Introduction to Classes, Objects, Member Functions and Strings

3

Objectives

In this chapter you'll:

- Begin programming with the object-oriented concepts introduced in Section 1.8.
- Define a class and use it to create an object.
- Implement a class's behaviors as member functions.
- Implement a class's attributes as data members.
- Call an object's member functions to make them perform their tasks.
- Access and manipulate `private` data members through their corresponding `public` *get* and *set* functions to enforce encapsulation of the data.
- Learn what local variables of a member function are and how they differ from data members of a class.
- Use a constructor to initialize an object's data.
- Validate the data passed to a constructor or member function.
- Become familiar with UML class diagrams.

Self-Review Exercises

3.1 Fill in the blanks in each of the following:

a) Every class definition contains the keyword ________ followed immediately by the class's name.

ANS: `class`.

b) A class definition is typically stored in a file with the ________ filename extension.

ANS: `.h`.

c) Each parameter in a function header specifies both a(n) ________ and a(n) ________.

ANS: type, name.

d) When each object of a class maintains its own version of an attribute, the variable that represents the attribute is also known as a(n) ________.

ANS: data member.

e) Keyword `public` is a(n) ________.

ANS: access specifier.

f) Return type ________ indicates that a function will perform a task but will not return any information when it completes its task.

ANS: `void`.

g) Function ________ from the `<string>` library reads characters until a newline character is encountered, then copies those characters into the specified `string`.

ANS: `getline`.

h) Any file that uses a class can include the class's header via a(n) ________ preprocessing directive.

ANS: `#include`.

3.2 State whether each of the following is *true* or *false*. If *false*, explain why.

a) By convention, function names begin with a capital letter and all subsequent words in the name begin with a capital letter.

ANS: False. Function names begin with a lowercase letter and all subsequent words in the name begin with a capital letter.

b) Empty parentheses following a function name in a function definition indicate that the function does not require any parameters to perform its task.

ANS: True.

c) Data members or member functions declared with access specifier `private` are accessible to member functions of the class in which they're declared.

ANS: True.

d) Variables declared in the body of a particular member function are known as data members and can be used in all member functions of the class.

ANS: False. Such variables are local variables and can be used only in the member function in which they're declared.

e) Every function's body is delimited by left and right braces (`{` and `}`).

ANS: True.

f) The types of arguments in a function call must be consistent with the types of the corresponding parameters in the function's parameter list.

ANS: True.

3.3 What is the difference between a local variable and a data member?

ANS: A local variable is declared in the body of a function and can be used only from its declaration to the closing brace of the block in which it's declared. A data member is declared in a class, but not in the body of any of the class's member functions. Every object of a class has each of the class's data members. Data members are accessible to all member functions of the class.

3.4 Explain the purpose of a function parameter. What's the difference between a parameter and an argument?

ANS: A parameter represents additional information that a function requires to perform its task. Each parameter required by a function is specified in the function header. An argument is the value supplied in the function call. When the function is called, the argument value is passed into the function parameter so that the function can perform its task.

Exercises

NOTE: Solutions to the programming exercises are located in the **`ch03solutions`** *folder.*

3.5 *(Default Constructor)* What's a default constructor? How are an object's data members initialized if a class has only a default constructor defined by the compiler?

ANS: A default constructor is the constructor that is called when you create an object without specifying any arguments to a constructor. You can write a default constructor that initializes data members any way you choose. The compiler provides a default constructor with no parameters in any class that does not explicitly include a constructor. The default constructor provided by the compiler creates an object without assigning any initial values to the object's fundamental type data members—data members must be initialized using the object's member functions. Every constructor implicitly calls the constructors of any data member objects before the body of the class's constructor executes.

3.6 *(Data Members)* Explain the purpose of a data member.

ANS: A class provides a data member (or several data members) when each object of the class must maintain data separately from all other objects of the class. For example, a class called `Account` that represents a bank account provides a data member to represent the balance of the account. Each `Account` object maintains its own balance but does not know the balances of the bank's other accounts.

3.7 *(Using a Class Without a `using` Directive)* Explain how a program could use class `string` without inserting a `using` directive.

ANS: A program could create `string` variables without a `using` declaration if each occurrence of the word `string` were prefixed by the namespace `std` and the binary scope resolution operator (`::`), as in `std::string`.

3.8 *(Set and Get Functions)* Explain why a class might provide a *set* function and a *get* function for a data member.

ANS: A data member is typically declared `private` in a class so that only the member functions of the class in which the data member is declared can manipulate the variable. A class typically provides a *set* function and a *get* function for a data member to allow clients of the class to manipulate the data member in a controlled manner. A *set* function should validate the data it is setting to ensure that invalid data is not placed in the object. A *get* function can return the value of a data member without allowing clients to interact with the data member directly. In addition, *set* and *get* functions hide the internal data representation. For example, a `Time` class might represent the time as the total number of seconds since midnight. Such a class can provide member functions `getHour`, `getMinute` and `getSecond` to calculate the hour, minute and second, respectively—even if the underlying data representation does not contain `hour`, `minute` and `second` data members. Similarly, a `Time` class could provide member functions `setHour`, `setMinute` and `setSecond` to set the current hour, minute or second value. This would require each *set* function to perform calculations that update the data member containing the total number of seconds since midnight.

3.14 *(C++11 List Initializers)* Write a statement that uses list initialization to initialize an object of class `Account` which provides a constructor that receives an `unsigned int`, two `strings` and a `double` to initialize the `accountNumber`, `firstName`, `lastName` and `balance` data members of a new object of the class.

ANS: Instructor note: We initially introduced the term "list initializer" in this chapter, but this was moved later before publication. This problem statement should have simply asked the reader to "Write a statement that initializes an object of class `Account` which provides a constructor that receives an `unsigned int`, two `strings` and a `double` to initialize the `accountNumber`, `firstName`, `lastName` and `balance` data members of a new object of the class."

ANS:
```
Account account1{100, "Jane", "Green", 50.00};
```