Chapter Summary

This chapter introduced enough of C++ to let you compile and execute simple C++ programs. We saw how to define a `main` function, which is the function that the operating system calls to execute our program. We also saw how to define variables, how to do input and output, and how to write `if`, `for`, and `while` statements. The chapter closed by introducing the most fundamental facility in C++: the class. In this chapter, we saw how to create and use objects of a class that someone else has defined. Later chapters will show how to define our own classes.

Defined Terms

argument Value passed to a function.

assignment Obliterates an object's current value and replaces that value by a new one.

block Sequence of zero or more statements enclosed in curly braces.

buffer A region of storage used to hold data. IO facilities often store input (or output) in a buffer and read or write the buffer independently from actions in the program. Output buffers can be explicitly flushed to force the buffer to be written. By default, reading `cin` flushes `cout`; `cout` is also flushed when the program ends normally.

built-in type Type, such as `int`, defined by the language.

cerr `ostream` object tied to the standard error, which often writes to the same device as the standard output. By default, writes to `cerr` are not buffered. Usually used for error messages or other output that is not part of the normal logic of the program.

character string literal Another term for string literal.

cin `istream` object used to read from the standard input.

class Facility for defining our own data structures together with associated operations. The class is one of the most fundamental features in C++. Library types, such as `istream` and `ostream`, are classes.

class type A type defined by a class. The name of the type is the class name.

clog `ostream` object tied to the standard error. By default, writes to `clog` are buffered. Usually used to report information about program execution to a log file.

comments Program text that is ignored by the compiler. C++ has two kinds of comments: single-line and paired. Single-line comments start with a `//`. Everything from the `//` to the end of the line is a comment. Paired comments begin with a `/*` and include all text up to the next `*/`.

condition An expression that is evaluated as true or false. A value of zero is false; any other value yields true.

cout `ostream` object used to write to the standard output. Ordinarily used to write the output of a program.

curly brace Curly braces delimit blocks. An open curly (`{`) starts a block; a close curly (`}`) ends one.

data structure A logical grouping of data and operations on that data.

edit-compile-debug The process of getting a program to execute properly.

end-of-file System-specific marker that indicates that there is no more input in a file.

expression The smallest unit of computation. An expression consists of one or more operands and usually one or more operators. Expressions are evaluated to produce a result. For example, assuming `i` and `j` are `int`s, then `i + j` is an expression and yields the sum of the two `int` values.

for statement Iteration statement that provides iterative execution. Often used to repeat a calculation a fixed number of times.

function Named unit of computation.

function body Block that defines the actions performed by a function.

function name Name by which a function is known and can be called.

header Mechanism whereby the definitions of a class or other names are made available to multiple programs. A program uses a header through a `#include` directive.

if statement Conditional execution based on the value of a specified condition. If the condition is true, the `if` body is executed. If not, the `else` body is executed if there is one.

initialize Give an object a value at the same time that it is created.

iostream Header that provides the library types for stream-oriented input and output.

istream Library type providing stream-oriented input.

library type Type, such as `istream`, defined by the standard library.

main Function called by the operating system to execute a C++ program. Each program must have one and only one function named `main`.

manipulator Object, such as `std::endl`, that when read or written "manipulates" the stream itself.

member function Operation defined by a class. Member functions ordinarily are called to operate on a specific object.

method Synonym for member function.

namespace Mechanism for putting names defined by a library into a single place. Namespaces help avoid inadvertent name clashes. The names defined by the C++ library are in the namespace `std`.

ostream Library type providing stream-oriented output.

parameter list Part of the definition of a function. Possibly empty list that specifies what arguments can be used to call the function.

return type Type of the value returned by a function.

source file Term used to describe a file that contains a C++ program.

standard error Output stream used for error reporting. Ordinarily, the standard output and the standard error are tied to the window in which the program is executed.

standard input Input stream usually associated with the window in which the program executes.

standard library Collection of types and functions that every C++ compiler must support. The library provides the types that support IO. C++ programmers tend to talk about "the library," meaning the entire standard library. They also tend to refer to particular parts of the library by referring to a library type, such as the "`iostream` library," meaning the part of the standard library that defines the IO classes.

standard output Output stream usually associated with the window in which the program executes.

statement A part of a program that specifies an action to take place when the program is executed. An expression followed by a semicolon is a statement; other kinds