When you declare a strongly typed enum, you can optionally specify any integer data type as the underlying type. You simply write a colon (:) after the enum name, followed by the desired data type. For example, the following statement declares an enum that uses the char data type for its enumerators:

```cpp
enum class Day : char { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY }
```

The following statement shows another example. This statement declares an enum named Water that uses unsigned as the data type of its enumerators. Additionally, values are assigned to the enumerators.

```cpp
enum class Water : unsigned { FREEZING = 32, BOILING = 212 }
```

**Checkpoint**

11.21 Look at the following declaration.

```cpp
enum Flower { ROSE, DAISY, PETUNIA }
```

In memory, what value will be stored for the enumerator ROSE? For DAISY? For PETUNIA?

11.22 What will the following code display?

```cpp
enum { HOBBIT, ELF = 7, DRAGON };
cout << HOBBIT << " " << ELF << " " << DRAGON << endl;
```

11.23 Does the enumerated data type declared in Checkpoint Question 11.22 have a name, or is it anonymous?

11.24 What will the following code display?

```cpp
enum Letters { Z, Y, X };
if (Z > X)
    cout << "Z is greater than X. \n";
else
    cout << "Z is not greater than X. \n";
```

11.25 Will the following code cause an error, or will it compile without any errors? If it causes an error, rewrite it so it compiles.

```cpp
enum Color { RED, GREEN, BLUE };
Color c;
c = 0;
```

11.26 Will the following code cause an error, or will it compile without any errors? If it causes an error, rewrite it so it compiles.

```cpp
enum Color { RED, GREEN, BLUE };
Color c = RED;
c++;
```

**NOTE:** For an additional example of this chapter’s topics, see the High Adventure Travel Part 2 Case Study on this book’s companion Web site at pearsonhighered.com/gaddis.