ase 0: // Fall to through to the next case
  c   ase 6: cout << "Weekend";
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

70

Problem: Chinese Zodiac

Write a program that prompts the user to enter a year and displays the animal for the year.



year % 12 =
0: monkey
1: rooster
2: dog
3: pig
4: rat
5: ox
6: tiger
7: rabbit
8: dragon
9: snake
10: horse
11: sheep

```
Enter a year: 1963
rabbit
```

```
Enter a year: 1877
ox
```

ChineseZodiac

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    cout << "Enter a year: ";
7    int year;
8    cin >> year;
9
10   switch (year % 12)
11   {
12     case 0: cout << "monkey" << endl; break;
13     case 1: cout << "rooster" << endl; break;
14     case 2: cout << "dog" << endl; break;
15     case 3: cout << "pig" << endl; break;
16     case 4: cout << "rat" << endl; break;
17     case 5: cout << "ox" << endl; break;
18     case 6: cout << "tiger" << endl; break;
19     case 7: cout << "rabbit" << endl; break;
20     case 8: cout << "dragon" << endl; break;
21     case 9: cout << "snake" << endl; break;
22     case 10: cout << "horse" << endl; break;
23     case 11: cout << "sheep" << endl; break;
24   }
25
26   return 0;
27 }
```

Conditional Operator

A conditional expression evaluates an expression based on a condition.

```
if (x > 0)
  y = 1
else
  y = -1;
```

is equivalent to

```
y = (x > 0) ? 1 : -1;
```

```
(booleanExpression) ? expression1 : expression2
```



✓ Ternary operator ✓ Binary operator ✓ Unary operator

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Conditional Operator

```
if (num % 2 == 0)
  cout << "num is even" << endl;
else
  cout << "num is odd" << endl;
```

```
cout << ((num % 2 == 0) ? "num is even" : "num is odd");
```

```
max = (num1 > num2) ? num1 : num2;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Operator Precedence

How to evaluate the following expression ?

```
3 + 4 * 4 > 5 * (4 + 3) – 1 && (4 – 3 > 5)
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Operator Precedence

TABLE 3.7 Operator Precedence Chart

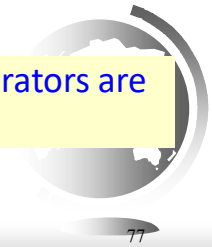
Precedence	*Operator*
↓	`var++` and `var--` (Postfix)
	`+`, `-` (Unary plus and minus), `++var` and `--var` (Prefix)
	`static_cast`<type>(v), (type) (Casting)
	`!` (Not)
	`*`, `/`, `%` (Multiplication, division, and remainder)
	`+`, `-` (Binary addition and subtraction)
	`<`, `<=`, `>`, `>=` (Relational)
	`==`, `!=` (Equality)
	`&&` (AND)
	`||` (OR)
	`=`, `+=`, `-=`, `*=`, `/=`, `%=` (Assignment operator)

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Operator Precedence and Associativity

- The expression in the parentheses is evaluated first.
  - Parentheses can be nested, in which case the expression in the inner parentheses is executed first.
- In expressions without parentheses, the precedence rule and the associativity rule are applied.
- If operators with the same precedence are next to each other, their associativity determines the order of evaluation.

All binary operators except assignment operators are left-associative.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Operator Associativity

`a - b + c - d` is equivalent to `((a - b) + c) - d`

Assignment operators are *right-associative*.

`a = b += c = 5` is equivalent to `a = (b += (c = 5))`



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example

- Applying the operator precedence and associativity rule, the expression 3 + 4 * 4 > 5 * (4 + 3) - 1 is evaluated as follows:

```
3 + 4 * 4 > 5 * (4 + 3) - 1
```
(1) inside parentheses first

```
3 + 4 * 4 > 5 * 7 - 1
```
(2) multiplication

```
3 + 16 > 5 * 7 - 1
```
(3) multiplication

```
3 + 16 > 35 - 1
```
(4) addition

```
19 > 35 - 1
```
(5) subtraction

```
19 > 34
```
(6) greater than

```
false
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Examples

```
bool found = true;
int age = 20;
double hours = 45.30;
double overTime = 15.00;
int count = 20;
char ch = 'B';
```

Expression	Value / Explanation
`!found`	`false` Because `found` is `true`, `!found` is `false`.
`hours > 40.00`	`true` Because `hours` is `45.30` and `45.30 > 40.00` is `true`, the expression `hours > 40.00` evaluates to `true`.
`!age`	`false` `age` is 20, which is nonzero, so `age` evaluates to `true`. Therefore, `!age` is `false`.
`!found && (age >= 18)`	`false` `!found` is `false`; `age > 18` is `20 > 18` is `true`. Therefore, `!found && (age >= 18)` is `false && true`, which evaluates to `false`.
`!(found && (age >= 18))`	`false` Now, `found && (age >= 18)` is `true && true`, which evaluates to `true`. Therefore, `!(found && (age >= 18))` is `!true`, which evaluates to `false`.

D.S. MALIK Book

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Examples

```
bool found = true;
int age = 20;
double hours = 45.30;
double overTime = 15.00;
int count = 20;
char ch = 'B';
```

Expression	Value / Explanation
`hours + overTime <= 75.00`	`true` Because `hours + overTime` is `45.30 + 15.00 = 60.30` and `60.30 <= 75.00` is `true`, it follows that `hours + overTime <= 75.00` evaluates to `true`.
`(count >= 0) && (count <= 100)`	`true` Now, `count` is `20`. Because `20 >= 0` is `true`, `count >= 0` is `true`. Also, `20 <= 100` is `true`, so `count <= 100` is `true`. Therefore, `(count >= 0) && (count <= 100)` is `true && true`, which evaluates to `true`.
`('A' <= ch && ch <= 'Z')`	`true` Here, `ch` is `'B'`. Because `'A' <= 'B'` is `true`, `'A' <= ch` evaluates to `true`. Also, because `'B' <= 'Z'` is `true`, `ch <= 'Z'` evaluates to `true`. Therefore, `('A' <= ch && ch <= 'Z')` is `true && true`, which evaluates to `true`.

D.S. MALIK Book

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Companion Website

Enumerated Types

```
enum Day {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY};
```

Once a type is defined, you can declare a variable of that type:

```
Day day;
```

The variable `day` can hold one of the values defined in the enumerated type.

For example, the following statement assigns enumerated value `MONDAY` to variable `day`:

```
day = MONDAY;
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Appendix

Enumerated Types

As with any other type, you can declare and initialize a variable in one statement:

```
Day day = MONDAY;
```

Furthermore, C++ allows you to declare an enumerated type and variable in one statement. For example,

```
enum Day {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY} day = MONDAY;
```



TestEnumeratedType

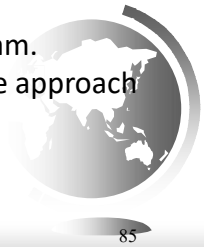
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    enum Day {MONDAY = 1, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY} day;
7
8    cout << "Enter a day (1 for Monday, 2 for Tuesday, etc): ";
9    int dayNumber;
10   cin >> dayNumber;
11
12   switch (dayNumber) {
13   case MONDAY:
14        cout << "Play soccer" << endl;
15        break;
16   case TUESDAY:
17        cout << "Piano lesson" << endl;
18        break;
19   case WEDNESDAY:
20        cout << "Math team" << endl;
21        break;
22   default:
23       cout << "Go home" << endl;
24   }
25
26   return 0;
27 }
```

Debugging

Debugging is the process of finding and fixing errors in a program.

- ✓ Logic errors are called *bugs*.
- ✓ A common approach to debugging is to use a combination of methods to narrow down to the part of the program where the bug is located.
  - ✓ You can hand-trace the program (i.e., catch errors by reading the program),
  - ✓ you can insert print statements in order to show the values of the variables or the execution flow of the program.
- ✓ This approach might work for a short, simple program.
- ✓ But for a large, complex program, the most effective approach for debugging is to use a debugger utility.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Debugger

Debugger is a program that facilitates debugging.

- ✓ You can use a debugger to
  - ✓ Execute a single statement at a time.
  - ✓ Trace into or stepping over a method.
  - ✓ Set breakpoints.
  - ✓ Display variables.
  - ✓ Display call stack.
  - ✓ Modify variables.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

References

- Introduction to Programming with C++, Daniel Liang, Pearson Education Limited 2014
- C++ Programming- From Problem Analysis to Program Design - D.S. MALIK - 2018



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

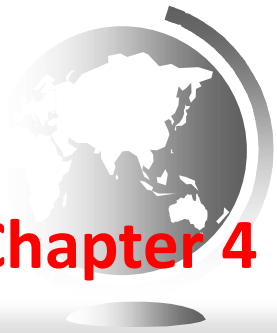
Computer Engineering Department
Islamic University of Gaza

Dr. Ayman Maliha
amaliha@iugaza.edu.ps
Homepage: http://site.iugaza.edu.ps/amaliha/

ECOM 2403
Computer Programming
Fall 2021-2022



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.



Chapter 4

Mathematical Functions, Characters, and Strings

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved. 2

Mathematical Functions

C++ provides many useful functions in the **cmath** header for performing common mathematical functions.

- They can be categorized as
  - ✓ *Trigonometric functions*
  - ✓ *Exponent functions*
  - ✓ *Service functions*



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Trigonometric Functions

TABLE 4.1 Trigonometric Functions in the `cmath` Header

Function	*Description*
`sin(radians)`	Returns the trigonometric sine of an angle in radians.
`cos(radians)`	Returns the trigonometric cosine of an angle in radians
`tan(radians)`	Returns the trigonometric tangent of an angle in radians.
`asin(a)`	Returns the angle in radians for the inverse of sine.
`acos(a)`	Returns the angle in radians for the inverse of cosine.
`atan(a)`	Returns the angle in radians for the inverse of tangent.

- ✓ The parameter for `sin`, `cos`, and `tan` is an angle in radians.
- ✓ The return value for `asin`, `acos`, and `atan` is an angle in radians in the range between -∏/2 and ∏ /2.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Examples

- ✓ `sin(0)` returns `0.0`
- ✓ `sin(270 * PI / 180)` returns `-1.0`
- ✓ `sin(PI / 6)` returns `0.5`
- ✓ `sin(PI / 2)` returns `1.0`
- ✓ `cos(0)` //returns `1.0`
- ✓ `cos(PI / 6)` returns `0.866`
- ✓ `cos(PI / 2)` returns `0`
- ✓ `asin(0.5)` returns `0.523599` (same as ∏ /6)
- ✓ `acos(0.5)` returns `1.0472` (same as ∏ /3)
- ✓ `atan(1.0)` returns `0.785398` (same as ∏ /4)

`PI` is a constant `3.14159`

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Exponent Functions

TABLE 4.2 Exponent Functions in the `cmath` Header

Function	*Description*
`exp(x)`	Returns e raised to power of x (e^x).
`log(x)`	Returns the natural logarithm of x ($\ln(x) = \log_e(x)$).
`log10(x)`	Returns the base 10 logarithm of x ($\log_{10}(x)$).
`pow(a, b)`	Returns a raised to the power of b (a^b).
`sqrt(x)`	Returns the square root of x ($\sqrt{x}$) for x >= 0.

- ✓ `exp(1.0)` returns `2.71828`
- ✓ `log(E)` returns `1.0`
- ✓ `log10(10.0)` returns `1.0`
- ✓ `pow(2.0, 3)` returns `8.0`
- ✓ `sqrt(4.0)` returns `2.0`
- ✓ `sqrt(10.5)` returns `3.24`

Assume `E` is a constant with value `2.71828`

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Rounding Functions

TABLE 4.3 Rounding Functions in the `cmath` Header

Function	*Description*
`ceil(x)`	x is rounded up to its nearest integer. This integer is returned as a double value.
`floor(x)`	x is rounded down to its nearest integer. This integer is returned as a double value.

```
✓ ceil(2.1)          returns 3.0
✓ ceil(2.0)          returns 2.0
✓ ceil(-2.0)         returns -2.0
✓ ceil(-2.1)         returns -2.0
✓ floor(2.1)         returns 2.0
✓ floor(2.0)         returns 2.0
✓ floor(-2.0)        returns -2.0
✓ floor(-2.1)        returns -3.0
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

The min, max, and abs Functions

The functions **`min`**, **`max`**, and **`abs`** are defined in the **`cstdlib`** header.

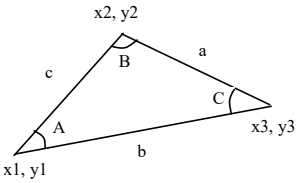
```
✓ max(2, 3) returns 3
✓ max(2.5, 3.0) returns 3.0
✓ min(2.5, 4.6) returns 2.5
✓ abs(-2) returns 2
✓ abs(-2.1) returns 2.1
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study: Computing Angles of a Triangle

Write a program that prompts the user to enter the x- and y-coordinates of the three corner points in a triangle and then displays the triangle's angles.




```
A = acos((a * a - b * b - c * c) / (-2 * b * c))
B = acos((b * b - a * a - c * c) / (-2 * a * c))
C = acos((c * c - b * b - a * a) / (-2 * a * b))
```

```
Enter three points: 1 1 6.5 1 6.5 2.5
The three angles are 15.2551 90.0001 74.7449
```

ComputeAngles

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int main()
6  {
7    // Prompt the user to enter three points
8    cout << "Enter three points: ";
9    double x1, y1, x2, y2, x3, y3;
10   cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
11
12   // Compute three sides
13   double a = sqrt((x2 - x3) * (x2 - x3) + (y2 - y3) * (y2 - y3));
14   double b = sqrt((x1 - x3) * (x1 - x3) + (y1 - y3) * (y1 - y3));
15   double c = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
16
17   // Obtain three angles in degrees
18   const double PI = 3.14159;
19   double A = acos((a * a - b * b - c * c) / (-2 * b * c)) * 180 / PI;
20   double B = acos((b * b - a * a - c * c) / (-2 * a * c)) * 180 / PI;
21   double C = acos((c * c - b * b - a * a) / (-2 * a * b)) * 180 / PI;
22
23   // Display the angles in degrees
24   cout << "The three angles are " << round(A * 100) / 100.0 << " "
25   << round(B * 100) / 100.0 << " " << round(C * 100) / 100.0 << endl;
26
27   return 0;
28 }
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Character Data Type

A character data type represents a single character.

```
char letter = 'A';          (ASCII)
char numChar = '4';         (ASCII)
```

- ✓ A character is stored in a computer as a sequence of 0s and 1s.
- ✓ Mapping a character to its binary representation is called *encoding*.
- ✓ There are different ways to encode a character.
- ✓ ASCII (American Standard Code for Information Interchange)
  - ✓ An 8-bit encoding scheme for representing all uppercase and lowercase letters, digits, punctuation marks, and control characters.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Character Data Type

TABLE 4.4 ASCII Code for Commonly Used Characters

Characters	*ASCII Code*
'0' to '9'	48 to 57
'A' to 'Z'	65 to 90
'a' to 'z'	97 to 122

Appendix B
The ASCII Character Set

- ➢ NOTE: The increment and decrement operators can also be used on **char** variables to get the next or preceding character.
- ➢ For example, the following statements display character b.

```
char ch = 'a';
cout << ++ch;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Read Characters

To read a character from the keyboard, use

```
cout << "Enter a character: ";
char ch;
cin >> ch;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Escape Sequences for Special Characters

- want to print a message with quotation marks in the output
  `"Programming is fun"`

```
cout << "He said "Programming is fun"" << endl;
```

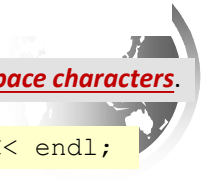
This statement has a compile error.

TABLE 4.5 Escape Sequences

Escape Sequence	*Name*	*ASCII Code*	*Escape Sequence*	*Name*	*ASCII Code*
`\b`	Backspace	8	`\r`	Carriage Return	13
`\t`	Tab	9	`\\`	Backslash	92
`\n`	Linefeed	10	`\"`	Double Quote	34
`\f`	Formfeed	12			

The characters `' '`, `\t`, `\f`, `\r`, or `\n` are known as ***whitespace characters***.

```
cout << "He said \"Programming is fun\"" << endl;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Output

```
cout << "Hello there.\nMy name is James.";

or:
cout << "Hello there.";
cout << "\nMy name is James.";

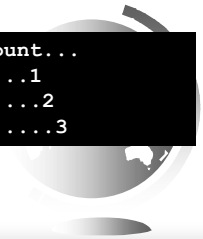
or:
cout << "Hello there.";
cout << endl << "My name is James.";
```

```
Hello there.
My name is James.
```

```
cout << "Count...\n....1\n.....2\n......3";

or:
cout << "Count..." << endl << "....1" << endl
<< ".....2" << endl << "......3";
```

```
Count...
....1
.....2
......3
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Casting between `char` and Numeric Types

A **`char`** can be cast into any numeric type, and vice versa.

```
int i = 'a';
//Same as int i = static_cast<int>('a');
char c = 97;
//Same as char c = static_cast<char>(97);
char c = 0XFF41;
// The lower 8 bits hex code 41 is assigned to c
// variable c is character A
char c = 65.25;
// 65 is assigned to variable c
// variable c is character A
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

16

Numeric Operators on Characters

- ✓ The `char` type is treated as if it is an integer of the byte size.
- ✓ All numeric operators can be applied to `char` operands.
- ✓ A `char` operand is automatically cast into a number if the other operand is a number or a character.

```
// (int)'2' is 50 and (int)'3' is 51
int i = '2' + '3';

// i is decimal 101
cout << "i is " << i << endl;

// (int)'a' is 97
int j = 2 + 'a';
cout << "j is " << j  << endl;

cout << j << " is the ASCII code for character "
     << static_cast<char>(j) << endl;
```

```
i is 101
```

```
j is 99
```

```
99 is the ASCII code for character c
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Note

- ✓ The ASCII for lowercase letters are consecutive integers starting from the code for `'a'`, then for `'b'`, `'c'`, …, and `'z'`.
  - ✓ The same is true for the uppercase letters.
- ✓ The ASCII code for `'a'` is greater than the code for `'A'`.
  - ✓ So `'a'` - `'A'` is the same as `'b'` - `'B'`.
- ✓ For a lowercase letter **`ch`**, its corresponding uppercase letter is

```
static_cast<char>('A' + (ch - 'a')).
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Problem: Converting a Lowercase to Uppercase

Write a program that prompts the user to enter a lowercase letter and finds its corresponding uppercase letter.

```
Enter a lowercase letter: b
The corresponding uppercase letter is B
```

ToUppercase

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

19

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    char lowercaseLetter;
7    cout << "Enter a lowercase letter: ";
8    cin >> lowercaseLetter;
9
10   char uppercaseLetter =
11   static_cast<char>('A' + (lowercaseLetter - 'a'));
12
13   cout << "The corresponding uppercase letter is "
14        << uppercaseLetter << endl;
15
16   return 0;
17 }
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

20

Comparing and Testing Characters

- Two characters can be compared using the comparison operators just like comparing two numbers.
- This is done by comparing the ASCII codes of the two characters.

✓ `'a' < 'b'` is true because the ASCII code for `'a'` (`97`) is less than the ASCII code for `'b'` (`98`).
✓ `'a' < 'A'` is false because the ASCII code for `'a'` (`97`) is greater than the ASCII code for `'A'` (`65`).
✓ `'1' < '8'` is true because the ASCII code for `'1'` (`49`) is less than the ASCII code for `'8'` (`56`).

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study: Generating Random Characters

- Every character has a unique ASCII code between 0 and 127.
- To generate a random character is to generate a random integer between 0 and 127.
- Recall that you can use the srand(seed) function to set a seed and use rand() to return a random integer.
- You can use it to write a simple expression to generate random numbers in any range.

```
Enter a starting character: a
Enter an ending character: z
The random character between a and z is p
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study: Generating Random Characters, cont.

Examples:

```
rand() % 10
```
→ Returns a random integer between 0 and 9.

```
50 + rand() % 50
```
→ Returns a random integer between 50 and 99.

In general,

```
a + rand() % b
```
→ Returns a random number between a and a + b, excluding a + b.

DisplayRandomCharacter

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4
5  int main()
6  {
7    cout << "Enter a starting character: ";
8    char startChar;
9    cin >> startChar;
10
11   cout << "Enter an ending character: ";
12   char endChar;
13   cin >> endChar;
14
15   // Get a random character
16   char randomChar = static_cast<char>(startChar + rand() %
17   (endChar - startChar + 1));
18
19   cout << "The random character between " << startChar << " and "
20   << endChar << " is " << randomChar << endl;
21
22   return 0;
23 }
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Character Functions

C++ provides several functions for testing a character and for converting a character in the `cctype` header file

Function	Description
`isdigit(ch)`	Returns true if the specified character is a digit.
`isalpha(ch)`	Returns true if the specified character is a letter.
`isalnum(ch)`	Returns true if the specified character is a letter or digit.
`islower(ch)`	Returns true if the specified character is a lowercase letter.
`isupper(ch)`	Returns true if the specified character is an uppercase letter.
`isspace(ch)`	Returns true if the specified character is a whitespace character.
`tolower(ch)`	Returns the lowercase of the specified character.
`toupper(ch)`	Returns the uppercase of the specified character.

CharacterFunctions

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  #include <cctype>
3  using namespace std;
4
5  int main()
6  {
7    cout << "Enter a character: ";
8    char ch;
9    cin >> ch;
10
11   cout << "You entered " << ch << endl;
12
13   if (islower(ch))
14   {
15   cout << "It is a lowercase letter " << endl;
16   cout << "Its equivalent uppercase letter is " <<
17   static_cast<char>(toupper(ch)) << endl;
18   }
19   else if (isupper(ch))
20   {
21   cout << "It is an uppercase letter " << endl;
22   cout << "Its equivalent lowercase letter is " <<
23   static_cast<char>(tolower(ch)) << endl;
24   }
25   else if (isdigit(ch))
26   {
27   cout << "It is a digit character " << endl;
28   }
29
30   return 0;
31 }
```

```
Enter a character: a
You entered a
It is a lowercase letter
Its equivalent uppercase letter is A
```

```
Enter a character: T
You entered T
It is an uppercase letter
Its equivalent lowercase letter is t
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study: Converting a Hexadecimal Digit to a Decimal Value

Write a program that converts a hexadecimal digit into a decimal value.

Self study from the text book

HexDigit2Dec



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

The `string` Type

A string is a sequence of characters

- To use the `string` type, you need to access its definition from the header file `string`
- Include the following preprocessor directive:

```
#include  <string>

string s;
string message = "Programming is fun";
```

- The **`string`** type is not a primitive type.
  - It is known as an object type.
- Here `message` represents a **`string`** object with contents `Programming is fun`.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

The `string` Type

TABLE 4.7 Simple Functions for `string` Objects

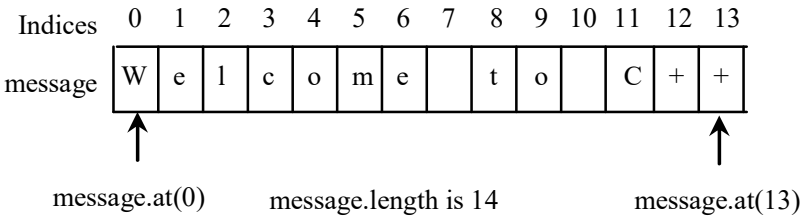
Function	*Description*
`length()`	Returns the number of characters in this string.
`size()`	Same as length().
`at(index)`	Returns the character at the specified index from this string.

```
string message = "ABCD";
cout << message.length() << endl;
cout << message.at(0) << endl;
string s = "Bottom";
cout << s.length() << endl;
cout << s.at(1) << endl;
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

29

String Subscript Operator



- ✓ C++ provides the subscript operator to access the character at a specified index in a string.
- ✓ Using the syntax **`stringName[index]`**.
- ✓ You can use this syntax to retrieve and modify the character in a string.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

30

string Subscript Operator

```
string s = "ABCD";
s[0] = 'P';
cout << s[0] << endl;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Concatenating Strings

```
string s3 = s1 + s2;
//concatenates strings s1 and s2 into s3
```

```
string message = "Welcome to C++"
message += " and programming is fun";
```

Appends the string `"and programming is fun"` with the string `"Welcome to C++"` in `message`.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Comparing Strings

- You can use the relational operators ==, !=, <, <=, >, >= to compare two strings.
- This is done by comparing their corresponding characters on by one from left to right.

```
string s1 = "ABC";
string s2 = "ABE";

cout << (s1 == s2) << endl;       // Displays 0 (means false)
cout << (s1 != s2) << endl;       // Displays 1 (means true)
cout << (s1 > s2) << endl;        // Displays 0 (means false)
cout << (s1 >= s2) << endl;       // Displays 0 (means false)
cout << (s1 < s2) << endl;        // Displays 1 (means true)
cout << (s1 <= s2) << endl;       // Displays 1 (means true)
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Reading Strings

```
1  string city;
2  cout << "Enter a city: ";
3  cin >> city;            // Read to string city
4  cout << "You entered " << city << endl;
```

If you want to enter New York, you have to use an alternative approach. C++ provides the `getline` function in the `string` header file

```
getline(cin, s, delimitCharacter)
```

```
1  string city;
2  cout << "Enter a city: ";
3  getline(cin, city, '\n'); // Same as getline(cin, city)
4  cout << "You entered " << city << endl;
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example

Write a program that prompts the user to enter two cities and displays them in alphabetical order.

```
Enter the first city: New York
Enter the second city: Boston
The cities in alphabetical order are Boston New York
```



OrderTwoCities

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7    string city1, city2;
8    cout << "Enter the first city: ";
9    getline(cin, city1);
10   cout << "Enter the second city: ";
11   getline(cin, city2);
12
13   cout << "The cities in alphabetical order are ";
14   if (city1 < city2)
15   cout << city1 << " " << city2 << endl;
16   else
17   cout << city2 << " " << city1 << endl;
18
19   return 0;
20 }
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Formatting Console Output

You can use the stream manipulators to display formatted output on the console.

TABLE 4.8 Frequently Used Stream Manipulators

Operator	*Description*
`setprecision(n)`	sets the precision of a floating-point number
`fixed`	displays floating-point numbers in fixed-point notation
`showpoint`	causes a floating-point number to be displayed with a decimal point with trailing zeros even if it has no fractional part
`setw(width)`	specifies the width of a print field
`left`	justifies the output to the left
`right`	justifies the output to the right

- ✓ **These functions are called stream manipulators**
- ✓ **Included in the `iomanip` header file**

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

37

`setprecision(n)` Manipulator

- ✓ Total number of digits displayed for a floating-point number

```
double number = 12.34567;
cout << setprecision(3) << number << " "
     << setprecision(4) << number << " "
     << setprecision(5) << number << " "
     << setprecision(6) << number << endl;

displays
12.3□12.35□12.346□12.3457
```

- ✓ The `setprecision` manipulator
  - ✓ Remains in effect until the precision is changed.
  - ✓ Ignored If the width is not sufficient for an `integer`

```
cout << setprecision(3) << 23456 << endl;
```

Displays `23456`

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

38

`fixed` Manipulator

- Large floating-point number can be displayed in scientific notation.
- **`fixed`** manipulator
  - Force the number to be displayed in nonscientific notation with a fixed number of digits after the decimal point (default 6)
  - You can change it using the **`fixed`** along with the **`setprecision`**
  - **`setprecision`** specifies the number of digits after the decimal point when it is used after the **`fixed`** manipulator,

```
cout << 232123434.357;
```
displays
```
2.32123e+08
```

```
cout << fixed << 232123434.357;
```
displays
```
232123434.357000
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

`fixed` Manipulator

```
double monthlyPayment = 345.4567;
double totalPayment = 78676.887234;
cout << fixed << setprecision(2)
     << monthlyPayment << endl
     << totalPayment << endl;
```

displays

```
345.46
78676.89
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

`showpoint` Manipulator

- By default, floating-point numbers that **do not have a fractional** part are **not displayed with a decimal point**.
- You can use the **`fixed`** manipulator **to force** the floating-point numbers to be displayed with a decimal point.
- Alternatively, you can use the **`showpoint`** manipulator together with the **`setprecision`** manipulator.

```
cout << setprecision(6);
cout << 1.23 << endl;
cout << showpoint << 1.23 << endl;
cout << showpoint << 123.0 << endl;
```

displays

```
1.23
1.23000
123.000
```

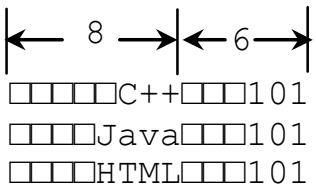
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

`setw(width)` Manipulator

Use **`setw(width)`** to specify the minimum number of columns for an output

```
cout << setw(8) << "C++" << setw(6) << 101 << endl;
cout << setw(8) << "Java" << setw(6) << 101 << endl;
cout << setw(8) << "HTML" << setw(6) << 101 << endl;
```

displays



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

`left` and `right` Manipulators

```
cout << right;
cout << setw(8) << 1.23 << endl;
cout << setw(8) << 351.34 << endl;
```

displays

```
□□□□1.23
□□351.34
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

`left` and `right` Manipulators

```
cout << left;
cout << setw(8) << 1.23;
cout << setw(8) << 351.34 << endl;
```

displays

```
1.23□□□□351.34□□
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example: Display a Table

```
Degrees    Radians    Sine       Cosine     Tangent
30         0.5236     0.5000     0.8660     0.5773
60         1.0472     0.8660     0.5000     1.7320
```

```
Degrees    Radians    Sine       Cosine     Tangent
30         0.5236     0.5000     0.8660     0.5773
60         1.0472     0.8660     0.5000     1.7320
```

FormatDemo

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  #include <cmath>
3  #include <iomanip>
4  using namespace std;
5
6  int main()
7  {
8   // Display the header of the table
9   cout << left << setw(10) << "Degrees" << setw(10) << "Radians"
10   << setw(10) << "Sine" << setw(10) << "Cosine" << setw(10)
11   << "Tangent" << endl;
12
13   // Display values for 30 degrees
14   const double PI = 3.14159;
15   double degrees = 30;
16   double radians = degrees * (PI / 180);
17   cout << setw(10) << degrees << setw(10) << fixed
18   << setprecision(4) << radians << setw(10) << sin(radians)
19   << setw(10) << cos(radians) << setw(10) << tan(radians) << endl;
20
21   // Display values for 60 degrees
22   degrees = 60;
23   radians = degrees * (PI / 180);
24   cout << setw(10) << setprecision(0) << degrees << setw(10)
25   << setprecision(4) << radians << setw(10)
26   << sin(radians) << setw(10) << cos(radians) << setw(10)
27   << tan(radians) << endl;
28
29   return 0;
30  }
```

Simple File Output

- To write data to a file, first declare a variable of the **ofstream** type:

```
ofstream output;
```

- To specify a file, invoke the **open** function from **output** object as follows:

```
output.open("numbers.txt");
```

- Optionally, you can create a file output object and open the file in one statement like this:

```
ofstream output("numbers.txt");
```

- To write data, use the stream insertion operator (**<<**) in the same way that you send data to the **cout** object.

```
output << 95 << " " << 56
       << " "<< 34 << endl;
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4
5  int main()
6  {
7    ofstream output;
8
9    // Create a file
10   output.open("scores.txt");
11
12   // Write numbers
13   output << 95 << " " << 56 << " " << 34;
14
15   // Close file
16   output.close();
17
18   cout << "Done" << endl;
19
20   return 0;
21 }
```

```
output << 95 << " " << 56 << " " << 34 << endl;

scores.txt
file      95 56 34
```

SimpleFileOutput

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Simple File Input

- To read data from a file, first declare a variable of the **`ifstream`** type:

```
ifstream input;
```

- To specify a file, invoke the **open** function from **input** as follows:

```
input.open("numbers.txt");
```

- Optionally, you can create a file output object and open the file in one statement like this:

```
ifstream input("numbers.txt");
```

- To read data, use the stream extraction operator (**>>**) in the same way that you read data from the **`cin`** object.

```
input << score1 << score2 << score3;
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4
5  int main()
6  {
7    ifstream input;
8
9    // Open a file
10   input.open("numbers.txt");
11
12   int score1, score2, score3;
13
14   // Read data
15   input >> score1;
16   input >> score2;
17   input >> score3;
18
19   cout << "Total score is " << score1 + score2 + score3 << endl;
20
21   // Close file
22   input.close();
23
24   cout << "Done" << endl;
25   return 0;
26 }
```

```
Total score is 185
Done
```

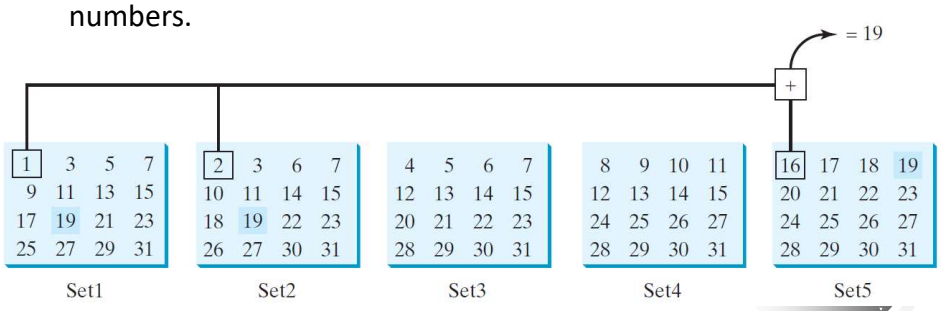
```
input >> score1; input >> score2; input >> score3;
```

scores.txt file: 95 56 34

SimpleFileInput

Case Study: Guessing Birthdays

- ✓ This section uses the if statements to write an interesting game program.
- ✓ You can find out the date of the month when your friend was born by asking five questions.
- ✓ Each question asks whether the day is in one of the five sets of numbers.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Mathematics Basis for the Game

19 is 10011 in binary. 7 is 111 in binary. 23 is 11101 in binary

```
  10000        00110        10000
     10           10         1000
+     1    +      1          100
  10011        00111    +     1
                          11101

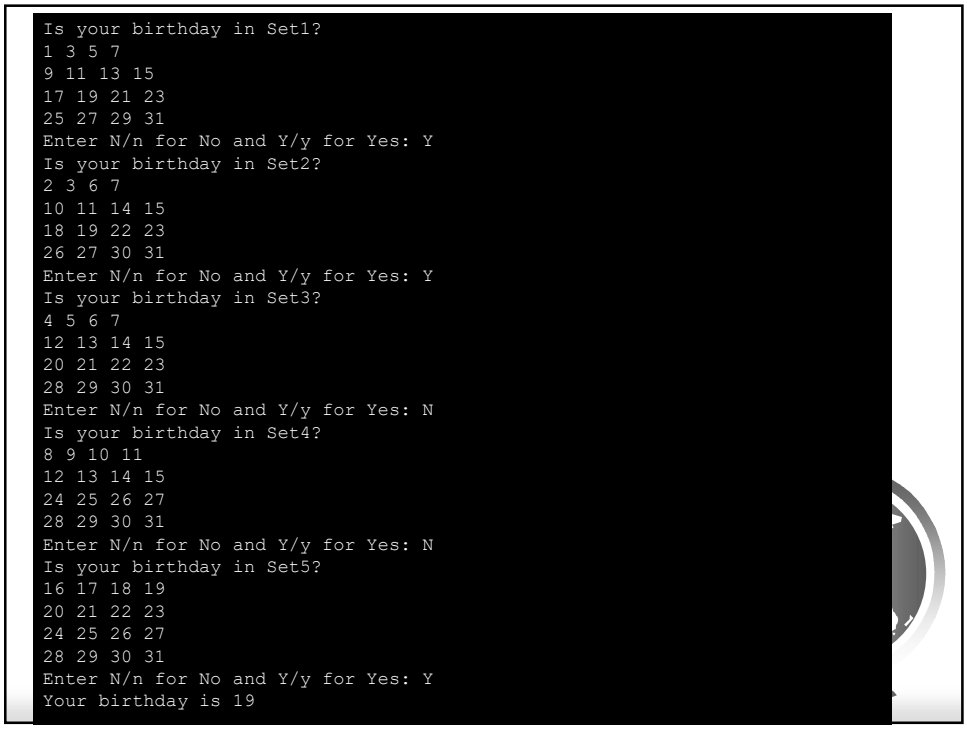
     19           7           23
```

Decimal	Binary
1	00001
2	00010
3	00011
...	
19	10011
...	
31	11111

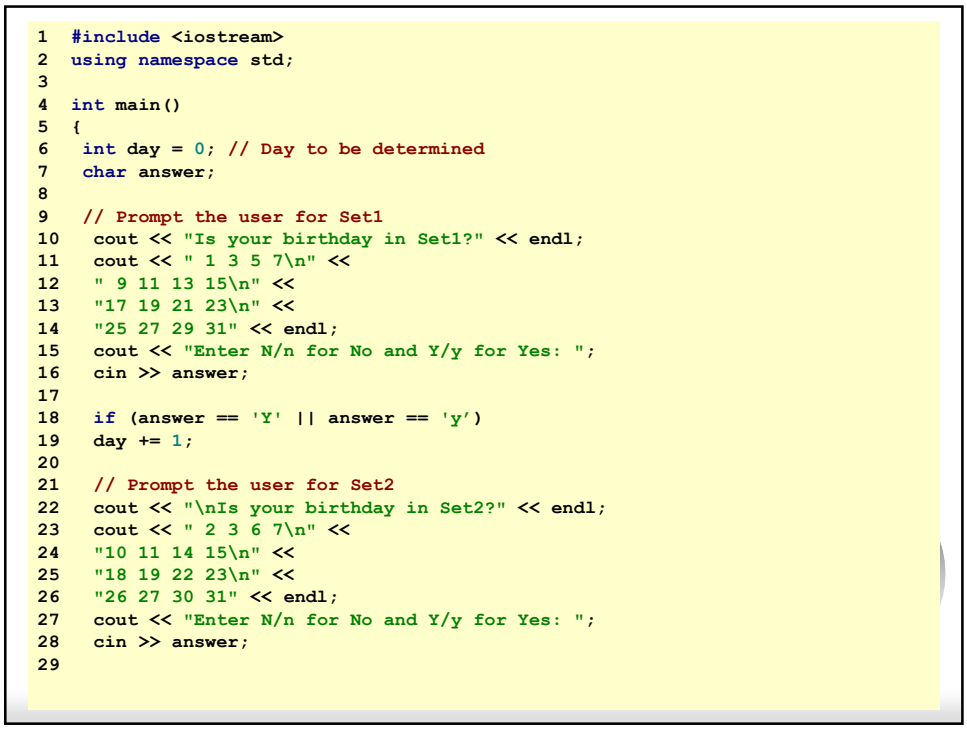
$$
\begin{array}{r}
b_5\ 0\ 0\ 0\ 0 \\
b_4\ 0\ 0\ 0 \\
b_3\ 0\ 0 \\
b_2\ 0 \\
+\qquad b_1 \\
\hline
b_5\, b_4\, b_3\, b_2\, b_1
\end{array}
$$

$$
\begin{array}{r}
10000 \\
10 \\
+\quad 1 \\
\hline
10011 \\
19
\end{array}
\qquad
\begin{array}{r}
10000 \\
1000 \\
100 \\
10 \\
+\quad 1 \\
\hline
11111 \\
31
\end{array}
$$

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.



```
Is your birthday in Set1?
1 3 5 7
9 11 13 15
17 19 21 23
25 27 29 31
Enter N/n for No and Y/y for Yes: Y
Is your birthday in Set2?
2 3 6 7
10 11 14 15
18 19 22 23
26 27 30 31
Enter N/n for No and Y/y for Yes: Y
Is your birthday in Set3?
4 5 6 7
12 13 14 15
20 21 22 23
28 29 30 31
Enter N/n for No and Y/y for Yes: N
Is your birthday in Set4?
8 9 10 11
12 13 14 15
24 25 26 27
28 29 30 31
Enter N/n for No and Y/y for Yes: N
Is your birthday in Set5?
16 17 18 19
20 21 22 23
24 25 26 27
28 29 30 31
Enter N/n for No and Y/y for Yes: Y
Your birthday is 19
```



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

int main()
{
 int day = 0; // Day to be determined
 char answer;

 // Prompt the user for Set1
  cout << "Is your birthday in Set1?" << endl;
  cout << " 1 3 5 7\n" <<
  " 9 11 13 15\n" <<
  "17 19 21 23\n" <<
  "25 27 29 31" << endl;
  cout << "Enter N/n for No and Y/y for Yes: ";
  cin >> answer;

  if (answer == 'Y' || answer == 'y')
  day += 1;

  // Prompt the user for Set2
  cout << "\nIs your birthday in Set2?" << endl;
  cout << " 2 3 6 7\n" <<
  "10 11 14 15\n" <<
  "18 19 22 23\n" <<
  "26 27 30 31" << endl;
  cout << "Enter N/n for No and Y/y for Yes: ";
  cin >> answer;

```

```
30   if (answer == 'Y' || answer == 'y')
31   day += 2;
32
33   // Prompt the user for Set3
34   cout << "\nIs your birthday in Set3?" << endl;
35   cout << " 4 5 6 7\n" <<
36   "12 13 14 15\n" <<
37   "20 21 22 23\n" <<
38   "28 29 30 31" << endl;
39   cout << "Enter N/n for No and Y/y for Yes: ";
40   cin >> answer;
41
42   if (answer == 'Y' || answer == 'y')
43   day += 4;
44
45   // Prompt the user for Set4
46   cout << "\nIs your birthday in Set4?" << endl;
47   cout << " 8 9 10 11\n" <<
48   "12 13 14 15\n" <<
49   "24 25 26 27\n" <<
50   "28 29 30 31" << endl;
51   cout << "Enter N/n for No and Y/y for Yes: ";
52   cin >> answer;
53
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

55

```
54   if (answer == 'Y' || answer == 'y')
55   day += 8;
56
57   // Prompt the user for Set5
58   cout << "\nIs your birthday in Set5?" << endl;
59   cout << "16 17 18 19\n" <<
60   "20 21 22 23\n" <<
61   "24 25 26 27\n" <<
62   "28 29 30 31" << endl;
63   cout << "Enter N/n for No and Y/y for Yes: ";
64   cin >> answer;
65
66   if (answer == 'Y' || answer == 'y')
67   day += 16;
68
69   cout << "Your birthday is " << day << endl;
70
71   return 0;
72 }
```

GuessBirthday

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

56

Case Study: Revising the Lottery Program Using Strings

- A problem can be solved using many different approaches. This section rewrites the lottery program in Listing 3.7 using strings.
- Using strings simplifies this program.



LotteryUsingStrings

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
Enter your lottery pick (two digits): 00
The lottery number is 00
Exact match: you win $10,000
```

```
Enter your lottery pick (two digits): 45
The lottery number is 54
Match all digits: you win $3,000
```

```
Enter your lottery pick: 23
The lottery number is 34
Match one digit: you win $1,000
```

```
Enter your lottery pick: 23
The lottery number is 14
Sorry, no match
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  #include <string> // for using strings
3  #include <ctime> // for time function
4  #include <cstdlib> // for rand and srand functions
5  using namespace std;
6
7  int main()
8  {
9   string lottery;
10   srand(time(0));
11   int digit = rand() % 10; // Generate first digit
12   lottery += static_cast<char>(digit + '0');
13   digit = rand() % 10; // Generate second digit
14   lottery += static_cast<char>(digit + '0');
15
16   // Prompt the user to enter a guess
17   cout << "Enter your lottery pick (two digits): ";
18   string guess;
19   cin >> guess;
20
21   cout << "The lottery number is " << lottery << endl;
22
23   // Check the guess
24   if (guess == lottery)
25   cout << "Exact match: you win $10,000" << endl;
26   else if (guess[1] == lottery[0] && guess[0] == lottery[1])
27   cout << "Match all digits: you win $3,000" << endl;
28   else if (guess[0] == lottery[0] || guess[0] == lottery[1]
29   || guess[1] == lottery[0] || guess[1] == lottery[1])
30   cout << "Match one digit: you win $1,000" << endl;
31   else
32   cout << "Sorry, no match" << endl;
33
34   return 0;
35  }
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.



59

References

- Introduction to Programming with C++, Daniel Liang, Pearson Education Limited 2014
- C++ Programming- From Problem Analysis to Program Design - D.S. MALIK - 2018



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

60

Computer Engineering Department
Islamic University of Gaza

Dr. Ayman Maliha
amaliha@iugaza.edu.ps
Homepage: http://site.iugaza.edu.ps/amaliha/

ECOM 2403
Computer Programming
Fall 2021-2022



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.



Chapter 5

Loops

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Motivations

- Suppose that you need to print a string (e.g., `"Welcome to C++!"`) a hundred times.
- It would be tedious to have to write the following statement a hundred times:

```
cout  << "Welcome to C++!" << endl;
```

So, how do you solve this problem?



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

3

Opening Problem

Problem:

```
                cout << "Welcome to C++!" << endl;
                cout << "Welcome to C++!" << endl;
                cout << "Welcome to C++!" << endl;
                cout << "Welcome to C++!" << endl;
                cout << "Welcome to C++!" << endl;

    100         …
    times       …
                …
                cout << "Welcome to C++!" << endl;
                cout << "Welcome to C++!" << endl;
                cout << "Welcome to C++!" << endl;
                cout << "Welcome to C++!" << endl;
                cout << "Welcome to C++!" << endl;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

4

Introducing `while` Loops

A loop can be used to tell a program to execute statements repeatedly

```
int count = 0;
while (count < 100)
{
  cout << "Welcome to C++!\n";
  count++;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

5

Loop Statements

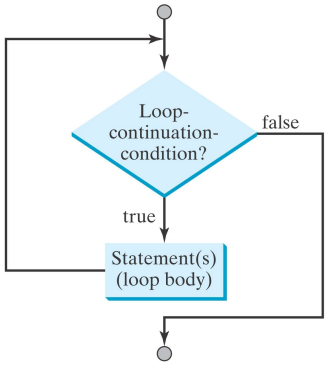
- *Loops are repetition statements* that allow us to execute a statement (or block of statements) multiple times
- Like conditional statements, they are controlled by boolean expressions
- Three types of loop statements:
  - the *while loop*
  - the *do-while loop*
  - the *for loop*
- The programmer should choose the right type of loop for the situation at hand



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

while Loop Flow Chart

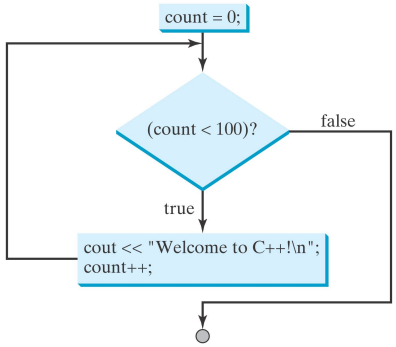
```
while (loop-continuation-condition)
{
  // loop-body;
  Statement(s);
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

7

while Loop Flow Chart




```
int count = 0;
while (count < 100)
{
  cout << "Welcome to C++!\n";
  count++;
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

8

animation

Trace `while` Loop

```
int count = 0;
while (count < 2)
{
  cout << "Welcome to C++!";
  count++;
}
```

Initialize count



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

9

animation

Trace `while` Loop, cont.

