Oluwamayokun Lawal

CSCI 2100 – Data Structures

Professor Wei Wang

Final Project

Section 1: Introduction

As the requirements asked my program should be able to Detect any erroneous data, print out the vehicles estimated speed at each second and use a hash table to facilitate the user to search for specific values. I have three files: main.cpp, bill.hpp and bill.cpp. Each serves a purpose.

1) Main.hpp:
   a) Initializes file names for our data source which are the txt files that were provided
   b) Creates an Instance Bill class
   c) Opens files and reads the data, then stores it into a vector
   d) Calls our function to output the speed, Erroneous data and carry out the search
2) Bill.hpp:
   a) Our Header file that contains a class called Bill
   b) That class, Bill, contains all our functions with their appropriate parameters
3) Bill.cpp:
   a) Gives purpose to the functions we initialized in our header file
   b) Contains algorithms for each function.
   c) The findErroneousData function should fish out potentially wrong data using threshold comparison
   d) the printSpeed function loops through the vector and prints out the speed at each index
   e) the searchSpeed function should look for whether a specific speed exists in either file, if it does it prints out, we do this by using s hash map.

Our whole code is expected to work because every role is clearly assigned to a function, and each process in each algorithm in the function is well thought out to have accuracy, efficiency and adaptability.

Section 2: Design Details

The Bill Class Implementation:

- Constructor Bill () prints “Let Us Begin” signaling the start of the code
- Destructor ~Bill () outputs “The end”
- This is primarily for user interaction, and it made it easier to debug

Error Detection (findErrenousData):

- It takes in the following parameters as inputs: A vector of speed data and sensor Name (*const vector<double>& data, const string& sensorName) const* {

- Initializes a valid vector to trach whether each data point is valid, and uses the first speed value as the baseline
- Loops through the data to calculate the absolute difference between values (*double change = abs(data[i] - lastValidValue*)
- If the change exceeds 10, it fishes out that datapoint as erroneous, it then prints out that data along with its corresponding time.

Speed Data Printing (printSpeed):

- It takes in the following parameters as inputs: A vector of speed data (lines) and file name *(const vector<double>& lines, const string& filename*)
- Loops through the data vector and prints each speed value along with its time stamp

Index Creation (createSpeedIndex)

- It takes in the following parameters as inputs: A vector of speed data *(const vector<double>& data)*
- Makes an unordered map where the {key} is the speed value, and the {value} is its index in the vector.
- It will return that unordered map that will later be used.

Speed Search (searchSpeed)

- It takes in the following parameters as inputs: An unordered map, the target speed and sensor name *(const unordered_map<double, size_t>& indexMap, double speed, const string& sensorName)*
- Uses ".find" to see if the speed exists in the map, if its found it prints the index it was found out , if not it prints a message saying it couldn't find the data.
- This cite was helpful in teaching me about the unordered map container - https://www.geeksforgeeks.org/unordered_map-find-in-c-stl/

Main Program (main.cpp)

The main program orchestrates the workflow by utilizing the Bill class's functionalities.

- File Reading
  - Logic:
    - Reads speed data from two files (SmartWatch.txt and RTK_GPS.txt). (learned how to read text files by watching this YouTube video : https://www.youtube.com/watch?v=Cz4fl-TUjVk)
    -
    - Stores the data in a vector of pairs, where each pair contains a filename and its corresponding data vector.
  - Error Handling:
    - If a file cannot be opened, the program prints an error message and exits.
  - Purpose:

        - Ensures compatibility with different data sources and scalability for additional datasets.
- Processing Data
    - Logic:
        - For each dataset:
            - Calls findErroneousData to detect outliers.
            - Calls printSpeed to display the data.
    - Purpose:
        - Validates and presents the data from each source.
- User Interaction: Searching for Speeds
    - Logic:
        - Prompts the user to input a speed value to search for.
        - For each dataset, calls createSpeedIndex to build the index map.
        - Calls searchSpeed to locate the user-specified value.
    - Purpose:
        - Provides interactive analysis capabilities.

