Lab 3 – Classes in C++

Computer Science 2413 – Data Structures – Fall 2019

Due by 11:59 pm CST on Wednesday, 11 September 2019

This lab is individual work. Each student must complete this assignment independently.

User Request:

"Create a simple program to read, store, and write drilling data."

Objectives:

1	Satisfy all requirements of Lab 2, except for the requirement to use a function.	*4 points*
2	Create a class that encapsulates dynamically resized arrays.	*8 points*
3	Create a class that encapsulates drilling data.	*4 points*
4	Use proper design, coding style, documentation, and formatting.	*4 points*

Description:

This lab builds on Lab 2 by requiring the use of an important building block for C++ programs—classes. Lab 3 does this by substituting a new implementation requirement for Lab 3 for an old one from Lab 2. This requirement substitution defines *Driller 0.3*. Other than this requirement substitution, Driller 0.3 must satisfy all requirements of Driller 0.2.

Operational Issues:

From the user's perspective, Driller 0.3 will perform exactly the same as Driller 0.2.

Implementation Issues:

In contrast to the implementation requirements from Lab 2 which required the use of a function to modularize the process of doubling the size of the array used to hold drilling data, Lab 3 requires that functionality to be encapsulated within a class. This must be a fully formed class with two constructors (a no argument constructor that starts at a default size of 10 and a constructor that takes a starting size as an argument), a destructor, accessor and mutator methods. The mutator method must allow for addition to the data stored (which must automatically double the size of the allocated memory when necessary). This class must be tested and shown to work correctly using unit testing in zyLabs, as well as work in the final, integrated program. In particular, you must be sure to test both constructors, the destructor, and the mutator that adds data, ensuring that this last method automatically doubles the size of the allocated memory, respectively.

Also in contrast with Lab 2 which required the use of a structure to hold data for each drilling record, Lab 3 requires the use of a class to encapsulate each drilling data record and to display it using an overloaded `operator<<`. This class will have two mutators and two accessors, one for each type of data that it holds (`int` and `std::string`).

The declaration/header (`.h`) files for these classes are provided for you. You are required to use them as they are, with no modifications. You will need to write the corresponding definition/implementation (`.cpp`) files for these classes, as well as the driver code including the `main` function, to satisfy all of these requirements. You do not need to implement error checking within this code beyond what was already required for Lab 2.

In addition to these specific requirements, you must carefully organize the data such that it is easy to manipulate (e.g., using looping constructs), easy to understand (by treating similar data in similar ways), and helps to preserve data integrity (e.g., by grouping related data together in structured ways).

You must use good programming style and documentation, including making your code modular, using explanatory comments for each section of code, using meaningful variable and method names, using consistent indentation, etc.

You must follow C++ conventions for compilation modules, including using a header (`.h`) file for constant variables, prototypes, etc. and a source (`.cpp`) file for implementations of functions, methods, etc.

You may write your program from scratch or may start from programs for which the source code is freely available on the web or through other sources (such as friends or student organizations). If you do not start from scratch, you must give a complete and accurate accounting of where all of your code came from and indicate which parts are original or changed, and which you got from which other source(s). Failure to give credit where credit is due is academic fraud and will be dealt with accordingly.

Due Date:

You must submit an electronic copy of your project to the appropriate lab in your zyBook by **11:59 pm CST on Wednesday, 11 September 2019**.