Data Structures (2028C) -- Spring 2025 – Lab 3

Topics covered: Classes, multi-file programs and operator overloading

Lab due: ***Sunday, Feb 9 at 11:55PM for Monday Section***
Tuesday, Feb 11 at 11:55 PM for Wednesday Section

Objective:

The objective of this Lab is to create a class including constructors, using correct class file separation and implement operator overloading.

Task 1: Design Class (.h file)

1. Create a new project. You can name this whatever you like.
2. Design a class to allow calculations on a new measurement standard (completely made up). This measurement standard consists of little, lot, and heap. Lot consists of 7 littles and heap consists of 23 lots.
    a. Declare a default constructor and an overload of a constructor that supports passing each of the attributes in littles, lots and heaps and in just littles (total of 3 constructor versions).
    b. Make all attributes (variables) private.
    c. Declare getters and setters for all attributes as appropriate.
    d. Declare a print member function to print out the value using cout.
3. Include in the submission your rationale for all members you have in your class.

Task 2: Implement the Class

1. Write the implementation code for the methods declared in Task 1. The definitions should be in a separate .cpp file.
2. Include in the lab report any changes you needed to make to the class declaration (.h file).

Task 3: Extend the class.

1. Modify the class declaration and definition to overload the +, -, *, / and == operators to correctly perform those calculations.
    a. The +, -, *, and / operators should accept an input parameter of the same class (Task 1) type and return void. This operation should update the contents of that instance. This should ensure littles and lots are below 7 and lots are below 23 when complete.
    b. The == operator should accept an input parameter of the same class type (Task 1) and return a bool indicating if the value is equivalent.
    c. Create an operator overload for converting the class to a string so it can be used in a cout.
2. Modify the class declaration and definition to include a member function returning the distance in meters.
3. Include in the lab report any changes you needed to make to the class declaration beyond

the new functions being defined in this task

Task 4: Test the class.

1. Create a program that tests the class.
    a. Prompt the user for x and y values for the distance.
    b. Prompt the user for the operation to perform.
    c. Prompt the user for the values for other class instance to be used in the calculation.
    d. Display the results.
    e. Ask the user if they wish to continue. If so, loop to step b.
2. Use your test program to test all member functions and ensure the class is working correctly.
3. Include in the lab report a screen shot(s) of the output of a test of all operator overload functions.

Lab Submission:

1. Write a lab report including the following information:
    a. A description of the objectives/concepts explored in this assignment including why you think they are important to this course and a career in CS and/or Engineering.
    b. Why you designed the class the way you did initially, what changes you made because of each task and what considerations you consider important when designing classes.
2. Include all source code from all tasks, input and output files (if any), and any special instructions to compile and run those programs.
3. Package all files in a single zip folder and submit the file to Canvas.

Lab Grading:

1. 20% - Lab attendance
2. 5% - Task 1 has been correctly implemented and meets all requirements.
3. 15% - Task 2 has been correctly implemented and meets all requirements.
4. 30% - Task 3 has been correctly implemented and meets all requirements.
5. 10% - Task 4 has been correctly implemented and meets all requirements.
6. 20% - Lab report contains all required information and is well written.

If program fails to compile, 0% will be given for that Task.