Section 3: Results

```
Detecting erroneous data in SmartWatch.txt
The Data 0.13977 at time 34 is wrong
The Data 79.1124 at time 54 is wrong
The Data 104.894 at time 75 is wrong
```

```
Detecting erroneous data in RTK_GPS.txt
The Data 120.244 at time 80 is wrong
```

We got these results as a result of our function that take out erroneous data, in the file these are the datapoints where the change exceeds 10 so it prints it out saying “The data” << data[i] << “at time” << i << “is wrong” <, endl;

```
Contents of the SmartWatch.txt file:
Time: 1s, Speed: 0.960361 m/s
Time: 2s, Speed: 2.06409 m/s
Time: 3s, Speed: 3.02897 m/s
Time: 4s, Speed: 4.00425 m/s
Time: 5s, Speed: 4.84046 m/s
Time: 6s, Speed: 6.03596 m/s
Time: 7s, Speed: 6.8914 m/s
Time: 8s, Speed: 7.98731 m/s
Time: 9s, Speed: 9.03048 m/s
Time: 10s, Speed: 9.91693 m/s
Time: 11s, Speed: 10.8489 m/s
Time: 12s, Speed: 12.0055 m/s
Time: 13s, Speed: 13.1012 m/s
Time: 14s, Speed: 14.033 m/s
Time: 15s, Speed: 14.9018 m/s
Time: 16s, Speed: 15.9974 m/s
Time: 17s, Speed: 16.8358 m/s
Time: 18s, Speed: 17.9927 m/s
Time: 19s, Speed: 19.023 m/s
Time: 20s, Speed: 19.8028 m/s
Time: 21s, Speed: 20.9657 m/s
Time: 22s, Speed: 22.1684 m/s
Time: 23s, Speed: 22.8502 m/s
Time: 24s, Speed: 24.0178 m/s
Time: 25s, Speed: 25.0141 m/s
Time: 26s, Speed: 25.9664 m/s
Time: 27s, Speed: 26.9012 m/s
Time: 28s, Speed: 28.099 m/s
Time: 29s, Speed: 29.0327 m/s
Time: 30s, Speed: 30.0379 m/s
Time: 31s, Speed: 31.0394 m/s
Time: 32s, Speed: 31.8765 m/s
```

```
Contents of the RTK_GPS.txt file:
Time: 1s, Speed: 0.88755 m/s
Time: 2s, Speed: 1.8914 m/s
Time: 3s, Speed: 2.87676 m/s
Time: 4s, Speed: 4.01674 m/s
Time: 5s, Speed: 4.94963 m/s
Time: 6s, Speed: 5.84687 m/s
Time: 7s, Speed: 7.09356 m/s
Time: 8s, Speed: 7.95922 m/s
Time: 9s, Speed: 8.96045 m/s
Time: 10s, Speed: 9.81924 m/s
Time: 11s, Speed: 10.8442 m/s
Time: 12s, Speed: 12.0964 m/s
Time: 13s, Speed: 12.9733 m/s
Time: 14s, Speed: 13.9925 m/s
Time: 15s, Speed: 14.9069 m/s
Time: 16s, Speed: 16.091 m/s
Time: 17s, Speed: 16.9065 m/s
Time: 18s, Speed: 17.9069 m/s
Time: 19s, Speed: 18.9967 m/s
Time: 20s, Speed: 19.9653 m/s
Time: 21s, Speed: 21.0314 m/s
Time: 22s, Speed: 22.0284 m/s
Time: 23s, Speed: 29.9029 m/s
Time: 24s, Speed: 24.2125 m/s
Time: 25s, Speed: 25.1577 m/s
Time: 26s, Speed: 26.0473 m/s
Time: 27s, Speed: 26.8526 m/s
Time: 28s, Speed: 28.045 m/s
Time: 29s, Speed: 29.0076 m/s
Time: 30s, Speed: 29.9908 m/s
Time: 31s, Speed: 30.8868 m/s
Time: 32s, Speed: 32.0544 m/s
```

For each file the program prints the speed data along with the timestamp.

```
Enter a value to search for: 25.966439231248803
Found it! The corresponding index in SmartWatch.txt is 26
Cannot find the data in RTK_GPS.txt
The end.
```

Prompts the user to put in a number, since that number we put in is in the SmartWatch.txt file it says it found it and also gives us the corresponding index, and since it wasn't in the RTK_GPS.txt file then it said it cant find the data