Note: *C How to Program*, Chapter 16 is a copy of *C++ How to Program* Chapter 3. We have not renumbered the PowerPoint Slides.

Chapter 3 Introduction to Classes, Objects and Strings

C++ How to Program, 8/e

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 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()
13    {
14       cout << "Welcome to the Grade Book!" << endl;
15    } // end function displayMessage
16 }; // end class GradeBook
17
```

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

```
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
```

```
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.)

3.3 Defining a Member Function with a Parameter (cont.)

- Fig. 3.3 redefines class `GradeBook` (lines 9–18) with a `displayMessage` member function (lines 13–17) that displays the course name as part of the welcome message.
  - The new version of `displayMessage` requires a parameter (`courseName` in line 13) that represents the course name to output.
- A variable of type `string` represents a string of characters.
- A string is actually an object of the C++ Standard Library class `string`.
  - Defined in header `<string>` and part of namespace `std`.
  - For now, you can think of `string` variables like variables of other types such as `int`.
  - Additional `string` capabilities in Section 3.9.

```
1  // Fig. 3.3: fig03_03.cpp
2  // Define class GradeBook with a member function that takes a parameter,
3  // create a GradeBook object and call its displayMessage function.
4  #include <iostream>
5  #include <string> // program uses C++ standard string class
6  using namespace std;
7
8  // GradeBook class definition
9  class GradeBook
10 {
11 public:
12    // function that displays a welcome message to the GradeBook user
13    void displayMessage( string courseName )
14    {
15       cout << "Welcome to the grade book for\n" << courseName << "!"
16          << endl;
17    } // end function displayMessage
18 }; // end class GradeBook
19
```

Fig. 3.3 | Define class `GradeBook` with a member function that takes a parameter, create a `GradeBook` object and call its `displayMessage` function. (Part 1 of 3.)

```
20  // function main begins program execution
21  int main()
22  {
23     string nameOfCourse; // string of characters to store the course name
24     GradeBook myGradeBook; // create a GradeBook object named myGradeBook
25
26     // prompt for and input course name
27     cout << "Please enter the course name:" << endl;
28     getline( cin, nameOfCourse ); // read a course name with blanks
29     cout << endl; // output a blank line
30
31     // call myGradeBook's displayMessage function
32     // and pass nameOfCourse as an argument
33     myGradeBook.displayMessage( nameOfCourse );
34  } // end main
```

Fig. 3.3 | Define class GradeBook with a member function that takes a parameter, create a GradeBook object and call its displayMessage function. (Part 2 of 3.)

```
Please enter the course name:
CS101 Introduction to C++ Programming

Welcome to the grade book for
CS101 Introduction to C++ Programming!
```

Fig. 3.3 | Define class GradeBook with a member function that takes a parameter, create a GradeBook object and call its displayMessage function. (Part 3 of 3.)

3.4 Data Members, *set Functions and get Functions (cont.)*

- An object has attributes that are carried with it as it's used in a program.
  - Such attributes exist throughout the life of the object.
  - A class normally consists of one or more member functions that manipulate the attributes that belong to a particular object of the class.
- Attributes are represented as variables in a class definition.
  - Such variables are called data members and are declared inside a class definition but outside the bodies of the class's member-function definitions.
- Each object of a class maintains its own copy of its attributes in memory.

3.4 Data Members, *set Functions and get Functions (cont.)*

- A typical instructor teaches multiple courses, each with its own course name.
- A variable that is declared in the class definition but outside the bodies of the class's member-function definitions is a data member.
- Every instance (i.e., object) of a class contains one copy of each of the class's data members.
- A benefit of making a variable a data member is that all the member functions of the class can manipulate any data members that appear in the class definition.

```
1  // Fig. 3.5: fig03_05.cpp
2  // Define class GradeBook that contains a courseName data member
3  // and member functions to set and get its value;
4  // Create and manipulate a GradeBook object with these functions.
5  #include <iostream>
6  #include <string> // program uses C++ standard string class
7  using namespace std;
8
9  // GradeBook class definition
10 class GradeBook
11 {
12 public:
13    // function that sets the course name
14    void setCourseName( string name )
15    {
16       courseName = name; // store the course name in the object
17    } // end function setCourseName
18
19    // function that gets the course name
20    string getCourseName()
21    {
22       return courseName; // return the object's courseName
23    } // end function getCourseName
```