```
int count = 0;
while (count < 2)
{
  cout << "Welcome to C++!";
  count++;
}
```

(count < 2) is true



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

10

animation

Trace `while` Loop, cont.

```
int count = 0;
while (count < 2)
{
  cout << "Welcome to C++!";
  count++;
}
```

Print Welcome to C++



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace `while` Loop, cont.

```
int count = 0;
while (count < 2)
{
  cout << "Welcome to C++!";
  count++;
}
```

Increase count by 1
count is 1 now



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace `while` Loop, cont.

```
int count = 0;
while (count < 2)
{
  cout << "Welcome to C++!";
  count++;
}
```

(count < 2) is still true since count is 1



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

13

animation

Trace `while` Loop, cont.

```
int count = 0;
while (count < 2)
{
  cout << "Welcome to C++!";
  count++;
}
```

Print Welcome to C++



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

14

animation

Trace `while` Loop, cont.

```
int count = 0;
while (count < 2)
{
  cout << "Welcome to C++!";
  count++;
}
```

Increase count by 1
count is 2 now



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

15

animation

Trace `while` Loop, cont.

```
int count = 0;
while (count < 2)
{
 cout << "Welcome to C++!";
  count++;
}
```

(count < 2) is false since count is 2 now



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

16

animation

Trace `while` Loop, cont.

```
int count = 0;
while (count < 2)
{
  cout << "Welcome to C++!"
  count++;
}
```

The loop exits. Execute the next statement after the loop.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

17

`while` Loop Example

- An example of a while statement:

```
int count = 1;
while (count <= 5) {
  cout << count;
  count++;
}
```

- If the condition is false initially, the statement (loop body) is never executed
  - Therefore, the body of a while loop will execute zero or more times



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Problem: Repeat Subtraction Until Correct

- Recall that SubtractionQuiz.cpp gives a program that prompts the user to enter an answer for a question on subtraction.
- Using a loop, you can rewrite the program to let the user enter a new answer until it is correct.

```
What is 4 - 3? 4
Wrong answer. Try again. What is 4 - 3? 5
Wrong answer. Try again. What is 4 - 3? 1
You got it!
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  #include <ctime> // for time function
3  #include <cstdlib> // for rand and srand functions
4  using namespace std;
5
6  int main()
7  {
8    // 1. Generate two random single-digit integers
9    srand(time(0));
10   int number1 = rand() % 10;
11   int number2 = rand() % 10;
12
13   // 2. If number1 < number2, swap number1 with number2
14   if (number1 < number2)
15   {
16     int temp = number1;
17     number1 = number2;
18     number2 = temp;
19   }
20
21   // 3. Prompt the student to answer "What is number1 - number2"
22   cout << "What is " << number1 << " - " << number2 << "? ";
23   int answer;
24   cin >> answer;
25
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
26   // 4. Repeatedly ask the user the question until it is correct
27   while (number1 - number2 != answer)
28   {
29     cout << "Wrong answer. Try again. What is "
30          << number1 << " - " << number2 << "? ";
31     cin >> answer;
32   }
33
34   cout << "You got it!" << endl;
35
36   return 0;
37 }
```

```
What is 4 - 3? 4
Wrong answer. Try again. What is 4 - 3? 5
Wrong answer. Try again. What is 4 - 3? 1
You got it!
```

RepeatSubtractionQuiz

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study: Guessing Numbers

- Write a program that randomly generates an integer between 0 and 100, inclusive.
- The program prompts the user to enter a number continuously until the number matches the randomly generated number.
- For each user input, the program tells the user whether the input is too low or too high, so the user can choose the next input intelligently.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Sample Run

```
Guess a magic number between 0 and 100
Enter your guess: 50
Your guess is too high
Enter your guess: 25
Your guess is too low
Enter your guess: 42
Your guess is too high
Enter your guess: 39
Yes, the number is 39
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Without loop

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <ctime> // Needed for the time function
4  using namespace std;
5
6  int main()
7  {
8    // Generate a random number to be guessed
9    srand(time(0));
10   int number = rand() % 101;
11
12   cout << "Guess a magic number between 0 and 100";
13
14   // Prompt the user to guess the number
15   cout << "\nEnter your guess: ";
16   int guess;
17   cin >> guess;
18
19   if (guess == number)
20      cout << "Yes, the number is " << number << endl;
21   else if (guess > number)
22      cout << "Your guess is too high" << endl;
23   else
24      cout << "Your guess is too low" << endl;
25
26   return 0;
27 }
```

GuessNumberOneTime

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <ctime> // Needed for the time function
4  using namespace std;
5
6  int main()
7  {
8    // Generate a random number to be guessed
9    srand(time(0));
10   int number = rand() % 101;
11
12   cout << "Guess a magic number between 0 and 100";
13
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
14   int guess = -1;
15   while (guess != number)
16   {
17     // Prompt the user to guess the number
18     cout << "\nEnter your guess: ";
19     cin >> guess;
20
21     if (guess == number)
22       cout << "Yes, the number is " << number << endl;
23     else if (guess > number)
24       cout << "Your guess is too high" << endl;
25     else
26       cout << "Your guess is too low" << endl;
27   } // End of loop
28
29   return 0;
30 }
```



GuessNumber

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Loop Design Strategies

- Step 1: Identify the statements that need to be repeated.
- Step 2: Wrap these statements in a loop as follows:

```
while (true) {
      Statements;
}
```

- Step 3: Code the loop-continuation-condition and add appropriate statements for controlling the loop.

```
while (loop-continuation-condition) {
  Statements;
  Additional statements for controlling the loop;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Problem: An Advanced Math Learning Tool

- The Math subtraction learning tool program generates just one question for each run.
- You can use a loop to generate questions repeatedly.
- This example gives a program that generates five questions and reports the number of the correct answers after a student answers all five questions.
- Note: calculate the time of the test



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
What is 9 - 2? 7
You are correct!
What is 3 - 0? 3
You are correct!
What is 3 - 2? 1
You are correct!
What is 7 - 4? 4
Your answer is wrong.
7 - 4 should be 3
What is 7 - 5? 4
Your answer is wrong.
7 - 5 should be 2
Correct count is 3
Test time is 201 seconds
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  #include <ctime> // Needed for time function
3  #include <cstdlib> // Needed for the srand and rand functions
4  using namespace std;
5
6  int main()
7  {
8    int correctCount = 0; // Count the number of correct answers
9    int count = 0; // Count the number of questions
10   long startTime = time(0);
11   const int NUMBER_OF_QUESTIONS = 5;
12
13   srand(time(0)); // Set a random seed
14
15   while (count < NUMBER_OF_QUESTIONS)
16   {
17     // 1. Generate two random single-digit integers
18     int number1 = rand() % 10;
19     int number2 = rand() % 10;
20
21     // 2. If number1 < number2, swap number1 with number2
22     if (number1 < number2)
23     {
24       int temp = number1;
25       number1 = number2;
26       number2 = temp;
27     }
```

```
29     // 3. Prompt the student to answer "what is number1 - number2?"
30     cout << "What is " << number1 << " - " << number2 << "? ";
31     int answer;
32     cin >> answer;
33
34     // 4. Grade the answer and display the result
35     if (number1 - number2 == answer)
36      {
37        cout << "You are correct!\n";
38        correctCount++; // Increase correct count
39      }
40     else
41      cout << "Your answer is wrong.\n" << number1 << " - " <<
42              number2 << " should be " << (number1 - number2) << endl;
43
44     // Increase the count
45     count++;
46   }
47
48   long endTime = time(0);
49   long testTime = endTime - startTime;
50
51   cout << "Correct count is " << correctCount << "\nTest time is "
52        << testTime << " seconds\n";
53
54   return 0;
55 }
```

SubtractionQuizLoop

Ending a Loop with a Sentinel Value

- Often the number of times a loop is executed is not predetermined.
- You may use an input value to signify the end of the loop. Such a value is known as a sentinel value.

```
char continueLoop = 'Y';
while (continueLoop == 'Y')
{
  // Execute body once
     ...
  // Prompt the user for confirmation
  cout << "Enter Y to continue and N to quit: ";
  cin >> continueLoop;
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Ending a Loop with a Sentinel Value

- Write a program that reads and calculates the sum of an unspecified number of integers.
- The input 0 signifies the end of the input.

```
Enter an integer (the input ends if it is 0): 2
Enter an integer (the input ends if it is 0): 3
Enter an integer (the input ends if it is 0): 4
Enter an integer (the input ends if it is 0): 0
The sum is 9
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    cout << "Enter an integer (the input ends " <<
7            "if it is 0): ";
8    int data;
9    cin >> data;
10
11   // Keep reading data until the input is 0
12   int sum = 0;
13   while (data != 0)
14   {
15     sum += data;
16
17     // Read the next data
18     cout << "Enter an integer (the input ends " <<
19             "if it is 0): ";
20     cin >> data;
21   }
22
23   cout << "The sum is " << sum << endl;
24
25   return 0;
26 }
```

SentinelValue

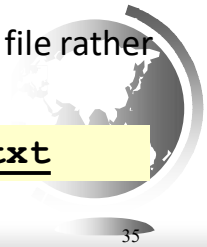
Input and Output Redirections

- if you have a large number of data to enter, it would be cumbersome to type from the keyboard.
- You may store the data separated by whitespaces in a text file, say **`input.txt`**, and run the program using the following command:

```
SentinelValue.exe < input.txt
```

- Output redirection can send the output to a file rather than displaying it on the console.

```
SentinelValue.exe > output.txt
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Reading Data from a File

- Listing 4.10 reads three numbers from the data file.
- If you have many numbers to read, you will need to write a loop to read all these numbers.
- If you don't know how many numbers are in the file and want to read them all, how do you know the end of file?

invoke the **`eof()`** function on the input object to detect it.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4
5  int main()
6  {
7    // Open a file
8    ifstream input("numbers.txt");
9
10   double sum = 0;
11   double number;
12   while (!input.eof()) // Read data to the end of file
13   {
14     input >> number; // Read data
15     cout << number << " "; // Display data
16     sum += number;
17   }
18
19   input.close();
20
21   cout << "\nTotal is " << sum << endl;
22
23   return 0;
24 }
```

```
95 56 34
Total score is 185
Done
```

ReadAllData

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

37

Caution

- Don't use floating-point values for equality checking in a loop control.
  - Recall that floating-point values are approximations
  - Using them could result in imprecise counter values and inaccurate results.

```
double item = 1; double sum = 0;
while (item != 0) // No guarantee item will be 0
{
  sum += item;
  item -= 0.1;
}
cout << sum << endl;
```

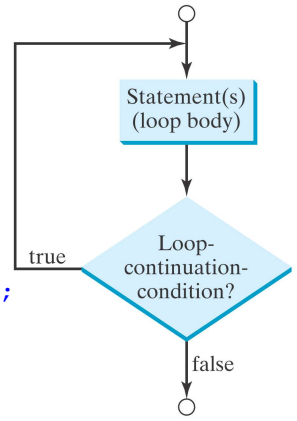
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

38

do-while Loop

Similar to while *loop except that it executes the loop body first and then checks the loop continuation condition.*

```
do
{
  // Loop body;
  Statement(s);
} while (loop-continuation-condition);
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

39

```
int count = 0;
while (count < 100)
{
  cout << "Welcome to C++!" << endl;
  count++;
}
```

```
int count = 0;
do
{
  cout << "Welcome to C++!" << endl;
  count++;
} while (count < 100);
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

40

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    // Initialize data and sum
7    int data = 0;
8    int sum = 0;
9
10   do
11   {
12     sum += data;
13
14     // Read the next data
15     cout << "Enter an integer (the input ends " <<
16             "if it is 0): ";
17     cin >> data; // Keep reading data until the input is 0
18   }
19   while (data != 0);
20
21   cout << "The sum is " << sum << endl;
22
23   return 0;
24 }
```

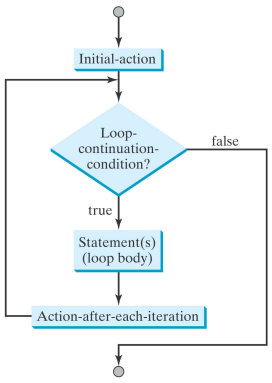
```
Enter an integer (the input ends if it is 0): 3
Enter an integer (the input ends if it is 0): 5
Enter an integer (the input ends if it is 0): 6
Enter an integer (the input ends if it is 0): 0
The sum is 14
```

TestDoWhile

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

for Loops

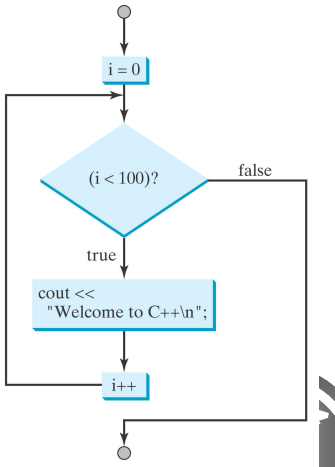
```
for (initial-action; loop-continuation-condition; action-after-each-iteration)
{
   // loop body;
   Statement(s);
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

for Loops

```
int i;
for (i = 0; i < 100; i++)
{
  cout << "Welcome to C++!\n";
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace for Loop

```
int i;
for (i = 0; i < 2; i++)
{
  cout <<  "Welcome to C++!";
}
```

Declare i

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace `for` Loop, cont.

```
int i;
for (i = 0; i < 2; i++)
{
  cout <<  "Welcome to C++!";
}
```

Execute initializer
i is now 0



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace `for` Loop, cont.

```
int i;
for (i = 0; i < 2; i++)
{
  cout <<  "Welcome to C++!";
}
```

(i < 2) is true
since i is 0



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace `for` Loop, cont.

Print Welcome to C++!

```
int i;
for (i = 0; i < 2; i++)
{
  cout <<  "Welcome to C++!";
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace `for` Loop, cont.

Execute adjustment statement
i now is 1

```
int i;
for (i = 0; i < 2; i++)
{
  cout <<  "Welcome to C++!";
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace for Loop, cont.

```
int i;
for (i = 0; i < 2; i++)
{
  cout <<  "Welcome to C++!";
}
```

(i < 2) is still true since i is 1



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace for Loop, cont.

```
int i;
for (i = 0; i < 2; i++)
{
  cout <<  "Welcome to C++!";
}
```

Print Welcome to C++



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace `for` Loop, cont.

```
int i;
for (i = 0; i < 2; i++)
{
  cout <<  "Welcome to C++!";
}
```

Execute adjustment statement
i now is 2



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

51

animation

Trace `for` Loop, cont.

```
int i;
for (i = 0; i < 2; i++)
{
  cout <<  "Welcome to C++!";
}
```

(i < 2) is false
since i is 2



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

52

animation

Trace `for` Loop, cont.

```
int i;
for (i = 0; i < 2; i++)
{
  cout <<  "Welcome to C++!";
}
```

Exit the loop. Execute the next statement after the loop



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

53

Note

- The initial-action in a for loop can be a list of zero or more comma-separated expressions.
- The action-after-each-iteration in a for loop can be a list of zero or more comma-separated statements.
- Therefore, the following for loops are correct. They are rarely used in practice, however.

```
for (int i = 1; i < 100; cout << i, i++);

for (int i = 1; i < 100; cout << (i++));

for (int i = 0, j = 0; (i + j < 10); i++, j++) {
  // Do something
}
```



Note

- If the loop-continuation-condition in a for loop is omitted, it is implicitly true.
- Thus the statement given below in (a), which is an infinite loop, is correct.
- Nevertheless, it is better to use the equivalent loop in (b) to avoid confusion:

```
for ( ; ; )
{
  // Do something
}
```

(a)

Equivalent

This is better

```
while (true)
{
  // Do something
}
```

(b)

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Which Loop to Use?

- The three forms of loop statements, while, do-while, and for, are expressively equivalent
- you can write a loop in any of these three forms.
- For example, a while loop in (a) in can always be converted into the following for loop in (b):

```
while (loop-continuation-condition)
{
  // Loop body
}
```

(a)

Equivalent

```
for ( ; loop-continuation-condition; )
{
  // Loop body
}
```

(b)

```
for (initial-action;
     loop-continuation-condition;
     action-after-each-iteration)
{
  // Loop body;
}
```

(a)

Equivalent

```
initial-action;
while (loop-continuation-condition)
{
  // Loop body;
  action-after-each-iteration;
}
```

(b)

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Recommendations

- Use the one that is most intuitive and comfortable for you.
- In general, a for loop may be used if the number of repetitions is known, as, for example, when you need to print a message 100 times.
- A while loop may be used if the number of repetitions is not known, as in the case of reading the numbers until the input is 0.
- A do-while loop can be used to replace a while loop if the loop body has to be executed before testing the continuation condition.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Caution

Adding a semicolon at the end of the for clause before the loop body is a common mistake, as shown below:

Logic Error

```
for (int i=0; i<10; i++);
{
  cout << "i is " << i << endl;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Caution, cont.

- Similarly, the following loop is also wrong:

```
int i=0;
while (i < 10);
{
  cout << "i is " << i << endl;
  i++;
}
```

Logic Error

- In the case of the do loop, the following semicolon is needed to end the loop.

```
int i=0;
do {
  cout << "i is " << i << endl;
  i++;
} while (i<10);
```

Correct



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

59

Nested Loops

Problem: Write a program that uses nested for loops to print a multiplication table.

```
          Multiplication Table
      1   2   3   4   5   6   7   8   9
-----------------------------------------
1 |   1   2   3   4   5   6   7   8   9
2 |   2   4   6   8  10  12  14  16  18
3 |   3   6   9  12  15  18  21  24  27
4 |   4   8  12  16  20  24  28  32  36
5 |   5  10  15  20  25  30  35  40  45
6 |   6  12  18  24  30  36  42  48  54
7 |   7  14  21  28  35  42  49  56  63
8 |   8  16  24  32  40  48  56  64  72
9 |   9  18  27  36  45  54  63  72  81
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

60

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  int main()
6  {
7    cout << " Multiplication Table\n";
8
9    // Display the number title
10   cout << "    ";
11   for (int j = 1; j <= 9; j++)
12     cout << setw(3) << j;
13   cout << "\n";
14
15   cout << "--------------------------------\n";
16
17   // Display table body
18   for (int i = 1; i <= 9; i++)
19   {
20     cout << i << " | ";
21     for (int j = 1; j <= 9; j++)
22     {
23       // Display the product and align properly
24       cout << setw(3) << i * j;
25     }
26     cout << "\n";
27   }
28
29   return 0;
30 }
```

MultiplicationTable

61

Minimizing Numeric Errors

Using floating-point numbers in the loop continuation condition may cause numeric errors.

Problem: Write a program that sums a series that starts with 0.01 and ends with 1.0. The numbers in the series will increment by 0.01, as follows: 0.01 + 0.02 + 0.03 and so on.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

62

```
1 #include <iomanip>
2 #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7    // Initialize sum
8    float sum = 0;
9
10    // Add 0.01, 0.02, ..., 0.99, 1 to sum
11   for (float i = 0.01f; i <= 1.0f; i = i + 0.01f)
12      sum += i;
13
14   // Display result
15   cout << "The sum is " << sum <<fixed<< setprecision(6) << sum << endl;
15
17   return 0;
18 }
```

```
The sum is 50.499985
```

But the correct result should be 50.5

How to fix it

TestSum

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

63

Fix the Problem

- What will happen if you change float in the program to double

```
1 #include <iomanip>
2 #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7    // Initialize sum
8    double sum = 0;
9
10    // Add 0.01, 0.02, ..., 0.99, 1 to sum
11   for (double i = 0.01; i <= 1.0; i = i + 0.01)
12      sum += i;
13
14   // Display result
15   cout << "The sum is " << sum <<fixed<< setprecision(16) << sum << endl;
16   return 0;
17 }
```

```
The sum is  49.5000000000000284
```

- i is incremented by 0.01 in each loop. (recall that floating-point numbers are approximate)
  - The i value is slightly larger than 1 (not exactly 1).
- This causes the last i value **not to be added** into sum.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

64

Fix the Problem

```
double currentValue = 0.01;
for (int count = 0; count < 100; count++)
{
   sum += currentValue;
   currentValue += 0.01;
}
cout << "The sum is " << sum << fixed << setprecision(16) << sum << endl;
```

```
The sum is  50.5000000000000284
```

```
double currentValue = 1.0;
for (int count = 0; count < 100; count++)
{
   sum += currentValue;
   currentValue -= 0.01;
}
cout << "The sum is " << sum <<fixed << setprecision(16) << sum << endl;
```

```
The sum is  50.4999999999999503
```

- This phenomenon is an artifact of the finite-precision arithmetic.
- To obtain more accurate results, carefully select the order of computation.
- **Adding smaller numbers before bigger numbers to sum is one way to minimize errors.**

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Loops are fundamental in programming. The ability to write loops is essential in learning programming.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study:
Finding the Greatest Common Divisor

Problem: Write a program that prompts the user to enter two positive integers and finds their greatest common divisor.

```
4   and 2      gcd is 2  ;
16 and 24     gcd is 8;
```

So, how do you find the greatest common divisor?



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

67

Case Study:
Finding the Greatest Common Divisor

Solution:

- Let the two input integers be n1 and n2.
- You know number 1 is a common divisor, but it may not be the greatest commons divisor.
- So you can check whether k (for k = 2, 3, 4, and so on) is a common divisor for n1 and n2, until k is greater than n1 or n2.

```
Enter first integer: 125
Enter second integer: 2525
The greatest common divisor for 125 and 2525 is 25
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

68

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    // Prompt the user to enter two integers
7    cout << "Enter first integer: ";
8    int n1;
9    cin >> n1;
10
11   cout << "Enter second integer: ";
12   int n2;
13   cin >> n2;
14
15   int gcd = 1;
16   int k = 2;
17   while (k <= n1 && k <= n2)
18   {
19     if (n1 % k == 0 && n2 % k == 0)
20       gcd = k;
21     k++;
22   }
23
24   cout << "The greatest common divisor for " << n1 << " and "
25        << n2 << " is " << gcd << endl;
26
27   return 0;
28 }
```

GreatestCommonDivisor

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Thinking Before Typing

- How would you write this program?
  - Beginning to write the code immediately is not the right way.
- *Thinking before typing is an important stage*.
  - Thinking enables you to generate a logical solution for the problem
  - Once you have a logical solution, translate the solution into a program.
- **The translation is not unique.**

```
int gcd = 1;
for (int k = 2; k <= m && k <= n; k++)
{
  if (m % k == 0 && n % k == 0)
      gcd = k;
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study: Predicting the Future Tuition

Problem: Suppose that the tuition for a university is $10,000 this year and tuition increases 7% every year. In how many years will the tuition be doubled?

```
Tuition will be doubled in 11 years
Tuition will be $21048.52 in 11 years
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

71

Solution

Consider how to solve it by hand.

```
double tuition = 10000; int year = 0;    // Year 0
tuition = tuition * 1.07; year++;        // Year 1
tuition = tuition * 1.07; year++;        // Year 2
tuition = tuition * 1.07; year++;        // Year 3
...
```

Keep computing the tuition for a new year until it is at least 20000

```
double tuition = 10000;        // Year 0
int year = 0;
while (tuition < 20000)
{
   tuition = tuition * 1.07;
   year++;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

72

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  int main()
6  {
7    double tuition = 10000 ; // Year 1
8    int year = 0;
9    while (tuition < 20000)
10   {
11     tuition = tuition * 1.07;
12     year++;;
13   }
14
15   cout << "Tuition will be doubled in " << year << " years" << endl;
16   cout << setprecision(2) << fixed << showpoint <<
17           "Tuition will be $" << tuition << " in "
18        << year << " years" << endl;
19
20   return 0;
21 }
```

FutureTuition

```
Tuition will be doubled in 11 years
Tuition will be $21048.52 in 11 years
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study: *Converting Decimals to Hexadecimals*

- Hexadecimals are often used in computer systems programming
  - Appendix F for an introduction to number systems
- How do you convert a decimal number to a hexadecimal number?
- To convert a decimal number *d* to a hexadecimal number is to find the hexadecimal digits h_n, h_{n-1}, h_{n-2}, ... , h_2, h_1, and h_0 such that

$$d = h_n \times 16^n + h_{n-1} \times 16^{n-1} + h_{n-2} \times 16^{n-2} + ... + h_2 \times 16^2 + h_1 \times 16^1 + h_0 \times 16^0$$

- These hexadecimal digits can be found by successively dividing *d* by 16 until the quotient is 0.
- The remainders are h_0, h_1, h_2, ... , h_{n-2}, h_{n-1}, and h_n.

```
Enter a decimal number: 1234
The hex number is 4D2
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7    // Prompt the user to enter a decimal integer
8    cout << "Enter a decimal number: ";
9    int decimal;
10   cin >> decimal;
11
12   // Convert decimal to hex
13   string hex = "";
14
15   while (decimal != 0)
16   {
17     int hexValue = decimal % 16;
18
19     // Convert a decimal value to a hex digit
20     char hexDigit = (hexValue <= 9 && hexValue >= 0) ?
21     static_cast<char>(hexValue + '0') :
22     static_cast<char>(hexValue - 10 + 'A');
23
24     hex = hexDigit + hex;
25   decimal = decimal / 16;
26   }
27
28   cout << "The hex number is " << hex << endl;
29
30   return 0;
31 }
```

Dec2Hex

break

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    int sum = 0;
7    int number = 0;
8
9    while (number < 20)
10   {
11     number++;
12     sum += number;
13     if (sum >= 100)
14       break;
15   }
16
17   cout << "The number is " << number << endl;
18   cout << "The sum is " << sum << endl;
19
20   return 0;
21 }
```

The `break` keyword provides additional controls in a loop

```
The number is 14
The sum is 105
```

TestBreak

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

76

continue

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    int sum = 0;
7    int number = 0;
8
9    while (number < 20)
10   {
11     number++;
12     if (number == 10 || number == 11)
13       continue;
14     sum += number;
15   }
16
17   cout << "The sum is " << sum << endl;
18
19   return 0;
20 }
```

The `continue` *keyword provide additional controls in a loop*

```
The sum is 189
```

```
Without line 12 and 13
The sum is 210
```

TestContinue

Education, Inc. All Rights Reserved.

77

```
int factor = 2;
while (factor <= n)
{
   if (n % factor == 0)
     break;
   factor++;
}
cout << "The smallest factor other than 1 for "
     << n << " is " << factor << endl;
```

find the smallest factor other than 1 for an integer n (assume n >= 2).

```
bool found = false;
int factor = 2;
while (factor <= n && !found)
{
  if (n % factor == 0)
    found = true;
  else
    factor++;
}
cout << "The smallest factor other than 1 for "
     << n << " is " << factor << endl;
```



© Copyright 2010 by Pearson Education, Inc. All Rights Reserved.

78

Case Study: Checking Palindromes

- Problem: Write a program that tests whether a string is a palindrome.
- A string is a palindrome if it reads the same forward and backward.
  - The words "**mom**," "**dad**," and *"**noon**,"* are all palindromes.
- How do you write a program to check whether a string is a palindrome?
- **One solution is**
  - Check whether the first character in the string is the same as the last character.
  - If so, check whether the second character is the same as the second-last character.
  - Continues until a mismatch is found or all the characters in the string are checked, except for the middle character if the string has an odd number of characters.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.


```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7    // Prompt user input
8    cout << "Enter a string: ";
9    string s;
10   getline(cin, s);
11
12   // low index
13   int low = 0;
14
15   // high index
16   int high = s.length() - 1;
17
18   bool isPalindrome = true;
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
19   while (low < high)
20   {
21     if (s[low] != s[high])
22     {
23       isPalindrome = false; // Not a palindrome
24       break;
25     }
26
27     low++;
28     high--;
29   }
30
31   if (isPalindrome)
32      cout << s << " is a palindrome" << endl;
33   else
34      cout << s << " is not a palindrome" << endl;
35
36   return 0;
37 }
```

```
Enter a string: abccba
abccba is a palindrome
```

```
Enter a string: abca
abca is not a palindrome
```

TestPalindrome

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study: Displaying Prime Numbers

- Problem: Write a program that displays the first 50 prime numbers in five lines, each of which contains 10 numbers. An integer greater than 1 is *prime* if its only positive divisor is 1 or itself.
- For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not.
- Solution: The problem can be broken into the following tasks:
  - For number = 2, 3, 4, 5, 6, ..., test whether the number is prime.
  - Determine whether a given number is prime.
  - Count the prime numbers.
  - Print each prime number, and print 10 numbers per line.

PrimeNumber

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  int main()
6  {
7    const int NUMBER_OF_PRIMES = 50; // Number of primes to display
8    const int NUMBER_OF_PRIMES_PER_LINE = 10; // Display 10 per line
9    int count = 0; // Count the number of prime numbers
10   int number = 2; // A number to be tested for primeness
11
12   cout << "The first " << NUMBER_OF_PRIMES << " prime numbers are \n";
13
14   // Repeatedly find prime numbers
15   while (count < NUMBER_OF_PRIMES)
16   {
17     // Assume the number is prime
18     bool isPrime = true; // Is the current number prime?
19
20     // Test if number is prime
21     for (int divisor = 2; divisor <= number / 2; divisor++)
22     {
23       if (number % divisor == 0)
24       {
25         // If true, the number is not prime
26         isPrime = false; // Set isPrime to false
27         break; // Exit the for loop
28       }
29     }
```

```
31     // Display the prime number and increase the count
32     if (isPrime)
33     {
34       count++; // Increase the count
35
36       if (count % NUMBER_OF_PRIMES_PER_LINE == 0)
37          // Display the number and advance to the new line
38          cout << setw(4) << number << endl;
39       else
40          cout << setw(4) << number;
41     }
42
43     // Check if the next number is prime
44     number++;
45   }
46
47   return 0;
48 }
```

```
The first 50 prime numbers are
2  3  5  7  11  13  17  19  23  29
31  37  41  43  47  53  59  61  67  71
73  79  83  89  97  101  103  107  109  113
127  131  137  139  149  151  157  163  167  173
179  181  191  193  197  199  211  223  227  229
```

PrimeNumber

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

References

- Introduction to Programming with C++, Daniel Liang, Pearson Education Limited 2014
- C++ Programming- From Problem Analysis to Program Design - D.S. MALIK - 2018



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Computer Engineering Department
Islamic University of Gaza

Dr. Ayman Maliha
amaliha@iugaza.edu.ps
Homepage: http://site.iugaza.edu.ps/amaliha/

ECOM 2403
Computer Programming
Fall 2020-2021



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.



Chapter 6

Functions

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Opening Problem

Find the sum of integers from 1 to 10, from 20 to 37, and from 35 to 49, respectively.

```
int sum = 0;
for (int i = 1; i <= 10; i++)
  sum += i;
cout << "Sum from 1 to 10 is " << sum << endl;

sum = 0;
for (int i = 20; i <= 37; i++)
  sum += i;
cout << "Sum from 20 to 37 is " << sum << endl;

sum = 0;
for (int i = 35; i <= 49; i++)
  sum += i;
cout << "Sum from 35 to 49 is " << sum << endl;
```

© Copyright 2010 by Pearson Education, Inc. All Rights Reserved.

Problem

```
int sum = 0;
for (int i = 1; i <= 10; i++)
  sum += i;
cout << "Sum from 1 to 10 is " << sum << endl;
```

```
sum = 0;
for (int i = 20; i <= 37; i++)
  sum += i;
cout << "Sum from 20 to 37 is " << sum << endl;
```

```
sum = 0;
for (int i = 35; i <= 49; i++)
  sum += i;
cout << "Sum from 35 to 49 is " << sum << endl;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Solution

```
int sum(int i1, int i2)
{
  int sum = 0;
  for (int i = i1; i <= i2; i++)
    sum += i;
  return sum;
}
```

```
int main()
{
  cout << "Sum from 1 to 10 is " << sum(1, 10) << endl;
  cout << "Sum from 20 to 37 is " << sum(20, 37) << endl;
  cout << "Sum from 35 to 49 is " << sum(35, 49) << endl;
  return 0;
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Predefined Functions

- In C++, a function is similar to that of a function in algebra
  - It has a name
  - It does some computation
- Some of the predefined mathematical functions are:

```
sqrt(x)
pow(x, y)
floor(x)
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Benefits of Functions

- Write a function once and reuse it anywhere
- Promotes Information hiding (hide the implementation from the user)
- Facilitate modularity (break the code into manageable modules)
- Reduce code complexity (better maintenance)



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Defining a Function

A function is a collection of statements that are grouped together to perform an operation.

Define a function

```
int max(int num1, int num2)
{

   int result;

   if (num1 > num2)
     result = num1;
   else
     result = num2;

   return result;
}
```

Invoke a funciton

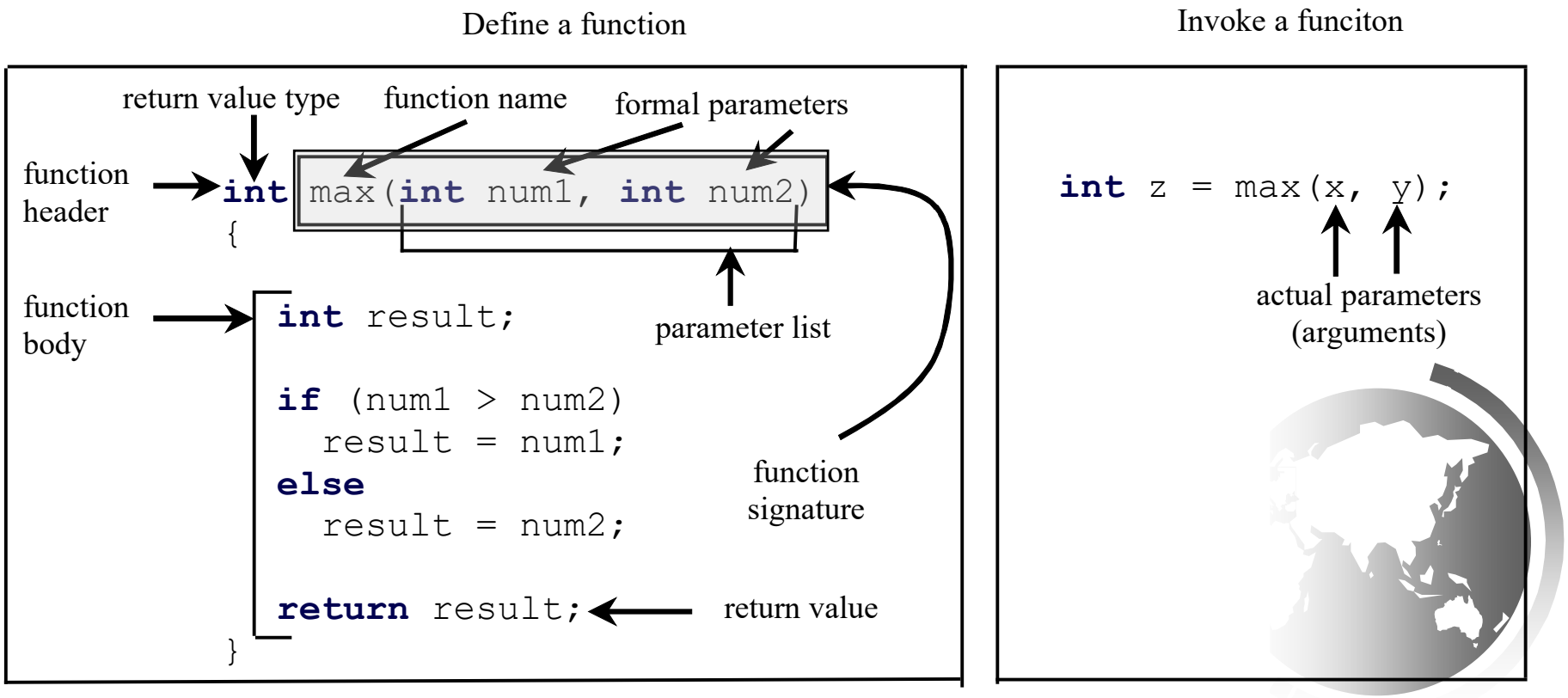
```
int z = max(x, y);
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Function Signature

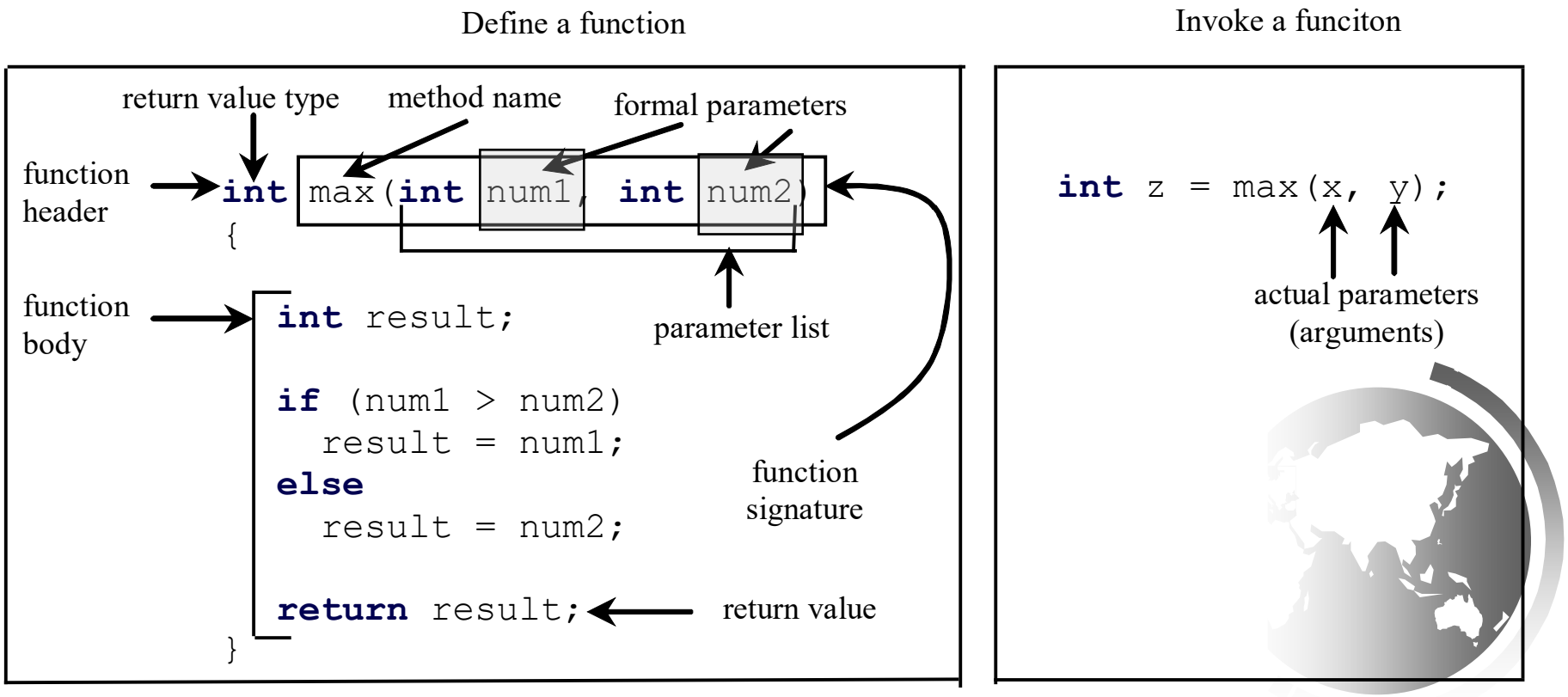
Function signature is the combination of the function name and the parameter list.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Formal Parameters

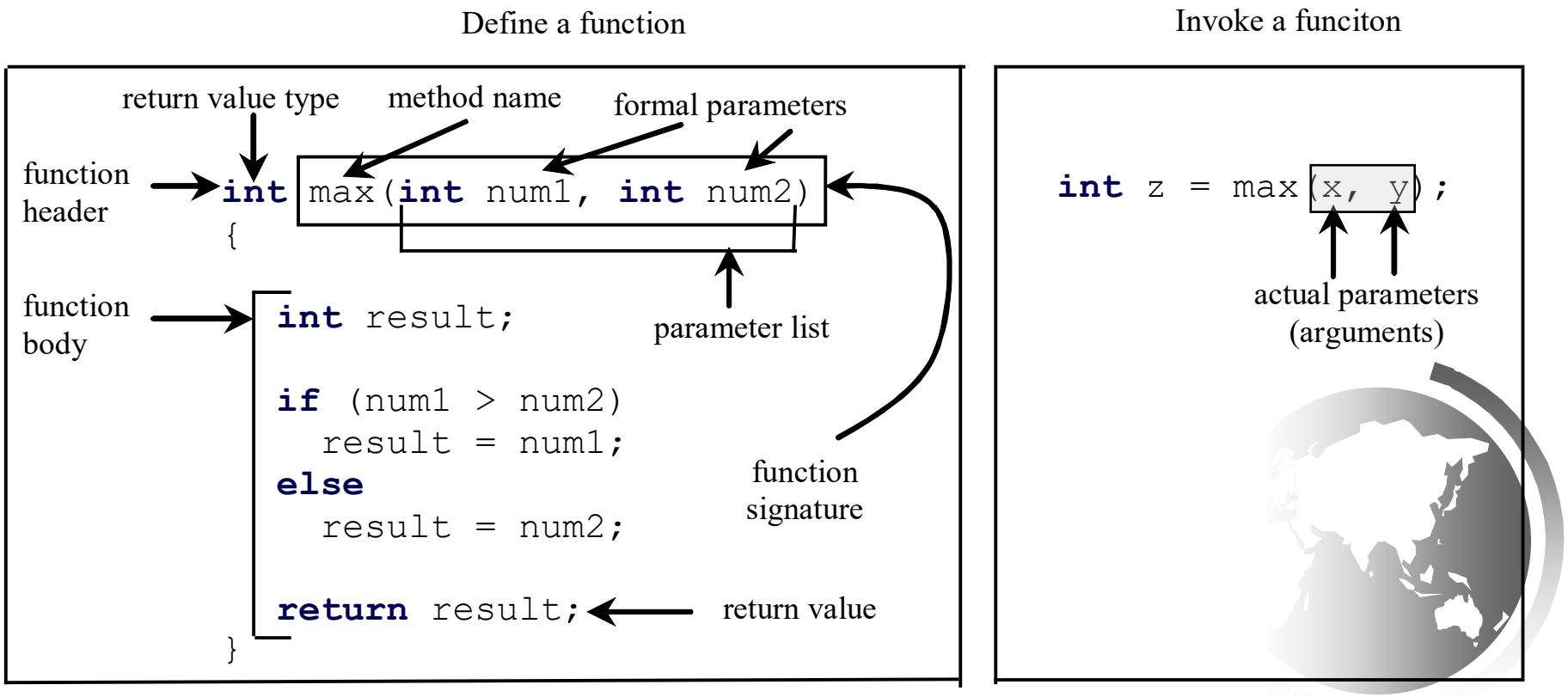
The variables defined in the function header are known as *formal parameters*.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Actual Parameters

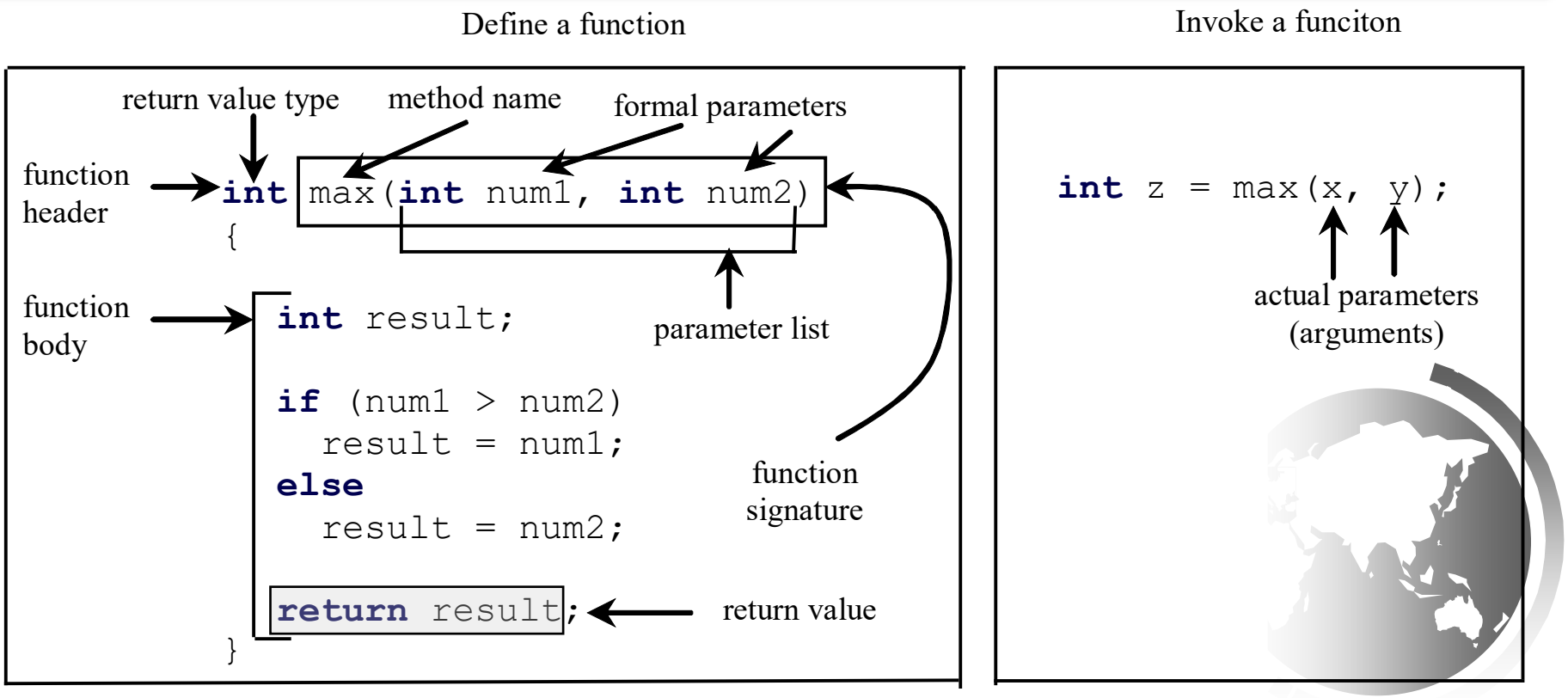
When a function is invoked, you pass a value to the parameter. This value is referred to as *actual parameter or argument*.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Return Value Type

- A function may return a value.
  - The returnValueType is the data type of the value the function returns.
  - If the function does not return a value, use the keyword `void`.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Calling Functions

