CSE3150: Lab 3 – Sensors: Arrays, Pointers, Const, Casting, and Exceptions

Justin Furuness

Overview

In this assignment, you will implement a tiny **sensor logger** that stores labeled readings. You will practice:

- raw arrays (no `std::vector`)
- pointers vs. references, and const-correctness
- numeric `static_cast`
- control flow (`if/else`, `switch`, loops)
- exceptions (throw/catch) using `std::string`

Specifications

You will maintain two parallel arrays:

- `std::string labels[capacity];`
- `double values[capacity];`

and a variable `int size` tracking the number of stored readings.

Functions to Implement

1.
   ```
   void addReading(const std::string& label, double value,
       std::string labels[], double values[], int& size, int capacity);
   ```
   - Append at index `size`. If full, **throw** `std::string("Full")`.
2. `void updateValue(double* valuePtr, double newValue);`
   - Use the pointer to write into an existing array element.
3. `void printReading(const std::string& label, const double& value);`
   - Print a single reading (read-only via const refs).

4. `double average(const double values[], int size);`
5. `double minValue(const double values[], int size);`
6. `double maxValue(const double values[], int size);`
    - For each of these, **throw** `std::string("Empty")` if `size==0`.

Menu-Driven `main()`

Implement a loop with `switch` that supports:

1. **Add** reading: prompt for `label` and `value`, then call `addReading`.
2. **Update** by index: prompt for an `index`. If out of range, **throw** `std::string("Bad index")`. Otherwise obtain a pointer to that element (`&values[index]`) and call `updateValue`.
3. **Print all**: iterate and call `printReading`.
4. **Compute** aggregate:
    - Submenu: `1=avg`, `2=min`, `3=max`
    - Directly call the chosen function and print the result.
    - Also print the **rounded integer** using `static_cast<int>` on the computed result.
5. **Exit**

Error Handling Wrap menu actions with `try/catch(const std::string&)` and print the message on error.

Deliverables

- Split your code into headers, .cpp files, and your main.cpp file. Put your headers under an include directory, your implementations in a .cpp file in src, and your main.cpp file in src.
- Initialize your github repo under cse3150_week_3_lab and push the starter code
- Checkout a new branch for your work called feature_123 and push updates to there. You can checkout a new branch using git checkout -b branch_name
- When you're done, make a pull request on GitHub and merge everything to the main branch (the branch name of the main branch is irrelevant). Make sure your code compiles and passes tests.