Fig. 3.5 | Defining and testing class GradeBook with a data member and *set* and *get* functions. (Part 1 of 3.)

```
24
25    // function that displays a welcome message
26    void displayMessage()
27    {
28       // this statement calls getCourseName to get the
29       // name of the course this GradeBook represents
30       cout << "Welcome to the grade book for\n" << getCourseName() << "!"
31          << endl;
32    } // end function displayMessage
33 private:
34    string courseName; // course name for this GradeBook
35 }; // end class GradeBook
36
37 // function main begins program execution
38 int main()
39 {
40    string nameOfCourse; // string of characters to store the course name
41    GradeBook myGradeBook; // create a GradeBook object named myGradeBook
42
```

Fig. 3.5 | Defining and testing class GradeBook with a data member and *set* and *get* functions. (Part 2 of 3.)

```
43      // display initial value of courseName
44      cout << "Initial course name is: " << myGradeBook.getCourseName()
45         << endl;
46
47      // prompt for, input and set course name
48      cout << "\nPlease enter the course name:" << endl;
49      getline( cin, nameOfCourse ); // read a course name with blanks
50      myGradeBook.setCourseName( nameOfCourse ); // set the course name
51
52      cout << endl; // outputs a blank line
53      myGradeBook.displayMessage(); // display message with new course name
54   } // end main
```

```
Initial course name is:

Please enter the course name:
CS101 Introduction to C++ Programming

Welcome to the grade book for
CS101 Introduction to C++ Programming!
```

Fig. 3.5 | Defining and testing class GradeBook with a data member and *set* and *get* functions. (Part 3 of 3.)

3.5 Initializing Objects with Constructors

- Each class can provide a constructor that can be used to initialize an object of the class when the object is created.
- A constructor is a special member function that must be defined with the same name as the class, so that the compiler can distinguish it from the class's other member functions.
- An important difference between constructors and other functions is that constructors cannot return values, so they cannot specify a return type (not even `void`).
- Normally, constructors are declared `public`.

3.5 Initializing Objects with Constructors (cont.)

- C++ requires a constructor call for each object that is created, which helps ensure that each object is initialized before it's used in a program.
- The constructor call occurs implicitly when the object is created.
- If a class does not explicitly include a constructor, the compiler provides a default constructor—that is, a constructor with no parameters.


```
1  // Fig. 3.7: fig03_07.cpp
2  // Instantiating multiple objects of the GradeBook class and using
3  // the GradeBook constructor to specify the course name
4  // when each GradeBook object is created.
5  #include <iostream>
6  #include <string> // program uses C++ standard string class
7  using namespace std;
8
9  // GradeBook class definition
10 class GradeBook
11 {
12 public:
13    // constructor initializes courseName with string supplied as argument
14    GradeBook( string name )
15    {
16       setCourseName( name ); // call set function to initialize courseName
17    } // end GradeBook constructor
18
```

Fig. 3.7 | Instantiating multiple objects of the GradeBook class and using the GradeBook constructor to specify the course name when each GradeBook object is created. (Part 1 of 3.)

```
19     // function to set the course name
20     void setCourseName( string name )
21     {
22        courseName = name; // store the course name in the object
23     } // end function setCourseName
24
25     // function to get the course name
26     string getCourseName()
27     {
28        return courseName; // return object's courseName
29     } // end function getCourseName
30
31     // display a welcome message to the GradeBook user
32     void displayMessage()
33     {
34        // call getCourseName to get the courseName
35        cout << "Welcome to the grade book for\n" << getCourseName()
36           << "!" << endl;
37     } // end function displayMessage
```

