PEN203

Introduction to Classes and Objects

C How to Program
Deitel & Deitel

Outline

- Defining a Class with a Member Function
- Defining a Member Function with a Parameter
- Data Members, set Functions and get Functions
- Initializing Objects with Constructors
- Separating Interface from Implementation

Defining a Class with a Member Function

- Class definition informs compiler about the member functions and the data members of the class.
- Keyword class is used with the class name to create a new class
- Class body is placed in braces { }
- Access specifier public shows that a member function or data member can be accessed by other functions.

Defining a Class with a Member Function

```
1  // Fig. 19.1: fig19_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 std::cout;
6  using std::endl;
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()
14    {
15       cout << "Welcome to the Grade Book!" << endl;
16    } // end function displayMessage
17 }; // end class GradeBook
18
19 // function main begins program execution
20 int main()
21 {
22    GradeBook myGradeBook; // create a GradeBook object named myGradeBook
23    myGradeBook.displayMessage(); // call object's displayMessage function
24    return 0; // indicate successful termination
25 } // end main
```

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

Defining a Class with a Member Function

- Member function
  - A return type should be provided.
  - void shows that the function does not return any value.
  - Function name must be a valid identifier.
  - Function body delimited by braces.
  - Defining a function inside another function is a syntax error.

Defining a Member Function with a Parameter

- Function parameters
  - Placed in parentheses that follows the function name
  - Additional information needed by a function to complete its task
  - Any number of parameters (comma separated list)
  - Number, order and types of arguments in a function call must match the parameters in function parameter list.
- Function arguments
  - Values passed by function call for each function parameter

Data Members, set Functions and get Functions

- Local variables are declared in function definition's body
- They cannot be used outside of that function body
- The values of local variables are lost when a function terminates

Data Members, set Functions and get Functions

- Attributes are the properties of the object
- They are represented as data members
- They are actually variables in a class definition
- Each object of class has its own copy of attributes

Data Members, set Functions and get Functions

- Access specifier private
  - private data members or member functions can only be used in member functions of the class.
  - private is the default access specifier
- Access specifiers public and private may be repeated and intermixed. However it is not a good programming practice.

Data Members, set Functions and get Functions

- public member functions enable user of the class to set or get the values of private data members
- set functions sometimes are called as mutators
- get functions sometimes are called as accessors
- set and get functions should be used by other member functions of the same class

Initializing Objects with Constructors

- Constructors
  - They are special functions.
  - They are called implicitly and automatically when an object of class is created.
  - Name of the constructor must be same with the class.
  - They have no return types. They do not return values, not even void.
  - Default constructor has no parameters.

Initializing Objects with Constructors

```
1  // Fig. 19.7: fig19_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  using std::cout;
7  using std::endl;
8
9  #include <string> // program uses C++ standard string class
10 using std::string;
11
12 // GradeBook class definition
13 class GradeBook
14 {
15 public:
16    // constructor initializes courseName with string supplied as argument
17    GradeBook( string name )
18    {
19       setCourseName( name ); // call set function to initialize courseName
20    } // end GradeBook constructor
21
22    // function to set the course name
23    void setCourseName( string name )
24    {
25       courseName = name; // store the course name in the object
26    } // end function setCourseName
27
```

Initializing Objects with Constructors

```
28    // function to get the course name
29    string getCourseName()
30    {
31       return courseName; // return object's courseName
32    } // end function getCourseName
33
34    // display a welcome message to the GradeBook user
35    void displayMessage()
36    {
37       // call getCourseName to get the courseName
38       cout << "Welcome to the grade book for\n" << getCourseName()
39          << "!" << endl;
40    } // end function displayMessage
41 private:
42    string courseName; // course name for this GradeBook
43 }; // end class GradeBook
44
```

Initializing Objects with Constructors

```
45 // function main begins program execution
46 int main()
47 {
48    // create two GradeBook objects
49    GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
50    GradeBook gradeBook2( "CS102 Data Structures in C++" );
51
52    // display initial value of courseName for each GradeBook
53    cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
54       << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
55       << endl;
56    return 0; // indicate successful termination
57 } // end main
```

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

Separating Interface from Implementation

- Interface
  - A class's interface consists of the class's public member functions
  - It describe which of the services are available and the way of requesting those services.
  - The implementation of the functions are not given to clients.
  - If implementation changes, the client code should not be affected.