Santa Clara University
School of Engineering

CSEN 79L: A container class for a course roster.

Assignment

Finish the implementation of the container Roster from the provided code. Note it uses C-style array to store data. Do not change that. We will practice with other representations in the future.

The provided "Person" class in `person.h` is a POD (plain old data) class that captures a person's ID, first name, and last name. The Person class is capable of handling input/out from stdin/stdout.

Also provided is a partially implemented "Roster" class in `roster.h` which is a container for Person. It has an array (initially size 30) that stores the roster for a class. You need to finish the test program in main as in the provided rostermain.cxx. A sample test data, `lab3sampleinput.txt`, is provided.

The provided Makefile compiles and builds the program. Modify it if you wish.

Specifically for this lab:

1. Implement the constructor for Roster.
2. Implement its "insert" and "erase" member functions. Must error check for array capacity.
3. Implement 3 more member functions for Roster class: `begin()`, `end()`, and `next()`
   The function begin() will return a pointer to the first record of the roster data; end() the final one, and next() the next one relative from the previous begin() call. Test these functions with a for-loop traversal like this:
   ```
   for (auto a = roster.begin(); a <= roster.end();
     a = roster.next())
     cout << a;
   ```
4. The executable "rostertest" processes from stdin line by line. It prints to stdout in plain text. Therefore, you can run your program as:
   `rosterTest < input_file > outout_file`
   a) Each input line begins with an "Command" character that is one of "A", "X", or "L".
   b) "A" will be followed by data of an ID, first, and last names. Your program should insert this data into the container.
   c) "X" is followed by an ID. Your program should remove this record, if exists, from the container.
   d) "L" has no followed. Your program is to print all members in the existing roster to stdout.
   Note the provided "main" has handled this part.
5. Submit all source files (everything not generated by the compiler or your test program).

Technical Explanations

The phrase `using csen79pods::Person`; introduce the name `Person` from the namespace `csen79pods`. This makes it possible to bring in only one name from the namespace instead of all the names.

CSEN 79L: A container class for a course roster.

Another use of the “using”, other than introducing the namespace is to create an alias for another type. In Roster “T” is an alias to “Person”. Think of this as a C++’s version of C’s “typedef.”

C++ compiler replaces the type of a variable declared “auto” with the most suitable type given the context of the code. By itself, auto is not a type.

C++ has a standard exception handling known as the “`try/catch/throw`” construct. This is facilitated by the standard library with the inclusion of `<stdexcept>`. This will be highly useful for the rest of the quarter.

Enum is nearly identical as its C counterpart. Our “Command” says the elements are to be encoded as `char` type.