Fig. 3.7 | Instantiating multiple objects of the GradeBook class and using the GradeBook constructor to specify the course name when each GradeBook object is created. (Part 2 of 3.)

```
38  private:
39     string courseName; // course name for this GradeBook
40  }; // end class GradeBook
41
42  // function main begins program execution
43  int main()
44  {
45     // create two GradeBook objects
46     GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
47     GradeBook gradeBook2( "CS102 Data Structures in C++" );
48
49     // display initial value of courseName for each GradeBook
50     cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
51        << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
52        << endl;
53  } // end main
```

```
gradeBook1 created for course: CS101 Introduction to C++ Programming
gradeBook2 created for course: CS102 Data Structures in C++
```

Fig. 3.7 | Instantiating multiple objects of the GradeBook class and using the GradeBook constructor to specify the course name when each GradeBook object is created. (Part 3 of 3.)

3.6 Placing a Class in a Separate File for Reusability (cont.)

- Each of the previous examples in the chapter consists of a single `.cpp` file, also known as a source-code file, that contains a `GradeBook` class definition and a `main` function.
- When building an object-oriented C++ program, it's customary to define reusable source code (such as a class) in a file that by convention has a `.h` filename extension—known as a header.
- Programs use `#include` preprocessor directives to include headers and take advantage of reusable software components.


```
1  // Fig. 3.9: GradeBook.h
2  // GradeBook class definition in a separate file from main.
3  #include <iostream>
4  #include <string> // class GradeBook uses C++ standard string class
5  using namespace std;
6
7  // GradeBook class definition
8  class GradeBook
9  {
10 public:
11    // constructor initializes courseName with string supplied as argument
12    GradeBook( string name )
13    {
14       setCourseName( name ); // call set function to initialize courseName
15    } // end GradeBook constructor
16
17    // function to set the course name
18    void setCourseName( string name )
19    {
20       courseName = name; // store the course name in the object
21    } // end function setCourseName
22
```

Fig. 3.9 | GradeBook class definition in a separate file from `main`. (Part 1 of 2.)

```
23     // function to get the course name
24     string getCourseName()
25     {
26        return courseName; // return object's courseName
27     } // end function getCourseName
28
29     // display a welcome message to the GradeBook user
30     void displayMessage()
31     {
32        // call getCourseName to get the courseName
33        cout << "Welcome to the grade book for\n" << getCourseName()
34           << "!" << endl;
35     } // end function displayMessage
36  private:
37     string courseName; // course name for this GradeBook
38  }; // end class GradeBook
```

Fig. 3.9 | GradeBook class definition in a separate file from `main`. (Part 2 of 2.)

```
1  // Fig. 3.10: fig03_10.cpp
2  // Including class GradeBook from file GradeBook.h for use in main.
3  #include <iostream>
4  #include "GradeBook.h" // include definition of class GradeBook
5  using namespace std;
6
7  // function main begins program execution
8  int main()
9  {
10    // create two GradeBook objects
11    GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
12    GradeBook gradeBook2( "CS102 Data Structures in C++" );
13
14    // display initial value of courseName for each GradeBook
15    cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
16       << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
17       << endl;
18 } // end main
```

```
gradeBook1 created for course: CS101 Introduction to C++ Programming
gradeBook2 created for course: CS102 Data Structures in C++
```

Fig. 3.10 | Including class GradeBook from file `GradeBook.h` for use in `main`.