```
1  #include <iostream>
2  using namespace std;
3
4  // Return the max between two numbers
5  int max(int num1, int num2)
6  {
7   int result;
8
9   if (num1 > num2)
10   result = num1;
11   else
12   result = num2;
13
14   return result;
15 }
16
17 int main()
18 {
19   int i = 5;
20   int j = 2;
21   int k = max(i, j);
22   cout << "The maximum between " << i <<
23            " and " << j << " is " << k << endl;
24
25   return 0;
26 }
```

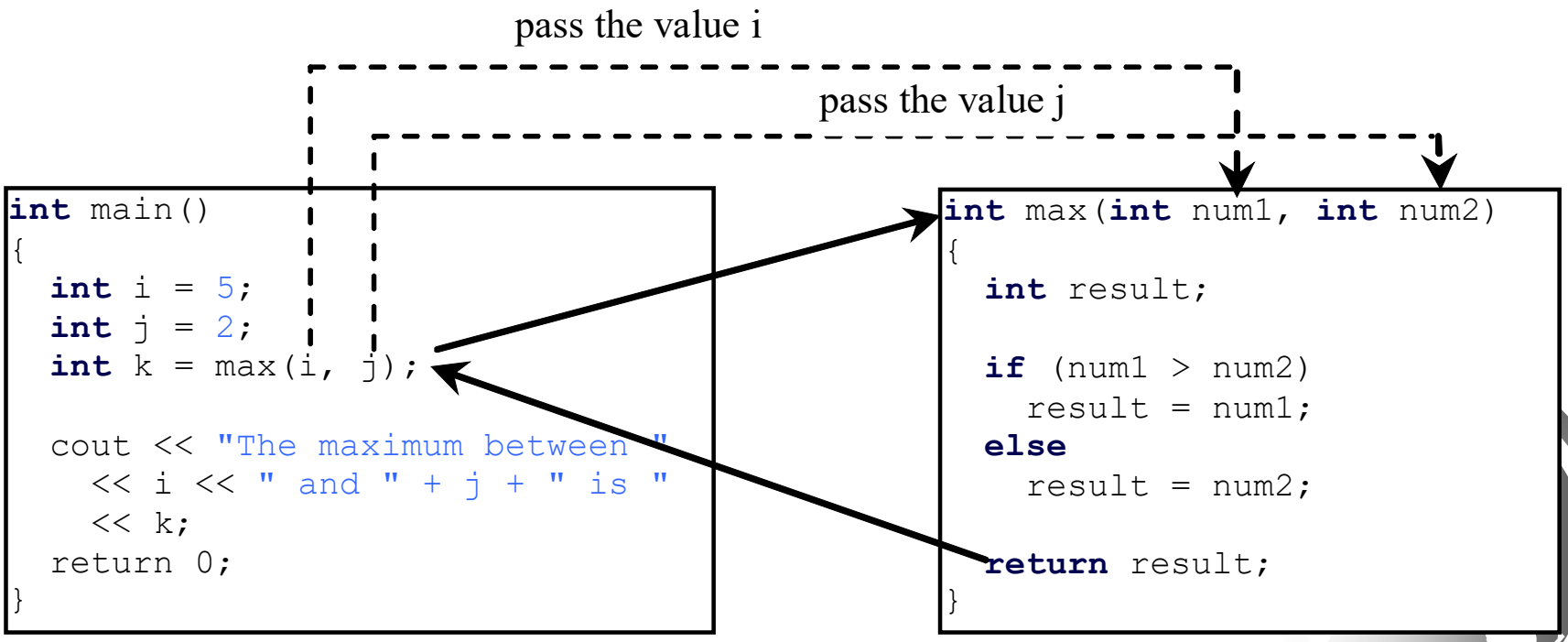
This program demonstrates calling a function max to return the largest of the int values

TestMax

© Copyright 2010 by Pearson Education, Inc. All Rights Reserved.

animation

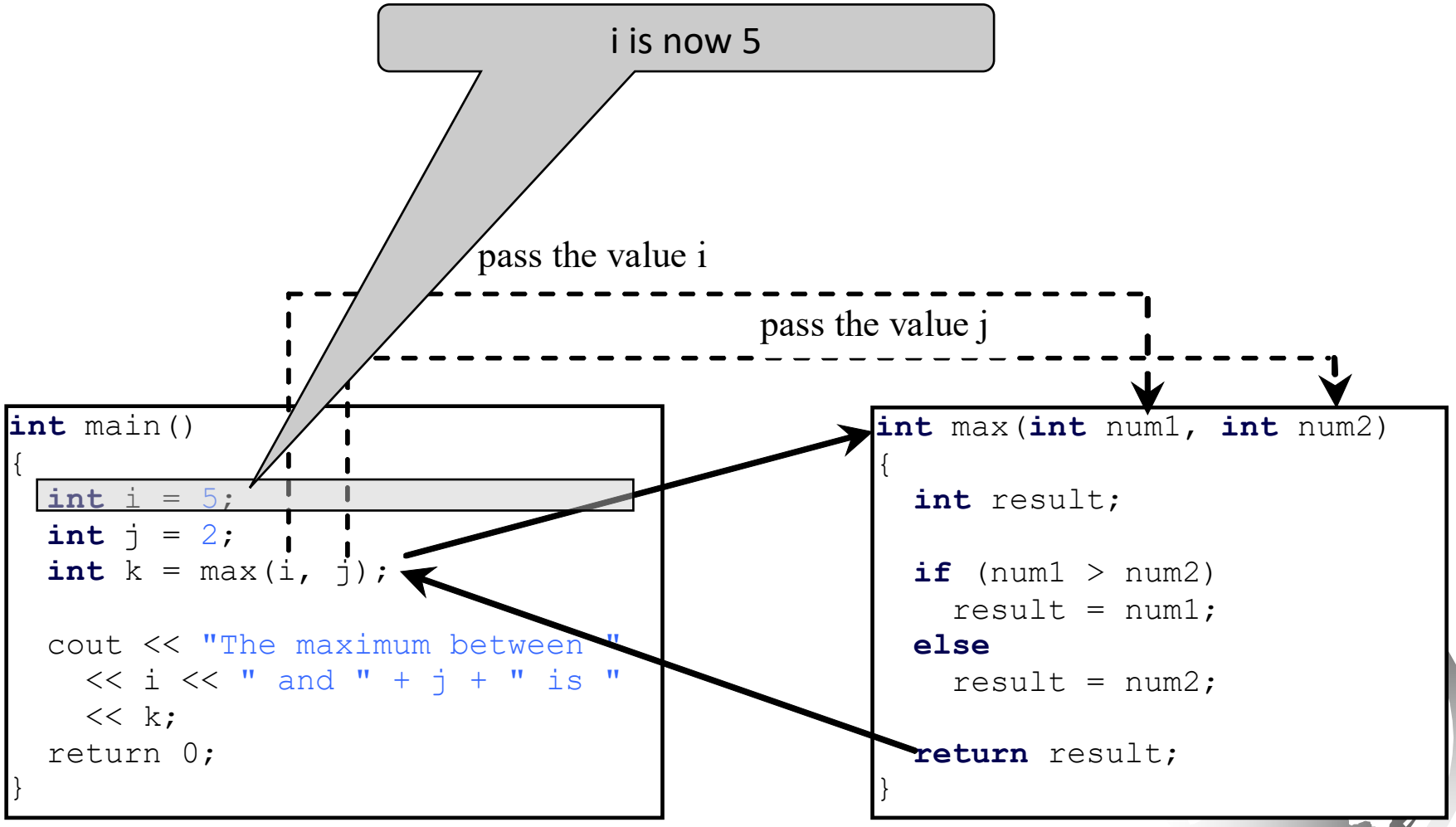
Calling Functions, cont.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

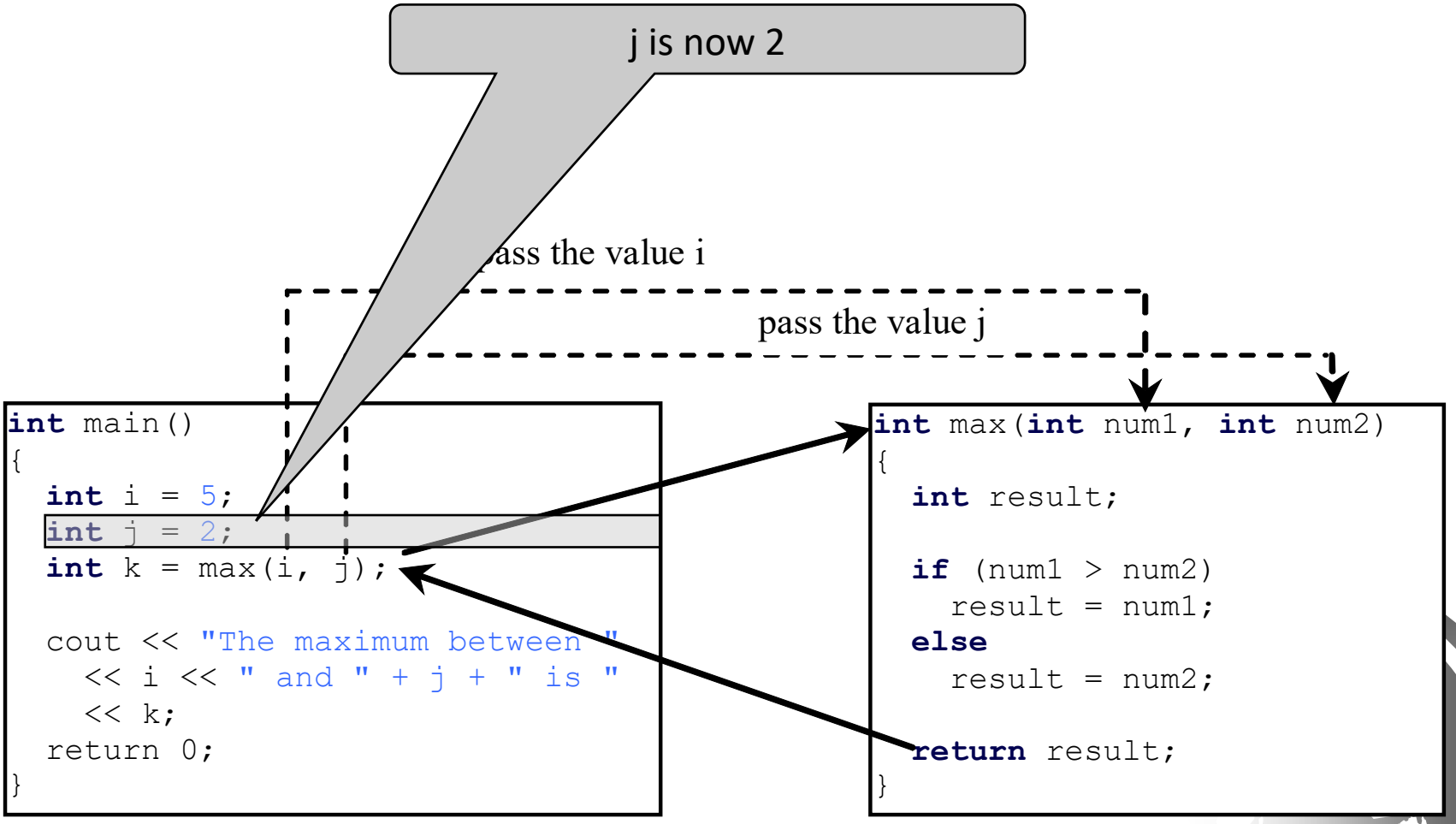
Trace Function Invocation



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

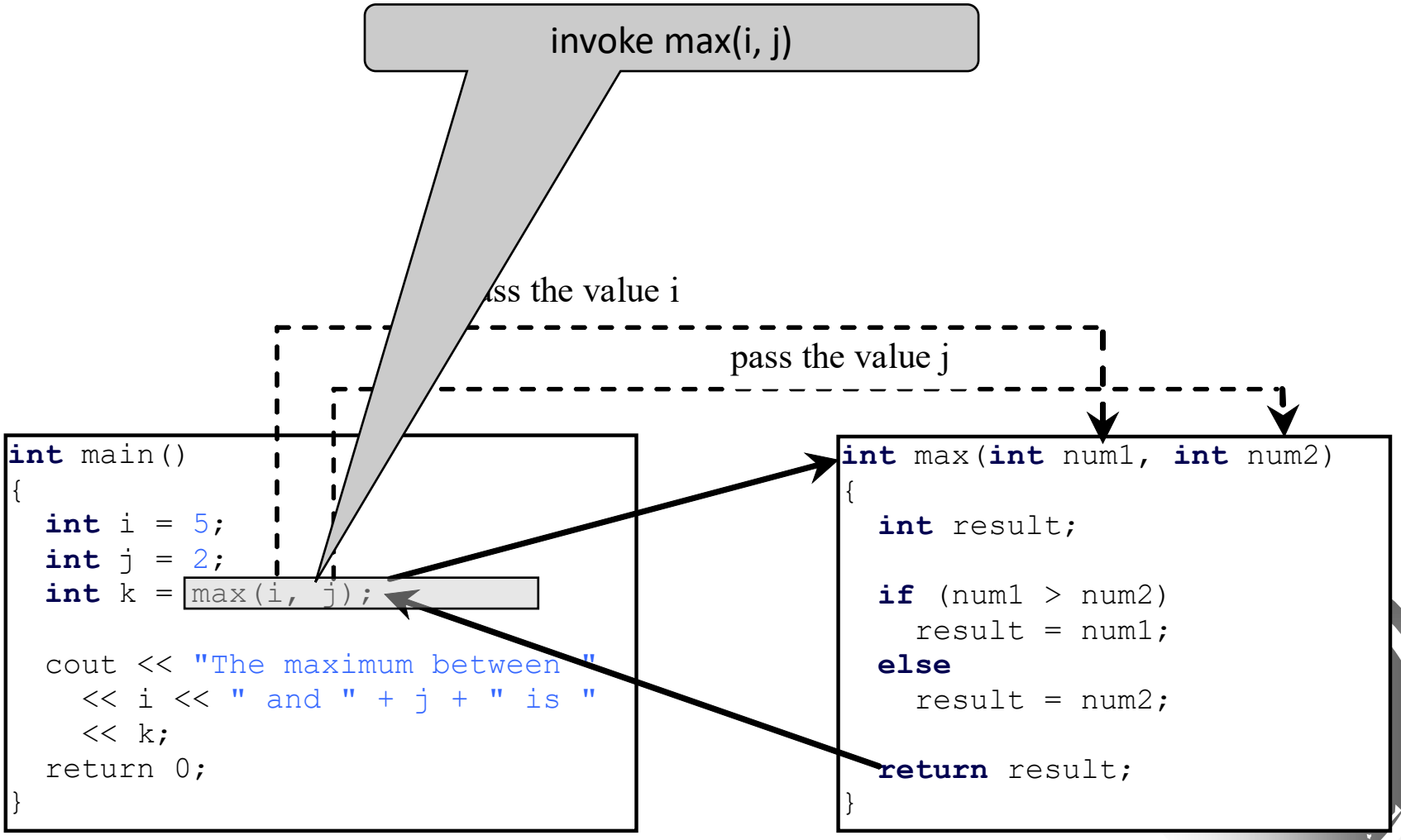
Trace Function Invocation



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

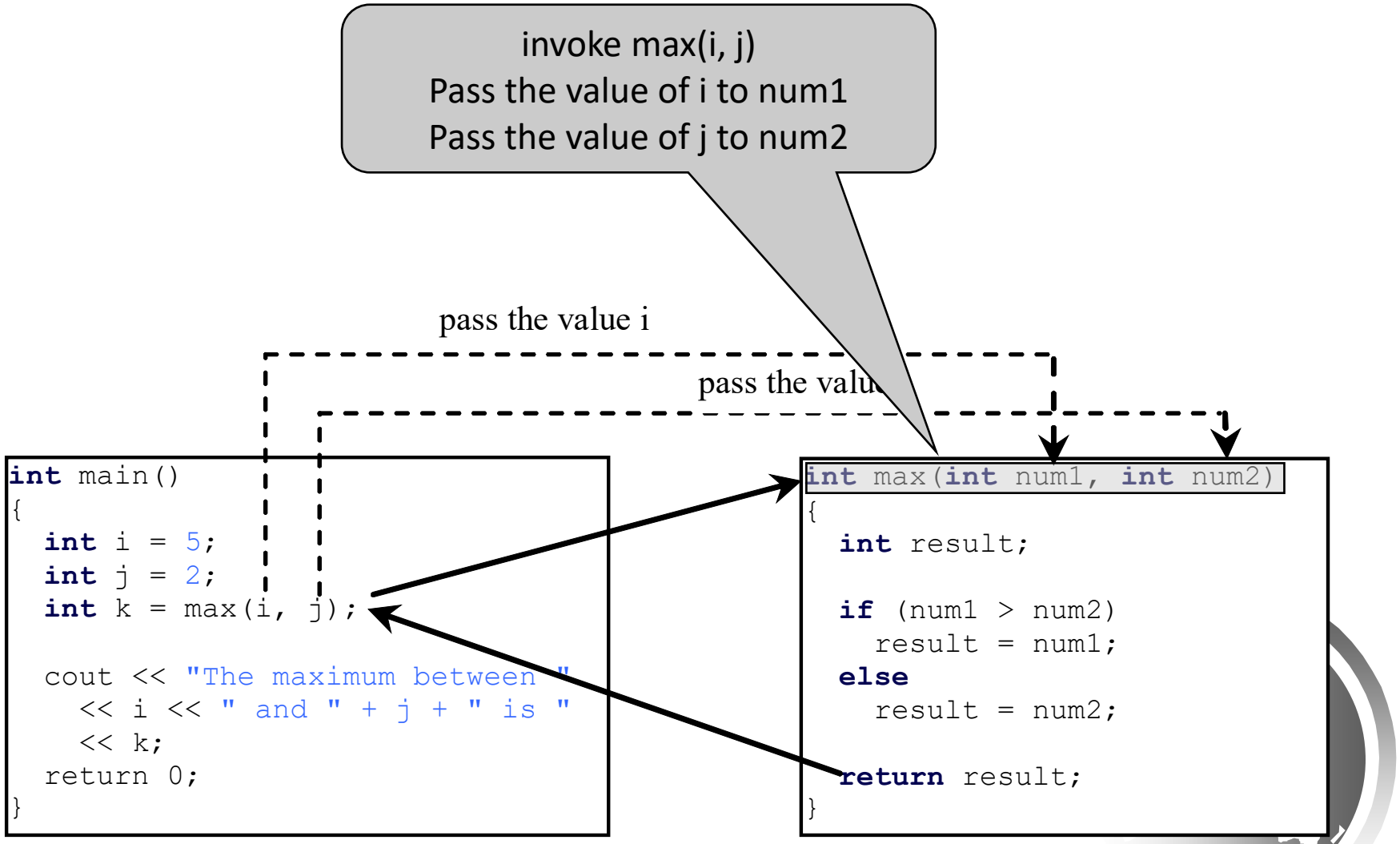
Trace Function Invocation



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

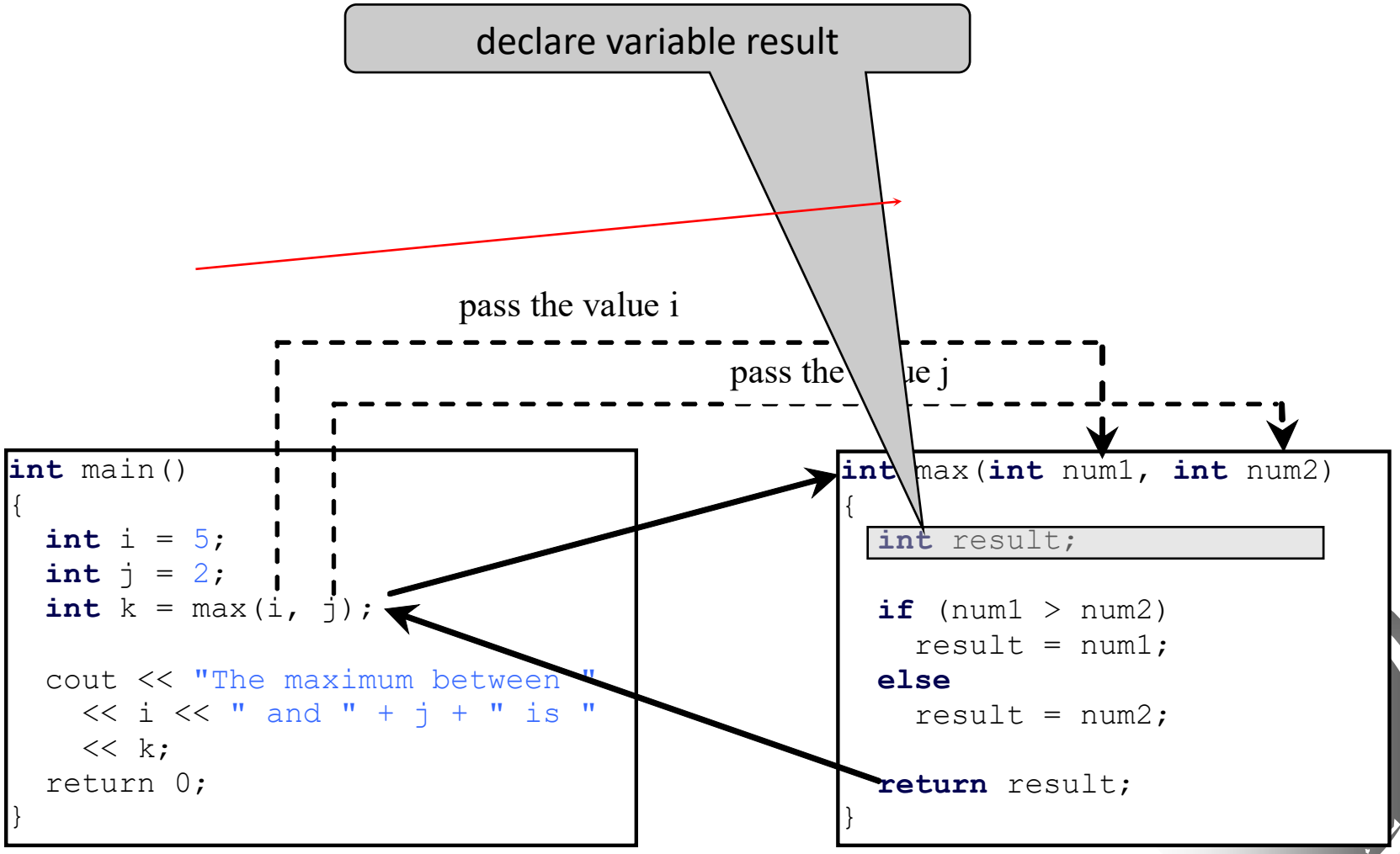
Trace Function Invocation



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

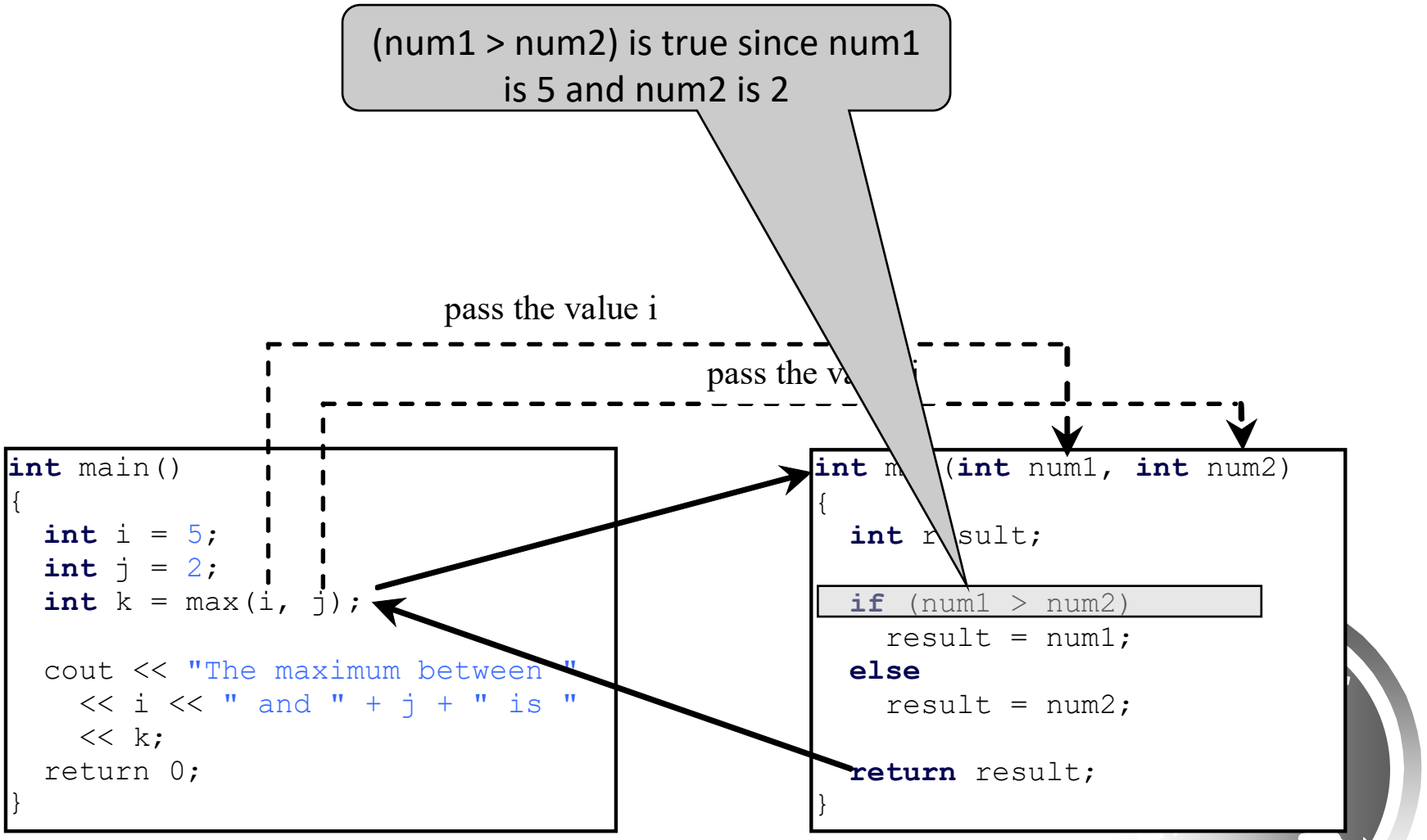
Trace Function Invocation



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

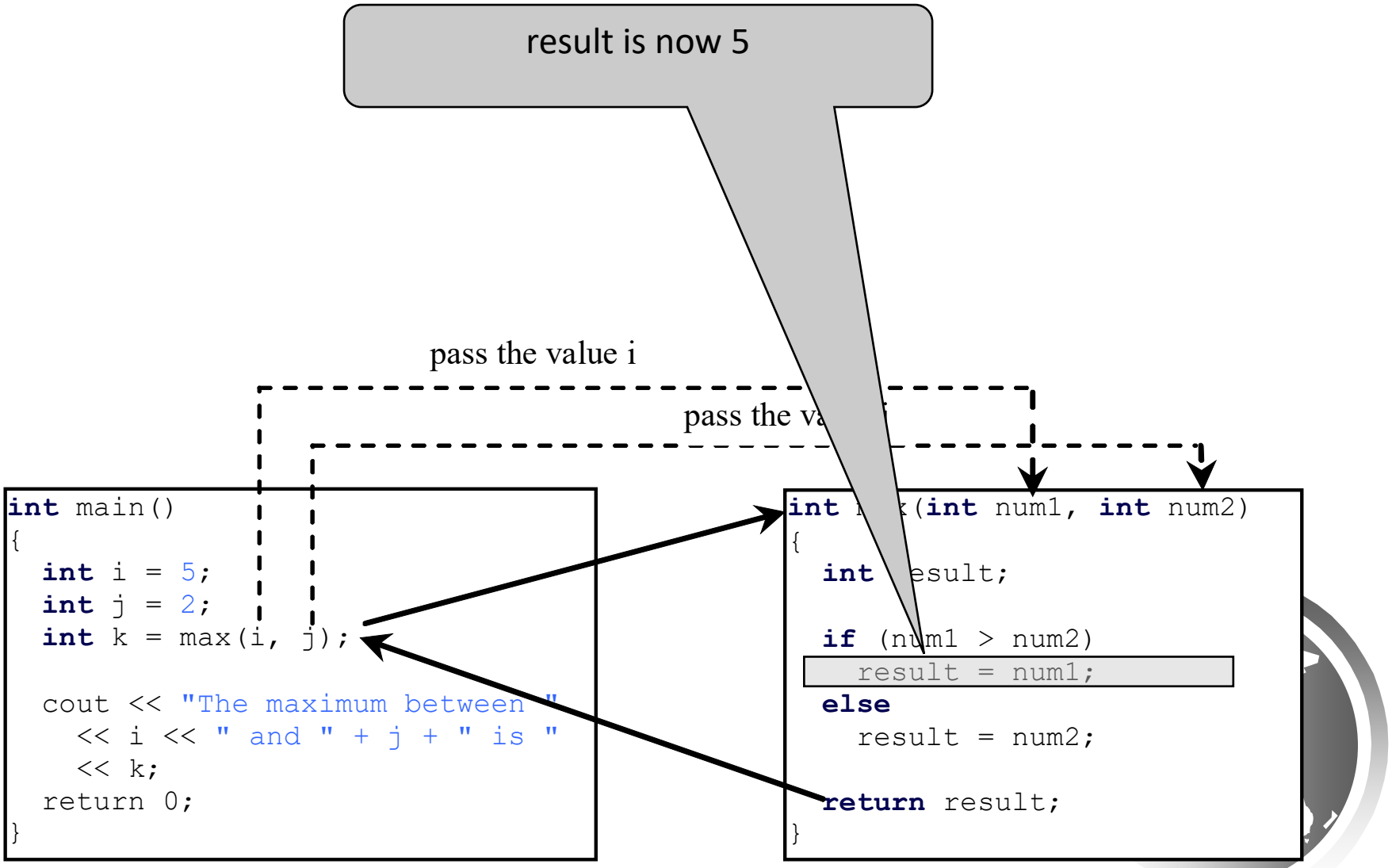
Trace Function Invocation



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

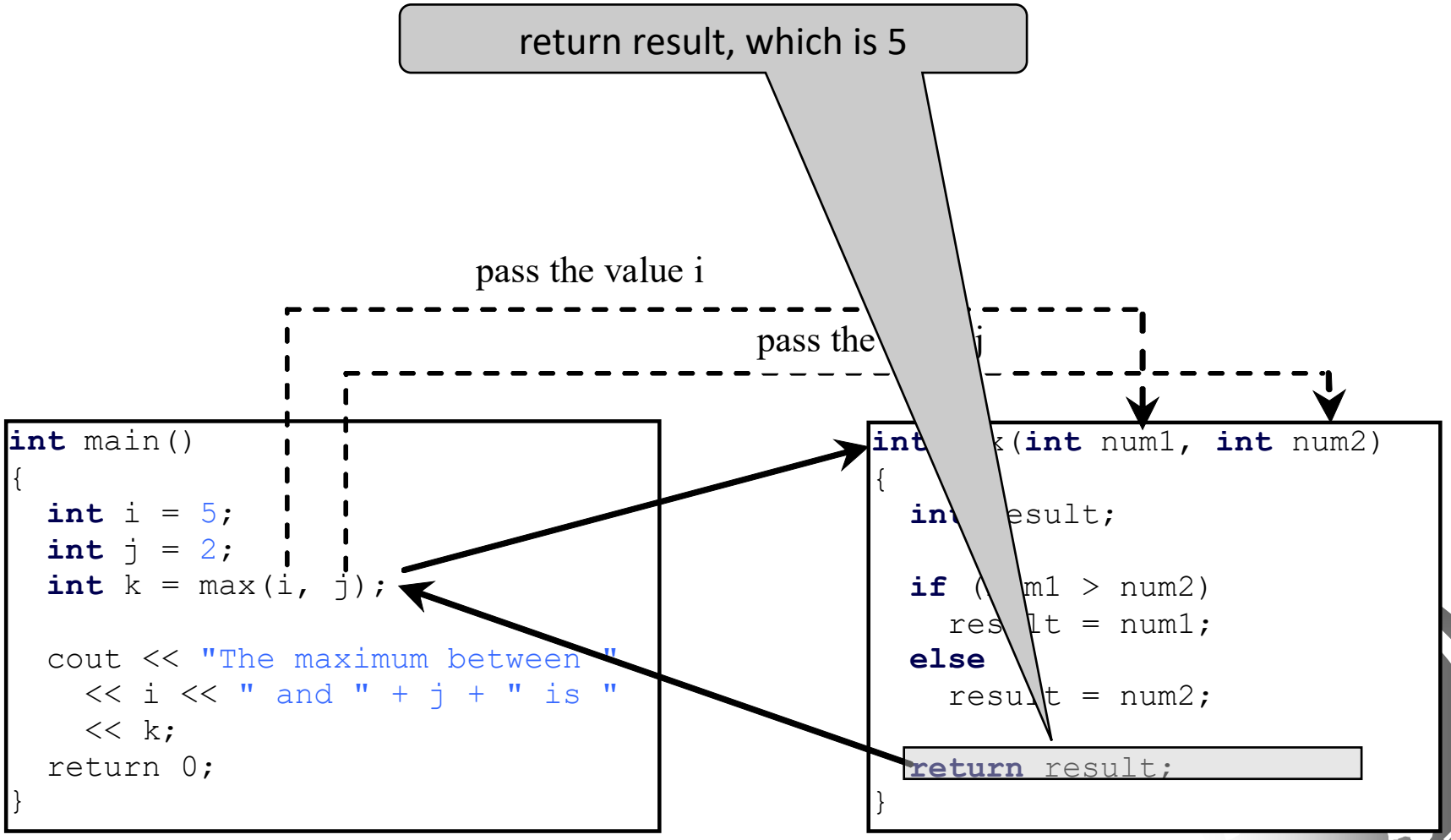
Trace Function Invocation



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

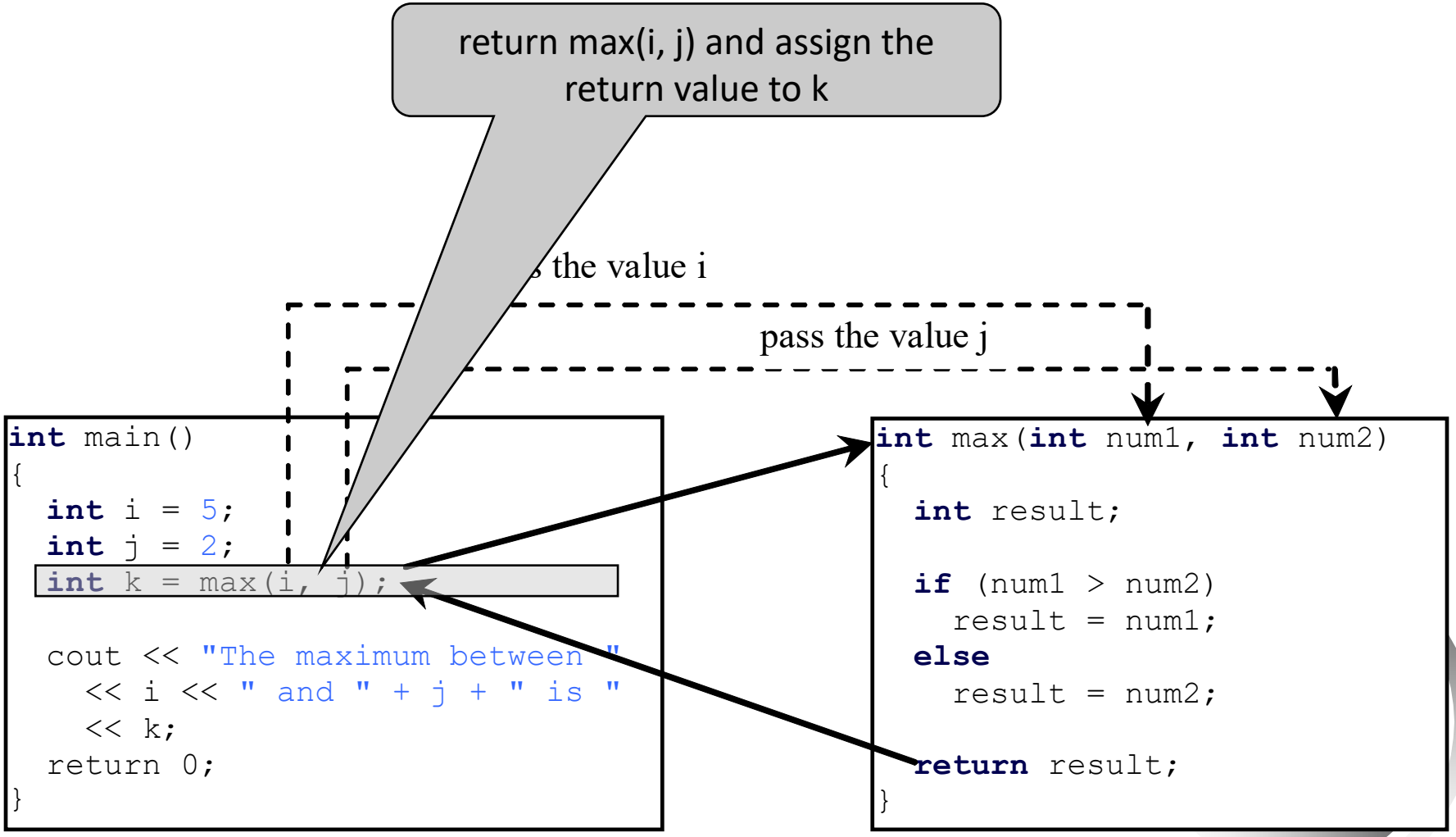
Trace Function Invocation



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

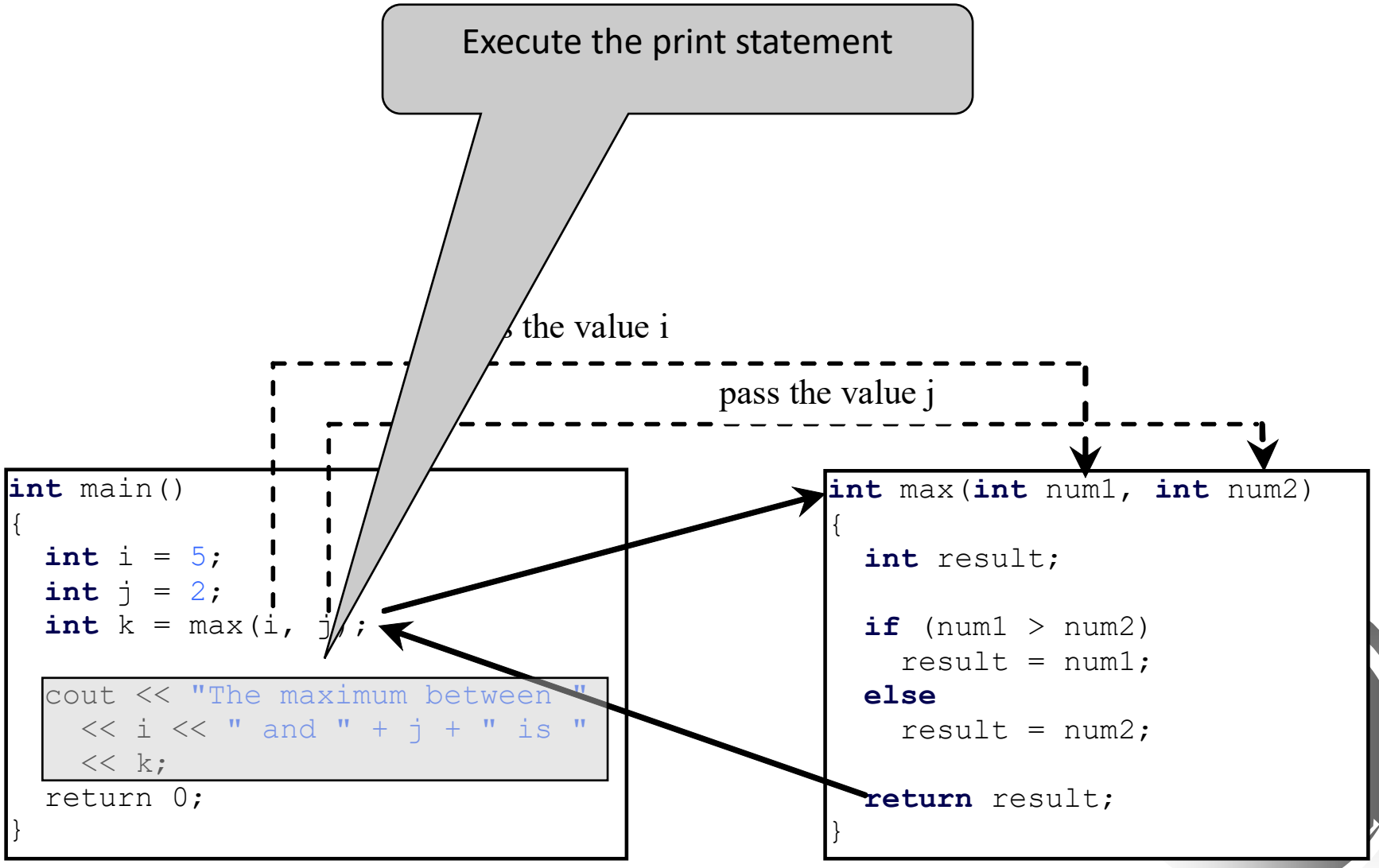
Trace Function Invocation



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

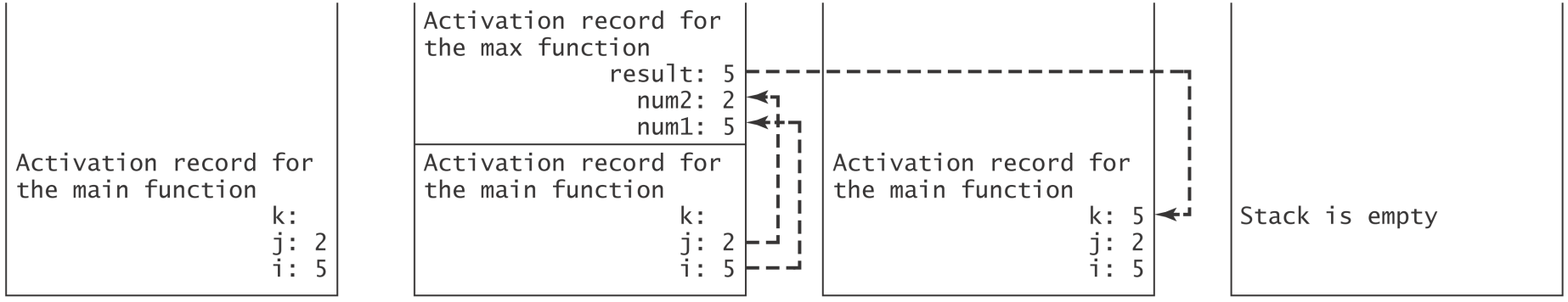
animation

Trace Function Invocation



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Call Stacks



(a) The main function is invoked.

(b) The max function is invoked.

(c) The max function is finished and the return value is sent to k.

(d) The main function is finished.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Call Stack

```
int main()
{
  int i = 5;
  int j = 2;
  int k = max(i, j);

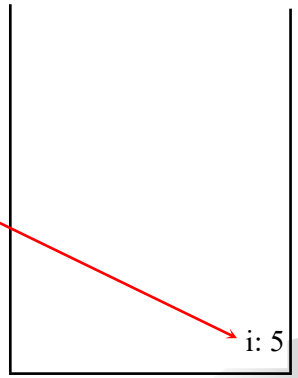
  cout << "The maximum between "
    << i << " and " + j + " is "
    << k;
  return 0;
}
```

i is declared and initialized

```
int max(int num1, int num2)
{
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



The main method is invoked.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Call Stack

```
int main()
{
  int i = 5;
  int j = 2;
  int k = max(i, j);

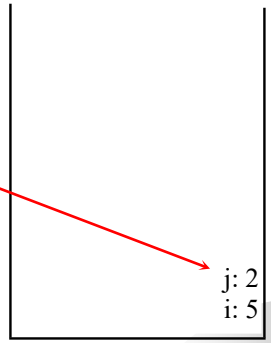
  cout << "The maximum between "
    << i << " and " + j + " is "
    << k;
  return 0;
}
```

j is declared and initialized

```
int max(int num1, int num2)
{
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



The main method is invoked.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Call Stack

```
int main()
{
  int i = 5;
  int j = 2;
  int k = max(i, j);

