Lab 7

Copy Constructor, Overloaded Assignment and Conversion

Introduction

The goal of these exercises is to reinforce your understanding of the role of copy constructors, overloaded assignment operators and conversions in C++ programs. You will study the starter program, and then you'll modify the **String** class to make it more efficient.

Suggested Time: 30 minutes.

Exercise 1

In this exercise you will study the starter code and its output to reinforce your understanding of the role of copy constructors, overloaded assignment operators and conversions in C++ programs.

1. Study the starter code. Before running the program try to predict the output, including all constructors, destructors, assignments, and overloaded cast operations. Build and run the program. Redirect the output to a file so that you can save and study it.
2. How many copy constructors are invoked?
3. What conversions are involved in executing **PrintString("Goodbye")**?
4. What conversions are involved in executing **PrintCharPtr(a)**?
5. Do you need the **GetString()** method any longer?

Exercise 2

In this exercise you'll modify the class by adding second overloaded assignment operator, which will make assigning a character pointer to a **String** more efficient. The first implementation is not completely adequate, so you'll provide a more robust implementation. Finally, you can remove the **GetString()** and **SetString()** methods, which are no longer needed.

1. Add a trace statement to the **GetString()** method so that you can compare its usage with the overloaded cast operator. Build and run the program. Note that **GetString()** is used in **PrintString()** and the overloaded cast operator is used in **Print2Strings()**. Save the output to a file so that you can study it and compare with other program variations.

2. Replace the code in **PrintString()** with code that uses the overloaded cast operator instead of **GetString()**. Build and run. Compare the output with that from the previous run.

3. It should be apparent that the results are equivalent. Remove **GetString()** from both the header and code files for the **String** class. Make sure you get a clean build and run.

4. Next, create a second overloaded assignment operator that will assign a character pointer to a **String**. .

5. Build and run. Save the output to a file and compare with the previous run. What do you notice?

6. The previous code required three calls to implement the assignment of a character pointer to a **String**.

7. You can now uncomment the line **PrintCharPtr(a)**; in DemoConv,cpp. Build and run.

8. 3. Replace the two lines
   a = "Hello";
   b = a;

   by the single line
   b = a = "Hello"

9. Build the program. What is wrong? Fix the problem.

10. When attempting to do the assignments on one line the compile fails, because the overloaded **operator=(const char *)** returns a **void**. To fix the problem, make it return a **String**, following the pattern of **operator=(const String&)**. (But we don't return a reference in this case.)