3.7 Separating Interface from Implementation

- Interfaces define and standardize the ways in which things such as people and systems interact with one another.
- The interface of a class describes what services a class's clients can use and how to request those services, but not how the class carries out the services.
- A class's `public` interface consists of the class's `public` member functions (also known as the class's `public` services).

```
1  // Fig. 3.11: GradeBook.h
2  // GradeBook class definition. This file presents GradeBook's public
3  // interface without revealing the implementations of GradeBook's member
4  // functions, which are defined in GradeBook.cpp.
5  #include <string> // class GradeBook uses C++ standard string class
6  using namespace std;
7
8  // GradeBook class definition
9  class GradeBook
10 {
11 public:
12    GradeBook( string ); // constructor that initializes courseName
13    void setCourseName( string ); // function that sets the course name
14    string getCourseName(); // function that gets the course name
15    void displayMessage(); // function that displays a welcome message
16 private:
17    string courseName; // course name for this GradeBook
18 }; // end class GradeBook
```

Fig. 3.11 | GradeBook class definition containing function prototypes that specify the interface of the class.

3.7 Separating Interface from Implementation (cont.)

- Source-code file `GradeBook.cpp` (Fig. 3.12) defines class `GradeBook`'s member functions, which were declared in lines 12–15 of Fig. 3.11.
- Notice that each member-function name in the function headers (lines 9, 15, 21 and 27) is preceded by the class name and `::`, which is known as the binary scope resolution operator.
- This "ties" each member function to the (now separate) `GradeBook` class definition (Fig. 3.11), which declares the class's member functions and data members.

```
1  // Fig. 3.12: GradeBook.cpp
2  // GradeBook member-function definitions. This file contains
3  // implementations of the member functions prototyped in GradeBook.h.
4  #include <iostream>
5  #include "GradeBook.h" // include definition of class GradeBook
6  using namespace std;
7
8  // constructor initializes courseName with string supplied as argument
9  GradeBook::GradeBook( string name )
10 {
11    setCourseName( name ); // call set function to initialize courseName
12 } // end GradeBook constructor
13
14 // function to set the course name
15 void GradeBook::setCourseName( string name )
16 {
17    courseName = name; // store the course name in the object
18 } // end function setCourseName
19
```

Fig. 3.12 | GradeBook member-function definitions represent the implementation of class `GradeBook`. (Part 1 of 2.)

```
20  // function to get the course name
21  string GradeBook::getCourseName()
22  {
23     return courseName; // return object's courseName
24  } // end function getCourseName
25
26  // display a welcome message to the GradeBook user
27  void GradeBook::displayMessage()
28  {
29     // call getCourseName to get the courseName
30     cout << "Welcome to the grade book for\n" << getCourseName()
31        << "!" << endl;
32  } // end function displayMessage
```

Fig. 3.12 | GradeBook member-function definitions represent the implementation of class `GradeBook`. (Part 2 of 2.)

```
1  // Fig. 3.13: fig03_13.cpp
2  // GradeBook class demonstration after separating
3  // its interface from its implementation.
4  #include <iostream>
5  #include "GradeBook.h" // include definition of class GradeBook
6  using namespace std;
7
8  // function main begins program execution
9  int main()
10 {
11    // create two GradeBook objects
12    GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
13    GradeBook gradeBook2( "CS102 Data Structures in C++" );
14
15    // display initial value of courseName for each GradeBook
16    cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
17       << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
18       << endl;
19 } // end main
```

Fig. 3.13 | GradeBook class demonstration after separating its interface from its implementation. (Part 1 of 2.)

```
gradeBook1 created for course: CS101 Introduction to C++ Programming
gradeBook2 created for course: CS102 Data Structures in C++
```

Fig. 3.13 | GradeBook class demonstration after separating its interface from its implementation. (Part 2 of 2.)

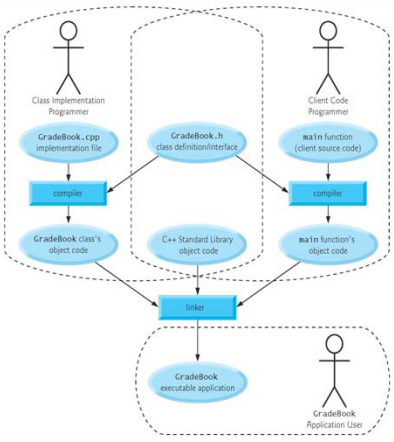

Fig. 3.14 | Compilation and linking process that produces an executable application.