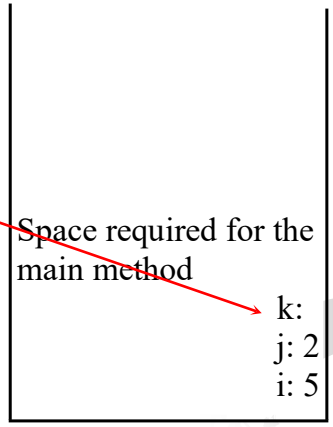
  cout << "The maximum between "
    << i << " and " + j + " is "
    << k;
  return 0;
}
```

```
int max(int num1, int num2)
{
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

Declare k



The main method is invoked.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Call Stack

```
int main()
{
  int i = 5;
  int j = 2;
  int k = max(i, j);

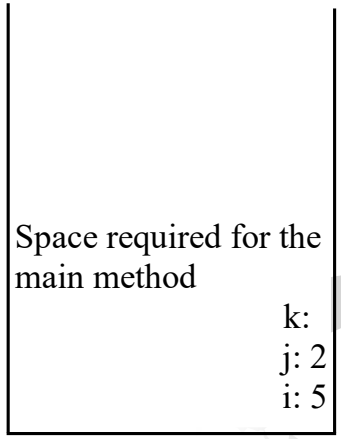
  cout << "The maximum between "
    << i << " and " + j + " is "
    << k;
  return 0;
}
```

```
int max(int num1, int num2)
{
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

Invoke max(i, j)



The main method is invoked.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Call Stack

```
int main()
{
  int i = 5;
  int j = 2;
  int k = max(i, j);

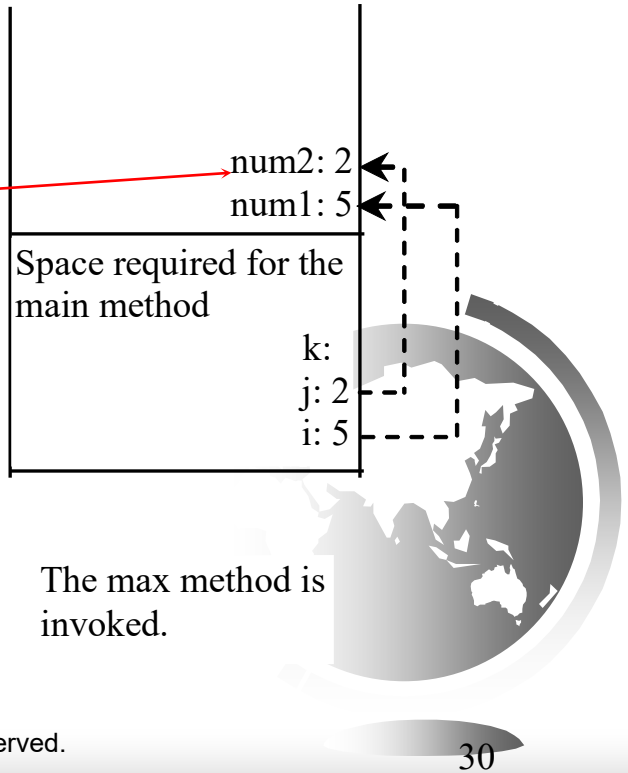
  cout << "The maximum between "
    << i << " and " + j + " is "
    << k;
  return 0;
}
```

pass the values of i and j to num1 and num2

```
int max(int num1, int num2)
{
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Call Stack

```
int main()
{
  int i = 5;
  int j = 2;
  int k = max(i, j);

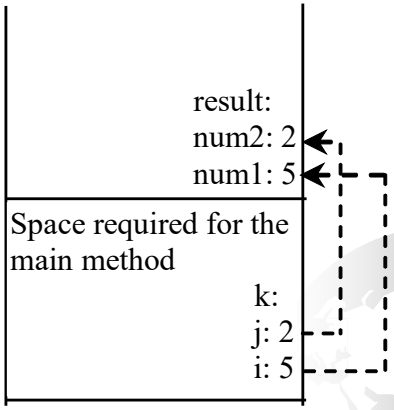
  cout << "The maximum between "
    << i << " and " + j + " is "
    << k;
  return 0;
}
```

```
int max(int num1, int num2)
{
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

(num1 > num2) is true



The max method is invoked.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Call Stack

```
int main()
{
  int i = 5;
  int j = 2;
  int k = max(i, j);

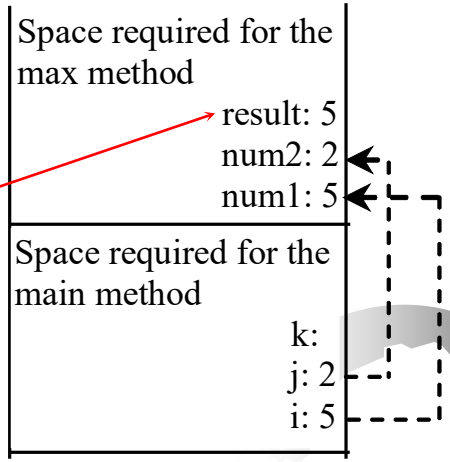
  cout << "The maximum between "
    << i << " and " + j + " is "
    << k;
  return 0;
}
```

```
int max(int num1, int num2)
{
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

Assign num1 to result



The max method is invoked.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Call Stack

```
int main()
{
  int i = 5;
  int j = 2;
  int k = max(i, j);

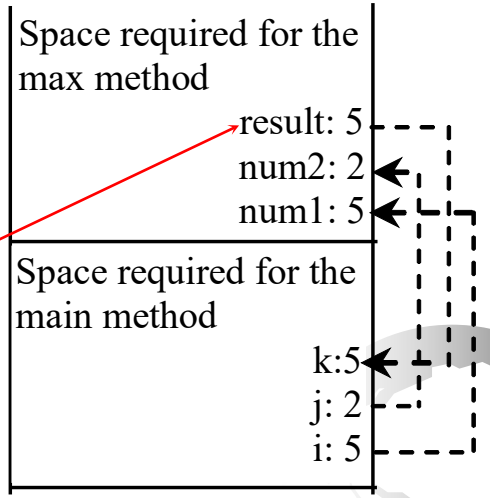
  cout << "The maximum between "
    << i << " and " + j + " is "
    << k;
  return 0;
}
```

```
int max(int num1, int num2)
{
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

Return result and assign it to k



The max method is invoked.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Call Stack

```
int main()
{
  int i = 5;
  int j = 2;
  int k = max(i, j);

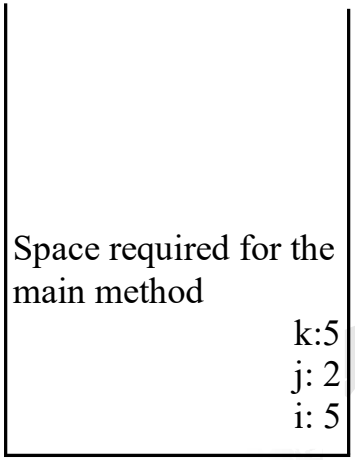
  cout << "The maximum between "
    << i << " and " + j + " is "
    << k;
  return 0;
}
```

Execute print statement

```
int max(int num1, int num2)
{
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



The main method is invoked.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

`void` Functions

- The preceding section gives an example of a nonvoid function.
- This section shows how to declare and invoke a void function.
- The following program declares a function named printGrade and invokes it to print the grade for a given score.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  // Print grade for the score
5  void printGrade(double score)
6  {
7    if (score >= 90.0)
8     cout << 'A' << endl;
9    else if (score >= 80.0)
10    cout << 'B' << endl;
11   else if (score >= 70.0)
12    cout << 'C' << endl;
13   else if (score >= 60.0)
14    cout << 'D' << endl;
15   else
16    cout << 'F' << endl;
17 }
18
19 int main()
20 {
21  cout << "Enter a score: ";
22  double score;
23  cin >> score;
24
25  cout << "The grade is ";
26  printGrade(score);
27
28  return 0;
29 }
```

```
Enter a score: 78.5
The grade is C
```

TestVoidFunction

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  // Return the grade for the score
5  char getGrade(double score)
6  {
7    if (score >= 90.0)
8      return 'A';
9    else if (score >= 80.0)
10     return 'B';
11   else if (score >= 70.0)
12     return 'C';
13   else if (score >= 60.0)
14     return 'D';
15   else
16     return 'F';
17 }
18
19 int main()
20 {
21   cout << "Enter a score: ";
22   double score;
23   cin >> score;
24
25   cout << "The grade is ";
26   cout << getGrade(score) << endl;
27
28   return 0;
29 }
```

```
Enter a score: 78.5
The grade is C
```

TestReturnGradeFunction

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Passing Arguments by Value

By default, the arguments are passed by value to parameters when invoking a function.

- The power of a function is its ability to work with parameters.
- When calling a function, you need to provide arguments
  - Must be given in the same order as their respective parameters in the function signature.
  - This is known as *parameter order association*.
  - For example, the following function prints a character **n** times.

```
void nPrint(char ch, int n)
{
  for (int i = 0; i < n; i++)
     cout << ch;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Modularizing Code

Modularizing makes the code easy to maintain and debug and enables the code to be reused.

- Modularization is software design concept that calls for writing code in modules.
- Functions can be used to reduce redundant coding and enable code reuse.
- Functions can also be used to modularize code and improve the quality of the program.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Greatest Common Divisor (gcd)

Write a program that prompts the user to enter two integers and displays their greatest common divisor.

You can write the program using a method called

```
gcd(int n1,int n2)
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  // Return the gcd of two integers
5  int gcd(int n1, int n2)
6  {
7    int gcd = 1; // Initial gcd is 1
8    int k = 2; // Possible gcd
9
10   while (k <= n1 && k <= n2)
11   {
12     if (n1 % k == 0 && n2 % k == 0)
13       gcd = k; // Update gcd
14     k++;
15   }
16
17   return gcd; // Return gcd
18 }
19
20 int main()
21 {
22   // Prompt the user to enter two integers
23   cout << "Enter first integer: ";
24   int n1;
25   cin >> n1;
26
27   cout << "Enter second integer: ";
28   int n2;
29   cin >> n2;
30
31   cout << "The greatest common divisor for " << n1 <<
32   " and " << n2 << " is " << gcd(n1, n2) << endl;
33
34   return 0;
35 }
```

```
Enter first integer: 45
Enter second integer: 75
The greatest common divisor for 45 and 75 is 15
```

GreatestCommonDivisorFunction

Advantages of Using Modularity on gcd Example

- It isolates computing the GCD from the rest of the code in the main function.
  - The logic becomes clear and the program is easier to read.
- If there are errors on computing GCD, they will be confined in the gcd function, which narrows the scope of debugging.
- The gcd function now can be reused by other programs.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  // Check whether number is prime
6  bool isPrime(int number)
7  {
8   for (int divisor = 2; divisor <= number / 2; divisor++)
9   {
10   if (number % divisor == 0)
11   {
12     // If true, number is not prime
13     return false; // number is not a prime
14   }
15  }
16
17   return true; // number is prime
18  }
19
20  void printPrimeNumbers(int numberOfPrimes)
21  {
21   const int NUMBER_OF_PRIMES_PER_LINE = 10 ; // Display 10 per line
23   int count = 0; // Count the number of prime numbers
24   int number = 2; // A number to be tested for primeness
25
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
26   // Repeatedly find prime numbers
27   while (count < numberOfPrimes)
28   {
29     // Print the prime number and increase the count
30     if (isPrime(number))
31     {
32       count++; // Increase the count
33
34       if (count % NUMBER_OF_PRIMES_PER_LINE == 0)
35       {
36         // Print the number and advance to the new line
37         cout << setw(4) << number << endl;
38       }
39       else
40         cout << setw(4) << number;
41      }
42
43     // Check if the next number is prime
44     number++;
45   }
46 }
47
48 int main()
49 {
50    cout << "The first 50 prime numbers are \n";
51    printPrimeNumbers(50);
52
53   return 0;
54 }
```

PrimeNumberFunction

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
The first 50 prime numbers are
2 3 5 7 11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Overloading Functions

- The max function that was used earlier works only with the int data type.
- But what if you need to find which of two floating-point numbers has the maximum value?
- The solution is to create another function with the same name but different parameters.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  // Return the max between two int values
5  int max(int num1, int num2)
6  {
7   if (num1 > num2)
8     return num1;
9   else
10    return num2;
11 }
12
13  // Find the max between two double values
14  double max(double num1, double num2)
15  {
16   if (num1 > num2)
17     return num1;
18   else
19     return num2;
20  }
21
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
23  double max(double num1, double num2, double num3)
24  {
25   return max(max(num1, num2), num3);
26  }
27
28  int main()
29  {
30    // Invoke the max function with int parameters
31    cout << "The maximum between 3 and 4 is " << max(3, 4) << endl;
32
33    // Invoke the max function with the double parameters
34    cout << "The maximum between 3.0 and 5.4 is "
35         << max(3.0, 5.4) << endl;
36
37    // Invoke the max function with three double parameters
38    cout << "The maximum between 3.0, 5.4, and 10.14 is "
39         << max(3.0, 5.4, 10.14) << endl;
40
41   return 0;
42  }
```

TestFunctionOverloading

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Ambiguous Invocation

- Sometimes there may be two or more possible matches for an invocation of a function, but the compiler cannot determine the most specific match.
- This is referred to as *ambiguous invocation*. Ambiguous invocation is a compilation error.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Ambiguous Invocation

```
#include <iostream>
using namespace std;
int maxNumber(int num1, double num2)
{
  if (num1 > num2)
    return num1;
  else
    return num2;
}
double maxNumber(double num1, int num2)
{
  if (num1 > num2)
    return num1;
  else
    return num2;
}
int main()
{
  cout << maxNumber(1, 2) << endl;
  return 0;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Function Prototypes

- Before a function is called, it must be declared first.
  - One way to ensure it is to place the declaration before all function calls.
  - Another way to approach it is to declare a function prototype before the function is called.
- A function prototype is a function declaration without implementation.
- The implementation can be given later in the program.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  // Function prototype
5  int max(int num1, int num2);
6  double max(double num1, double num2);
7  double max(double num1, double num2, double num3);
8
9  int main()
10  {
11   // Invoke the max function with int parameters
12   cout << "The maximum between 3 and 4 is " <<
13   max(3, 4) << endl;
14
15   // Invoke the max function with the double parameters
16   cout << "The maximum between 3.0 and 5.4 is "
17        << max(3.0, 5.4) << endl;
18
19   // Invoke the max function with three double parameters
20   cout << "The maximum between 3.0, 5.4, and 10.14 is "
21        << max(3.0, 5.4, 10.14) << endl;
22
23   return 0;
24  }
25
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
26  // Return the max between two int values
27  int max(int num1, int num2)
28  {
29   if (num1 > num2)
30     return num1;
31   else
32     return num2;
33  }
34
35  // Find the max between two double values
36  double max(double num1, double num2)
37  {
38   if (num1 > num2)
39     return num1;
40   else
41     return num2;
42  }
43
44  // Return the max among three double values
45  double max(double num1, double num2, double num3)
46  {
47    return max(max(num1, num2), num3);
48  }
```

TestFunctionPrototype

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Default Arguments

- C++ allows you to declare functions with default argument values.
- The default values are passed to the parameters when a function is invoked without the arguments.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  // Display area of a circle
5  void printArea(double radius = 1)
6  {
7    double area = radius * radius * 3.14159;
8    cout << "area is " << area << endl;
9  }
10
11 int main()
12 {
13   printArea();
14   printArea(4);
15
16   return 0;
17 }
```

```
area is 3.14159
area is 50.2654
```

DefaultArgumentDemo

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Default Arguments

- When a function contains a mixture of parameters with and without default values, those with default values must be declared last

```
void t1(int x, int y = 0, int z); // Illegal
void t2(int x = 0, int y = 0, int z); // Illegal

void t3(int x, int y = 0, int z = 0); // Legal
void t4(int x = 0, int y = 0, int z = 0); // Legal
```

- When an argument is left out of a function, all arguments that come after it must be left out as well.

```
t3(1, , 20); // Illegal
t4(, , 20); // Illegal

t3(1); // Parameters y and z are assigned a default value
t4(1, 2); // Parameter z is assigned a default value
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

`Inline` Functions

- functions makes the program easy to read and easy to maintain
- but function calls involve runtime overhead (i.e., pushing arguments and CPU registers into the stack and transferring control to and from a function).
- C++ provides *inline functions* to avoid function calls.
- Inline functions are not called
- The compiler copies the function code *in line* at the point of each invocation.
- To specify an inline function, precede the function declaration with the `inline` keyword, as shown in Listing 6.18.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  inline void f(int month, int year)
5  {
6   cout << "month is " << month << endl;
7   cout << "year is " << year << endl;
8  }
9
10 int main()
11 {
12   int month = 10, year = 2008;
13   f(month, year); // Invoke inline function
14   f(9, 2010); // Invoke f with month 9 and year 2010
15
16   return 0;
17 }
```

```
month is 10
year is 2008
month is 9
year is 2010
```

InlineDemo

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Inline Expanded

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6     int month = 10, year = 2008;
7     cout << "month is " << month << endl;
8     cout << "year is " << year << endl;
9     cout << "month is " << 9 << endl;
10    cout << "year is " << 2010 << endl; 11
12    return 0;
13  }
```

InlineExpandedDemo



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Short Functions, Not for long Functions
Compiler Decision

- Inline functions are desirable for short functions
  - Not suitable for long functions that are called in multiple places in a program
  - Long inline functions will dramatically increase the executable code size when it is copied in multiple places.
- For this reason, C++ allows the compilers to ignore the inline keyword if the function is too long.
- The inline keyword is merely a request to the compiler
  - It is up for the compiler to make the decision whether to honor it or ignore it.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Scope of Variables

- Scope
  - The part of the program where the variable can be referenced.
  - The scope of a variable starts from its declaration and continues to the end of the block that contains the variable.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Local Variables

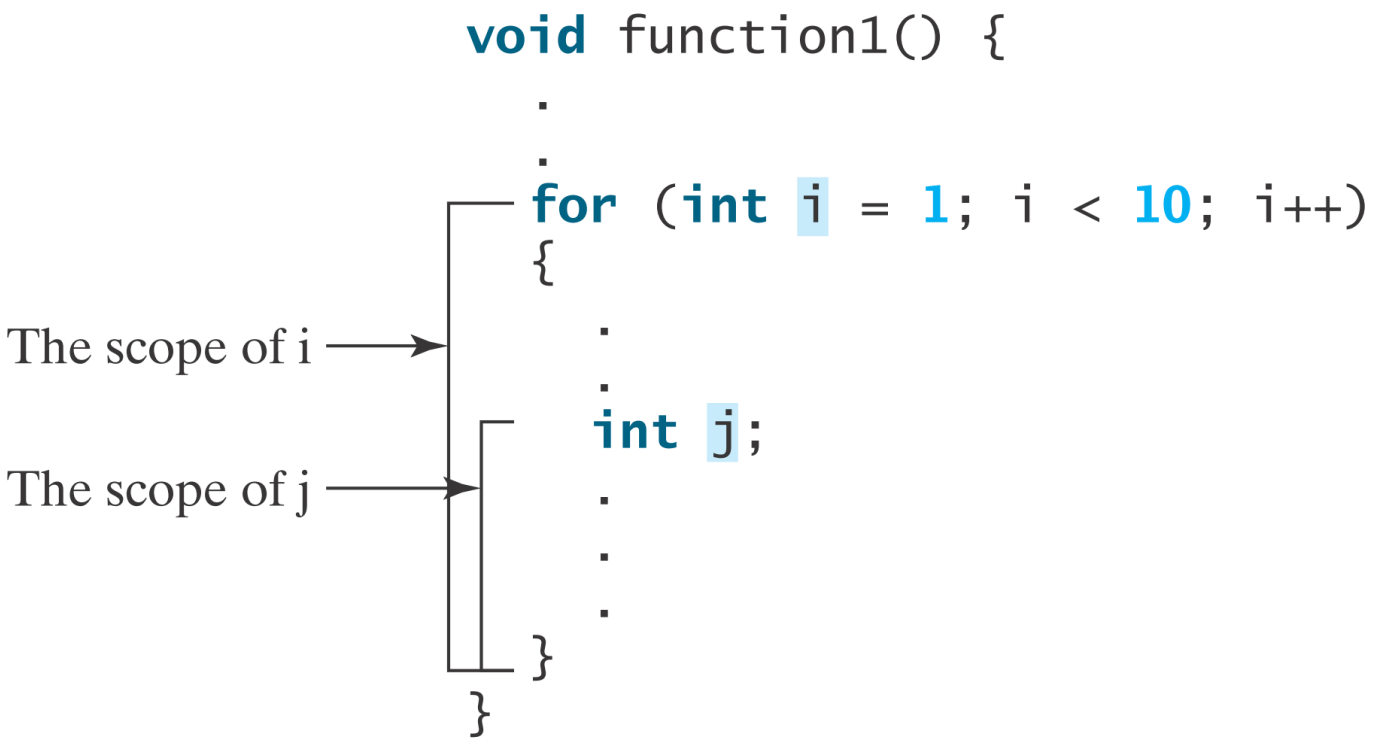
- A local variable:
  - A variable defined inside a function.
  - Local variables do not have default values
  - Must be declared and assigned a value before it can be used.
  - Scope starts from its declaration and continues to the end of the block that contains the variable



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Scope of Local Variables, cont.

- A variable declared in the initial action part of a for loop header has its scope in the entire loop.
- A variable declared inside a for loop body has its scope from its declaration to the end of the block that contains the variable.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Scope of Local Variables, cont.

It is fine to declare i in two nonnesting blocks

```
void function1()
{
  int x = 1;
  int y = 1;

  for (int i = 1; i < 10; i++)
  {
    x += i;
  }

  for (int i = 1; i < 10; i++)
  {
    y += i;
  }
}
```

It is not a good practice to declare i in two nesting blocks

```
void function2()
{
  int i = 1;
  int sum = 0;

  for (int i = 1; i < 10; i++)
  {
    sum += i;
  }

  cout << i << endl;
  cout << sum << endl;
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Global Variables

- Global variables:
  - Declared outside all functions and are accessible to all functions in their scope.
  - Defaulted to zero.
  - Scope starts from its declaration and continues to the end of the program



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  void t1(); // Function prototype
5  void t2(); // Function prototype
6
7  int main()
8  {
9   t1();
10   t2();
11
12   return 0;
13  }
14
15  int y; // Global variable, default to 0
16
17  void t1()
18  {
19   int x = 1;
20   cout << "x is " << x << endl;
21   cout << "y is " << y << endl;
22   x++;
23   y++;
24  }
25
26  void t2()
27  {
28   int x = 1;
29   cout << "x is " << x << endl;
30   cout << "y is " << y << endl;
31  }
```

VariableScopeDemo

Unary Scope Resolution

- If a local variable name is the same as a global variable name
  - You can access the global variable using ::globalVariable.
- The :: operator is known as the *unary scope resolution*.

```
#include <iostream>
using namespace std;
int v1 = 10;
int main()
{
  int v1 = 5;
  cout << "local variable v1 is " << v1 << endl;
  cout << "global variable v1 is " << ::v1 << endl;
  return 0;
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Static Local Variables

- After a function completes its execution, all its local variables are destroyed.
- Sometimes, it is desirable to retain the value stored in local variables so that they can be used in the next call.
- C++ allows you to declare **`static`** local variables.
- **`Static`** local variables are permanently allocated in the memory for the lifetime of the program.
- To declare a static variable, use the keyword **`static`**.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  void t1(); // Function prototype
5
6  int main()
7  {
8   t1();
9   t1();
10
11   return 0;
12  }
13
14  void t1()
15  {
16   static int x = 1;
17   int y = 1;
18   x++;
19   y++;
20   cout << "x is " << x << endl;
21   cout << "y is " << y << endl;
22  }
```

```
x is 2
y is 2
x is 3
y is 2
```

StaticVariableDemo

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Pass by Value

- When you invoke a function with a parameter, the value of the argument is passed to the parameter.
- This is referred to as *pass-by-value*.
- If the argument is a variable rather than a literal value, the value of the variable is passed to the parameter.
- The variable is not affected, regardless of the changes made to the parameter inside the function.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1   #include <iostream>
2   using namespace std;
3
4   void increment(int n)
5   {
6    n++;
7    cout << "\tn inside the function is " << n << endl;
8   }
9
10   int main()
11   {
12    int x = 1;
13    cout << "Before the call, x is " << x << endl;
14    increment(x);
15    cout << "after the call, x is " << x << endl;
16
17    return 0;
18   }
```

```
Before the call, x is 1
   n inside the function is 2
after the call, x is 1
```

Increment

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Pass by Value, cont.

- Pass-by-value has serious limitations.
- Next slide shows a program that shows the effect and limitation of passing by value.
- The program creates a function for swapping two variables.
- The swap function is invoked by passing two arguments.
- Interestingly, the values of the arguments are not changed after the function is invoked.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  // Attempt to swap two variables does not work!
5  void swap(int n1, int n2)
6  {
7    cout << "\tInside the swap function" << endl;
8    cout << "\tBefore swapping n1 is " << n1 <<
9            " n2 is " << n2 << endl;
10
11   // Swap n1 with n2
12   int temp = n1;
13   n1 = n2;
14   n2 = temp;
15
16   cout << "\tAfter swapping n1 is " << n1 <<
17           " n2 is " << n2 << endl;
18 }
19
20 int main()
21 {
22   // Declare and initialize variables
23   int num1 = 1;
24   int num2 = 2;
25
26   cout << "Before invoking the swap function, num1 is "
27        << num1 << " and num2 is " << num2 << endl;
28
29   // Invoke the swap function to attempt to swap two variables
30   swap(num1, num2);
31
32   cout << "After invoking the swap function, num1 is " << num1 <<
33           " and num2 is " << num2 << endl;
34
35   return 0;
36 }
```

Sample Output

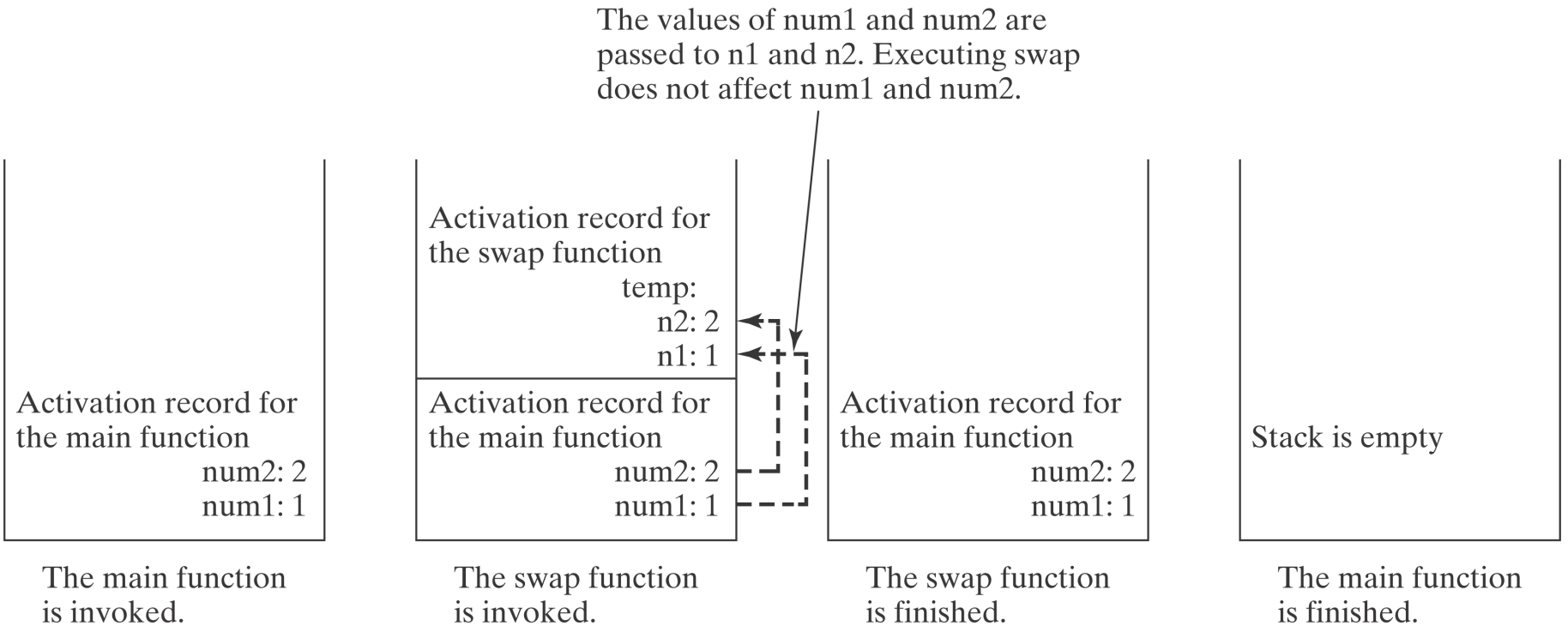
```
Before invoking the swap function, num1 is 1 and num2 is 2
Inside the swap function
Before swapping n1 is 1 n2 is 2
After swapping n1 is 2 n2 is 1
After invoking the swap function, num1 is 1 and num2 is 2
```

SwapByValue



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Pass by Value, cont.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Reference Variables

- C++ provides a special type of variable, called a *reference variable*
- Can be used as a function parameter to reference the original variable.
- A reference variable is an alias for another variable.
- Any changes made through the reference variable are actually performed on the original variable.
- To declare a reference variable, place the ampersand (&) in front of the name.

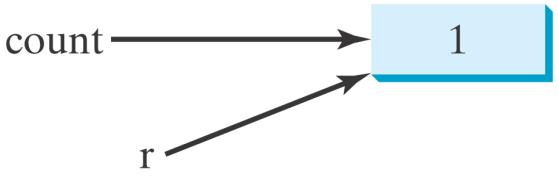


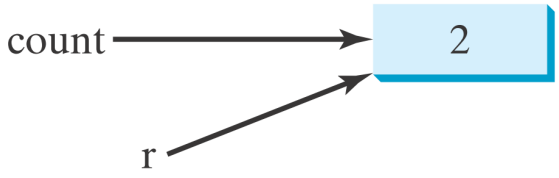
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    int count = 1;
7    int& r = count;
8    cout << "count is " << count << endl;
9    cout << "r is " << r << endl;
10
11   r++;
12   cout << "count is " << count << endl;
13   cout << "r is " << r << endl;
14
15   count = 10;
16   cout << "count is " << count << endl;
17   cout << "r is " << r << endl;
18
19   return 0;
20 }
```

```
count is 1
r is 1
count is 2
r is 2
count is 10
r is 10
```

TestReferenceVariable







© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Pass by Reference

```
1  #include <iostream>
2  using namespace std;
3
4  void increment(int& n)
5  {
6   n++;
7   cout << "\tn inside the function is " << n << endl;
8  }
9
10  int main()
11  {
12   int x = 1;
13   cout << "Before the call, x is " << x << endl;
14   increment(x);
15   cout << "after the call, x is " << x << endl;
16
17   return 0;
18  }
```

```
Before the call, x is 1
n inside the function is 2
After the call, x is 2
```

IncrementWithPassByReference



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Pass By Reference

- You can use a reference variable as a parameter in a function and pass a regular variable to invoke the function.
- The parameter becomes an alias for the original variable.
- This is known as *pass-by-reference*.
- When you change the value through the reference variable, the original value is actually changed.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  // Attempt to swap two variables does not work!
5  void swap(int& n1, int& n2)
6  {
7    cout << "\tInside the swap function" << endl;
8    cout << "\tBefore swapping n1 is " << n1 <<
9            " n2 is " << n2 << endl;
10
11   // Swap n1 with n2
12   int temp = n1;
13   n1 = n2;
14   n2 = temp;
15
16   cout << "\tAfter swapping n1 is " << n1 <<
17           " n2 is " << n2 << endl;
18 }
19
20 int main()
21 {
22   // Declare and initialize variables
23   int num1 = ;
24   int num2 = ;
25
26   cout << "Before invoking the swap function, num1 is "
27        << num1 << " and num2 is " << num2 << endl;
28
29   // Invoke the swap function to attempt to swap two variables
30   swap(num1, num2);
31
32   cout << "After invoking the swap function, num1 is " << num1 <<
33           " and num2 is " << num2 << endl;
34
35   return 0;
36 }
```

Sample Output

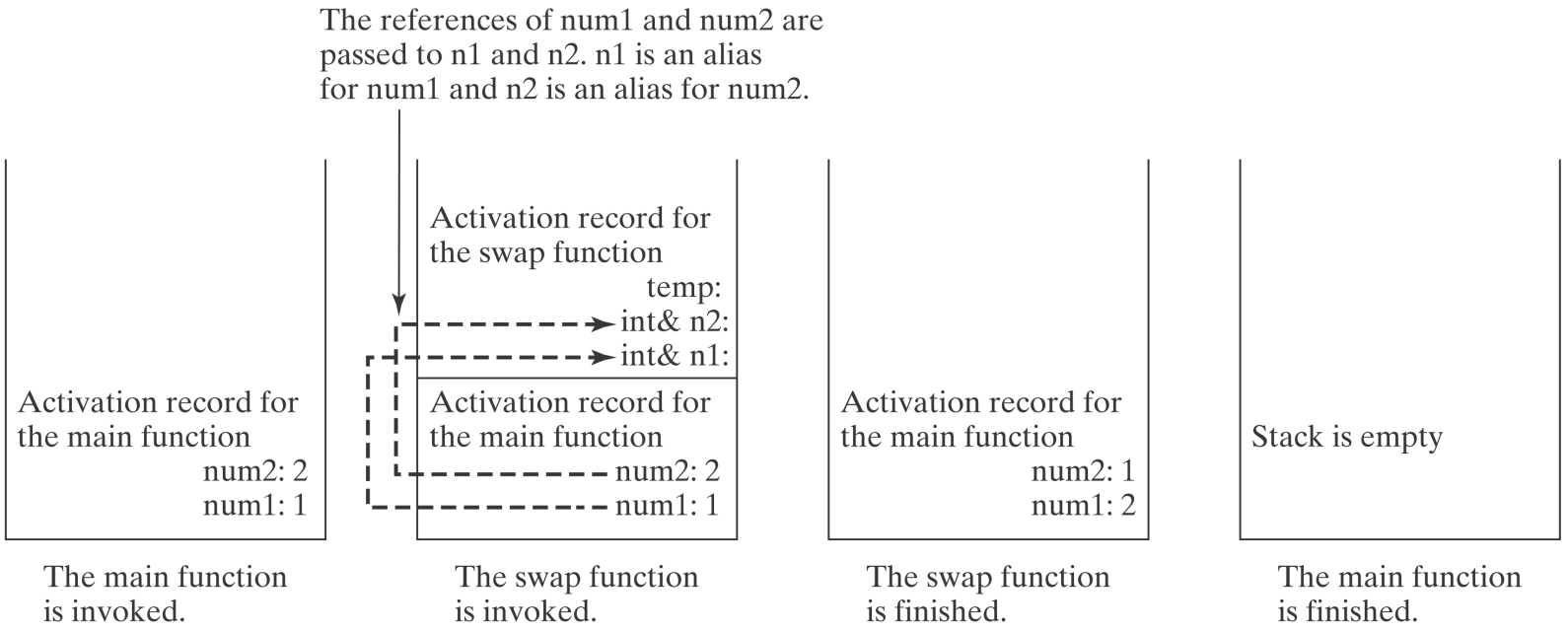
```
Before invoking the swap function, num1 is 1 and num2 is 2
Inside the swap function
Before swapping n1 is 1 n2 is 2
After swapping n1 is 2 n2 is 1
After invoking the swap function, num1 is 2 and num2 is 1
```

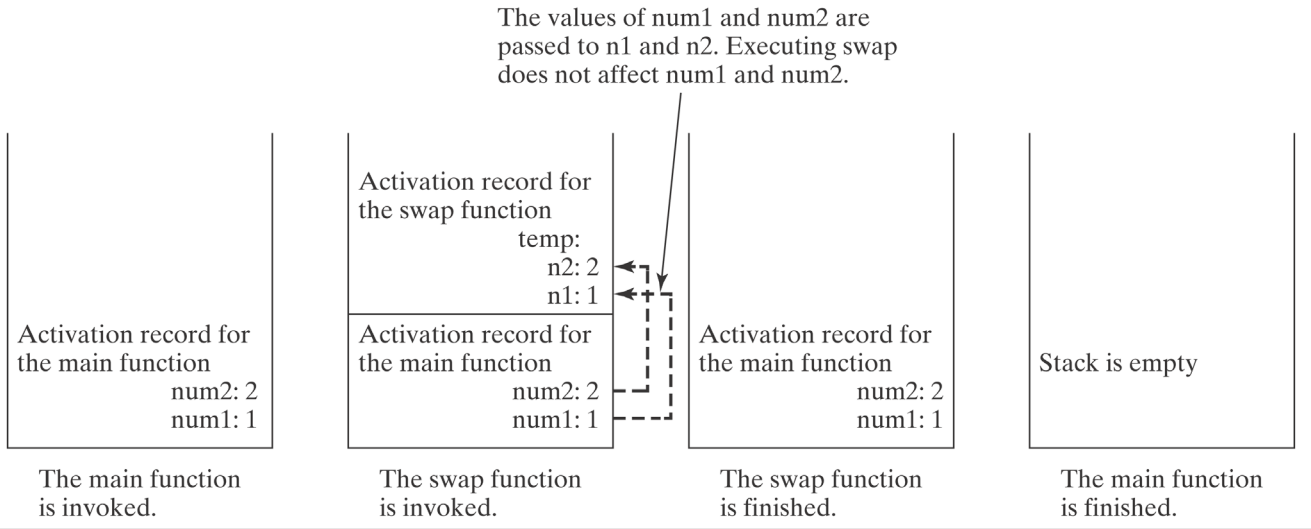
SwapByReference



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Pass by Reference, cont.





Pass-by-Value or Pass-by-Reference

- In pass-by-value, the actual parameter and its formal parameter are independent variables.
- In pass-by-reference, the actual parameter and its formal parameter refer to the same variable.
- Pass-by-reference is more efficient than pass-by-value.
- However, the difference is negligible for parameters of primitive types such as int and double.

if a primitive data type parameter is not changed in the function, you should declare it as pass-by-value parameter.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Constant Reference Parameters

You can specify a constant reference parameter to prevent its value from being changed by accident.

```
// Return the max between two numbers
int max(const int& num1, const int& num2)
{
  int result;
  if (num1 > num2)
    result = num1;
  else
    result = num2;
  return result;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study: Converting Hexadecimals to Decimals

Write a program that converts a hexadecimal number into a decimal number.

ABCD =>

A*16^3 + B*16^2 + C*16^1+ D*16^0

= ((A*16 + B)*16 + C)*16+D

= ((10*16 + 11)*16 + 12)*16+13 = ?



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study Cont.

Our program will prompt the user to enter a hex number as a string and convert it into a decimal using the following method:

```
int hex2Dec(const string& hex);

int hexCharToDecimal(char ch);
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  #include <string>
3  #include <cctype>
4  using namespace std;
5
6  // Converts a hex number as a string to decimal
7  int hex2Dec(const string& hex);
8
9  // Converts a hex character to a decimal value
10 int hexCharToDecimal(char ch);
11
12 int main()
13 {
14   // Prompt the user to enter a hex number as a string
15   cout << "Enter a hex number: ";
16   string hex;
17   cin >> hex;
18
19   cout << "The decimal value for hex number " << hex
20        << " is " << hex2Dec(hex) << endl;
21
22   return 0;
23 }
24
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
25 int hex2Dec(const string& hex)
26 {
27   int decimalValue = 0;
28   for (unsigned i = 0; i < hex.size(); i++)
29     decimalValue = decimalValue * 16 + hexCharToDecimal(hex[i]);
30
31   return decimalValue;
32 }
33
34 int hexCharToDecimal(char ch)
35 {
36   ch = toupper(ch); // Change it to uppercase
37   if (ch >= 'A' && ch <= 'F')
38     return 10 + ch - 'A';
39   else // ch is '0', '1', ..., or '9'
40     return ch - '0';
41 }
```

```
Enter a hex number: af71
The decimal value for hex number
af71 is 44913
```

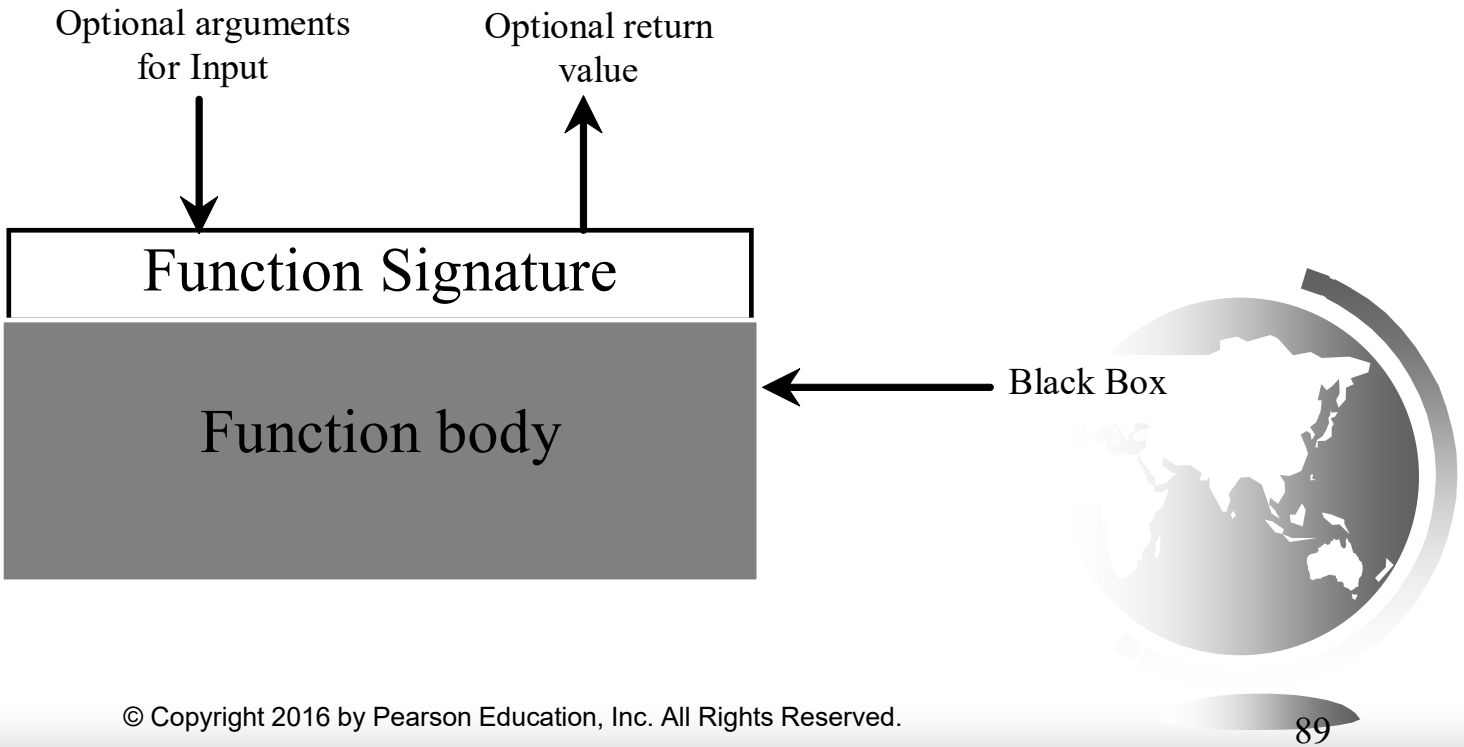
```
Enter a hex number: AB8C
The decimal value for hex number
AB8C is 43916
```

Hex2Dec

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Function Abstraction

You can think of the function body as a black box that contains the detailed implementation for the method.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Benefits of Functions

- Write a function once and reuse it anywhere.
- Information hiding. Hide the implementation from the user.
- Reduce complexity.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Stepwise Refinement

- The concept of function abstraction can be applied to the process of developing programs.
- When writing a large program, you can use the "divide and conquer" strategy, also known as *stepwise refinement*, to decompose it into subproblems.
- The subproblems can be further decomposed into smaller, more manageable problems.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

PrintCalender Case Study

Let us use the `PrintCalendar` example to demonstrate the stepwise refinement approach.

```
Enter full year (e.g., 2012): 2012
Enter month as a number between 1 and 12: 3
March 2012
-----------------------------
Sun Mon Tue Wed Thu Fri Sat
                  1   2   3
  4   5   6   7   8   9  10
 11  12  13  14  15  16  17
 18  19  20  21  22  23  24
 25  26  27  28  29  30  31
```

PrintCalendar



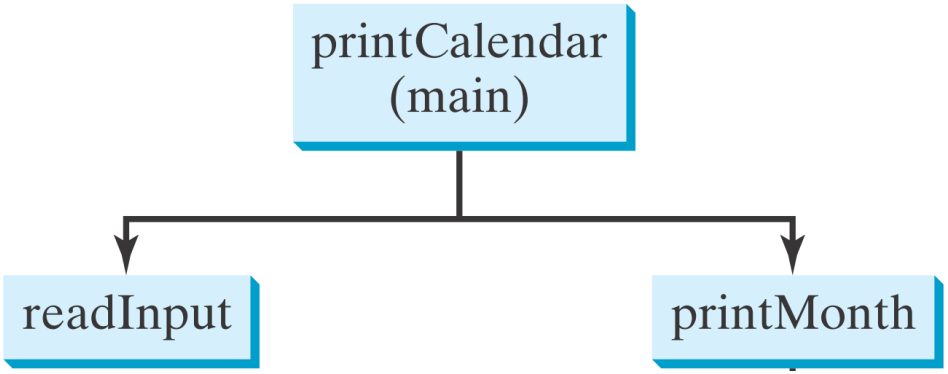
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

printCalendar
(main)

Design Diagram

At this stage, you should be concerned with what the subproblems will achieve, not with how to get input and print the calendar for the month.

