Introduction to Classes, Objects and Strings

OBJECTIVES

In this chapter you'll learn:

- How to define a class and use it to create an object.
- How to implement a class's behaviors as member functions.
- How to implement a class's attributes as data members.
- How to call a member function of an object to perform a task.
- The differences between data members of a class and local variables of a function.
- How to use a constructor to initialize an object's data when the object is created.
- How to engineer a class to separate its interface from its implementation and encourage reuse.
- How to use objects of class `string`.

3.1 Introduction

- In this chapter, you'll begin writing programs that employ the basic concepts of *object-oriented programming*.
- Typically, the programs you develop in this book will consist of function `main` and one or more *classes*, each containing *data members* and *member functions*.
- In this chapter, we develop a simple, well-engineered framework for organizing object-oriented programs in C++.

3.2 Defining a Class with a Member Function

- We begin with an example (Fig. 3.1) that consists of class `GradeBook` (lines 8–16), which, when it is fully developed in Chapter 7, will represent a grade book that an instructor can use to maintain student test scores, and a `main` function (lines 19–23) that creates a `GradeBook` object.
- Function `main` uses this object and its `displayMessage` member function to display a message on the screen welcoming the instructor to the grade-book program.

```
1  // Fig. 3.1: fig03_01.cpp
2  // Define class GradeBook with a member function displayMessage,
3  // create a GradeBook object, and call its displayMessage function.
4  #include <iostream>
5  using namespace std;
6
7  // GradeBook class definition
8  class GradeBook
9  {
10 public:
11    // function that displays a welcome message to the GradeBook user
12    void displayMessage() const
13    {
14       cout << "Welcome to the Grade Book!" << endl;
15    } // end function displayMessage
16 }; // end class GradeBook
17
18 // function main begins program execution
19 int main()
20 {
21    GradeBook myGradeBook; // create a GradeBook object named myGradeBook
22    myGradeBook.displayMessage(); // call object's displayMessage function
23 } // end main
```

Fig. 3.1 | Define class GradeBook with a member function displayMessage, create a GradeBook object and call its displayMessage function. (Part 1 of 2.)

```
Welcome to the Grade Book!
```

Fig. 3.1 | Define class `GradeBook` with a member function `displayMessage`, create a `GradeBook` object and call its `displayMessage` function. (Part 2 of 2.)

3.2 Defining a Class with a Member Function (cont.)

- The `GradeBook` class definition (lines 8–16) begins with keyword `class` and contains a member function called `displayMessage` (lines 12–15) that displays a message on the screen (line 14).
- Need to make an object of class `GradeBook` (line 21) and call its `displayMessage` member function (line 22) to get line 14 to execute and display the welcome message.
- The class definition begins with the keyword `class` followed by the class name `GradeBook`.

3.2 Defining a Class with a Member Function (cont.)

- By convention, the name of a user-defined class begins with a capital letter, and for readability, each subsequent word in the class name begins with a capital letter.
- Often referred to as Pascal case.
- The occasional uppercase letters resemble a camel's humps. More generally, camel case capitalization style allows the first letter to be either lowercase or uppercase

- Every class's body is enclosed in a pair of left and right braces ({ and }), as in lines 9 and 16.

Common Programming Error 3.1

Forgetting the semicolon at the end of a class definition is a syntax error.

3.2 Defining a Class with a Member Function (cont.)

- Function `main` is always called automatically when you execute a program.
- Most functions do not get called automatically.
- You must call member function `displayMessage` explicitly to tell it to perform its task.
- The access-specifier label `public:` contains the keyword `public` is an access specifier.
  - Indicates that the function is "available to the public"—that is, it can be called by other functions in the program (such as `main`), and by member functions of other classes (if there are any)

3.2 Defining a Class with a Member Function (cont.)

- Each function in a program performs a task and may *return a value* when it completes its task.
- When you define a function, you must specify a return type to indicate the type of the value returned by the function when it completes its task.
- Keyword `void` to the left of the function name `displayMessage` is the function's return type.
  - Indicates that `displayMessage` will *not* return any data to its calling function when it completes its task.
- The name of the member function, `displayMessage`, follows the return type.
- By convention, our function names use the *camel case* style with a lowercase first letter.
- The parentheses after the member function name indicate that it is a *function*.

3.2 Defining a Class with a Member Function (cont.)

- Empty parentheses indicate that a member function does not require additional data to perform its task.
- The first line of a function definition is commonly called the function header.
- Every function's *body* is delimited by left and right braces ({ and }).
- The *function body* contains statements that perform the function's task.

3.2 Defining a Class with a Member Function (cont.)

Testing Class `GradeBook`

- Typically, you cannot call a member function of a class until you create an object of that class.
- First, create an object of class `GradeBook` called `myGradeBook`.
  - The variable's type is `GradeBook`.
  - The compiler does not automatically know what type `GradeBook` is—it's a user-defined type.
  - Tell the compiler what `GradeBook` is by including the class definition.
  - Each class you create becomes a new type that can be used to create objects.

3.2 Defining a Class with a Member Function (cont.)

- Call the member function displayMessage- by using variable myGradeBook followed by the dot operator (.), the function name display-Message and an empty set of parentheses.
- Causes the displayMessage function to perform its task.