3.8 Validating Data with *set Functions*

- The program of Figs. 3.15–3.17 enhances class `GradeBook`'s member function `setCourseName` to perform validation (also known as validity checking).
- Since the interface of the clas remains unchanged, clients of this class need not be changed when the definition of member function `setCourseName` is modified.
- This enables clients to take advantage of the improved `GradeBook` class simply by linking the client code to the updated `GradeBook`'s object code.

```
1  // Fig. 3.16: GradeBook.cpp
2  // Implementations of the GradeBook member-function definitions.
3  // The setCourseName function performs validation.
4  #include <iostream>
5  #include "GradeBook.h" // include definition of class GradeBook
6  using namespace std;
7
8  // constructor initializes courseName with string supplied as argument
9  GradeBook::GradeBook( string name )
10 {
11    setCourseName( name ); // validate and store courseName
12 } // end GradeBook constructor
13
```

Fig. 3.16 | Member-function definitions for class GradeBook with a *set* function that validates the length of data member courseName. (Part 1 of 3.)

```
14 // function that sets the course name;
15 // ensures that the course name has at most 25 characters
16 void GradeBook::setCourseName( string name )
17 {
18    if ( name.length() <= 25 ) // if name has 25 or fewer characters
19       courseName = name; // store the course name in the object
20
21    if ( name.length() > 25 ) // if name has more than 25 characters
22    {
23       // set courseName to first 25 characters of parameter name
24       courseName = name.substr( 0, 25 ); // start at 0, length of 25
25
26       cout << "Name \"" << name << "\" exceeds maximum length (25).\n"
27          << "Limiting courseName to first 25 characters.\n" << endl;
28    } // end if
29 } // end function setCourseName
30
```

Fig. 3.16 | Member-function definitions for class GradeBook with a *set* function that validates the length of data member courseName. (Part 2 of 3.)

```
1  // Fig. 3.17: fig03_17.cpp
2  // Create and manipulate a GradeBook object; illustrate validation.
3  #include <iostream>
4  #include "GradeBook.h" // include definition of class GradeBook
5  using namespace std;
6
7  // function main begins program execution
8  int main()
9  {
10    // create two GradeBook objects;
11    // initial course name of gradeBook1 is too long
12    GradeBook gradeBook1( "CS101 Introduction to Programming in C++" );
13    GradeBook gradeBook2( "CS102 C++ Data Structures" );
14
15    // display each GradeBook's courseName
16    cout << "gradeBook1's initial course name is: "
17       << gradeBook1.getCourseName()
18       << "\ngradeBook2's initial course name is: "
19       << gradeBook2.getCourseName() << endl;
20
21    // modify myGradeBook's courseName (with a valid-length string)
22    gradeBook1.setCourseName( "CS101 C++ Programming" );
23
```

Fig. 3.17 | Creating and manipulating a GradeBook object in which the course name is limited to 25 characters in length. (Part 1 of 2.)

```
24     // display each GradeBook's courseName
25     cout << "\ngradeBook1's course name is: "
26        << gradeBook1.getCourseName()
27        << "\ngradeBook2's course name is: "
28        << gradeBook2.getCourseName() << endl;
29  } // end main
```

```
Name "CS101 Introduction to Programming in C++" exceeds maximum length (25).
Limiting courseName to first 25 characters.

gradeBook1's initial course name is: CS101 Introduction to Pro
gradeBook2's initial course name is: CS102 C++ Data Structures

gradeBook1's course name is: CS101 C++ Programming
gradeBook2's course name is: CS102 C++ Data Structures
```

Fig. 3.17 | Creating and manipulating a GradeBook object in which the course name is limited to 25 characters in length. (Part 2 of 2.)