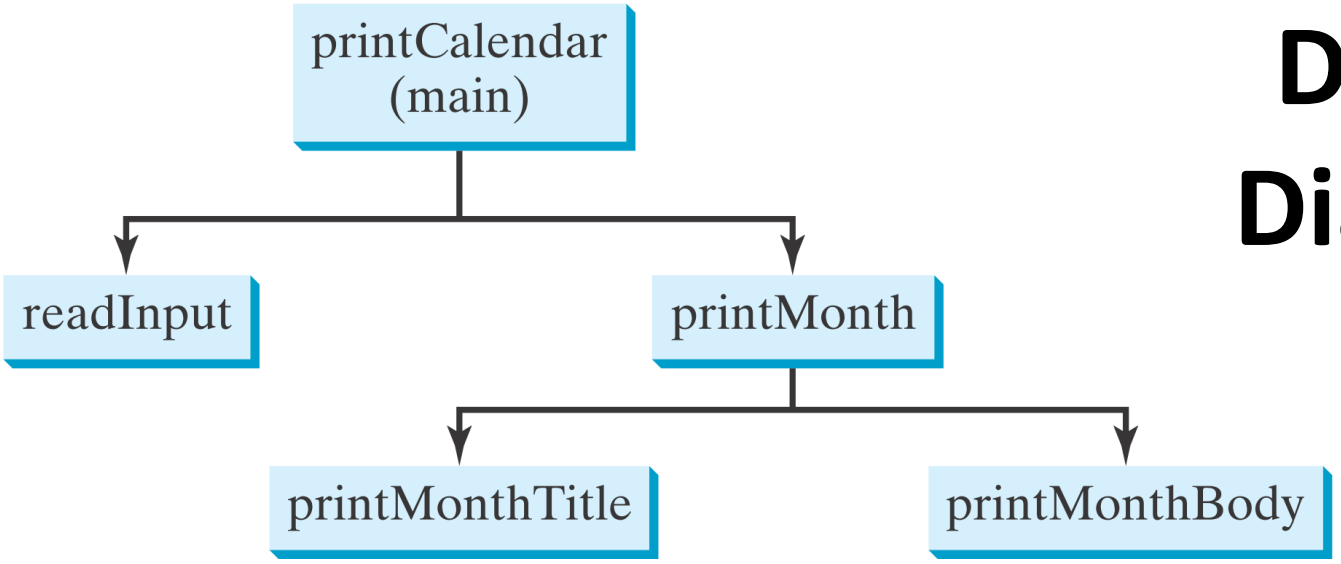
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.



Design Diagram

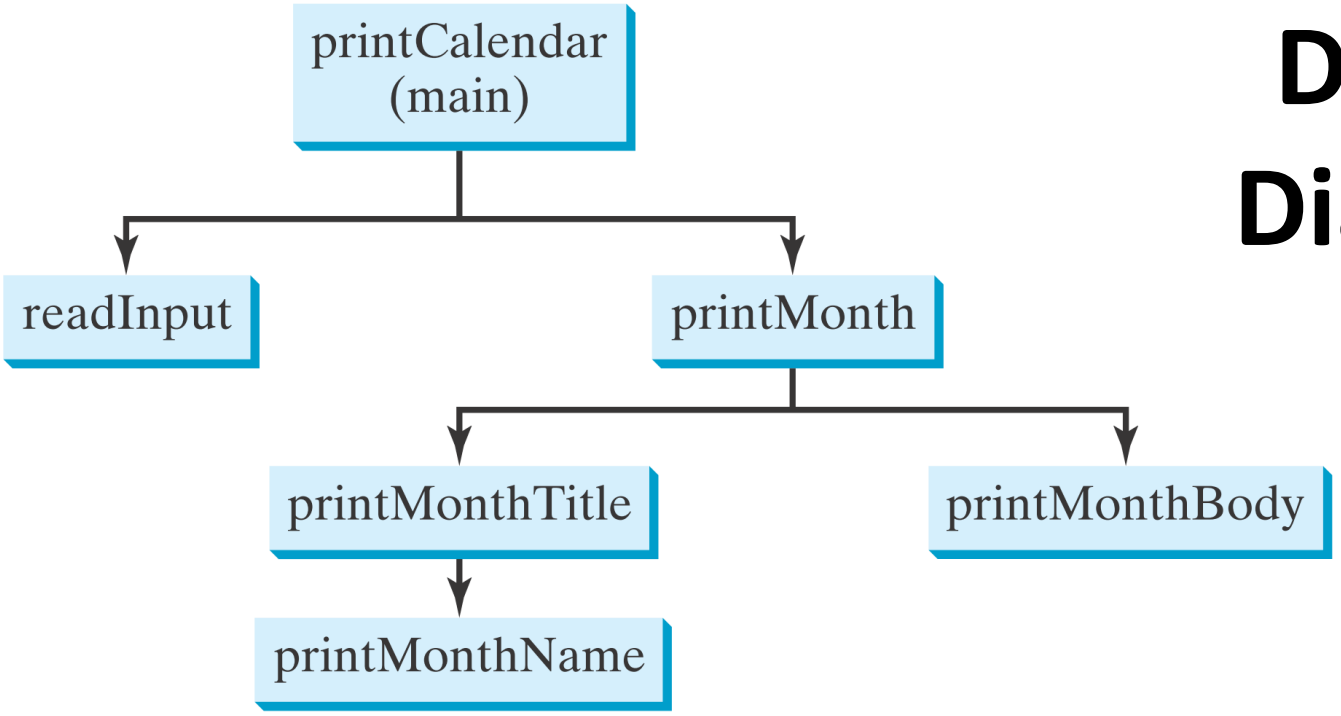
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Design Diagram



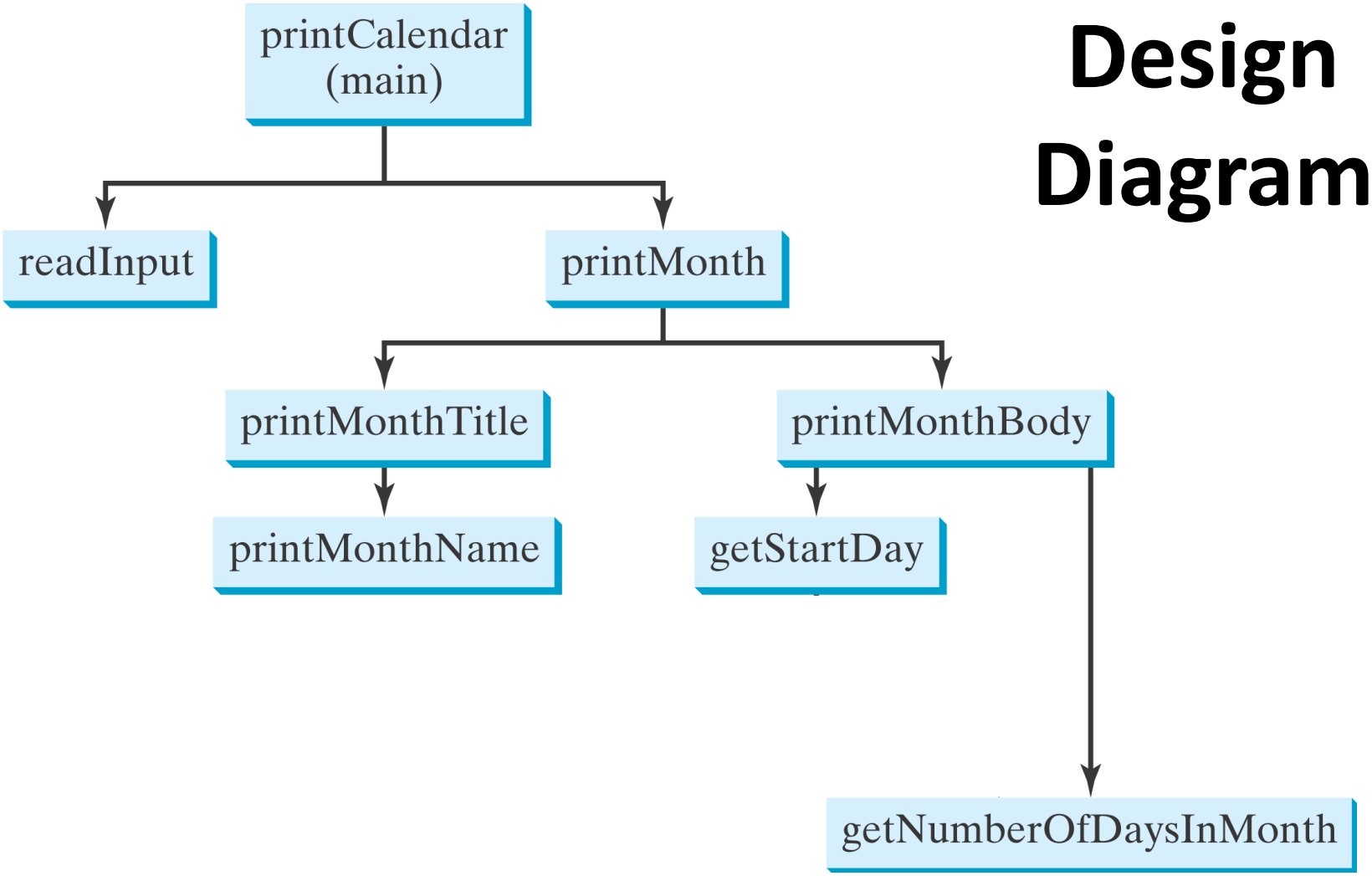
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Design Diagram



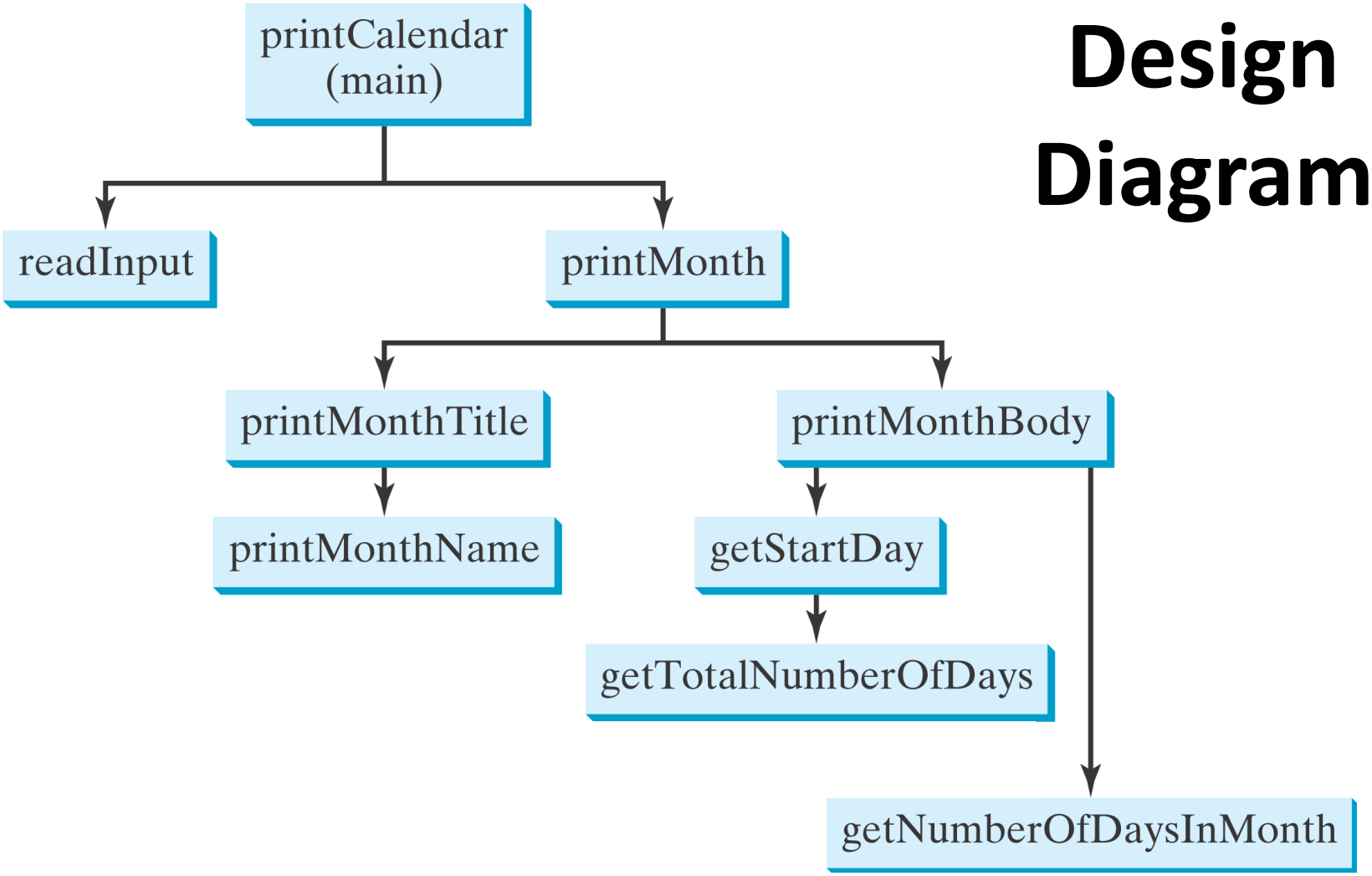
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Design Diagram



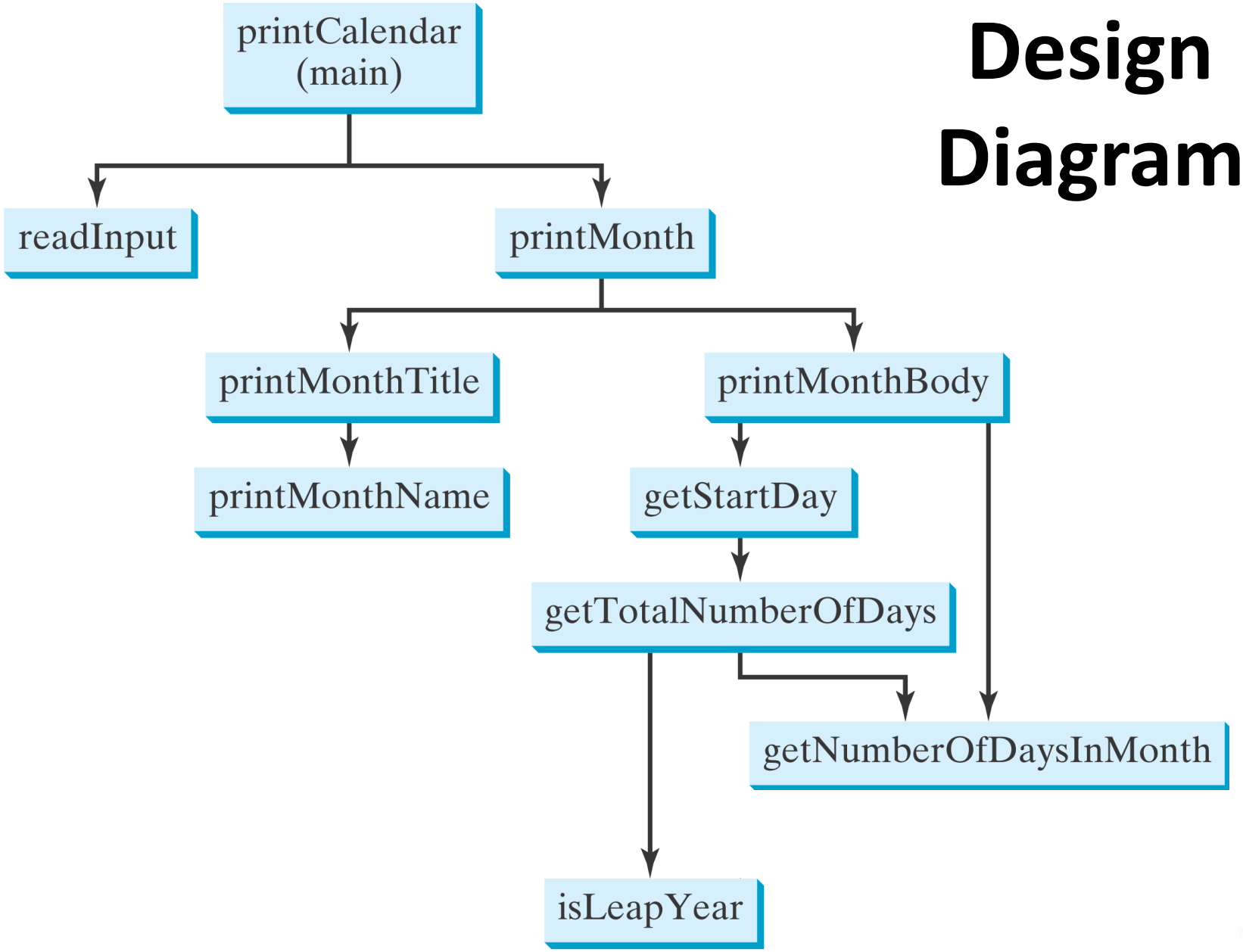
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Design Diagram



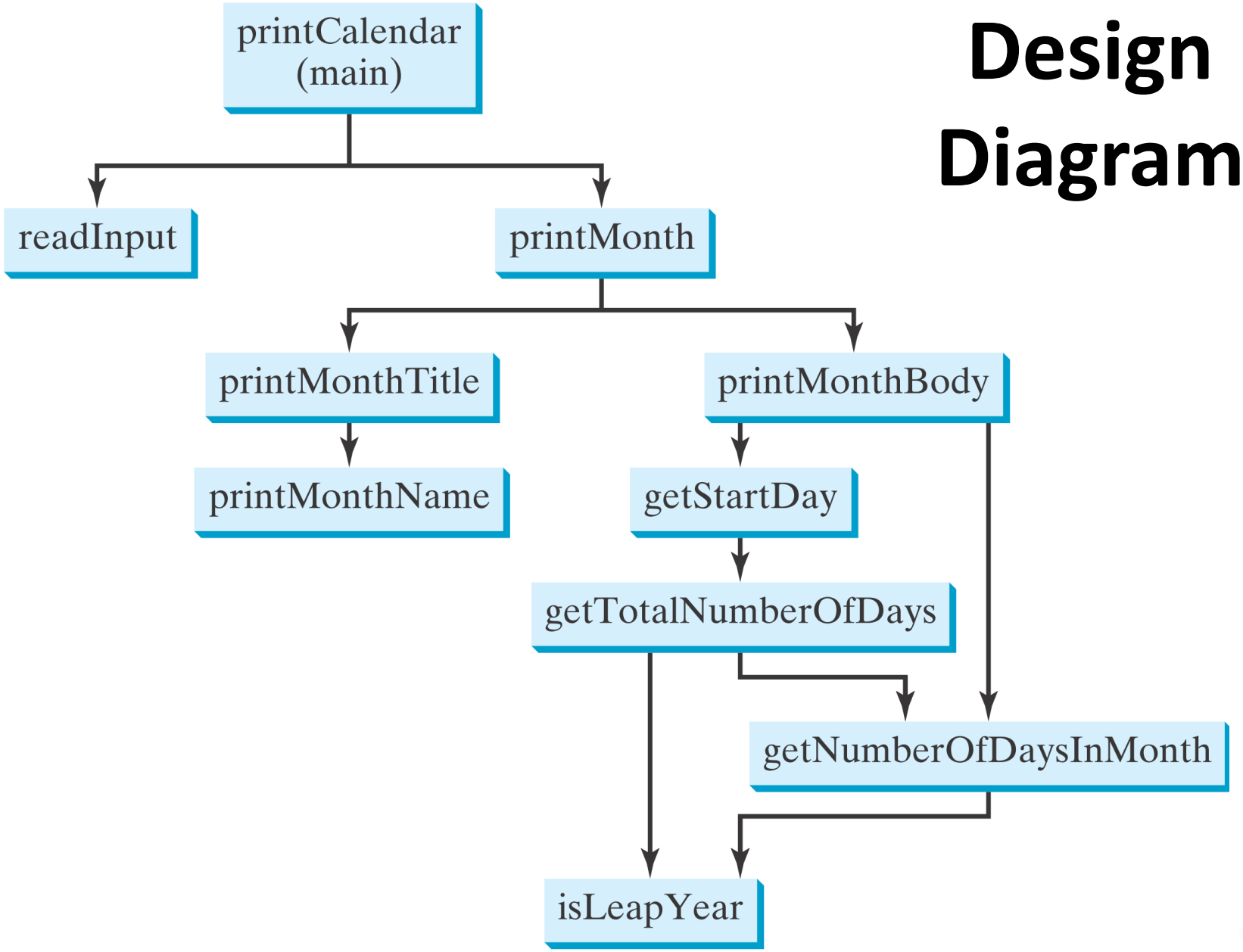
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Design Diagram



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

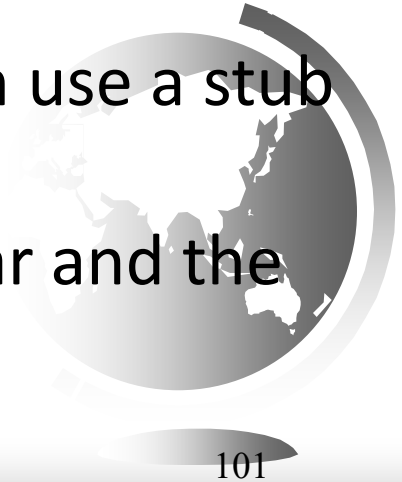
Design Diagram



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Implementation: Top-Down

- Top-down approach is to implement one Function in the structure chart at a time from the top to the bottom.
- Stubs can be used for the Functions waiting to be implemented.
- A stub is a simple but incomplete version of a Function.
- The use of stubs enables you to test invoking the Function from a caller.
- Implement the main Function first and then use a stub for the printMonth Function.
- For example, let printMonth display the year and the month in the stub.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  void printMonth(int year, int month);
6  void printMonthTitle(int year, int month);
7  void printMonthName(int month);
8  void printMonthBody(int year, int month);
9  int getStartDay(int year, int month);
10 int getTotalNumberOfDays(int year, int month);
11 int getNumberOfDaysInMonth(int year, int month);
12 bool isLeapYear(int year);
13
14 int main()
15 {
16   // Prompt the user to enter year
17   cout << "Enter full year (e.g., 2001): ";
18   int year;
19   cin >> year;
20
21   // Prompt the user to enter month
22   cout << "Enter month in number between 1 and 12: ";
23   int month;
24   cin >> month;
25
26   // Print calendar for the month of the year
27   printMonth(year, month);
28
29   return 0;
30 }
31
32 void printMonth(int year, int month)
33 {
34   cout << month << " " << year << endl;
35 }
```

PrintCalendarSkeleton

Implementation: Bottom-Up

- Bottom-up approach is to implement one Function in the structure chart at a time from the bottom to the top.
- For each Function implemented, write a test program to test it.
- Both top-down and bottom-up Functions are fine.
- Both approaches implement the Functions incrementally and help to isolate programming errors and makes debugging easy.
- Sometimes, they can be used together.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Benefits of Stepwise Refinement

- Simpler Program
- Reusing Functions
- Easier Developing, Debugging, and Testing
- Better Facilitating Teamwork



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

References

- Introduction to Programming with C++, Daniel Liang, Pearson Education Limited 2014
- C++ Programming- From Problem Analysis to Program Design - D.S. MALIK - 2018



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Computer Engineering Department
Islamic University of Gaza

Dr. Ayman Maliha
amaliha@iugaza.edu.ps
Homepage: http://site.iugaza.edu.ps/amaliha/

ECOM 2403
Computer Programming
Fall 2020-2021



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.



Chapter 7
Single-Dimensional Arrays and C-Strings

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Opening Problem

- Read one hundred numbers
- Compute their average
- Find out how many numbers are above the average.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Introduction

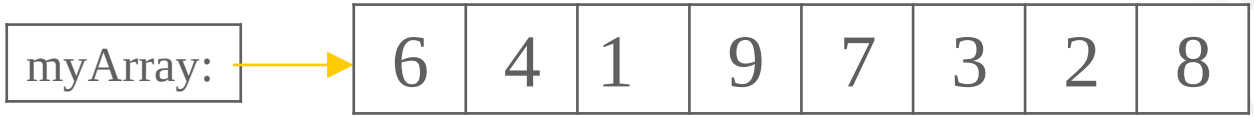
- Simple data type: variables of these types can store only one value at a time
- Structured data type: a data type in which each data item is a collection of other data items



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Arrays

- Array:
  - A collection of a fixed number of components, all of the same data type
- One-dimensional array:
  - Components are arranged in a list form



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Declaring Array Variables

```
datatype arrayName[arraySize];
```

Example:

```
double myList[10];
```

- When an array is created, its elements are assigned with arbitrary values.
- C++ requires array size to be a constant expression.

```
int size = 4;
double myList[size]; // Wrong
```

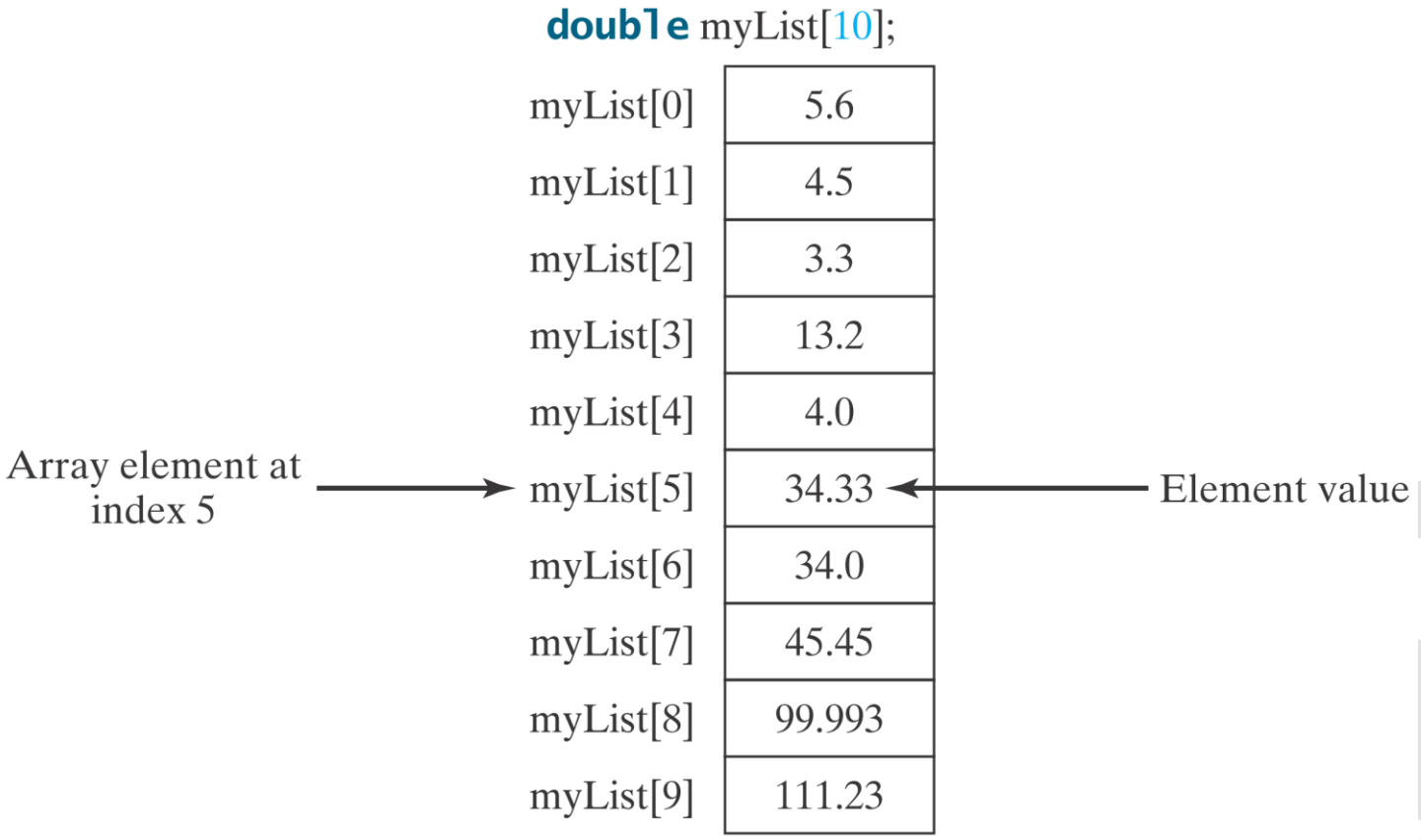
But it would be OK, if size is a constant as follow:

```
const int size = 4;
double myList[size]; // Correct
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Arrays

Array is a data structure that represents a collection of the same types of data.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Accessing Array Components

- General syntax:

```
arrayName[index];
```

- Value of the index is the position of the item in the array
- Array index always starts at 0
- []: array subscripting operator
- After an array is created, an indexed variable can be used in the same way as a regular variable.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Accessing Array Components (cont'd.)

```
int list[10];
```

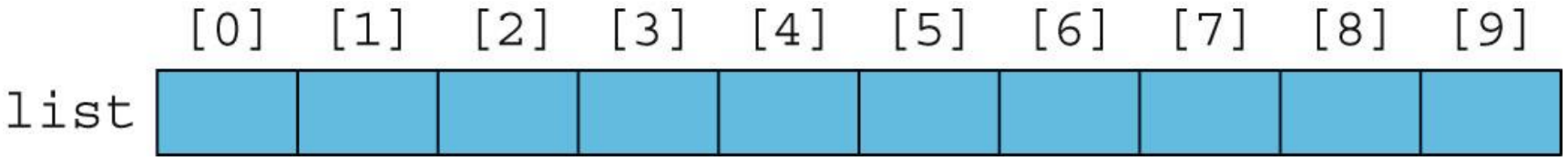


FIGURE 8-3 Array list

D.S. MALIK Book

```
list[5] = 34;
```

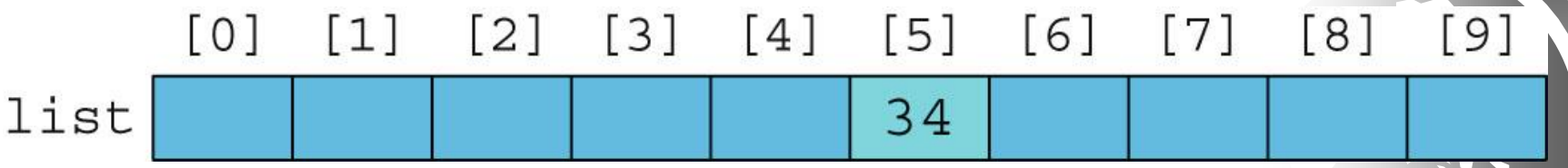


FIGURE 8-4 Array list after execution of the statement list[5]= 34;

D.S. MALIK Book

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Accessing Array Components (cont'd.)

```
list[3] = 10;
list[6] = 35;
list[5] = list[3] + list[6];
```

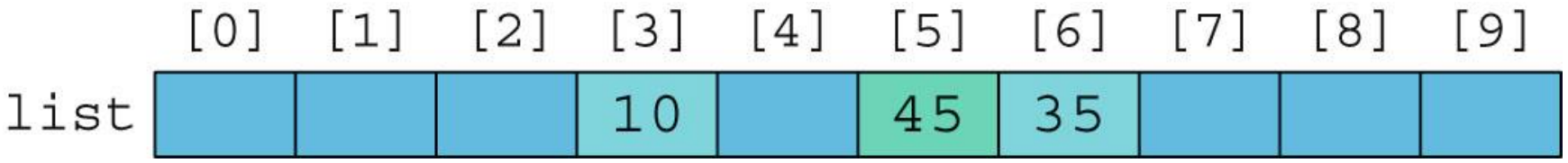


FIGURE 8-5 Array `list` after execution of the statements `list[3]= 10;`, `list[6]= 35;`, and `list[5] = list[3] + list[6];`

D.S. MALIK Book

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

No Bound Checking

- C++ does not check array's boundary.
  - So, accessing array elements beyond the boundary does not does cause syntax errors
  - But the operating system might report a memory access violation.

(e.g., myList[-1] and myList[11])



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Array Initializers

Declaring, creating, initializing in one step:

```
dataType arrayName[arraySize] = {value0, value1,
..., valuek};
```

```
double myList[4] = {1.9, 2.9, 3.4, 3.5};
```

This shorthand notation is equivalent to the following statements:

```
double myList[4];
myList[0] = 1.9;
myList[1] = 2.9;
myList[2] = 3.4;
myList[3] = 3.5;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

CAUTION

- ✓ Using the shorthand notation, you have to declare, create, and initialize the array all in one statement.
- ✓ Splitting it would cause a syntax error.
- ✓ For example, the following is wrong:

```
double myList[4];
myList = {1.9, 2.9, 3.4, 3.5};
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Implicit Size

- C++ allows you to omit the array size when declaring and creating an array using an initializer.

```
double myList[] = {1.9, 2.9, 3.4, 3.5};
```

- C++ automatically figures out how many elements are in the array.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Partial Initialization

- C++ allows you to initialize a part of the array.
- For example, the following statement assigns values 1.9, 2.9 to the first two elements of the array.

```
double myList[4] = {1.9, 2.9};
```

- The other two elements will be set to zero.
- Note that if an array is declared, but not initialized, all its elements will contain “garbage”, like all other local variables.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

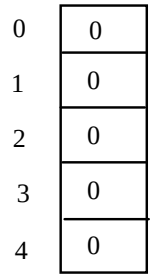
animation

Trace Program with Arrays

Declare array variable values, create an array, and assign its reference to values

```
int main()
{
    int values[5]={0};
    for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
}
```

After the array is created





© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Program with Arrays

i becomes 1

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Program with Arrays

i (=1) is less than 5

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

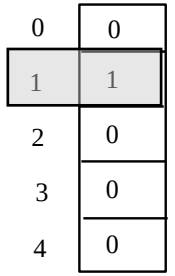
animation

Trace Program with Arrays

After this line is executed, value[1] is 1

```
int main()
{
     int values[5]=    ;
     for (int i = 1   i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the first iteration



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Program with Arrays

After i++, i becomes 2

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the first iteration

0	0
1	1
2	0
3	0
4	0



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Program with Arrays

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

i (= 2) is less than 5

After the first iteration

0	0
1	1
2	0
3	0
4	0



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

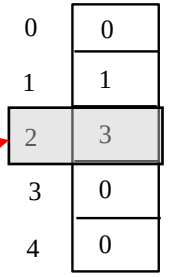
animation

Trace Program with Arrays

After this line is executed, values[2] is 3 (2 + 1)

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the second iteration



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Program with Arrays

After this, i becomes 3.

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Program with Arrays

i (=3) is still less than 5.

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

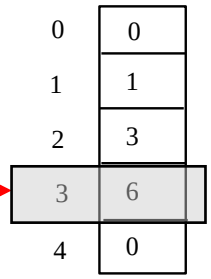
animation

Trace Program with Arrays

After this line, values[3] becomes 6 (3 + 3)

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the third iteration



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Program with Arrays

After this, i becomes 4

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Program with Arrays

i (=4) is still less than 5

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Program with Arrays

After this, values[4] becomes 10 (4 + 6)

```
int main()
{
     int values[5]=    };
     for (int i = 1    i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Program with Arrays

After i++, i becomes 5

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Program with Arrays

i (=5) < 5 is false. Exit the loop

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = i + values[i-1];
     }
     values[0] = values[1] + values[4];
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace Program with Arrays

After this line, values[0] is 11 (1 + 10)

```
int main()
{
     int values[5]={0};
     for (int i = 1; i < 5; i++)
    {
       values[i] = I + values[i-1];
     }
    values[0] = values[1] + values[4];
}
```

0	11
1	1
2	3
3	6
4	10



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Initializing arrays with random values

The following loop initializes the array myList with random values between 0 and 99:

```
for (int i = 0; i < ARRAY_SIZE; i++)
{
  myList[i] = rand() % 100;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Printing arrays

To print an array, you have to print each element in the array using a loop like the following:

```
for (int i = 0; i < ARRAY_SIZE; i++)
{
  cout << myList[i] << " ";
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Copying Arrays

- Can you copy array using a syntax like this?

```
list = myList;
```

- This is not allowed in C++.
- You have to copy individual elements from one array to the other as follows:

```
for (int i = 0; i < ARRAY_SIZE; i++)
{
  list[i] = myList[i];
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Summing All Elements

- Use a variable named total to store the sum.
- Initially total is 0.
- Add each element in the array to total using a loop like this:

```
double total = 0;
for (int i = 0; i < ARRAY_SIZE; i++)
{
  total += myList[i];
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Finding the Largest Element

- Use a variable named `max` to store the largest element.
- Initially `max` is `myList[0]`.
- To find the largest element, compare each element in `myList` with `max`
- Update `max` if the element is greater than `max`.

```
double max = myList[0];
for (int i = 1; i < ARRAY_SIZE; i++)
{
  if (myList[i] > max) max = myList[i];
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Finding the smallest index of the largest element

```
double max = myList[0];
int indexOfMax = 0;
for (int i = 1; i < ARRAY_SIZE; i++)
{
  if (myList[i] > max)
  {
    max = myList[i];
    indexOfMax = i;
  }
}
```

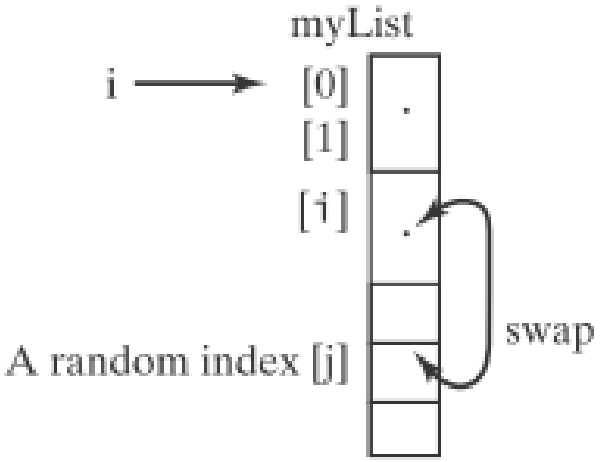
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Random Shuffling

```
srand(time(0));

for (int i=0; i< ARRAY_SIZE-1; i++)
{
  // Generate an index randomly
  int index = rand() % ARRAY_SIZE;

  double temp = myList[i];
  myList[i] = myList[index];
  myList[index] = temp;
}
```



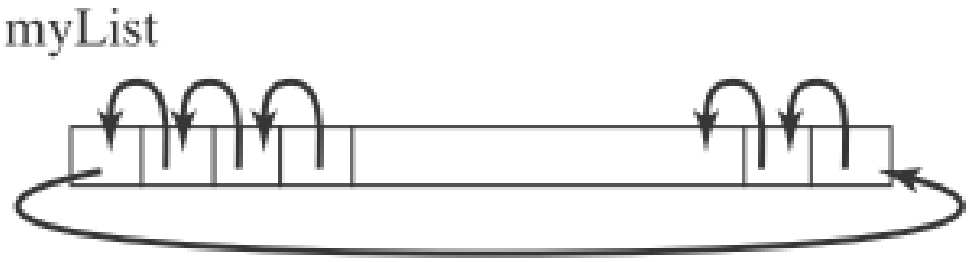
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Shifting Elements

```
double temp = myList[0]; // Retain the first element

// Shift elements left
for (int i = 1; i < ARRAY_SIZE; i++)
{
   myList[i - 1] = myList[i];
}

// Move the first element to fill in the last position
myList[ARRAY_SIZE - 1] = temp;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Foreach Loops

```
for (double e: myList) {
  cout << e << endl;
}
```

C++11: Foreach loops are defined in C++11



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

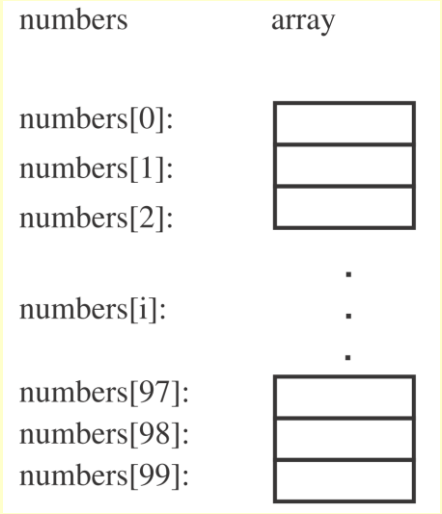
Analyze Numbers

Read one hundred numbers, compute their average, and find out how many numbers are above the average.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6   const int NUMBER_OF_ELEMENTS = 3; // For simplicity, use 3 (not 100) for demo
7   double numbers[NUMBER_OF_ELEMENTS];
8   double sum = 0;
9
10  for (int i = 0; i < NUMBER_OF_ELEMENTS; i++)
11  {
12   cout << "Enter a new number: ";
13   cin >> numbers[i];
14   sum += numbers[i];
15  }
16
17  double average = sum / NUMBER_OF_ELEMENTS;
18
19  int count = 0; // The number of elements above average
20  for (int i = 0; i < NUMBER_OF_ELEMENTS; i++)
21   if (numbers[i] > average)
22     count++;
23
24  cout << "Average is " << average << endl;
25  cout << "Number of elements above the average " << count << endl;
26
27  return 0;
28 }
```



AnalyzeNumbers

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

supplement

Problem: Lotto Numbers

The problem is to write a program that checks if all the input numbers cover 1 to 99.

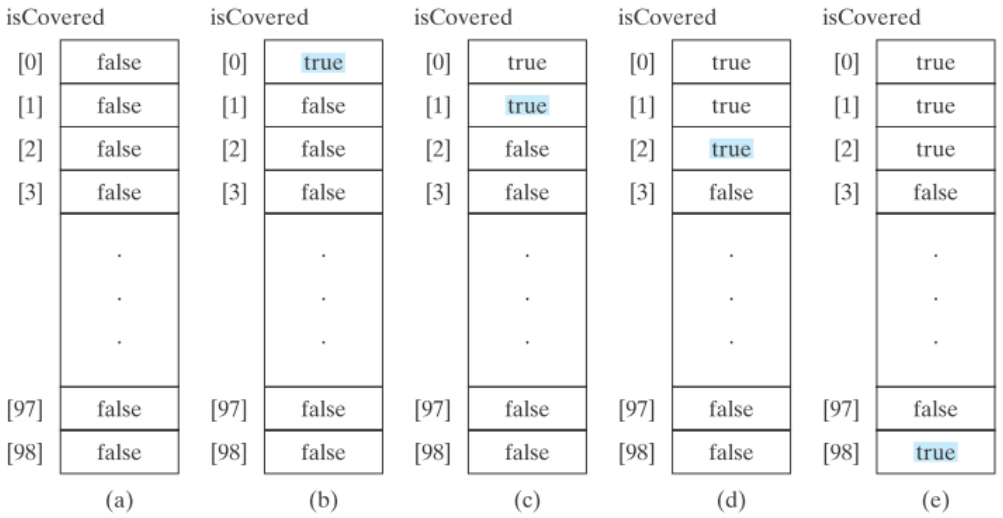
Write a program that reads the ticket numbers from a file and checks whether all numbers are covered. Assume the last number in the file is 0.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

If the file contains the numbers

```
80 3 87 62 30 90 10 21 46 27
12 40 83 9 39 88 95 59 20 37
80 40 87 67 31 90 11 24 56 77
11 48 51 42 8 74 1 41 36 53
52 82 16 72 19 70 44 56 29 33
54 64 99 14 23 22 94 79 55 2
60 86 34 4 31 63 84 89 7 78
43 93 97 45 25 38 28 26 85 49
47 65 57 67 73 69 32 71 24 66
92 98 96 77 6 75 17 61 58 13
35 81 18 15 5 68 91 50 76
0
```



Your program should display

`The tickets cover all numbers`

Suppose the file contains the numbers

```
11 48 51 42 8 74 1 41 36 53
52 82 16 72 19 70 44 56 29 33
0
```

Your program should display

`The tickets don't cover all numbers`



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    bool isCovered[99];
7    int number; // number read from a file
8
9    // Initialize the array
10   for (int i = 0; i < 99; i++)
11     isCovered[i] = false;
12
13   // Read each number and mark its corresponding element covered
14   cin >> number;
15   while (number != 0)
16   {
17     isCovered[number - 1] = true;
18     cin >> number;
19   }
20
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
21    // Check if all covered
22    bool allCovered = true; // Assume all covered initially
23    for (int i = 0; i < 99; i++)
24    if (!isCovered[i])
25    {
26      allCovered = false; // Find one number not covered
27      break;
28    }
29
30    // Display result
31    if (allCovered)
32      cout << "The tickets cover all numbers" << endl;
33    else
34      cout << "The tickets don't cover all numbers" << endl;
35
36    return 0;
37  }
```

LottoNumbers

```
g++ LottoNumbers.cpp -o LottoNumbers.exe
LottoNumbers.exe < LottoNumbers.txt
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Problem: Deck of Cards

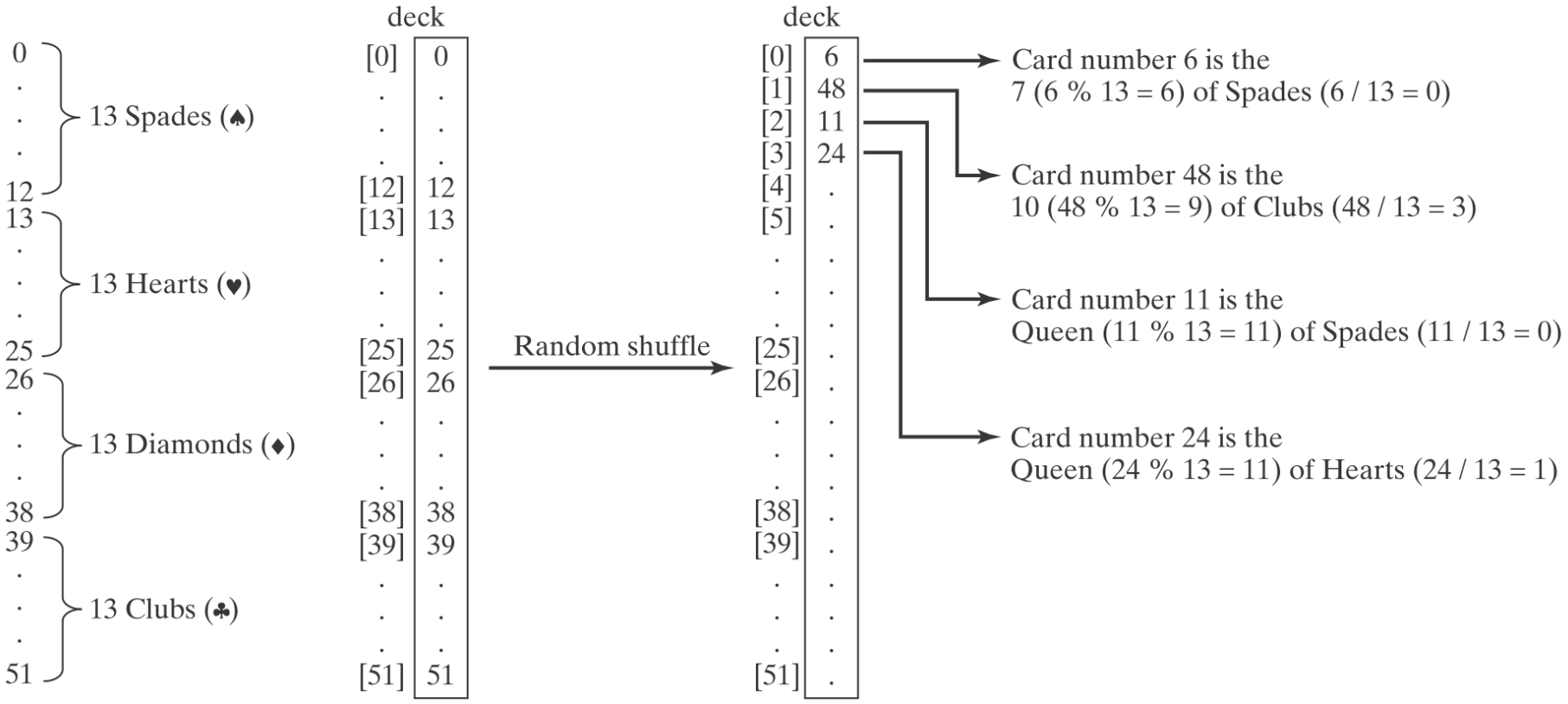
- The problem is to write a program that picks four cards randomly from a deck of 52 cards.
- All the cards can be represented using an array named deck, filled with initial values 0 to 51, as follows:

```
int deck[52];
// Initialize cards
for (int i = 0; i < NUMBER_OF_CARDS; i++)
  deck[i] = i;
```



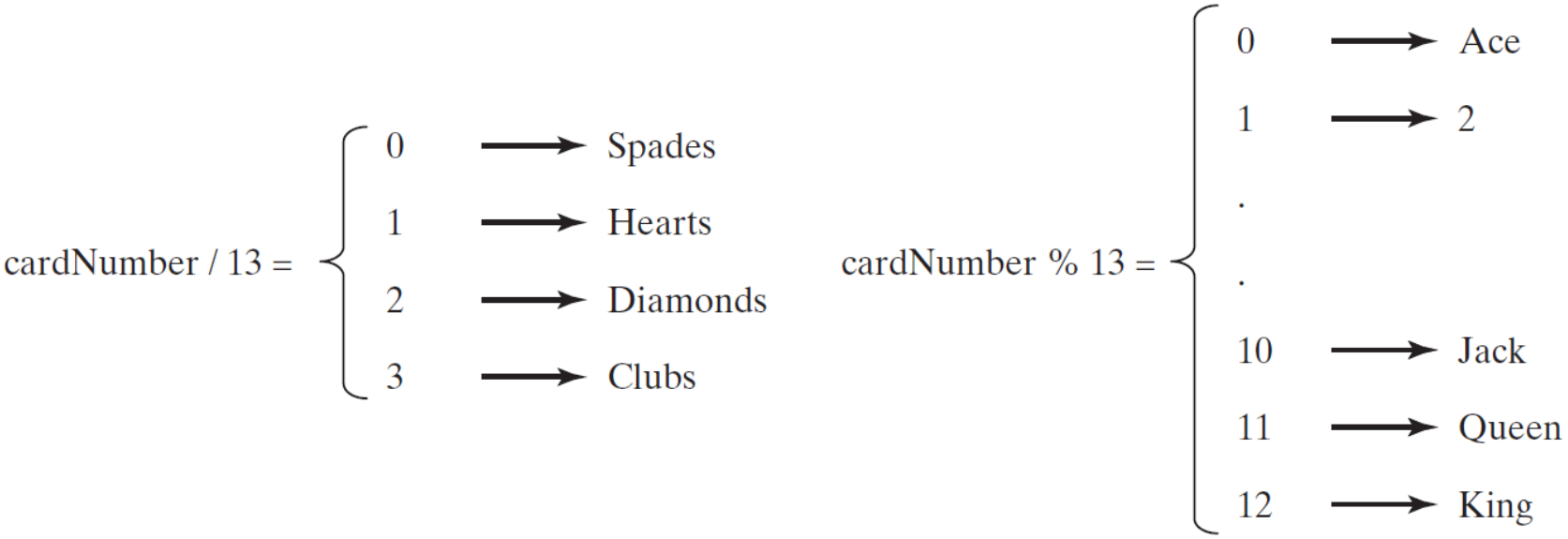
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Problem: Deck of Cards, cont.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Problem: Deck of Cards, cont.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  #include <ctime>
3  #include <cstdlib>
4  #include <string>
5  using namespace std;
6
7  int main()
8  {
9    const int NUMBER_OF_CARDS = 52;
10   int deck[NUMBER_OF_CARDS];
11   string suits[] = {"Spades", "Hearts", "Diamonds", "Clubs"};
12   string ranks[] = {"Ace", "2", "3", "4", "5", "6", "7", "8",
13                     "9", "10", "Jack", "Queen", "King"};
14
15   // Initialize cards
16   for (int i = 0; i < NUMBER_OF_CARDS; i++)
17     deck[i] = i;
18
19   // Shuffle the cards
20   srand(time(0));
21   for (int i = 0; i < NUMBER_OF_CARDS; i++)
22   {
23     // Generate an index randomly
24     int index = rand() % NUMBER_OF_CARDS;
25     int temp = deck[i];
26     deck[i] = deck[index];
27     deck[index] = temp;
28   }
```

```
29
30    // Display the first four cards
31    for (int i = 0; i < 4; i++)
32    {
33      string suit = suits[deck[i] / 13];
34      string rank = ranks[deck[i] % 13];
35      cout << "Card number " << deck[i] << ": "
36           << rank << " of " << suit << endl;
37    }
38
39    return 0;
40  }
```

```
Card number 6: 7 of Spades
Card number 48: 10 of Clubs
Card number 11: Queen of Spades
Card number 24: Queen of Hearts
```

DeckOfCards

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Passing Arrays to Functions

- Just as you can pass single values to a function, you can also pass an entire array to a function.
- When an array argument is passed to a function, its starting address is passed to the array parameter in the function.
- Both parameter and argument refer to the same array.

