

CSE 332 Studio Session on Basic C++ Memory Management

These studio exercises are intended to introduce basic memory management concepts and techniques in C++, and to give you experience using those concepts and techniques in the Visual C++ environment.

In this studio you will again work in self-selected small groups of 2 or 3 people. As before, students who are more familiar with the material are encouraged to help those for whom it is less familiar, and asking questions during the studio sessions is highly encouraged as well. Please record your answers as you work through the following exercises. After you have finished please post your answers to the required exercises, and to any of the enrichment exercises you completed, to the course message board as a reply to my posting titled “Basic Memory Management Studio”. The enrichment exercises are optional but are a good way to dig into the material a little deeper, especially if you breeze through the required ones.

Please make sure as you work through these exercises that each team member has a chance to participate actively – e.g., take turns coding, looking up details, debugging, etc., and please also refer to the slides and the posted code examples as you work – working through this material is crucial to your understanding and success with much of the material that will follow in the course.

PART I: REQUIRED EXERCISES

1. Find your team members in the studio area, sit down at/around and log in to one of the Windows machines, open up Visual Studio and create a new Visual C++ Win32 Console Application project. Change the signature of the main function in the source file that Visual C++ generated to match the one that was specified for the previous studios, and write down the names of the team members who are present (please catch up anyone arriving late on the work, and also add their name) as the answer to this exercise.
2. Add a new header file and source file to your project, and in those respectively declare and define a class that you will use throughout this studio to track object creation and destruction. The class should have a private unsigned integer member variable, another private unsigned integer member variable that is *static*, a public default constructor, copy constructor, assignment operator, and destructor, and a public accessor method for the **non-static** member variable: (1) The default constructor should initialize the non-static member variable with the value of the static member variable, then increment the value of the static member variable, and print out a message with the method name, the object’s address, and the value of the initialized non-static member variable, to `cout`. (2) The copy constructor should do the same thing as the default constructor, but should print out the non-static member variable values of both the current object and the object that was passed to it, to `cout`. (3) The destructor should print out the method name, address, and the value of the non-static member variable, to `cout`, and should do nothing else. (4) The assignment operator should print out the method name, address, and non-static member variable values of both the current object and the object that was passed to it, to `cout`, and should do nothing else. As the answer to this exercise, please give the name of the class you declared and defined.

3. In your main function put one output statement at the very beginning and one at the very end of the function to mark when the function begins and ends (don't forget to use **endl** to make sure the message is flushed out to the screen). In between the output statements, declare local objects of the class you developed in exercise 3 using both default construction and copy construction. Also assign one object to another. Build and run your program, and give its output as the answer to this exercise.

4. Using your code from exercise 4 as the starting point, above (and outside) your main function, declare a global object of the class you developed in exercise 3. Build and run the program, and as the answer to this exercise, describe how the lifetimes of the global vs. local variables in this exercise differ. (**hint:** in this exercise and the exercises that follow it, you may want to use additional output statements to narrow down exactly when objects are being created and destroyed).

5. Between the beginning and ending output statements in your main function, declare a pointer to the class type and initialize it using **new** (which will obtain and return the address of a new object from the heap). After that but before the ending output statement in your main function, call **delete** on the pointer (which will destroy the object and return the memory it used to the heap). As the answer to this exercise, please answer the following questions: (a) if you move the creation and destruction of the heap object to different positions relative to the declaration of the local object? (b) if you leave off the line that deletes the pointer, what happens (or more specifically, what doesn't happen)?

6. Before starting this exercise, comment out all of the lines in your main function. Declare a pointer to the class type and initialize it using **new[]** (the array version of **new**), so that the pointer points to the beginning of an array of 3 class objects. As the answer to this exercise, please answer the following questions: (a) after allocating the array, what happens if you call **delete[]** on the pointer? (b) what happens instead if you instead use **delete** (without the square brackets)? (c) when using **new[]** and **delete[]**, what is the relationship between the order in which the objects are created vs. destroyed?

PART II: ENRICHMENT EXERCISES (optional, please do the ones that interest you).

7. Again comment out all of the lines in your main function. In your main source file, define a function that has a local object of the class you developed in exercise 2, and returns that object by value. In your main function, declare a local variable and initialize it using the result of calling the function you just developed. As the answer to this exercise, say (a) how many objects are created and destroyed, and explain (b) when and (c) why each of them was created and destroyed.

8. Change the function you developed in exercise 7 to return by reference instead of by value, and in your main function, change the local variable to be a reference instead of an object. As the answer to this exercise, say (a) how many objects were created and destroyed, and (b) whether the reference in your main function refers to an existing object or not, once it has been initialized.