Ekaba Bisong
Programming in C++
University of Calabar

Tutorial Notes #15
May 13, 2015

Placing a Class in a Separate File for Reusability

Placing a Class in a Separate File for Reusability

One of the benefits of creating class definitions is that, when packaged properly, your classes can be reused by other programmers. For example, you can reuse C++ Standard Library type string in any C++ program by including the header <string> (and, as you'll see, by being able to link to the library's object code).

Headers

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, such as type string provided in the C++ Standard Library and user-defined types.

```
1  //
2  //  GradeBook.h
3  //  GradeBook
4  //  GradeBook class definition in a separate file from main.
5
6  #include <iostream>
7  #include <string> // class GradeBook uses C++ standard string class
8  using namespace std;
9
10 // GradeBook class definition
11 class GradeBook
12 {
13     public:
14     // constructor initializes courseName with string supplied as argument
15     GradeBook( string name )
16     {
17         setCourseName( name ); // call set function to initialize courseName
18     } // end GradeBook constructor
19
20
21     // function to set the course name
22     void setCourseName( string name )
23     {
24         courseName = name; // store the course name in the object
25     } // end function setCourseName
26
27     // function to get the course name
28     string getCourseName()
29     {
30         return courseName; // return object's courseName
31     } // end function getCourseName
32
33     // display a welcome message to the GradeBook user
34     void displayMessage() {
35         // call getCourseName to get the courseName
36         cout << "Welcome to the grade book for\n" << getCourseName() << "!" << endl;
37     } // end function displayMessage
38
39 private:
40     string courseName; // course name for this GradeBook
41 }; // end class GradeBook
```

```
1  //
2  //  main.cpp
3  //  GradeBook
4  //
5  //  Define class GradeBook with a member function displayMessage,
6  //  create a GradeBook object, and call its displayMessage function.
7  //
8
9  #include <iostream>
10 #include "GradeBook.h" // include definition of class GradeBook
11 using namespace std;
12
13 // function main begins program execution
14 int main()
15 {
16     // create two GradeBook objects
17     GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
18     GradeBook gradeBook2( "CS102 Data Structures in C++" );
19
20     // display initial value of courseName for each GradeBook
21     cout << "gradeBook1 created for course: " << gradeBook1.getCourseName()
22     << "\ngradeBook2 created for course: " << gradeBook2.getCourseName()
23     << endl;
24 }   //endmain
```

How Headers Are Located

- Notice that the name of the GradeBook.h header in line 10 is enclosed in quotes (" ") rather than angle brackets (< >).
- Normally, a program's source-code files and user- defined headers are placed in the same directory.
- When the preprocessor encounters a header name in quotes, it attempts to locate the header in the same directory as the file in which the ***#include*** directive appears.
- If the preprocessor cannot find the header in that directory, it searches for it in the same location(s) as the C++ Standard Library headers.
- When the preprocessor encounters a header name in angle brackets (e.g., <iostream>), it assumes that the header is part of the C++ Standard Library and does not look in the directory of the program that's being preprocessed.