PassArrayDemo



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  void printArray(int list[], int arraySize); // Function prototype
5
6  int main()
7  {
8    int numbers[6] = {1, 4, 3, 6, 8, 9};
9    printArray(numbers, 6); // Invoke the function
10
11   return 0;
12  }
13
14  void printArray(int list[], int arraySize)
15  {
16   for (int i = 0; i < arraySize; i++)
17   {
18      cout << list[i] << " ";
19   }
20  }
```

```
1 4 3 6 8
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Passing Size along with Array

- Normally when you pass an array to a function, you should also pass its size in another argument.
- So the function knows how many elements are in the array.
- Otherwise, you will have to hard code this into the function or declare it in a global variable.
- Neither is flexible or robust.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Pass-by-Value

- Passing an array variable means that the starting address of the array is passed to the formal parameter by value.
- The parameter inside the function references to the same array that is passed to the function.
- No new arrays are created.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  void m(int, int []);
5
6  int main()
7  {
8    int x = 1; // x represents an int value
9    int y[10] = {0}; // y represents an array of int values
10
11   m(x, y); // Invoke m with arguments x and y
12
13   cout << "x is " << x << endl;
14   cout << "y[0] is " << y[0] << endl;
15
16   return 0;
17  }
18
19  void m(int number, int numbers[])
20  {
21   number = 1001; // Assign a new value to number
22   numbers[0] = 5555; // Assign a new value to numbers[0]
23  }
```

```
x is 1
y[0] is 5555
```

EffectOfPassArrayDemo

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

`const` Parameters

- Passing arrays by reference makes sense for performance reasons.
- If an array is passed by value
  - All its elements must be copied into a new array.
  - Additional time and memory space.
- However, passing arrays by its reference
  - Could lead to errors if the function changes the array accidentally.
- To prevent it from happening
  - Use `const` keyword before the array parameter
  - The compiler will report errors if the code in the function attempts to modify the array.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  void p(const int list[], int arraySize)
5  {
6   // Modify array accidentally
7   list[0] = 100; // Compile error!
8  }
9
10  int main()
11  {
12   int numbers[5] = {1, 4, 3, 6, 8};
13   p(numbers, 5);
14
15   return 0;
16  }
```

ConstArrayDemo

Compiled using Visual C++ 2012

```
error C3892: "list": you cannot assign to a variable that is const
```

Compiled using GNU C++

```
ConstArrayDemo.cpp:7: error: assignment of read-only location
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Returning an Array from a Function

To return an array from a function, pass it as a parameter in a function.

- Returning an array from a function using a similar syntax is not allowed.
- If you may attempt to declare a function that returns a new array that is a reversal of an array as follows:

```
// Return the reversal of list
int[] reverse(const int list[], int size)
```

This is not allowed in C++.

- How can we do that !!!

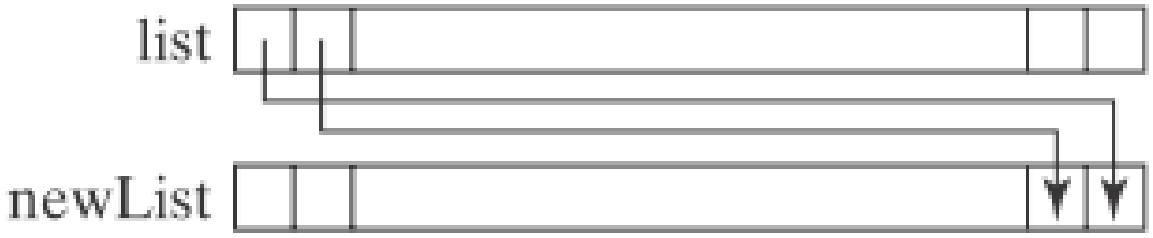


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Modifying Arrays in Functions, cont.

However, you can circumvent this restriction by passing two array arguments in the function, as follows:

```
// newList is the reversal of list
void reverse(const int list[], int newList[], int size)
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  // newList is the reversal of list
5  void reverse(const int list[], int newList[], int size)
6  {
7   for (int i = 0, j = size - 1; i < size; i++, j--)
8   {
9     newList[j] = list[i];
10  }
11 }
12
13  void printArray(const int list[], int size)
14  {
15   for (int i = 0; i < size; i++)
16     cout << list[i] << " ";
17  }
18
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
19  int main()
20  {
21    const int SIZE = 6;
22    int list[] = {1, 2, 3, 4, 5, 6};
23    int newList[SIZE];
24
25    reverse(list, newList, SIZE);
26
27    cout << "The original array: ";
28    printArray(list, SIZE);
29    cout << endl;
30
31    cout << "The reversed array: ";
32    printArray(newList, SIZE);
33    cout << endl;
34
35    return 0;
36  }
```

```
The original array: 1 2 3 4 5 6
The reversed array: 6 5 4 3 2 1
```

ReverseArray

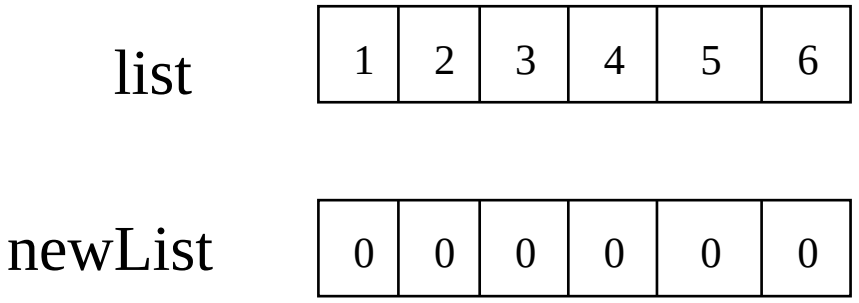
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace the reverse Function

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
     newList[j] = list[i];
   }
}
```





© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

i = 0 and j = 5

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
   {
      newList[j] = list[i];
   }
}
```

list	1	2	3	4	5	6

newList	0	0	0	0	0	0



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
     newList[j] = list[i];
   }
}
```

i (= 0) is less than 6

list	1	2	3	4	5	6

newList	0	0	0	0	0	0



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

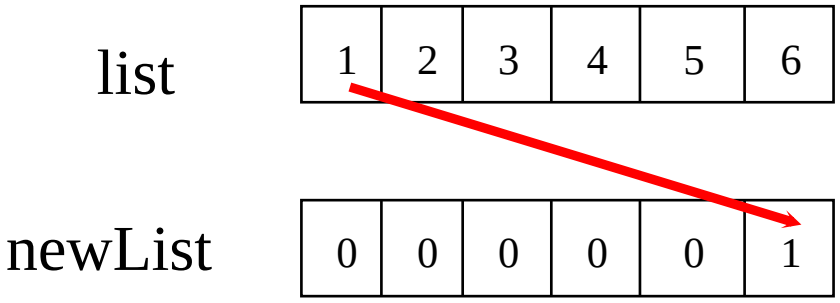
animation

Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);


void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
    newList[j] = list[i];
    }
}
```

i = 0 and j = 5
Assign list[0] to result[5]





© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
     newList[j] = list[i];
   }
}
```

After this, i becomes 1 and j becomes 4

list	1	2	3	4	5	6

newList	0	0	0	0	0	1



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

i (=1) is less than 6

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
     newList[j] = list[i];
   }
}
```

list	1	2	3	4	5	6

newList	0	0	0	0	0	1



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

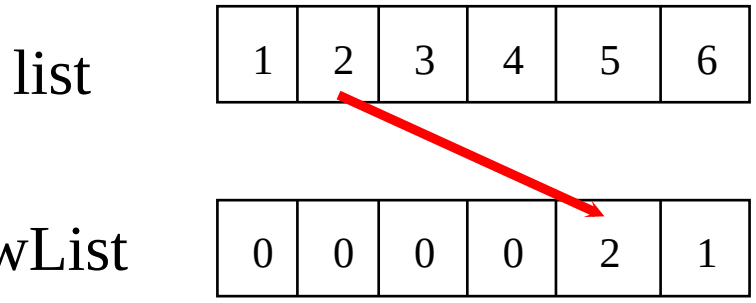
animation

Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
    newList[j] = list[i];
   }
}
```

i = 1 and j = 4
Assign list[1] to result[4]



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
     newList[j] = list[i];
   }
}
```

After this, i becomes 2 and j becomes 3

list	1	2	3	4	5	6

newList	0	0	0	0	2	1



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

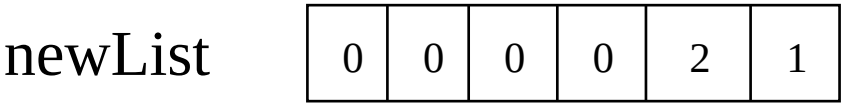
Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
     newList[j] = list[i];
   }
}
```

i (=2) is still less than 6







© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

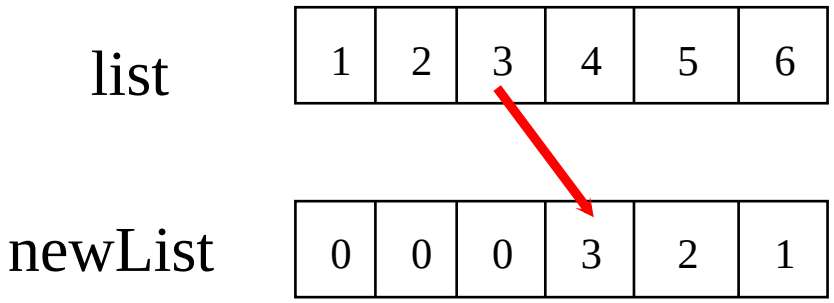
animation

Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
    newList[j] = list[i];
    }
}
```

i = 2 and j = 3
Assign list[i] to result[j]



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
     newList[j] = list[i];
   }
}
```

After this, i becomes 3 and j becomes 2

list	1	2	3	4	5	6

newList	0	0	0	3	2	1



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
     newList[j] = list[i];
   }
}
```

i (=3) is still less than 6

list

1	2	3	4	5	6

newList

0	0	0	3	2	1

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

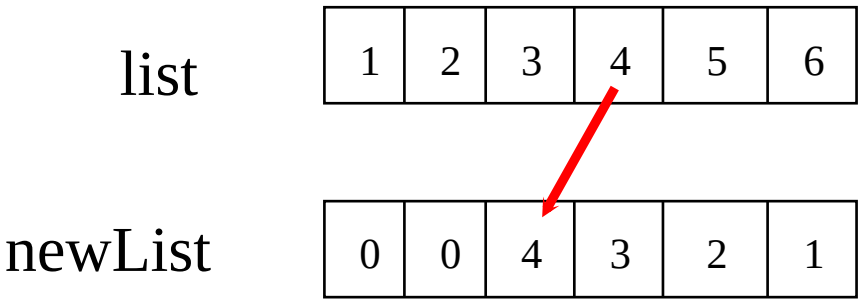
animation

Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
   {
     newList[j] = list[i];
   }
}
```

i = 3 and j = 2
Assign list[i] to result[j]



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

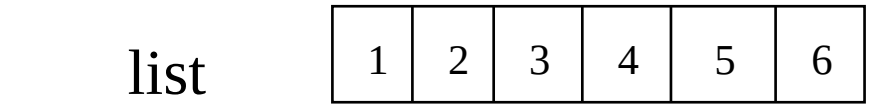
animation

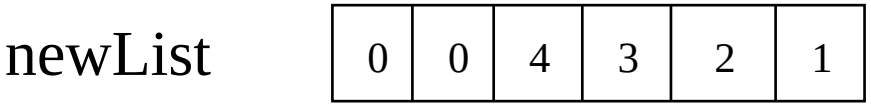
Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);

void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
     newList[j] = list[i];
   }
}
```

After this, i becomes 4 and j becomes 1





© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace the reverse Function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
     newList[j] = list[i];
   }
}
```

i (=4) is still less than 6

list

1	2	3	4	5	6

newList

0	0	4	3	2	1

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

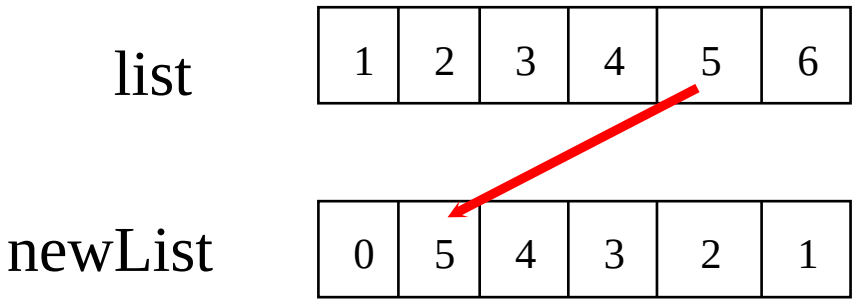
animation

Trace the reverse Function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
    newList[j] = list[i];
   }
}
```

i = 4 and j = 1
Assign list[i] to result[j]



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace the reverse Function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);

void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
   {
      newList[j] = list[i];
   }
}
```

After this, i becomes 5 and j becomes 0

list

1	2	3	4	5	6

newList

0	5	4	3	2	1

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

Trace the reverse Function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
     newList[j] = list[i];
   }
}
```

i (=5) is still less than 6





© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

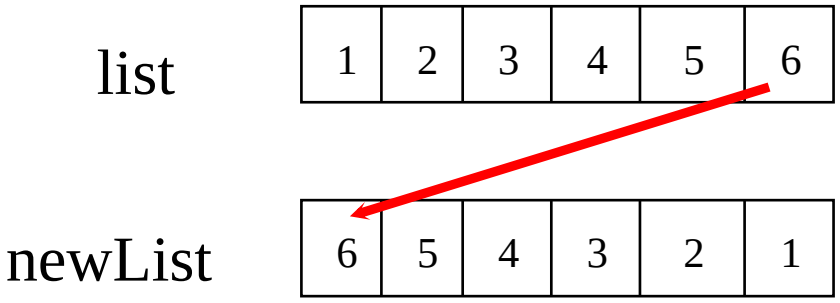
animation

Trace the reverse Function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);

void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
   {
     newList[j] = list[i];
   }
}
```

i = 5 and j = 0
Assign list[i] to result[j]



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

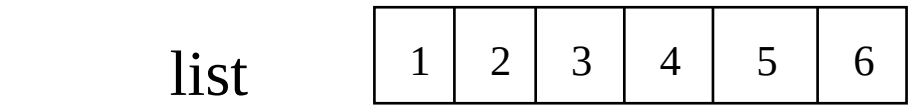
animation

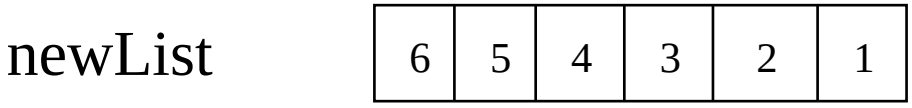
Trace the reverse Function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
   {
      newList[j] = list[i];
   }
}
```

After this, i becomes 6 and j becomes -1





© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

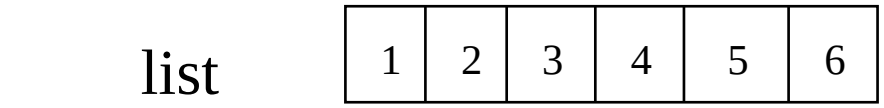
animation

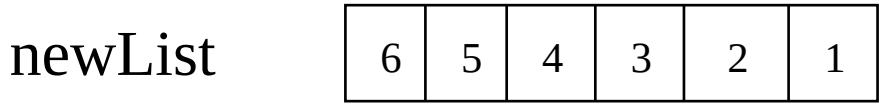
Trace the reverse function, cont.

```
int list1[] = {1, 2, 3, 4, 5, 6};
reverse(list1, list2);
```

```
void reverse(const int list[], int newList[], int size)
{
   for (int i = 0, j = size - 1; i < size; i++, j--)
  {
     newList[j] = list[i];
   }
}
```

i (=6) < 6 is false. So exit the loop.



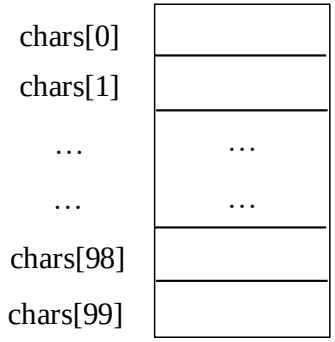


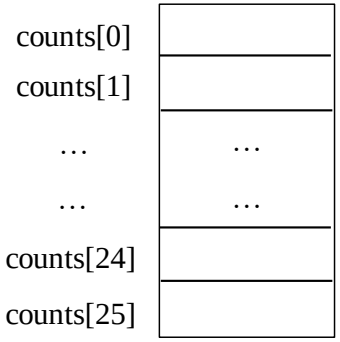


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Problem: Counting Occurrence of Each Letter

- Generate 100 lowercase letters randomly and assign to an array of characters.
- Count the occurrence of each letter in the array.






```
The lowercase letters are:
p y a o u n s u i b t h y g w q l b y o
x v b r i g h i x w v c g r a s p y i z
n f j v c j c a c v l a j r x r d t w q
m a y e v m k d m e m o j v k m e v t a
r m o u v d h f o o x d g i u w r i q h

The occurrences of each letter are:
6 a 3 b 4 c 4 d 3 e 2 f 4 g 4 h 6 i 4 j
2 k 2 l 6 m 2 n 6 o 2 p 3 q 6 r 2 s 3 t
4 u 8 v 4 w 4 x 5 y 1 z
```

CountLetterInArray

© Copyright 2010 by Pearson Education, Inc. All Rights Reserved.

```
 1  #include <iostream>
 2  #include <ctime>
 3  using namespace std;
 4
 5  const int NUMBER_OF_LETTERS = 26;
 6  const int NUMBER_OF_RANDOM_LETTERS = 100;
 7  void createArray(char []);
 8  void displayArray(const char []);
 9  void countLetters(const char [], int []);
10  void displayCounts(const int []);
11
12  int main()
13  {
14    // Declare and create an array
15    char chars[NUMBER_OF_RANDOM_LETTERS];
16
17    // Initialize the array with random lowercase letters
18    createArray(chars);
19
20    // Display the array
21    cout << "The lowercase letters are: " << endl;
22    displayArray(chars);
23
24    // Count the occurrences of each letter
25    int counts[NUMBER_OF_LETTERS];
26
27    // Count the occurrences of each letter
28    countLetters(chars, counts);
29
30    // Display counts
31    cout << "\nThe occurrences of each letter are: " << endl;
32    displayCounts(counts);
33
34    return 0;
35  }
```

```
37  // Create an array of characters
38  void createArray(char chars[])
39  {
40    // Create lowercase letters randomly and assign
41    // them to the array
42    srand(time(0));
43    for (int i = 0; i < NUMBER_OF_RANDOM_LETTERS; i++)
44      chars[i] = static_cast<char>('a' + rand() % ('z' - 'a' + 1));
45  }
46
47  // Display the array of characters
48  void displayArray(const char chars[])
49  {
50    // Display the characters in the array 20 on each line
51    for (int i = 0; i < NUMBER_OF_RANDOM_LETTERS; i++)
52    {
53      if ((i + 1) % 20 == 0)
54        cout << chars[i] << " " << endl;
55      else
56        cout << chars[i] << " ";
57    }
58  }
59
60  // Count the occurrences of each letter
61  void countLetters(const char chars[], int counts[])
62  {
63    // Initialize the array
64    for (int i = 0; i < NUMBER_OF_LETTERS; i++)
65      counts[i] = 0;
66
67    // For each lowercase letter in the array, count it
68    for (int i = 0; i < NUMBER_OF_RANDOM_LETTERS; i++)
69      counts[chars[i] - 'a'] ++;
70  }
```

```
72  // Display counts
73  void displayCounts(const int counts[])
74  {
75    for (int i = 0; i < NUMBER_OF_LETTERS; i++)
76    {
77      if ((i + 1) % 10 == 0)
78        cout << counts[i] << " " << static_cast<char>(i + 'a') << endl;
79      else
80        cout << counts[i] << " " << static_cast<char>(i + 'a') << " ";
81    }
82  }
```

```
The lowercase letters are:
p y a o u n s u i b t h y g w q l b y o
x v b r i g h i x w v c g r a s p y i z
n f j v c j c a c v l a j r x r d t w q
m a y e v m k d m e m o j v k m e v t a
r m o u v d h f o o x d g i u w r i q h

The occurrences of each letter are:
6 a 3 b 4 c 4 d 3 e 2 f 4 g 4 h 6 i 4 j
2 k 2 l 6 m 2 n 6 o 2 p 3 q 6 r 2 s 3 t
4 u 8 v 4 w 4 x 5 y 1 z
```



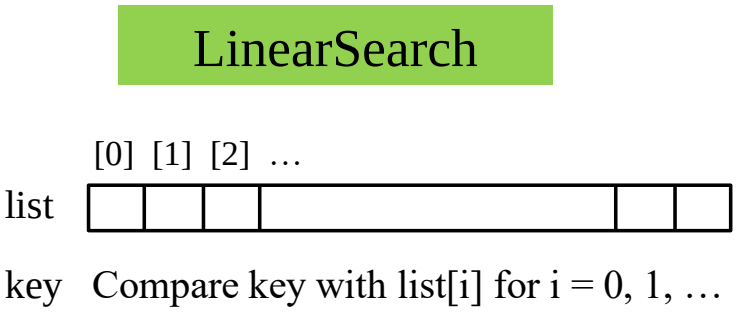
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Searching Arrays

- Searching is the process of looking for a specific element in an array
- Searching is a common task in computer programming.
- Different algorithms and data structures devoted to searching.
- Two commonly used approaches are discussed, *linear search* and *binary search*.

```
int linearSearch(const int list[], int key, int arraySize)
{
  for (int i = 0; i < arraySize; i++)
  {
    if (key == list[i])
      return i;
  }

  return -1;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Linear Search

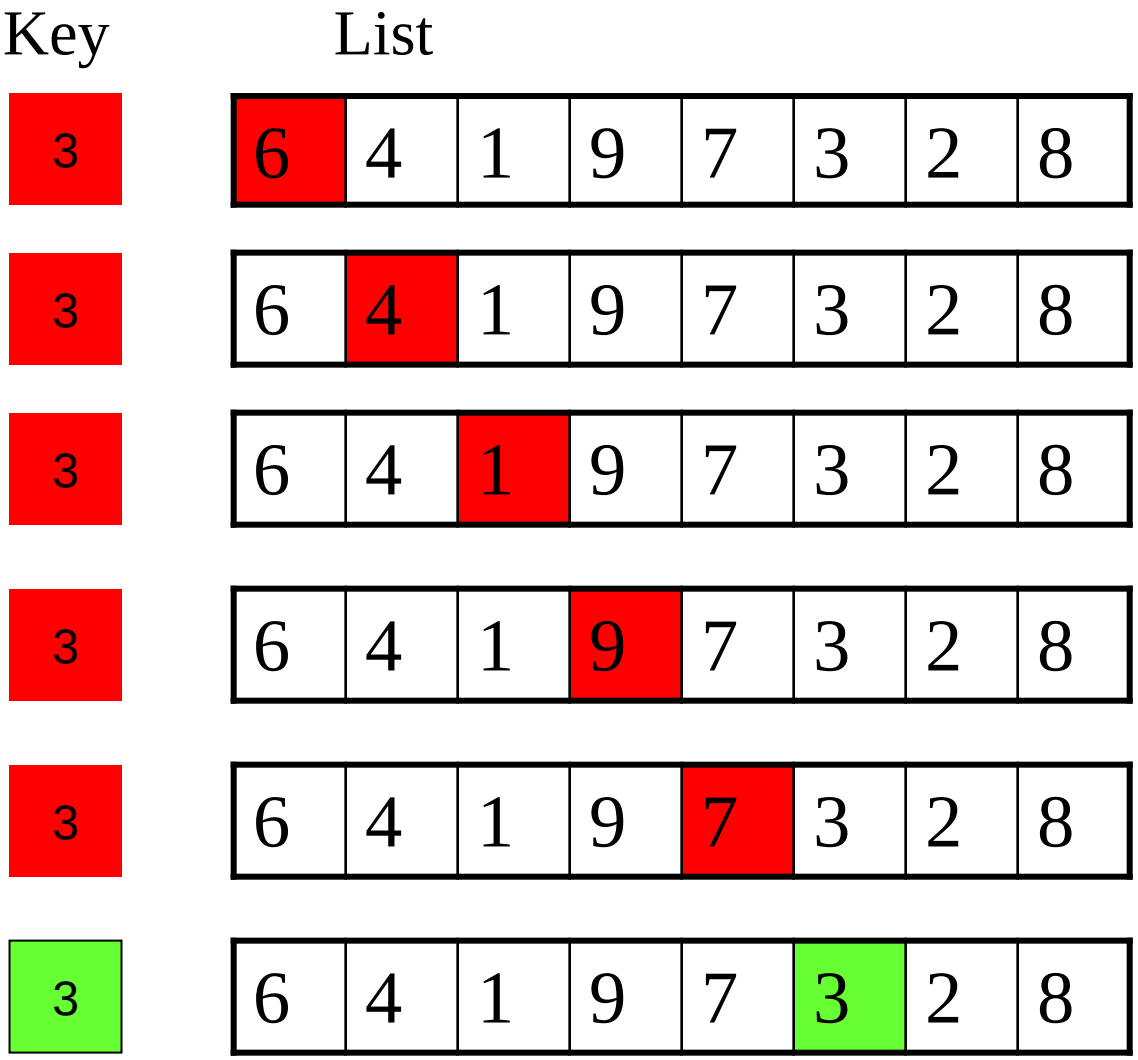
- The linear search approach compares the key element, ***key***, ***sequentially*** with each element in the array ***list***.
- The Function continues to do so until the key matches an element in the list or the list is exhausted without a match being found.
- If a match is made, the index of the element is returned.
- If no match is found, the search returns -1.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

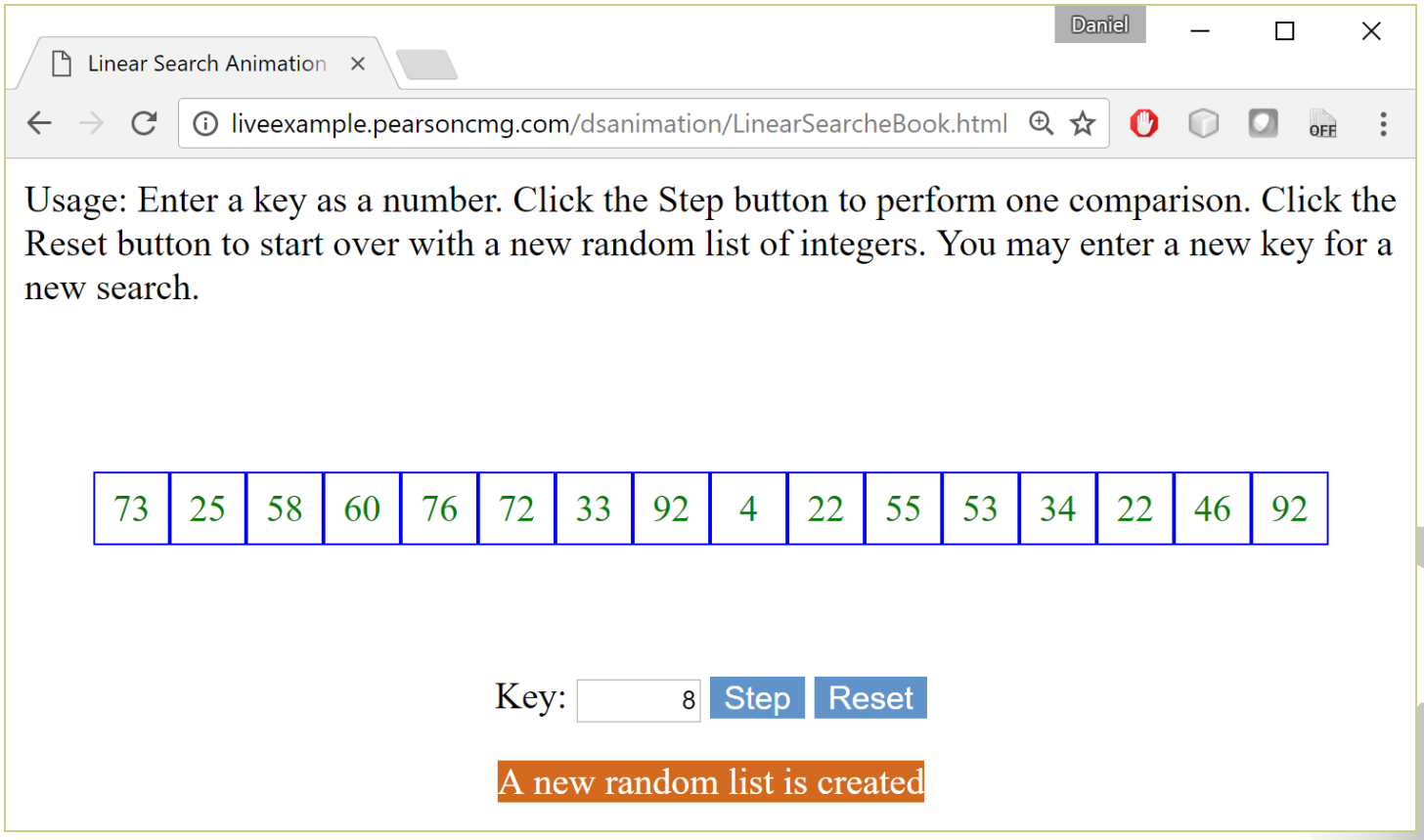
Linear Search Animation



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Linear Search Animation

liveexample.pearsoncmg.com/dsanimation/LinearSearcheBook.html

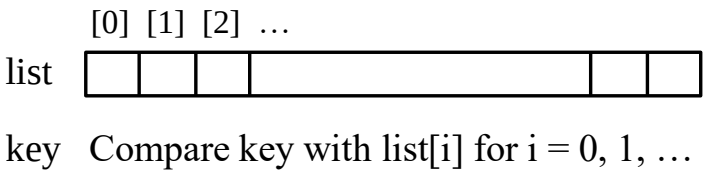


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

From Idea to Solution

```
int linearSearch(const int list[], int key, int arraySize)
{
  for (int i = 0; i < arraySize; i++)
  {
    if (key == list[i])
      return i;
  }

  return -1;
}
```



Trace the function

```
int[] list = {1, 4, 4, 2, 5, -3, 6, 2};
int i = linearSearch(list, 4);  // returns 1
int j = linearSearch(list, -4); // returns -1
int k = linearSearch(list, -3); // returns 5
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Binary Search

- For binary search to work, the elements in the array must already be ordered.
- Without loss of generality, assume that the array is in ascending order.

  e.g., 2 4 7 10 11 45 50 59 60 66 69 70 79
- The binary search first compares the key with the element in the middle of the array.



BinarySearch

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Binary Search, cont.

Consider the following three cases:

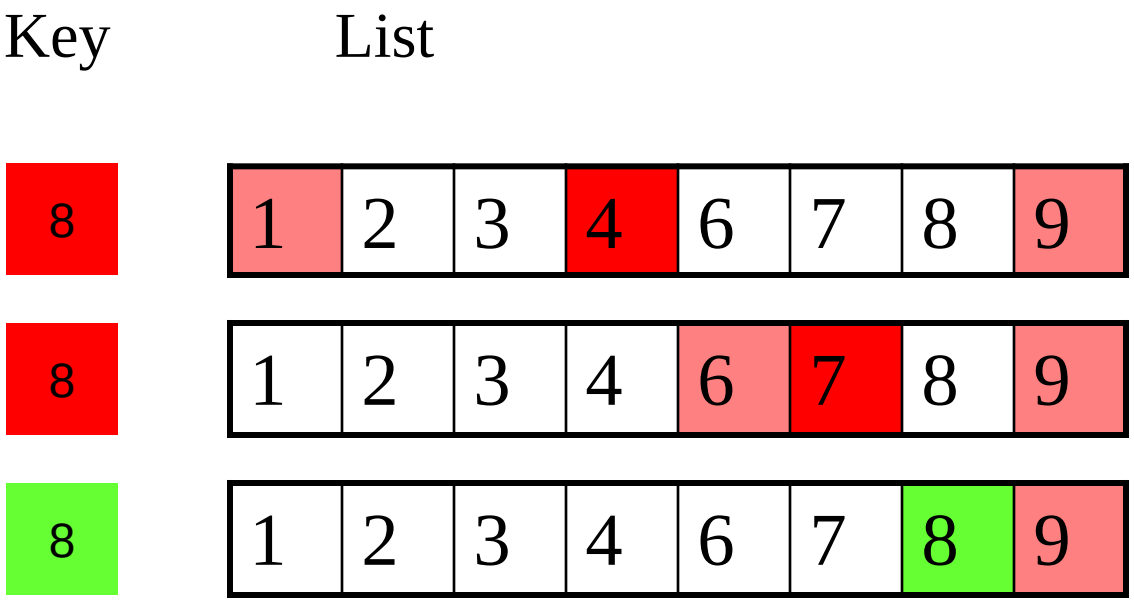
- If the key is less than the middle element, you only need to search the key in the first half of the array.
- If the key is equal to the middle element, the search ends with a match.
- If the key is greater than the middle element, you only need to search the key in the second half of the array.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

animation

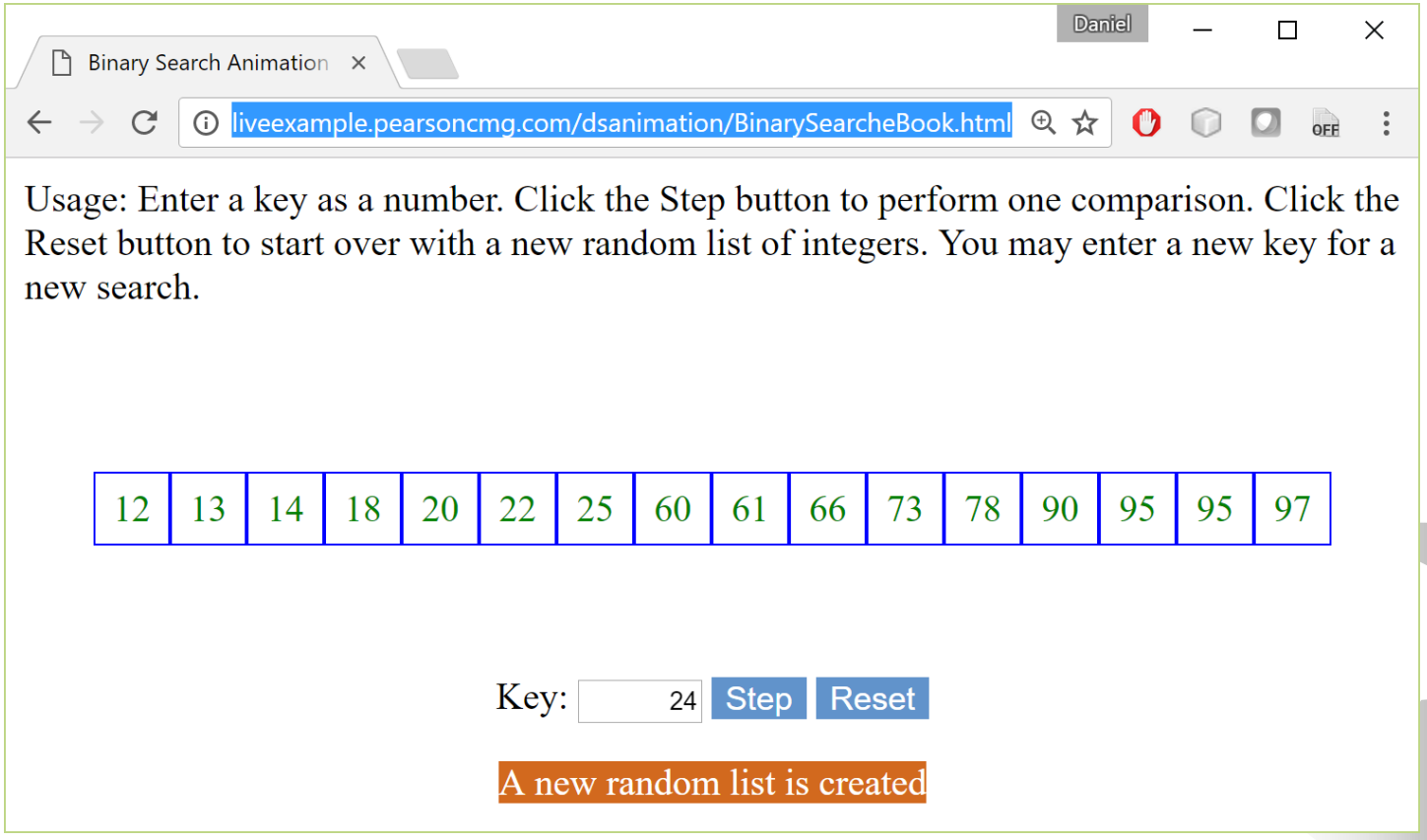
Binary Search



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

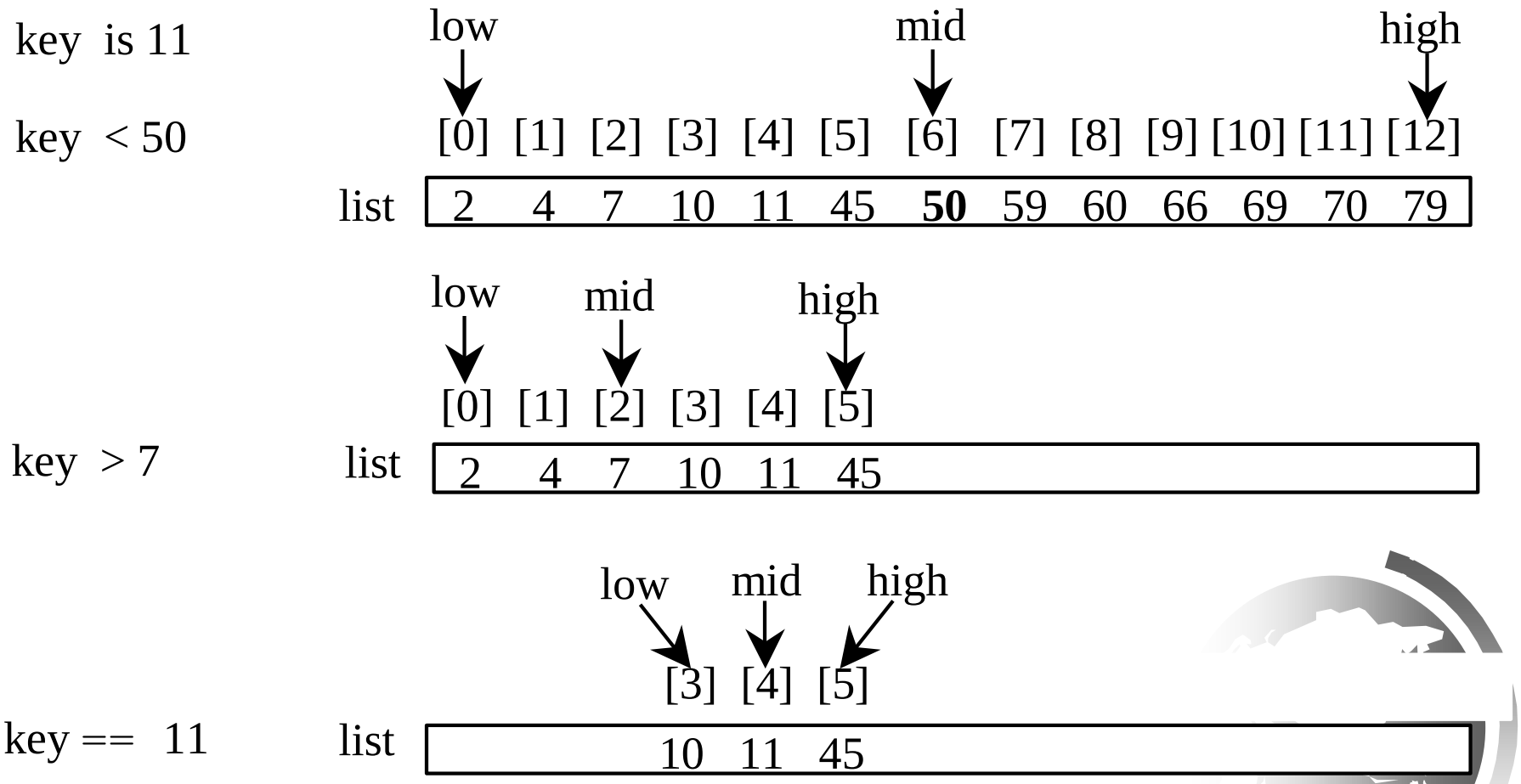
Binary Search Animation

http://liveexample.pearsoncmg.com/dsanimation/BinarySearcheBook.html



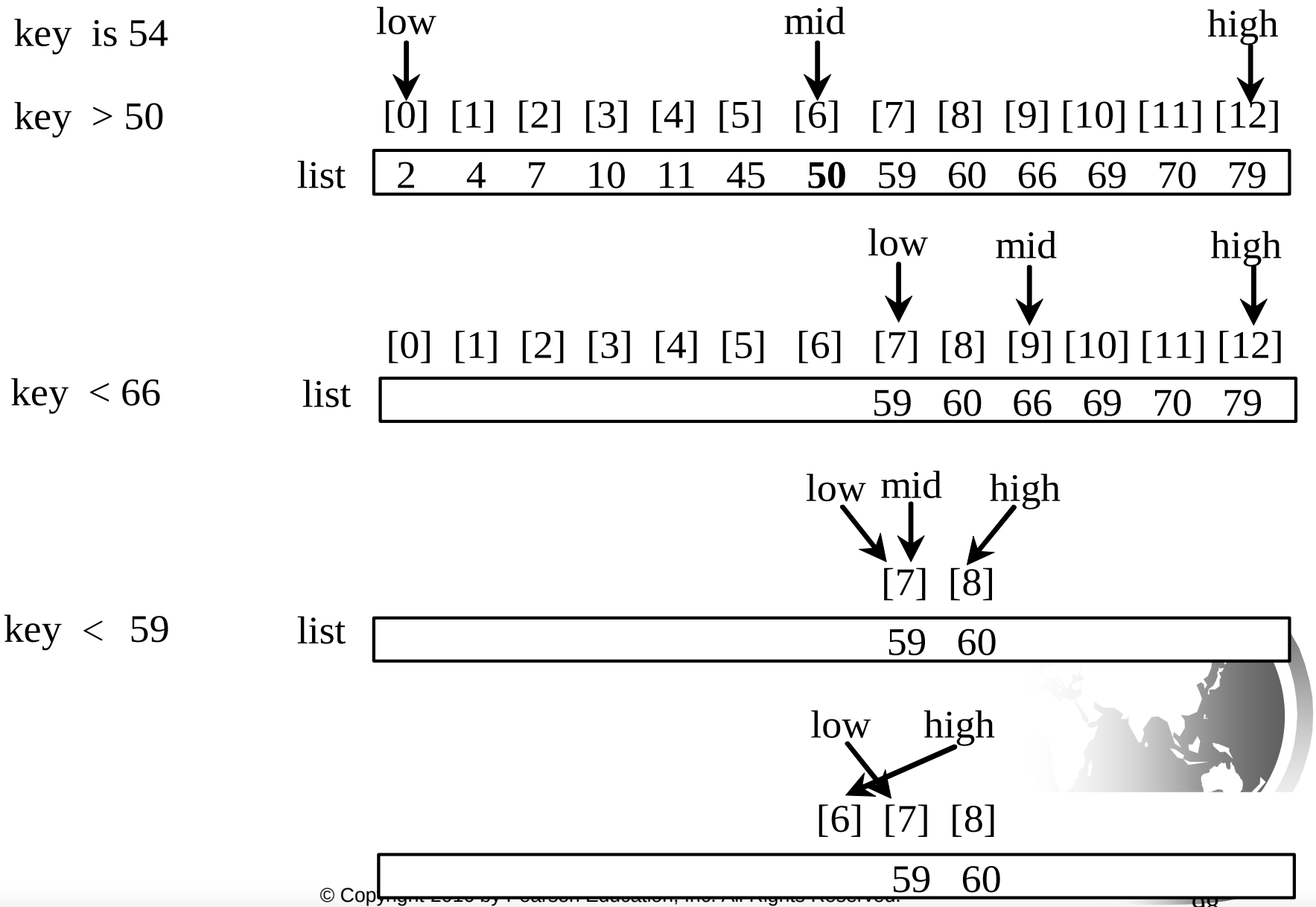
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Binary Search, cont.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Binary Search, cont.



© Copyright 2018 by Pearson Education, Inc. All Rights Reserved.

Binary Search, cont.

- The binarySearch Function returns the index of the search key if it is contained in the list.
- Otherwise, it returns –insertion point - 1.
- The insertion point is the point at which the key would be inserted into the list.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

From Idea to Solution

