Lab 2 – Arrays, Pointers, and Functions in C++

Computer Science 2413 – Data Structures – Fall 2016
Due by 11:00 pm CST on Thursday, 8 September 2016

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

User Request:

"Create a simple program to read and write merchant vessel data, with additional error checking."

Objectives:

1	Satisfy all requirements of Lab 1.	*5 points*
2	Use arrays to store the data as it is read in.	*5 points*
3	Use pointers to reference space that is dynamically allocated for storing the data.	*5 points*
►	Develop and use an appropriate design, including the use of at least one function.	*5 points*
►	Use proper coding style, documentation, formatting, and unit testing.	*5 points*

Description:

This lab builds on Lab 1 by requiring the use of three important building blocks for C++ programs—arrays, pointers, and functions.[1] Lab 2 does this by adding two operational requirements and four implementation requirements to those already present in Lab 1. These additional requirements define MVP 0.2. Other than these requirements, MVP 0.2 must satisfy all requirements of MVP 0.1.

Operational Issues:

In addition to the row checking accomplished by MVP 0.1, MVP 0.2 must check to ensure that each numeric column total is correct by summing all data entries for each numeric column for each country and comparing these totals to the totals for each numeric column provided in the data file itself. For example, data column two is the total number of ships. For each country row, this is the number of ships registered to that country. For the final row, this is the total number of ships registered worldwide. The sum of all of the ships for all of the countries should equal the worldwide total. If MVP 0.2 encounters an error in a column total, it must provide an error message indicating at which column the error was found, and move on to the next column of data.

In contrast to MVP 1.0, MVP 2.0 will read and store all data before sending any output to `cout`. Once the data is read in and checked for internal consistency (checking the row and column totals, as described), the data will be sent to `cout` in reverse country order. Note that the two header lines will still be output first and the total line will be output last. However, all of the country lines will be reversed in order, with Yemen first and Albania last.

Implementation Issues:

In addition to the implementation requirements from Lab 1 (using MS Visual Studio, etc.), you must use arrays, pointers, functions, and unit tests in this assignment. Arrays will be used to store the data and

1 Note that arrays, pointers, and functions are part of C as well as C++. In fact, these building blocks are less important in C++ than they were in C, because C++ provides important alternatives to these, such as vectors, references, and methods. However, we're focusing on understanding the basic building blocks from which more complex structures can be built.

pointers will be used to refer to memory that is dynamically allocated. Note that the memory must be dynamically allocated, because in many cases it is not possible to know how much data will be found in a file at the time your code is written. Therefore, it is important to know how to dynamically allocate memory.

MVP0.2 must initially allocate enough space to hold the data for 10 countries. As it reads the file, if that allocation proves to be insufficient, MVP0.2 will ask for a new allocation twice that large (using `new`), copy the old data to the new space, deallocate the old data space (using `delete`), and continue reading from the file. If that allocation also proves to be insufficient, MVP0.2 will repeat the doubling, copying, deallocating, and resuming process, continuing in this way until the entire file is read in. This process of dynamically doubling the array size as the data is read in must be modularized within a function. This function must be tested and shown to work correctly using unit tests that you must also develop and code, as well as work in the final, integrated program.

In addition to these specific requirements, you must carefully consider how to 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 will create simple design document that explains and justifies your design choices.** Please make this a PDF file and name it "**design.pdf**" in your submission.

The only libraries you may use for this assignment are `iostream`, `iomanip`, and `string` (`#include <iostream>`, `#include <iomanip>`, `#include <string>`).

Note that this lab does not require you to create any classes or to use object-oriented design at all. In fact, you don't even need to create any functions except the `main` function and the array size doubling function. Nonetheless, 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. 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 Visual Studio project **and your design document** to the appropriate dropbox in D2L by **11:00 pm CST on Thursday, 8 September 2016**.