```
int binarySearch(const int list[], int key, int arraySize)
{
  int low = 0;
  int high = arraySize - 1;

  while (high >= low)
  {
    int mid = (low + high) / 2;
    if (key < list[mid])
      high = mid - 1;
    else if (key == list[mid])
      return mid;
    else
      low = mid + 1;
  }

  return -low - 1;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Sorting Arrays

- ✓ Sorting, like searching, is also a common task in computer programming.
- ✓ It would be used, for instance, if you wanted to display the grades from in alphabetical order.
- ✓ Many different algorithms have been developed for sorting.
- ✓ This section introduces selection sort.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

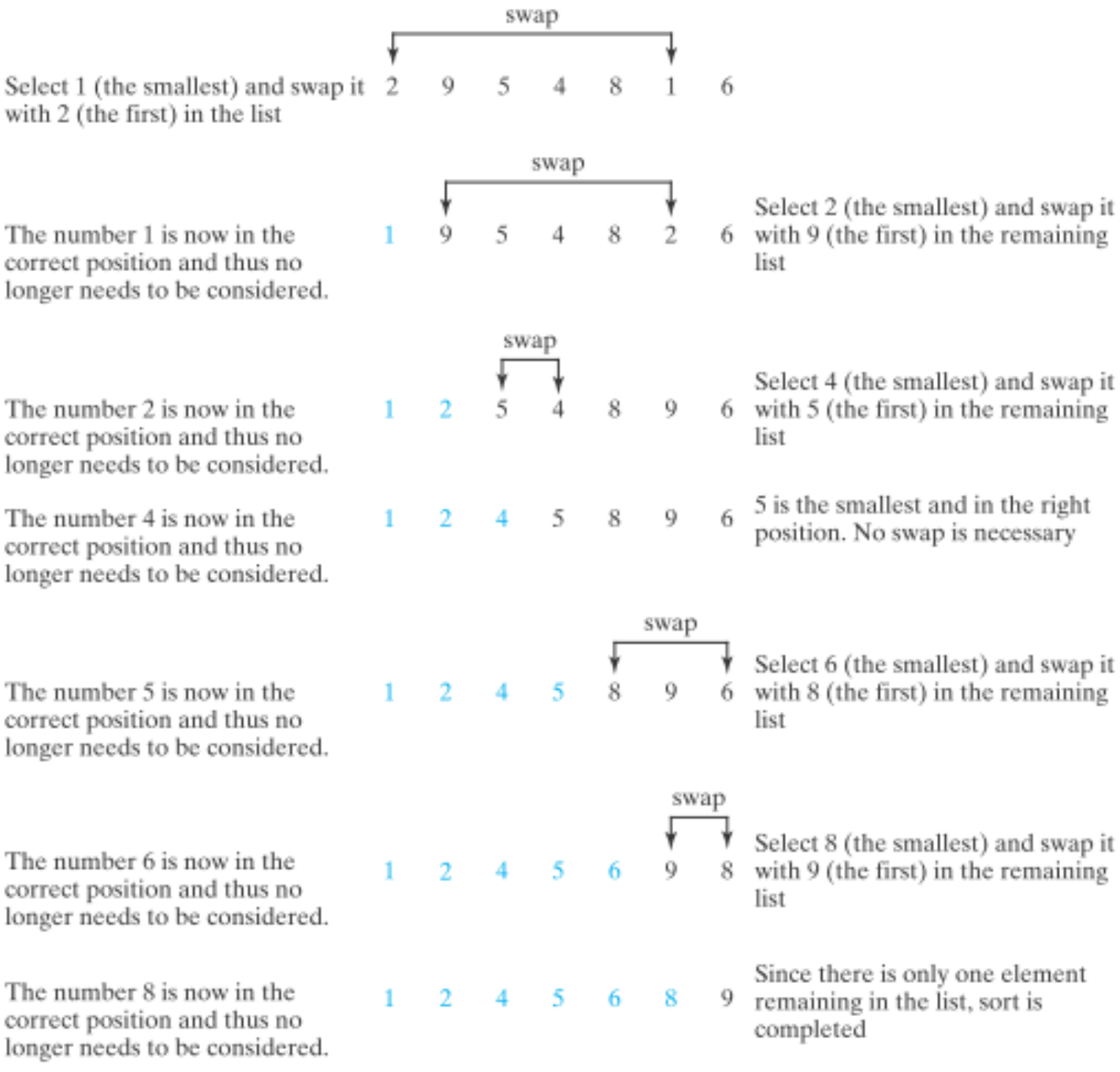
Selection Sort

- ✓ Selection sort finds the smallest number in the list and swaps it with the first.
- ✓ It then finds the smallest number remaining and swaps it with the next to first, and so on.
- ✓ The process continues until only a single number remains



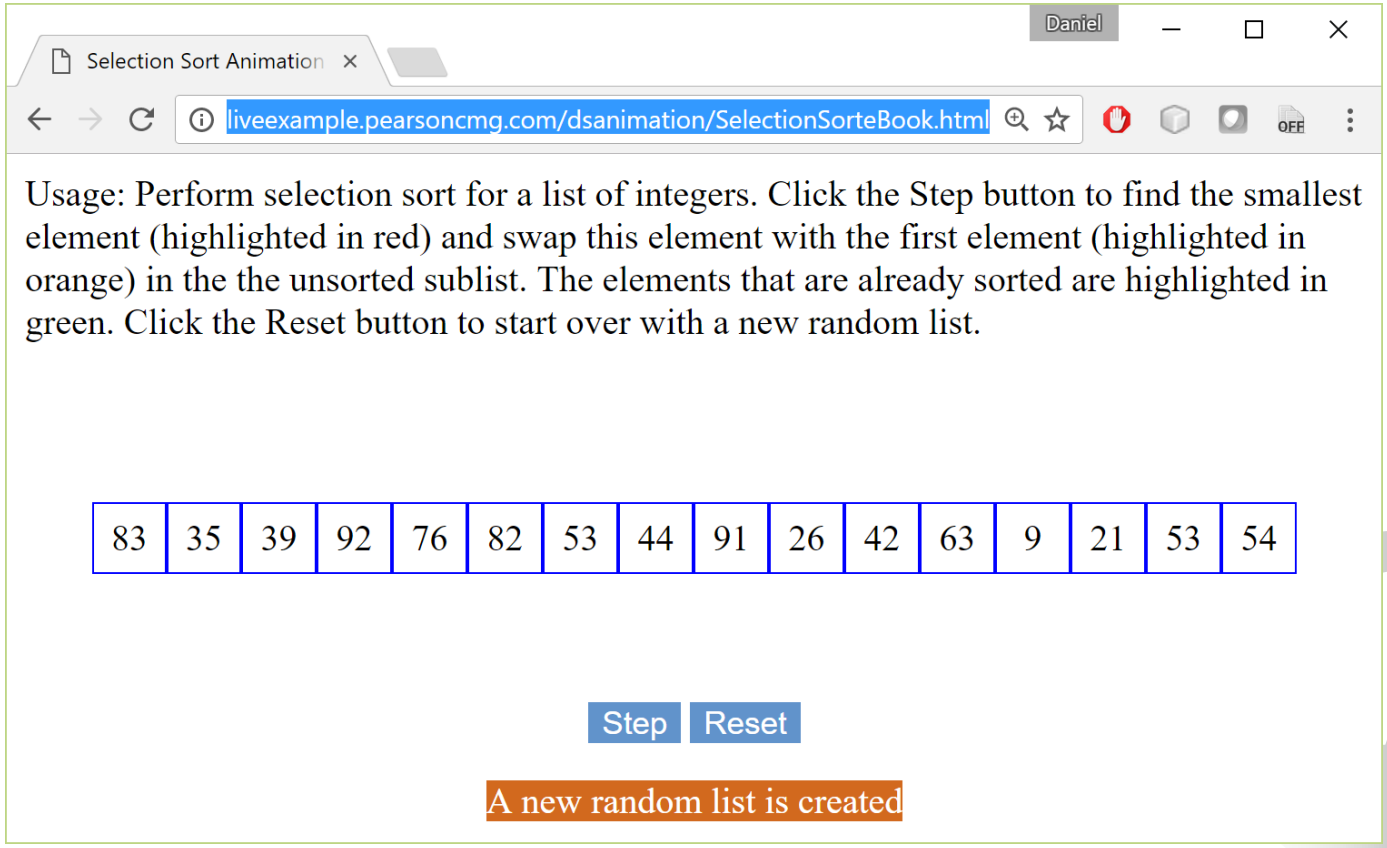
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Selection Sort



Selection Sort Animation

http://liveexample.pearsoncmg.com/dsanimation/SelectionSorteBook.html



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

From Idea to Solution

```
for (int i = 0; i < listSize; i++)
{
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i..listSize-1]
}
```

```
list[0] list[1] list[2] list[3] ...                 list[10]

list[0] list[1] list[2] list[3] ...                 list[10]

list[0] list[1] list[2] list[3] ...                 list[10]

list[0] list[1] list[2] list[3] ...                 list[10]

list[0] list[1] list[2] list[3] ...                 list[10]

                                ...

list[0] list[1] list[2] list[3] ...                 list[10]
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
for (int i = 0; i < listSize; i++)
{
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i..listSize-1]
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
for (int i = 0; i < listSize; i++)
{
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i..listSize-1]
}
```

Expand

```
double currentMin = list[i];
int currentMinIndex = i;
for (int j = i; j < listSize; j++)
{
  if (currentMin > list[j])
  {
    currentMin = list[j];
    currentMinIndex = j;
  }
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
for (int i = 0; i < listSize; i++)
{
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i..listSize-1]
}
```



Expand

```
if (currentMinIndex != i)
{
   list[currentMinIndex] = list[i];
   list[i] = currentMin;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  void selectionSort(double [], int);
5
6  int main()
7  {
8    double list[] = {-2, 4.5, 5, 1, 2, -3};
9    selectionSort(list, 6);
10
11   for (int i = 0; i < 6; i++)
12   {
13     cout << list[i] << " ";
14   }
15
16   return 0;
17  }
18
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
19  void selectionSort(double list[], int listSize)
20  {
21   for (int i = 0; i < listSize - 1; i++)
22   {
23     // Find the minimum in the list[i..listSize-1]
24     double currentMin = list[i];
25     int currentMinIndex = i;
26
27     for (int j = i + 1; j < listSize; j++)
28     {
29       if (currentMin > list[j])
30       {
31         currentMin = list[j];
32         currentMinIndex = j;
33       }
34     }
35
36     // Swap list[i] with list[currentMinIndex] if necessary;
37     if (currentMinIndex != i)
38     {
39       list[currentMinIndex] = list[i];
40       list[i] = currentMin;
41     }
42   }
43  }
```

```
-3.3 1 2 2 4.5 5
```

SelectionSort

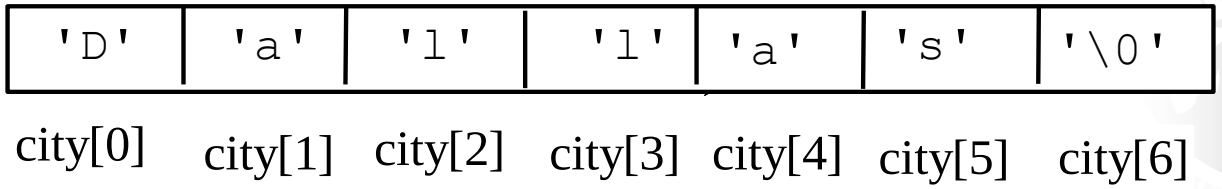
© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Initializing Character Arrays

```
char city[] =  {'D', 'a', 'l', 'l', 'a', 's'};

char city[] =  "Dallas";
```

- ✓ This statement is equivalent to the preceding statement, except that C++ adds the character **'\0'**.
  - ✓ **'\0'** called the *null terminator*, to indicate the end of the string.
- ✓ Recall that a character that begins with the back slash symbol (\) is an escape character.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Reading C-Strings

- You can read a string from the keyboard using the cin object.
- For example, see the following code:

```
char city[10];
cout << "Enter a city: ";
cin >> city; // read to array city
cout << "You entered " << city << endl;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Printing Character Array

- For a character array, it can be printed using one print statement.
- For example, the following code displays Dallas:

```
char city[] = "Dallas";
cout << city;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Reading C-Strings Using getline

- C++ provides the cin.getline function in the iostream header file, which reads a string into an array.
- The syntax of the function is:

```
cin.getline(char array[], int size, char delimitChar)
```

- The function stops reading characters when
  - The delimiter character is encountered
  - The size - 1 number of characters are read.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Reading C-Strings Using getline

- The last character in the array is reserved for the null terminator ('\0').
- If the delimiter is encountered, it is read, but not stored in the array.
- The third argument delimitChar has a default value ('\n').

```
cin.getline(char array[], int size, char delimitChar)
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

C-String Functions

Function	*Description*
`size_t strlen(char s[])`	Returns the length of the string, i.e., the number of the characters before the null terminator.
`strcpy(char s1[], const char s2[])`	Copies string s2 to string s1.
`strncpy(char s1[], const char s2[], size_t n)`	Copies the first n characters from string s2 to string s1.
`strcat(char s1[], const char s2[])`	Appends string s2 to s1.
`strncat(char s1[], const char s2[], size_t n)`	Appends the first n characters from string s2 to s1.
`int strcmp(char s1[], const char s2[])`	Returns a value greater than 0, 0, or less than 0 if s1 is greater than, equal to, or less than s2 based on the numeric code of the characters.
`int strncmp(char s1[], const char s2[], size_t n)`	Same as strcmp, but compares up to n number of characters in s1 with those in s2.
`int atoi(char s[])`	Returns an int value for the string.
`double atof(char s[])`	Returns a double value for the string.
`long atol(char s[])`	Returns a long value for the string.
`void itoa(int value, char s[], int radix)`	Obtains an integer value to a string based on specified radix.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

String Examples - `strcpy`

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  int main()
6  {
7    char s1[20];
8    char s2[20] = "Dallas, Texas";
9    char s3[20] = "AAAAAAAAAA";
10
11   strcpy(s1, s2);
12   strncpy(s3, s2, 6);
13   s3[6] = '\0'; // Insert null terminator
14
15   cout << "The string in s1 is " << s1 << endl;
16   cout << "The string in s2 is " << s2 << endl;
17   cout << "The string in s3 is " << s3 << endl;
18   cout << "The length of string s3 is " << strlen(s3) << endl;
19
20   return 0;
21 }
```

```
The string in s1 is Dallas, Texas
The string in s2 is Dallas, Texas
The string in s3 is Dallas
The length of string s3 is 6
```

CopyString

c. All Rights Reserved.

String Examples - `strcat`

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  int main()
6  {
7    char s1[20] = "Dallas";
8    char s2[20] = "Texas, USA";
9    char s3[20] = "Dallas";
10
11   strcat(strcat(s1, ", "), s2);
12   strncat(strcat(s3, ", "), s2, 5);
13
14   cout << "The string in s1 is " << s1 << endl;
15   cout << "The string in s2 is " << s2 << endl;
16   cout << "The string in s3 is " << s3 << endl;
17   cout << "The length of string s1 is " << strlen(s1) << endl;
18   cout << "The length of string s3 is " << strlen(s3) << endl;
19
20   return 0;
21 }
```

```
The string in s1 is Dallas, Texas, USA
The string in s2 is Texas, USA
The string in s3 is Dallas, Texas
The length of string s1 is 18
The length of string s3 is 13
```

CombineString

, Inc. All Rights Reserved.

String Examples - strcmp

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  int main()
6  {
7   char s1[] = "abcdefg";
8   char s2[] = "abcdg";
9   char s3[] = "abcdg";
10
11   cout << "strcmp(s1, s2) is " << strcmp(s1, s2) << endl;
12   cout << "strcmp(s2, s1) is " << strcmp(s2, s1) << endl;
13   cout << "strcmp(s2, s3) is " << strcmp(s2, s3) << endl;
14   cout << "strncmp(s1, s2, 3) is " << strncmp(s1, s2, 3) << endl;
15
16   return 0;
17  }
```

```
strcmp(s1, s2) is -1
strcmp(s2, s1) is 1
strcmp(s2, s3) is 0
strncmp(s1, s2, 3) is 0
```

CompareString

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

String Examples - atoi

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  int main()
6  {
7    cout << atoi("4") + atoi("5") << endl;
8    cout << atof("4.5") + atof("5.5") << endl;
9
10   char s[10];
11   itoa(42, s, 8);
12   cout << s << endl;
13
14   itoa(42, s, 10);
15   cout << s << endl;
16
17   itoa(42, s, 16);
18   cout << s << endl;
19
20   return 0;
21 }
```

```
9
10
52
42
2a
```

StringConversion

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Converting Numbers to Strings

```
int x = 15;
double y = 1.32;
long long int z = 10935;
string s = "Three numbers: " + to_string(x) + ", " +
  to_string(y) + ", and " + to_string(z);
cout << s << endl;
```

C++11: the to_string function is defined in C++11



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

References

- Introduction to Programming with C++, Daniel Liang, Pearson Education Limited 2014
- C++ Programming- From Problem Analysis to Program Design - D.S. MALIK - 2018



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Computer Engineering Department
Islamic University of Gaza

Dr. Ayman Maliha
amaliha@iugaza.edu.ps
Homepage: http://site.iugaza.edu.ps/amaliha/

ECOM 2403
Computer Programming
Fall 2020-2021



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.



Chapter 8

Multidimensional Arrays

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Motivations

- ✓ One-dimensional arrays are used model linear collections of elements.
- ✓ Data in a table or a matrix can be represented using a two-dimensional array.

Distance Table (in miles)

	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1375	967	1087
Boston	983	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	1187
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Motivations

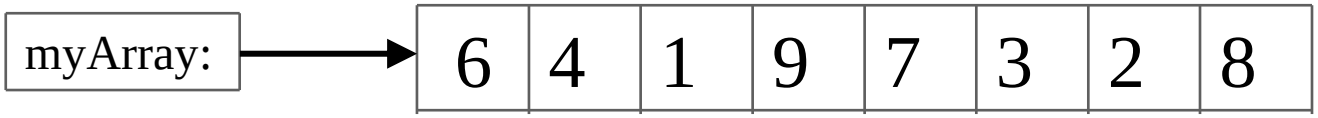
```
double distances [][] = {
      {0, 983, 787, 714, 1375, 967, 1087},
      {983, 0, 214, 1102, 1763, 1723, 1842},
      {787, 214, 0, 888, 1549, 1548, 1627},
      {714, 1102, 888, 0, 661, 781, 810},
      {1375, 1763, 1549, 661, 0, 1426, 1187},
      {967, 1723, 1548, 781, 1426, 0, 239},
      {1087, 1842, 1627, 810, 1187, 239, 0},
    };
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

1-Dimentional and 2-Dimentional Arrays

- In the previous chapter we used 1-dimensional arrays to model linear collections of elements.



- Now think of each element in the array to be a 1-dimentional array. This gives us a matrix.

1	4	1	5	7	1	8	8
5	4	3	9	1	3	3	5
3	5	2	5	0	7	4	3
9	7	1	9	9	8	6	2

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Declaring Two-dimensional Arrays

- ✓ An element in a two-dimensional array is accessed through a row and column index.

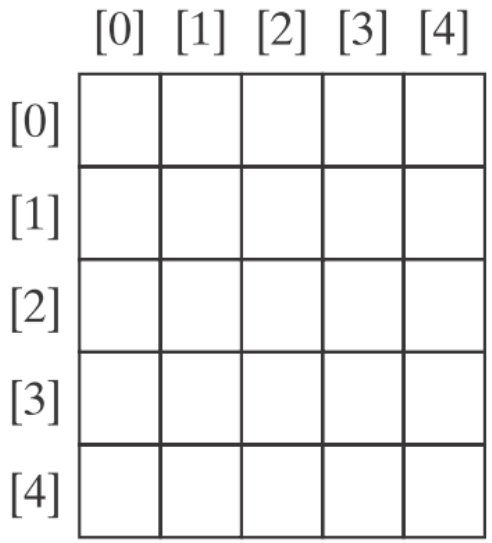
```
elementType arrayName[rowSize][columnSize];
```

```
int matrix[5][5];
```

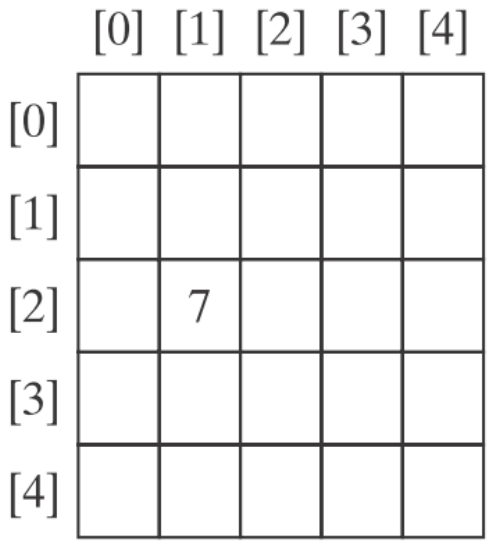


© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

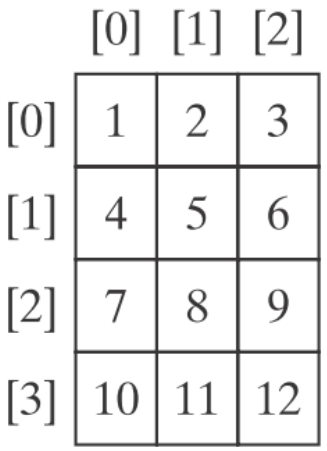
Two-dimensional Array Illustration




```
int matrix[5][5];
```




```
matrix[2][1] = 7;
```




```
int m[4][3] =
  {{1, 2, 3},
   {4, 5, 6},
   {7, 8, 9},
   {10, 11, 12}
  };
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Declaring, Creating, and Initializing Using Shorthand Notations

- You can also use an array initializer to declare, create and initialize a two-dimensional array. For example,

```
int array[4][3] =
{
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

(a)

Equivalent

```
int array[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

(b)



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Processing Two-Dimensional Arrays

- Initializing arrays with input values
- Initializing arrays with random values
- Displaying arrays
- Summing all elements
- Summing elements by column
- Which row has the largest sum
- Random shuffling



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Processing Two-Dimensional Arrays

- Nested for loops are often used to process a two-dimensional array.
- Suppose an array matrix is declared as follows:

```
const int ROW_SIZE = 10;
const int COLUMN_SIZE = 10;
int matrix[ROW_SIZE][COLUMN_SIZE];
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Initializing Arrays with Input Values

```
cout << "Enter " << ROW_SIZE << " rows and "
     << COLUMN_SIZE << " columns: " << endl;

for (int i = 0; i < ROW_SIZE; i++)
  for (int j = 0; j < COLUMN_SIZE; j++)
    cin >> matrix[i][j];
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Initializing Arrays with Random Values

```
for (int row = 0; row < ROW_SIZE; row++)
{
  for (int column = 0; column < COLUMN_SIZE; column++)
  {
    matrix[row][column] = rand() % 100;
  }
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Displaying Arrays

```
for (int row = 0; row < ROW_SIZE; row++)
{
  for (int column = 0; column < COLUMN_SIZE; column++)
  {
    cout << matrix[row][column] << " ";
  }

  cout << endl;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Summing All Elements

```
int total = 0;
for (int row = 0; row < ROW_SIZE; row++)
{
  for (int column = 0; column < COLUMN_SIZE; column++)
  {
    total += matrix[row][column];
  }
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Summing Elements by Column

```
for (int column = 0; column < COLUMN_SIZE; column++)
{
  int total = 0;
  for (int row = 0; row < ROW_SIZE; row++)
    total += matrix[row][column];
  cout << "Sum for column " << column
       << " is " << total << endl;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Which Row Has the Largest Sum

```
int maxRow = 0;
int indexOfMaxRow = 0;

// Get sum of the first row in maxRow
for (int column = 0; column < COLUMN_SIZE; column++)
  maxRow += matrix[0][column];

for (int row = 1; row < ROW_SIZE; row++)
{
  int totalOfThisRow = 0;
  for (int column = 0; column < COLUMN_SIZE; column++)
    totalOfThisRow += matrix[row][column];

  if (totalOfThisRow > maxRow)
  {
    maxRow = totalOfThisRow;
    indexOfMaxRow = row;
  }
}
cout << "Row " << indexOfMaxRow
     << " has the maximum sum of " << maxRow << endl;
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Random Shuffling

```
srand(time(0));

for (int i = 0; i < ROW_SIZE; i++)
{
  for (int j = 0; j < COLUMN_SIZE; j++)
  {
    int i1 = rand() % ROW_SIZE;
    int j1 = rand() % COLUMN_SIZE;

    // Swap matrix[i][j] with matrix[i1][j1]
    double temp = matrix[i][j];
    matrix[i][j] = matrix[i1][j1];
    matrix[i1][j1] = temp;
  }
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Passing Two-Dimensional Arrays to Functions

- You can pass a two-dimensional array to a function
- C++ requires that the column size to be specified in the function declaration.
- Write a function that returns the sum of all the elements in a matrix.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  const int COLUMN_SIZE = 4;
5
6  int sum(const int a[][COLUMN_SIZE], int rowSize)
7  {
8   int total = 0;
9   for (int row = 0; row < rowSize; row++)
10  {
11   for (int column = 0; column < COLUMN_SIZE; column++)
12   {
13     total += a[row][column];
14   }
15  }
16
17   return total;
18 }
19
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
20  int main()
21  {
22   const int ROW_SIZE = 3;
23   int m[ROW_SIZE][COLUMN_SIZE];
24   cout << "Enter " << ROW_SIZE << " rows and "
25   << COLUMN_SIZE << " columns: " << endl;
26   for (int i = 0; i < ROW_SIZE; i++)
27   for (int j = 0; j < COLUMN_SIZE; j++)
28   cin >> m[i][j];
29
30   cout << "\nSum of all elements is " << sum(m, ROW_SIZE) << endl;
31
32   return 0;
33  }
```

```
Enter 3 rows and 4 columns:
1 2 3 4
5 6 7 8
9 10 11 12
Sum of all elements is 78
```



PassTwoDimensionalArray

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example: Grading Multiple-Choice Test

Objective: write a program that grades multiple-choice test.

Students' Answers to the Questions:

	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

Key to the Questions:

	0	1	2	3	4	5	6	7	8	9
key	D	B	D	C	C	D	A	E	A	D



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6   const int NUMBER_OF_STUDENTS = 8;
7   const int NUMBER_OF_QUESTIONS = 10;
8
9   // Students' answers to the questions
10   char answers[NUMBER_OF_STUDENTS][NUMBER_OF_QUESTIONS] =
11   {
12    {'A', 'B', 'A', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
13    {'D', 'B', 'A', 'B', 'C', 'A', 'E', 'E', 'A', 'D'},
14    {'E', 'D', 'D', 'A', 'C', 'B', 'E', 'E', 'A', 'D'},
15    {'C', 'B', 'A', 'E', 'D', 'C', 'E', 'E', 'A', 'D'},
16    {'A', 'B', 'D', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
17    {'B', 'B', 'E', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
18    {'B', 'B', 'A', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
19    {'E', 'B', 'E', 'C', 'C', 'D', 'E', 'E', 'A', 'D'}
20   };
21
22   // Key to the questions
23   char keys[] = {'D', 'B', 'D', 'C', 'C', 'D', 'A', 'E', 'A', 'D'};
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
25    // Grade all answers
26    for (int i = 0; i < NUMBER_OF_STUDENTS; i++)
27    {
28      // Grade one student
29      int correctCount = 0;
30      for (int j = 0; j < NUMBER_OF_QUESTIONS; j++)
31      {
32        if (answers[i][j] == keys[j])
33          correctCount++;
34      }
35
36    cout << "Student " << i << "'s correct count is " <<
37            correctCount << endl;
38    }
39
40    return 0;
41  }
```

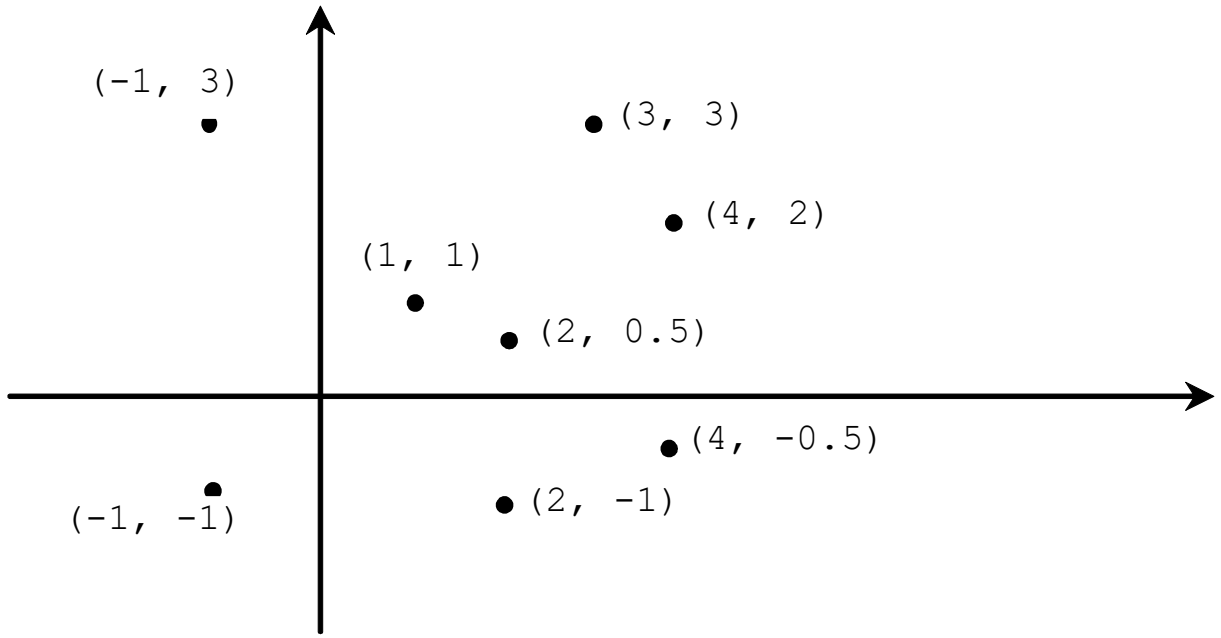
```
Student 0's correct count is 7
Student 1's correct count is 6
Student 2's correct count is 5
Student 3's correct count is 4
Student 4's correct count is 8
Student 5's correct count is 7
Student 6's correct count is 7
Student 7's correct count is 7
```

GradeExam



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Problem: Finding Two Points Nearest to Each Other



	x	y
0	-1	3
1	-1	-1
2	1	1
3	2	0.5
4	2	-1
5	3	3
6	4	2
7	4	-0.5

```
Enter 8 points: -1 3  -1 -1  1 1  2 0.5  2 -1  3 3 4 2 4 -0.5
The closest two points are (1, 1) and (2, 0.5)
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

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

// Compute the distance between two points (x1, y1) and (x2, y2)
double getDistance(double x1, double y1, double x2, double y2)
{
 return sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
}

int main()
{
  const int NUMBER_OF_POINTS = 8;

  // Each row in points represents a point
  double points[NUMBER_OF_POINTS][2];

  cout << "Enter " << NUMBER_OF_POINTS << " points: ";
  for (int i = 0; i < NUMBER_OF_POINTS; i++)
    cin >> points[i][0] >> points[i][1];

  // p1 and p2 are the indices in the points array
  int p1 = 0, p2 = 1; // Initial two points
  double shortestDistance = getDistance(points[p1][0], points[p1][1],
  points[p2][0], points[p2][1]); // Initialize shortestDistance

```

```
27   // Compute distance for every two points
28   for (int i = 0; i < NUMBER_OF_POINTS; i++)
29   {
30    for (int j = i + 1; j < NUMBER_OF_POINTS; j++)
31    {
32      double distance = getDistance(points[i][0], points[i][1],
33              points[j][0], points[j][1]); // Find distance
34
35      if (shortestDistance > distance)
36      {
37        p1 = i; // Update p1
38        p2 = j; // Update p2
39        shortestDistance = distance; // Update shortestDistance
40      }
41     }
42   }
43
44   // Display result
45   cout << "The closest two points are " <<
46   "(" << points[p1][0] << ", " << points[p1][1] << ") and (" <<
47   points[p2][0] << ", " << points[p2][1] << ")" << endl;
48
49   return 0;
50 }
```

FindNearestPoints

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study: Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

The objective is to fill the grid (see the left figure) so that every row, every column, and every 3×3 box contain the numbers 1 to 9, as shown in the right figure.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Case Study: Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	1	2	7	4	8	9	
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	1	2	7	4	9		
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

- ✓ Rule 1: Fill in an empty cell from the first to the last.
- ✓ Rule 2: Fill in a smallest number possible.
- ✓ Rule 3: If no number can fill in a cell, backtrack.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

What is Sudoku?

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Every row contains the numbers 1 to 9

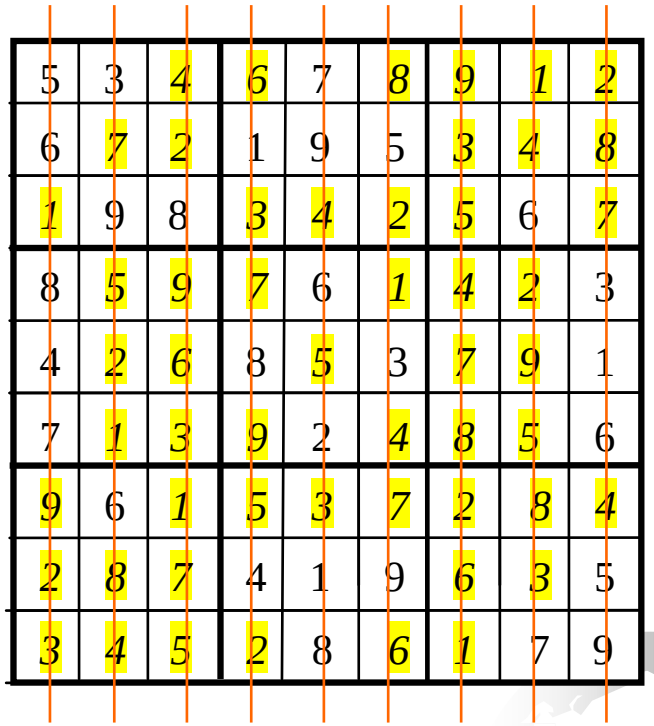
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Every column contains the numbers 1 to 9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Every 3×3 box contains the numbers 1 to 9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Checking Whether a Solution Is Correct

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

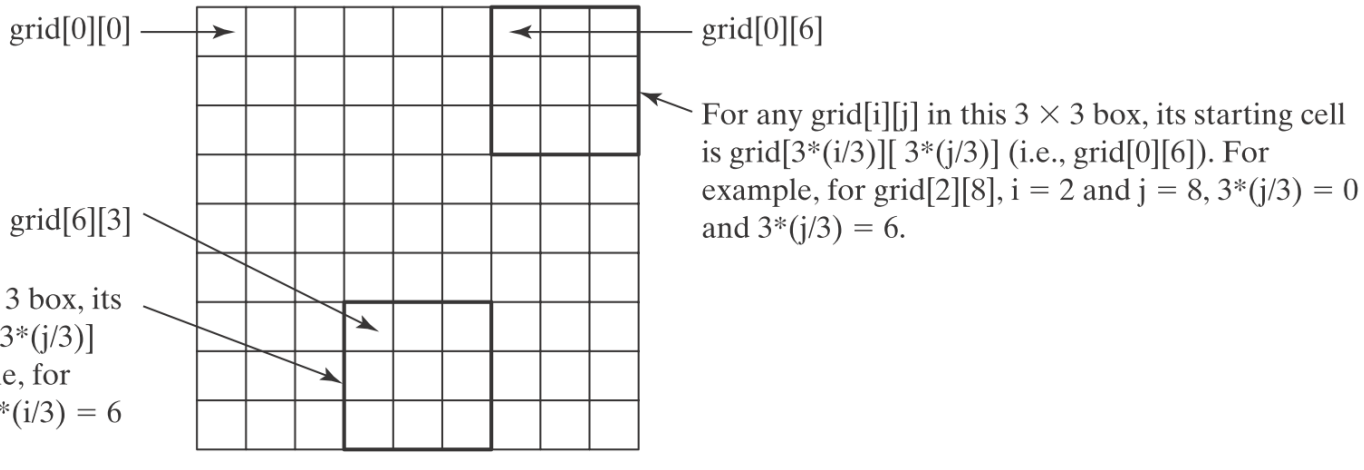
- ✓ Check if every row has numbers from 1 to 9, every column has numbers from 1 to 9, and every small box has numbers from 1 to 9.
- ✓ Check each cell. Each cell must be a number from 1 to 9 and the cell must be unique on every row, every column, and every small box.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  void readASolution(int grid[][9]);
5  bool isValid(const int grid[][9]);
6  bool isValid(int i, int j, const int grid[][9]);
7
8  int main()
9  {
10   // Read a Sudoku puzzle
11   int grid[9][9];
12   readASolution(grid);
13
14   cout << (isValid(grid) ? "Valid solution" : "Invalid solution");
15
16   return 0;
17  }
18
19  /** Read a Sudoku puzzle from the keyboard */
20  void readASolution(int grid[][9])
21  {
22   cout << "Enter a Sudoku puzzle solution:" << endl;
23   for (int i = 0; i < 9; i++)
24     for (int j = 0; j < 9; j++)
25       cin >> grid[i][j];
26  }
27
28  // Check whether the fixed cells are valid in the grid
29  bool isValid(const int grid[][9])
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
30  {
31   for (int i = 0; i < 9; i++)
32     for (int j = 0; j < 9; j++)
33        if (grid[i][j] < 1 || grid[i][j] > 9 ||
34           !isValid(i, j, grid))
35         return false;
36
37   return true; // The fixed cells are valid
38  }
39
40  // Check whether grid[i][j] is valid in the grid
41  bool isValid(int i, int j, const int grid[][9])
42  {
43   // Check whether grid[i][j] is valid at the i's row
44   for (int column = 0; column < 9; column++)
45      if (column != j && grid[i][column] == grid[i][j])
46        return false;
47
48   // Check whether grid[i][j] is valid at the j's column
49   for (int row = 0; row < 9; row++)
50     if (row != i && grid[row][j] == grid[i][j])
51        return false;
52
53   // Check whether grid[i][j] is valid in the 3-by-3 box
54   for (int row = (i / 3) * 3; row < (i / 3) * 3 + 3; row++)
55     for (int col = (j / 3) * 3; col < (j / 3) * 3 + 3; col++)
56        if (row != i && col != j && grid[row][col] == grid[i][j])
57           return false;
58
59   return true; // The current value at grid[i][j] is valid
60 }
```




```
Enter a Sudoku puzzle solution:
9 6 3 1 7 4 2 5 8
1 7 8 3 2 5 6 4 9
2 5 4 6 8 9 7 3 1
8 2 1 4 3 7 5 9 6
4 9 6 8 5 2 3 1 7
7 3 5 9 6 1 8 2 4
5 8 9 7 1 3 4 6 2
3 1 7 2 4 6 9 8 5
6 4 2 5 9 8 1 7 3
Valid solution
```

CheckSudokuSolution

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Multidimensional Arrays

- In the preceding section, you used a two-dimensional array to represent a matrix or a table.
- Occasionally, you will need to represent *n*-dimensional data structures.
- In C++, you can create *n*-dimensional arrays for any integer *n*.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Multidimensional Arrays

- The way to declare two-dimensional array can be generalized to declare *n*-dimensional array for *n* >= 3.
- you may use a three-dimensional array to store exam scores
- The class has 10 students with five exams and each exam has two parts (multiple-choice and essay)

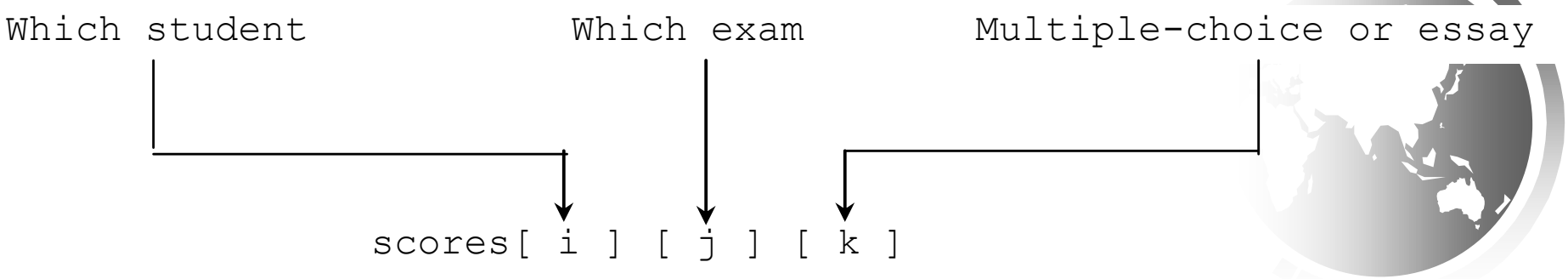
```
double scores[6][5][2];
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Multidimensional Arrays

```
double scores [][][] = {
  {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
  {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
  {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
  {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
  {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
  {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}}
};
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Problem: Daily Temperature and Humidity

- ✓ Suppose a meteorology station records the temperature and humidity at each hour of every day.
- ✓ The data for the past ten days are stored in a text file named weather.txt.
- ✓ Each line of the file consists of four numbers that indicates the day, hour, temperature, and humidity.

Day	Hour	Temperature	Humidity
1	1	76.4	0.92
1	2	77.7	0.93
...			
10	23	97.7	0.71
10	24	98.7	0.74

Day	Hour	Temperature	Humidity
10	24	98.7	0.74
1	2	77.7	0.93
...			
10	23	97.7	0.71
1	1	76.4	0.92

Your task is to write a program that calculates the average daily temperature and humidity for the 10 days.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6   const int NUMBER_OF_DAYS = 10;
7   const int NUMBER_OF_HOURS = 24;
8   double data[NUMBER_OF_DAYS][NUMBER_OF_HOURS][2];
9
10   // Read input using input redirection from a file
11   int day, hour;
12   double temperature, humidity;
13   for (int k = 0; k < NUMBER_OF_DAYS * NUMBER_OF_HOURS; k++)
14   {
15     cin >> day >> hour >> temperature >> humidity;
16     data[day - 1][hour - 1][0] = temperature;
17     data[day - 1][hour - 1][1] = humidity;
18   }
19
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
20    // Find the average daily temperature and humidity
21    for (int i = 0; i < NUMBER_OF_DAYS; i++)
22    {
23      double dailyTemperatureTotal = 0, dailyHumidityTotal = 0;
24      for (int j = 0; j < NUMBER_OF_HOURS; j++)
25      {
26        dailyTemperatureTotal += data[i][j][0];
27        dailyHumidityTotal += data[i][j][1];
28      }
29
30      // Display result
31      cout << "Day " << i + 1 << "'s average temperature is "
32           << dailyTemperatureTotal / NUMBER_OF_HOURS << endl;
33      cout << "Day " << i + 1 << "'s average humidity is "
34           << dailyHumidityTotal / NUMBER_OF_HOURS << endl;
35    }
36
37    return 0;
38  }
```

```
Day 0's average temperature is 77.7708
Day 0's average humidity is 0.929583
Day 1's average temperature is 77.3125
Day 1's average humidity is 0.929583
...
Day 9's average temperature is 79.3542
Day 9's average humidity is 0.9125
```

Weather

Inc. All Rights Reserved.

Problem: Guessing Birthday

- Listing 4.4, GuessBirthday.cpp, gives a program that guesses a birthday.
- The program can be simplified by storing the numbers in five sets in a three-dimensional array, and it prompts the user for the answers using a loop.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  int main()
6  {
7   int day = 0; // Day to be determined
8   char answer;
9
10   int dates[5][4][4] = {
11   {{ 1, 3, 5, 7},
12    { 9, 11, 13, 15},
13    {17, 19, 21, 23},
14    {25, 27, 29, 31}},
15   {{ 2, 3, 6, 7},
16    {10, 11, 14, 15},
17    {18, 19, 22, 23},
18    {26, 27, 30, 31}},
19   {{ 4, 5, 6, 7},
20    {12, 13, 14, 15},
21    {20, 21, 22, 23},
22    {28, 29, 30, 31}},
23   {{ 8, 9, 10, 11},
24    {12, 13, 14, 15},
25    {24, 25, 26, 27},
26    {28, 29, 30, 31}},
27   {{16, 17, 18, 19},
28    {20, 21, 22, 23},
29    {24, 25, 26, 27},
30    {28, 29, 30, 31}}};
```

```
32   for (int i = 0; i < 5; i++)
33   {
34     cout << "Is your birthday in Set" << (i + 1) << "?" << endl;
35     for (int j = 0; j < 4; j++)
36     {
37       for (int k = 0; k < 4; k++)
38         cout << setw(3) << dates[i][j][k] << " ";
39       cout << endl;
40     }
41     cout << "\nEnter N/n for No and Y/y for Yes: ";
42     cin >> answer;
43     if (answer == 'Y' || answer == 'y')
44       day += dates[i][0][0];
45   }
46
47   cout << "Your birthday is " << day << endl;
48
49   return 0;
50 }
```



GuessBirthdayUsingArray

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

References

- Introduction to Programming with C++, Daniel Liang, Pearson Education Limited 2014
- C++ Programming- From Problem Analysis to Program Design - D.S. MALIK - 2018



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Computer Engineering Department
Islamic University of Gaza

Dr. Ayman Maliha
amaliha@iugaza.edu.ps
Homepage: http://site.iugaza.edu.ps/amaliha/

ECOM 2403
Computer Programming
Fall 2020-2021



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.



Chapter 9

Objects and Classes

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Motivations

- After learning the preceding chapters
  - you are capable of solving many programming problems using selections, loops, methods (functions), and arrays.
  - However, these features are **not sufficient for developing** large scale software systems.

object-oriented programming enables you to develop large-scale software systems effectively.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Object-Oriented Programming

- Procedural programming
  - Uses variables to store data, and focuses on the processes/ functions that occur in a program.
  - Data and functions are separate and distinct.
- Object-oriented programming
  - Based on objects that encapsulate the data and the functions that operate with and on the data.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

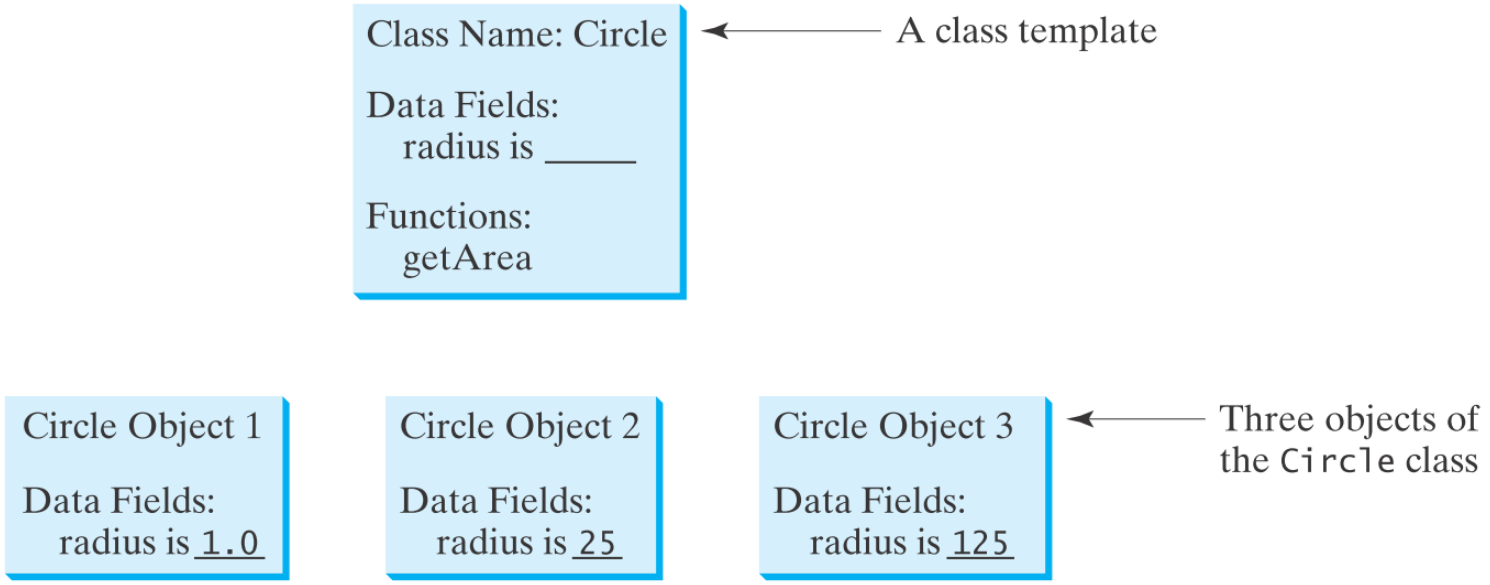
OO Programming Concepts

- An ***object*** represents an entity in the real world that can be distinctly identified.
  - For example, a student, a desk, a circle, a button, and even a loan can all be viewed as objects.
- An ***object*** has a unique ***identity***, ***state***, and ***behaviors***.
  - The ***state*** of an object consists of a set of ***data fields*** (also known as ***properties***) with their current values.
  - The ***behavior*** (also known as ***actions***) of an object is defined by a set of methods.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Objects



- An object (an instance of a class) has both a state and behavior.
  - The state defines the object
  - The behavior defines what the object does.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Classes

- *Classes* are constructs that define objects of the same type.
- A class uses variables to define data fields and functions to define behaviors.
- A special type of functions, known as constructors, are invoked to construct objects from the class.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Classes

```
class Circle
{
public:
  // The radius of this circle
  double radius;

  // Construct a circle object
  Circle()
  {
    radius = 1;
  }

  // Construct a circle object
  Circle(double newRadius)
  {
    radius = newRadius;
  }

  // Return the area of this circle
  double getArea()
  {
    return radius * radius * 3.14159;
  }
};
```

Data field

Constructors

Function



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Classes

C++11: data field can be initialized in C++

```
class Circle
{
public:
  // The radius of this circle
  double radius = 1;

  // Construct a circle object
  Circle()
  {

  }

  // Construct a circle object
  Circle(double newRadius)
  {
    radius = newRadius;
  }

  // Return the area of this circle
  double getArea()
  {
    return radius * radius * 3.14159;
  }
};
```

Data field

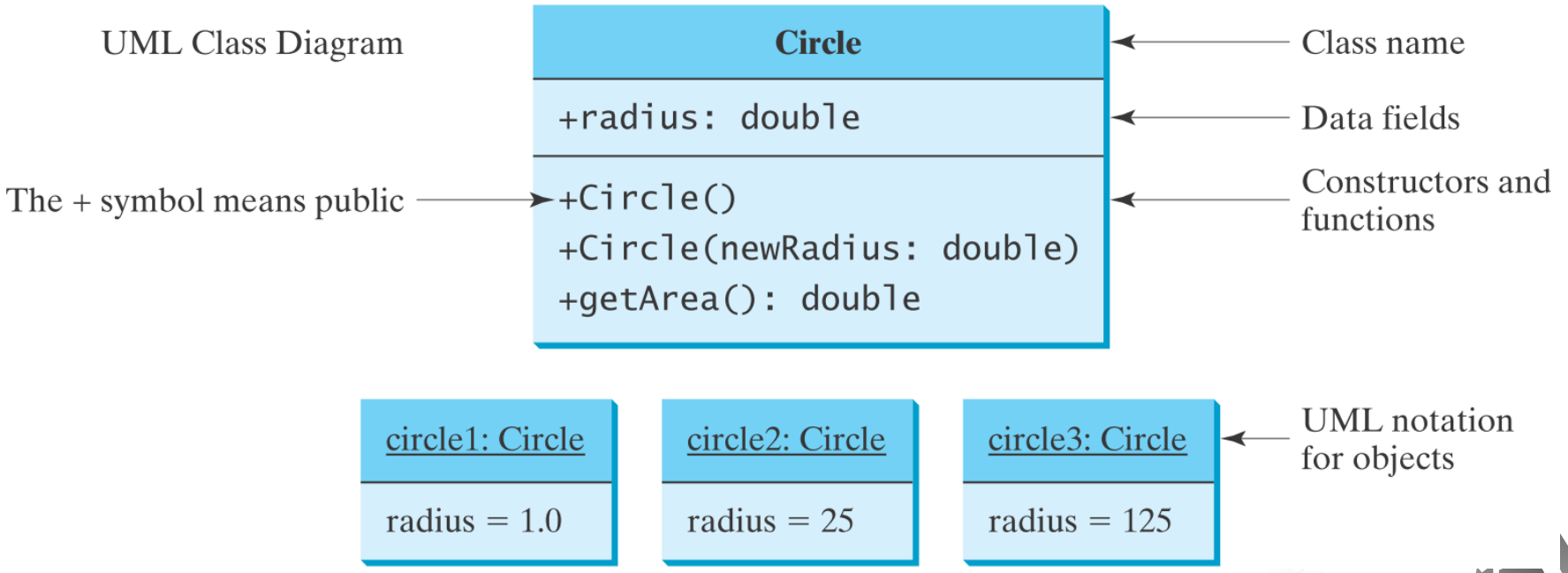
Constructors

Function



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

UML Class Diagram



Unified Modeling Language

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

A Simple Circle Class

- Objective: Demonstrate creating objects, accessing data, and using functions.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  #include <iostream>
2  using namespace std;
3
4  class Circle
5  {
6  public:
7    // The radius of this circle
8    double radius;
9
10   // Construct a default circle object
11   Circle()
12   {
13     radius = 1;
14   }
15
16   // Construct a circle object
17   Circle(double newRadius)
18   {
19     radius = newRadius;
20   }
21
22   // Return the area of this circle
23   double getArea()
24   {
25     return radius * radius * 3.14159;
26   }
```



```
28   // Return the perimeter of this circle
29   double getPermeter()
30   {
31      return 2 * radius * 3.14159;
32   }
33
34   // Set new radius for this circle
35   void setRadius(double newRadius)
36   {
37     radius = newRadius;
38   }
39 }; // Must place a semicolon here
40
41 int main()
42 {
43   Circle circle1(1.0);
44   Circle circle2(25);
45   Circle circle3(125);
46
47   cout << "The area of the circle of radius "
48        << circle1.radius << " is " << circle1.getArea() << endl;
49   cout << "The area of the circle of radius "
50        << circle2.radius << " is " << circle2.getArea() << endl;
51   cout << "The area of the circle of radius "
52        << circle3.radius << " is " << circle3.getArea() << endl;
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
54    // Modify circle radius
55    circle2.radius = 100;
56    cout << "The area of the circle of radius "
57         << circle2.radius << " is " << circle2.getArea() << endl;
58
59    return 0;
60  }
```

```
The area of the circle of radius 1 is 3.14159
The area of the circle of radius 25 is 1963.49
The area of the circle of radius 125 is 49087.3
The area of the circle of radius 100 is 31415.9
```

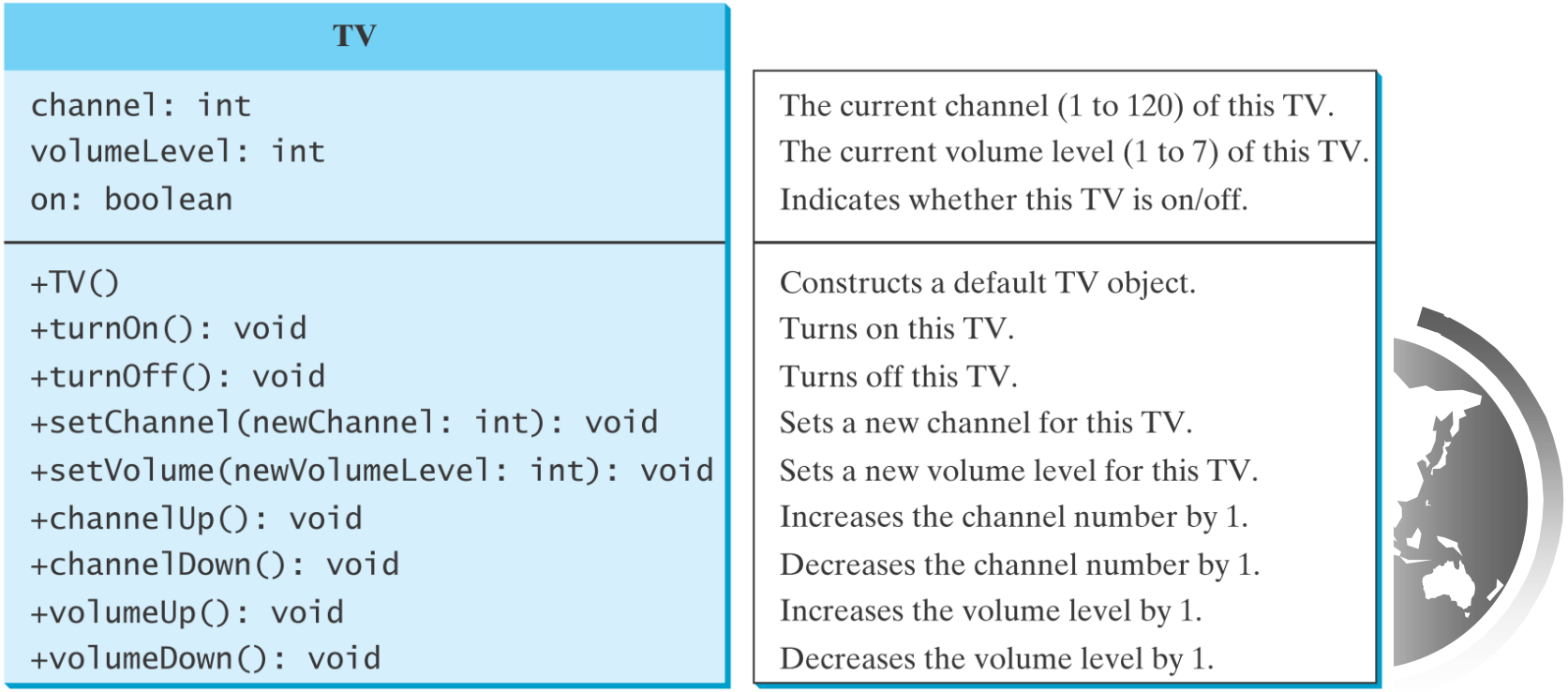
TestCircle



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example: Defining Classes and Creating Objects

- ✓ TV is an object with
  - ✓ state such as current channel, current volume level, and power on or off
  - ✓ behaviors such as change channels, adjust volume, and turn on/off.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

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

class TV
{
public:
  int channel;
  int volumeLevel; // Default volume level is 1
  bool on; // By default TV is off

  TV()
  {
    channel = 1; // Default channel is 1
    volumeLevel = 1; // Default volume level is 1
    on = false; // By default TV is off
  }

  void turnOn()
  {
    on = true;
  }

  void turnOff()
  {
    on = false;
  }

```

```
28   void setChannel(int newChannel)
29   {
30     if (on && newChannel >= 1 && newChannel <= 120)
31       channel = newChannel;
32   }
33
34   void setVolume(int newVolumeLevel)
35   {
36     if (on && newVolumeLevel >= 1 && newVolumeLevel <= 7)
37       volumeLevel = newVolumeLevel;
38   }
39
40   void channelUp()
41   {
42     if (on && channel < 120)
43       channel++;
44   }
45
46   void channelDown()
47   {
48     if (on && channel > 1)
49       channel--;
50   }
51
52   void volumeUp()
53   {
54     if (on && volumeLevel < 7)
55       volumeLevel++;
56   }
57
```

```
58   void volumeDown()
59   {
60     if (on && volumeLevel > 1)
61      volumeLevel--;
62   }
63  };
64
65  int main()
66  {
67   TV tv1;
68   tv1.turnOn(); // Turn tv1 on
69   tv1.setChannel(30);
70   tv1.setVolume(3);
71
72   TV tv2;
73   tv2.turnOn();
74   tv2.channelUp();
75   tv2.channelUp();
76   tv2.volumeUp(); // Increase tv2 volume up 1 level
77
78   cout << "tv1's channel is " << tv1.channel
79        << " and volume level is " << tv1.volumeLevel << endl;
80   cout << "tv2's channel is " << tv2.channel
81        << " and volume level is " << tv2.volumeLevel << endl;
82
83   return 0;
84  }
```

```
tv1's channel is 30 and volume level is 3
tv2's channel is 3 and volume level is 2
```

Constructors

Constructors are used to guarantee that member variables of a class are initialized

- ✓ Constructors must have the same name as the class itself.
- ✓ Constructors do **not have a return type**—not even void.
- ✓ A constructor with no parameters is referred to as a ***no-arg constructor***.
- ✓ A class may be declared without constructors.
  - ✓ A no-arg constructor with an empty body is implicitly declared in the class.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Constructors (cont'd.)

- A class can have more than one constructor
  - Each must have a different formal parameter list
- Constructors execute automatically when a class object enters its scope
- They cannot be called like other functions
- Which constructor executes depends on the types of values passed to the class object when the class object is declared



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Constructors

- Data fields may be initialized in the constructor using an initializer list in the following syntax:

```
ClassName(parameterList)
  : datafield1(value1), datafield2(value2) // Initializer list
{
  // Additional statements if needed
}
```

- For example

```
Circle::Circle()
  : radius(1)
{
}
```

```
Circle::Circle()
{
 radius = 1;
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Object Creation

- In C++, you can assign a name when creating an object.
- A constructor is invoked when an object is created.
- The syntax to create an object using the no-arg constructor is

```
ClassName objectName;
```

- For example,

```
Circle circle1;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Constructing with Arguments

- The syntax to declare an object using a constructor with arguments is

```
ClassName objectName(arguments);
```

- For example, the following declaration creates an object named circle2 by invoking the Circle class's constructor with a specified radius 5.5.

```
Circle circle2(5.5);
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Access Operator

- After an object is created, its data can be accessed and its functions invoked using the dot operator (.)
- Also known as the *object member access operator*:

```
objectName.dataField
```

```
objectName.function(arguments)
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Naming Objects and Classes

- When you declare a custom class, capitalize the first letter of each word in a class name
- For example, the class names Circle, Rectangle, and Desk.
- The class names in the C++ library are named in lowercase.
- The objects are named like variables.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Class is a Type

- You can use primitive data types to declare variables.
- You can also use class names to declare object names.
- In this sense, a class is also a data type.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Memberwise Copy

- In C++, you can also use the assignment operator = to copy the contents from one object to the other.
- By default, each data field of one object is copied to its counterpart in the other object.
- For example,

```
circle2 = circle1;
```

Copies the radius in circle1 to circle2.

- After the copy, circle1 and circle2 are still two different objects, but with the same radius.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Constant Object Name

- Object names are like array names.
- Once an object name is declared, it represents an object.
- It cannot be reassigned to represent another object.
- In this sense, an object name is a constant, though the contents of the object may change.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Anonymous Object

- When you create a named object, you can access the members of the object through its name.
- If you want to use the object only once, you can create *anonymous objects (objects without name)*.
- The syntax using the no-arg constructor is

```
ClassName()
```

- The syntax to using the constructor with arguments is

```
ClassName(arguements)
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Anonymous Object

- Creates a Circle object and copies its contents to circle1.

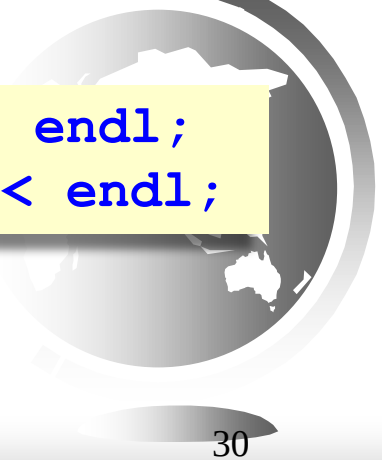
```
circle1 = Circle();
```

- Creates a Circle object with radius 5 and copies its contents to circle1.

```
circle1 = Circle(5);
```

- Creates Circle objects and invokes their **`getArea()`** function.

```
cout << "Area is " << Circle().getArea() << endl;
cout << "Area is " << Circle(5).getArea() << endl;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Separating Definition from Implementation

- C++ allows you to separate class definition from implementation.
- The class definition describes the contract of the class and the class implementation implements the contract.
- The class declaration simply lists all the data fields, constructor prototypes, and the function prototypes.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Separating Definition from Implementation

- The class implementation implements the constructors and functions.
- The class declaration and implementation are in two separate files.
- Both files should have the same name, but with different extension names.
- The class declaration file has an extension name .h and the class implementation file has an extension name .cpp.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
1  class Circle
2  {
3  public:
4    // The radius of this circle
5    double radius;
6
7    // Construct a default circle object
8    Circle();
9
10   // Construct a circle object
11   Circle(double);
12
13   // Return the area of this circle
14   double getArea();
15 };
```

Circle.h

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```cpp
#include "Circle.h"

// Construct a default circle object
Circle::Circle()
{
 radius = 1;
}

// Construct a circle object
Circle::Circle(double newRadius)
{
 radius = newRadius;
}

// Return the area of this circle
double Circle::getArea()
{
 return radius * radius * 3.14159;
}
```

Circle.cpp

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
 1  #include <iostream>
 2  #include "Circle.h"
 3  using namespace std;
 4
 5  int main()
 6  {
 7    Circle circle1;
 8    Circle circle2(5.0);
 9
10    cout << "The area of the circle of radius "
11      << circle1.radius << " is " << circle1.getArea() << endl;
12    cout << "The area of the circle of radius "
13      << circle2.radius << " is " << circle2.getArea() << endl;
14
15    // Modify circle radius
16    circle2.radius = 100;
17    cout << "The area of the circle of radius "
18      << circle2.radius << " is " << circle2.getArea() << endl;
19
20    return 0;
21  }
```

```
The area of the circle of radius 1 is 3.14159
The area of the circle of radius 5 is 78.5397
The area of the circle of radius 100 is 31415.9
```

TestCircleWithHeader.cpp

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Preventing Multiple Inclusions

- It is a common mistake to include the same header file in a program multiple times inadvertently.
- Suppose Head1.h includes Circle.h and TestHead1.cpp includes both Head1.h and Circle.h, as shown in the next slide.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Preventing Multiple Declarations

Head1.h

```
#include "Circle.h"
// Other code in Head1.h omitted
```

TestHead1.cpp

```
#include "Circle.h"
#include "Head1.h"
int main()
{
  // Other code in TestHead.cpp
  // omitted
}
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

inclusion guard

```
#ifndef CIRCLE_H
#define CIRCLE_H
class Circle
{
public:
  // The radius of this circle
  double radius;
  // Construct a default circle object
  Circle();
  // Construct a circle object
  Circle(double);
  // Return the area of this circle
  double getArea();
}; // Semicolon required
#endif
```

The preprocessor directive #ifndef stands for “if not defined.

If not, define the symbol in line 2 using the #define directive and the rest of the header file is included

otherwise, the rest of the header file is skipped.



The #endif directive is needed to indicate the end of header file.

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Inline Declaration

- Chapter 6 introduced how to improve function efficiency using inline functions.
- Inline functions play an important role in class declarations.
- When a function is implemented inside a class declaration, it automatically becomes an inline function.
- This is also known as *inline declaration*.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Inline Declaration Example

```
class A
{
public:
  A()
  {
    // do something;
  }
  double f1()
  {
    // return a number
  }

  double f2();
};
```

In class A

The constructor and function f1 are automatically inline functions, but function f2 is a regular function.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Inline Functions in Implementation File

- You may declare inline functions in the class's implementation file.
- For example, to declare function f2 as an inline function, precede the inline keyword in the function header as follows:

```
// Implement function as inline
inline double A::f2()
{
  // return a number
}
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Inline Declarations?

Short functions are good candidates for inline functions, but long functions are not.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Data Field Encapsulation

- The data fields radius in the Circle class can be modified directly (e.g., circle1.radius = 5).
- This is **not a good practice** for two reasons:
  - First, data may be tampered.
  - Second, it makes the class difficult to maintain and vulnerable to bugs.
    - Suppose you want to modify the Circle class to ensure that the radius is non-negative after other programs have already used the class.
    - You have to change not only the Circle class, but also the programs that use the Circle class.
    - This is because the clients may have modified the radius directly (e.g., myCircle.radius = -5).



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Accessor and Mutator

- A get function is referred to as a *getter* (or *accessor)*
- A set function is referred to as a *setter* (or *mutator*).
- A get function has the following signature:

```
returnType getPropertyName()
```

- If the returnType is `bool`, the get function should be defined as follows by convention:

```
bool isPropertyName()
```

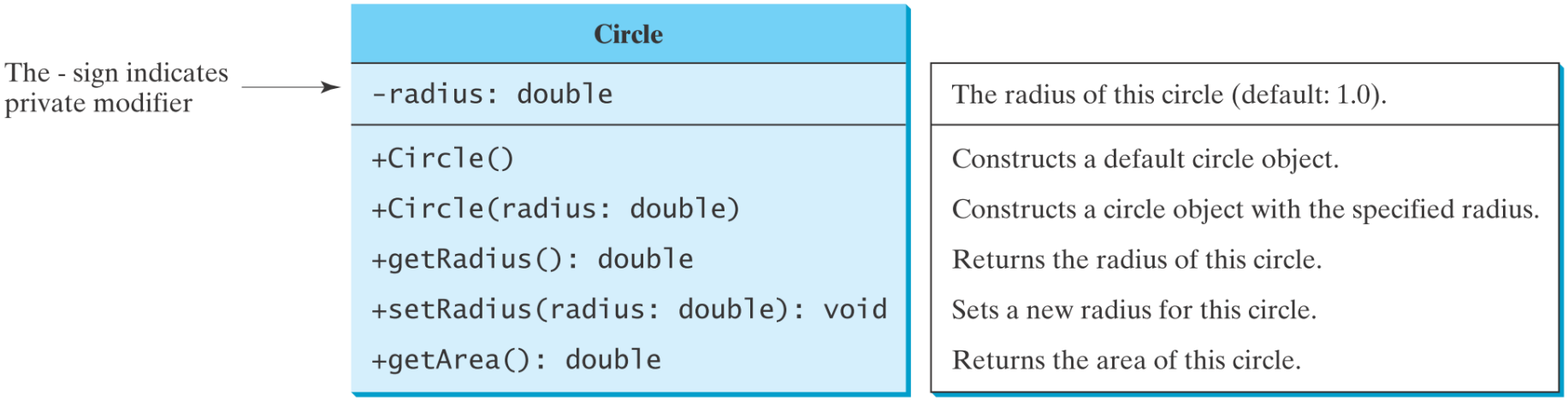
- A set function has the following signature:

```
public void setPropertyName(dataType propertyValue)
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example: New Circle Class





© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
 1  #ifndef CIRCLE_H
 2  #define CIRCLE_H
 3
 4  class Circle
 5  {
 6  public:
 7    Circle();
 8    Circle(double);
 9    double getArea();
10    double getRadius();
11    void setRadius(double);
12
13  private:
14    double radius;
15  };
16
17  #endif
```

CircleWithPrivateDataFields.h

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
 1  #include "CircleWithPrivateDataFields.h"
 2
 3  // Construct a default circle object
 4  Circle::Circle()
 5  {
 6    radius = 1;
 7  }
 8
 9  // Construct a circle object
10  Circle::Circle(double newRadius)
11  {
12    radius = newRadius;
13  }
14
15  // Return the area of this circle
16  double Circle::getArea()
17  {
18    return radius * radius * 3.14159;
19  }
20
21  // Return the radius of this circle
22  double Circle::getRadius()
23  {
24    return radius;
25  }
26
27  // Set a new radius
28  void Circle::setRadius(double newRadius)
29  {
30    radius = (newRadius >= 0) ? newRadius : 0;
31  }
```

CircleWithPrivateDataFields.cpp

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
 1  #include <iostream>
 2  #include "CircleWithPrivateDataFields.h"
 3  using namespace std;
 4
 5  int main()
 6  {
 7    Circle circle1;
 8    Circle circle2(5.0);
 9
10    cout << "The area of the circle of radius "
11      << circle1.getRadius() << " is " << circle1.getArea() << endl;
12    cout << "The area of the circle of radius "
13      << circle2.getRadius() << " is " << circle2.getArea() << endl;
14
15    // Modify circle radius
16    circle2.setRadius(100);
17    cout << "The area of the circle of radius "
18      << circle2.getRadius() << " is " << circle2.getArea() << endl;
19
20    return 0;
21  }
```

TestCircleWithPrivateDataFields

```
The area of the circle of radius 1 is 3.14159
The area of the circle of radius 5 is 78.5397
The area of the circle of radius 100 is 31415.9
```

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

The Scope of Variables

- Chapter 6, “Functions,” discussed the scope of global variables and local variables.
- Global variables
  - Declared outside all functions and are accessible to all functions in its scope.
  - The scope of a global variable starts from its declaration and continues to the end of the program.
- Local variables
  - Defined inside functions.
  - The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

The Scope of Variables

- ✓ The data fields are declared as variables and are accessible to all constructors and functions in the class.
- ✓ In this sense, data fields are like global variables.
- ✓ However, data fields and functions can be declared in any order in a class.
- ✓ For example, all the following declarations are the same:

```
class Circle
{
public:
  Circle();
  Circle(double);
  double getArea();
  double getRadius();
  void setRadius(double);

private:
  double radius;
};
```

(a)

```
class Circle
{
public:
  Circle();
  Circle(double);

private:
  double radius;

public:
  double getArea();
  double getRadius();
  void setRadius(double);
};
```

(b)

```
class Circle
{
private:
  double radius;

public:
  double getArea();
  double getRadius();
  void setRadius(double);

public:
  Circle();
  Circle(double);
};
```

(c)

© Copyright 2016 by Pearson Edu Inc. All Rights Reserved.

The Scope of Variables

```
 1  #include <iostream>
 2  using namespace std;
 3
 4  class Foo
 5  {
 6  public:
 7    int x; // Data field
 8    int y; // Data field
 9
10    Foo()
11    {
12      x = 10;
13      y = 10;
14    }
15
16    void p()
17    {
18      int x = 20; // Local variable
19      cout << "x is " << x << endl;
20      cout << "y is " << y << endl;
21    }
22  };
23
24  int main()
25  {
26    Foo foo;
27    foo.p();
28
29    return 0;
30  }
```

- ✓ Local variables are declared and used inside a function locally.
- ✓ If a local variable has the same name as a data field, the local variable takes precedence and the data field with the same name is hidden.

HideDataField

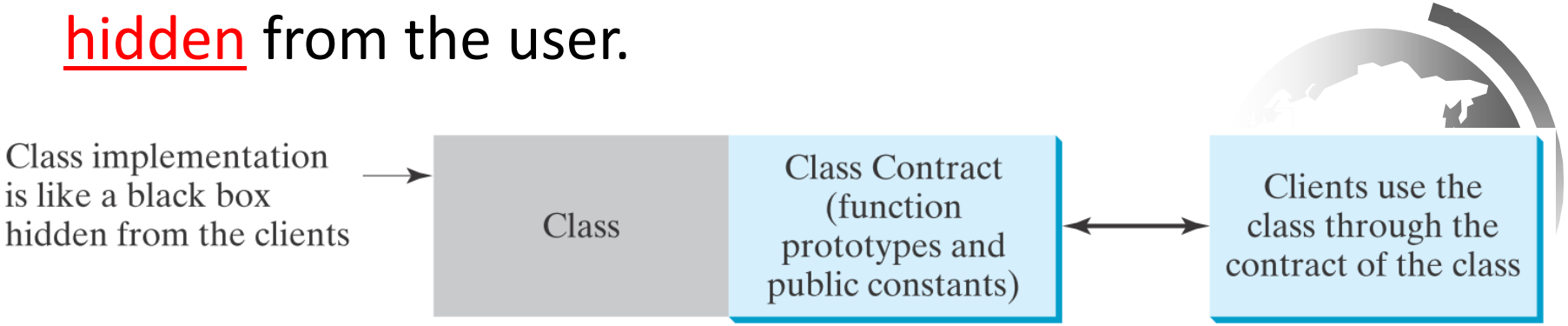
```
x is 20
y is 10
```



n, Inc. All Rights Reserved.

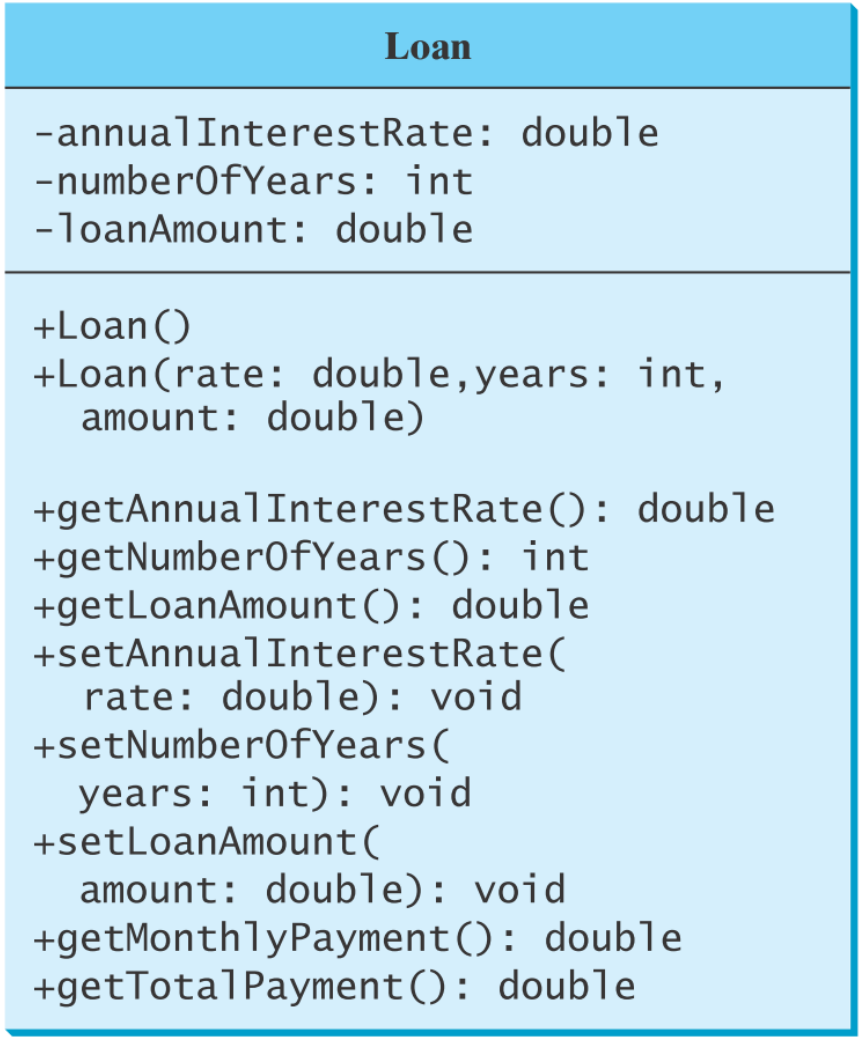
Class Abstraction and Encapsulation

- Class abstraction means to separate class implementation from the use of the class.
- The creator of the class provides a description of the class and let the **user know how** the class can be used.
- The **user** of the class **does not need** to know how the class is implemented.
- The detail of implementation is encapsulated and hidden from the user.



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Example: The Loan Class



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
 1  #ifndef LOAN_H
 2  #define LOAN_H
 3
 4  class Loan
 5  {
 6  public:
 7    Loan();
 8    Loan(double rate, int years, double amount);
 9    double getAnnualInterestRate();
10    int getNumberOfYears();
11    double getLoanAmount();
12    void setAnnualInterestRate(double rate);
13    void setNumberOfYears(int years);
14    void setLoanAmount(double amount);
15    double getMonthlyPayment();
16    double getTotalPayment();
17
18  private:
19    double annualInterestRate;
20    int numberOfYears;
21    double loanAmount;
22  };
23
24  #endif
```

Loan.h



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

```
 1  #include <iostream>
 2  #include <iomanip>
 3  #include "Loan.h"
 4  using namespace std;
 5
 6  int main()
 7  {
 8    // Enter annual interest rate
 9    cout << "Enter yearly interest rate, for example 8.25: ";
10    double annualInterestRate;
11    cin >> annualInterestRate;
12
13    // Enter number of years
14    cout << "Enter number of years as an integer, for example 5: ";
15    int numberOfYears;
16    cin >> numberOfYears;
17
18    // Enter loan amount
19    cout << "Enter loan amount, for example 120000.95: ";
20    double loanAmount;
21    cin >> loanAmount;
22
23    // Create Loan object
24    Loan loan(annualInterestRate, numberOfYears, loanAmount);
25
26    // Display results
27    cout << fixed << setprecision(2);
28    cout << "The monthly payment is "
29      << loan.getMonthlyPayment() << endl;
30    cout << "The total payment is " << loan.getTotalPayment() << endl;
31
32    return 0;
33  }
34
```

TestLoanClass

```
 1  #include "Loan.h"
 2  #include <cmath>
 3  using namespace std;
 4
 5  Loan::Loan()
 6  {
 7    annualInterestRate = 9.5;
 8    numberOfYears = 30;
 9    loanAmount = 100000;
10  }
11
12  Loan::Loan(double rate, int years, double amount)
13  {
14    annualInterestRate = rate;
15    numberOfYears = years;
16    loanAmount = amount;
17  }
18
19  double Loan::getAnnualInterestRate()
20  {
21    return annualInterestRate;
22  }
23
24  int Loan::getNumberOfYears()
25  {
26    return numberOfYears;
27  }
28
29  double Loan::getLoanAmount()
30  {
31    return loanAmount;
32  }
33
```

```
34  void Loan::setAnnualInterestRate(double rate)
35  {
36    annualInterestRate = rate;
37  }
38
39  void Loan::setNumberOfYears(int years)
40  {
41    numberOfYears = years;
42  }
43
44  void Loan::setLoanAmount(double amount)
45  {
46    loanAmount = amount;
47  }
48
49  double Loan::getMonthlyPayment()
50  {
51    double monthlyInterestRate = annualInterestRate / 1200;
52    return loanAmount * monthlyInterestRate / (1 -
53      (pow(1 / (1 + monthlyInterestRate), numberOfYears * 12)));
54  }
55
56  double Loan::getTotalPayment()
57  {
58    return getMonthlyPayment() * numberOfYears * 12;
59  }
```

Loan.cpp

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Records (`structs`)

- `struct`: collection of a fixed number of components (members), accessed by name
  - Members may be of different types
- Syntax:

```
struct structName
{
    dataType1 identifier1;
    dataType2 identifier2;
       .
       .
       .
    dataTypen identifiern;
};
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Records (`structs`) (cont'd.)

- A `struct` is a definition, not a declaration
  - Must declare a variable of that type to use it

```
struct houseType
{
    string style;
    int numOfBedrooms;
    int numOfBathrooms;
    int numOfCarsGarage;
    int yearBuilt;
    int finishedSquareFootage;
    double price;
    double tax;
};

  //variable declaration
houseType newHouse;
```



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Records (`structs`) (cont'd.)

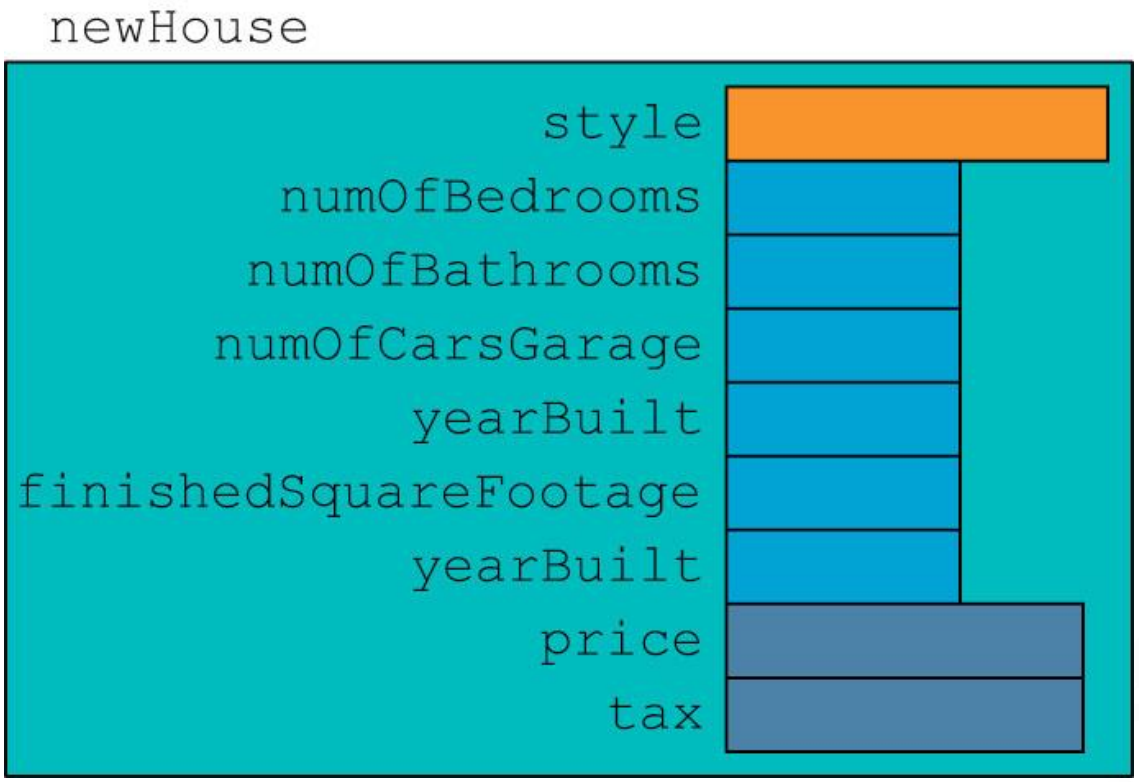


FIGURE 9-1 `struct` newHouse

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Accessing `struct` Members

- Syntax to access a `struct` member:

```
structVariableName.memberName
```

- The dot (.) is called the member access operator



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

Class Replaces `struct`

- ✓ The C language has the `struct` type for representing records.
- ✓ For example, you may define a `struct` type for representing students as shown in (a).

```
struct Student
{
  int id;
  char firstName[30];
  char mi;
  char lastName[30];
};
```

(a)

```
class Student
{
public:
  int id;
  char firstName[30];
  char mi;
  char lastName[30];
};
```

(b)

© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

A `struct` Versus a `class`

- By default, members of a `struct` are `public`
  - `private` specifier can be used in a `struct` to make a member private
- By default, the members of a `class` are `private`
- `class`es and `struct`s have the same capabilities



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

A `struct` Versus a `class` (cont'd.)

- In C++, the definition of a `struct` was expanded to include member functions, constructors, and destructors
- If all member variables of a `class` are `public` and there are no member functions
  - Use a `struct`



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.

References

- Introduction to Programming with C++, Daniel Liang, Pearson Education Limited 2014
- C++ Programming- From Problem Analysis to Program Design - D.S. MALIK - 2018



© Copyright 2016 by Pearson Education, Inc. All Rights Reserved.