Student: Adam Behun

Course: Data Structures

Instructor: Dr. Wei Wang

Final Project

Introduction

The goal of my program is to calculate the accurate speed of a vehicle constantly accelerating on a straight road having data from *SmartWatch.txt* and *RTK_GPS.txt*. One of the challenges of this program was to read the data from the two files and then distinguish between accurate and inaccurate data. Inaccuracy in data exists due to interference with the environment and inaccuracy of the actual sensors. My program is firstly able to decide which data points are likely not representing the accurate reality of a vehicle accelerating at a constant pace; secondly, it is able to compute the correct speed of the vehicle at any given time using the average of the two sensors, and thirdly, it allows the user to search for data within the three datasets, e.g. *SmartWatch.txt*, *RTK_GPS.txt, and final_speed_data* (created as an average of *Smartwatch and RTK_GPS datasets*).

Design Details

I have started my design by opening and reading, line by line, the two different datasets and saving them into vectors of doubles data type. Once the data from these two datasets is read and saved into appropriate containers (C++ vectors) I close the files as they are no longer needed for my implementation. My next step was to create a third container (C++ vector) which is going to save the results of my compute speed method. This method computes the average of the vectors at all given times, and I consider this to be the accurate representation of the vehicle's actual speed. At this point in my program, we have 3 vectors which all store 99 double data type values. The first one is called smartwatch_data (data read from the SmartWatch.txt), the second one is called rtk_gps_data (data read from the RTK_GPS.txt), and the third one called final_speed _data (computed average of the values at every index (0 – 99) in the smartwatch_data and the rtk_gps_data) representing the most accurate approximation of the vehicle's actual speed. Once these datasets are ready to work with, I have implemented an algorithm which finds the data points in each dataset that do not represent the reality accurately. I

have decided to point out the possible mistakes made by the errors due the interference from the environment or the inaccuracy of the sensors themselves by printing out all values that do not follow the pattern ($n < n + 1$) or if any data point is much bigger than the previous one. After I had computed the most accurate approximation of the vehicle's speed, pointed out the data that most likely do not represent the reality, I have implemented a binary search algorithm from the C++ algorithm library. In my binary search implementation, the program first asks the user to choose which dataset they want to search from (smartwatch_data, rtk_gps_data, final_speed_data) and after that they can choose the specific integer they would like to search for. This method utilizing binary search algorithm returns the speed and its corresponding index in the corresponding dataset if it found the target speed searched or returns "Speed not found within this dataset" message.

Libraries used

For my design I have used some of the available C++ libraries and here I will explain what functionality they offered to my code. I have decided to use the *fstream* library to open the external files with data, allowing my code to read the data points into memory and work with it in the form of vectors. Another library I used is *string* which allowed me to convert the previously read line of data from string datatype into a double datatype. Then, I used the *cmath* library to round to computed speed of a vehicle to 4 decimal places utilizing the *round()* function. Lastly, I took advantage of the *algorithm* library which contains a binary search algorithm which I used in my implementation of a search program.

Functions designs

read_data function enables to read data from .txt files and utilizes the *fstream* library. This function takes in 2 vectors as parameters and checks whether they were properly open (prints out an error if not). Then, iterates over the file and converts, line-by-line, each read string into a double value and pushes it into the appropriate vector. Then, it closes the files as they are no longer needed.

compute_speed function enables me to create a third vector with double data type. This function iterates over the 2 created vectors and computes a mean with each iteration, then it stores this

mean as a data point in the final_speed_data vector. After this function is completed, the program stores the vector in memory as my closest approximation to the vehicle's actual speed.

find_wrong_smartwatch_data function checks the smartwatch vector for data that likely does not follow normal distribution and so is likely not an accurate representation of the vehicle which is accelerating at a constant speed. This is achieved by comparing whether the current element is smaller with the previous one and if yes, then printing out the message that this point likely is an error.

find_wrong_rtkgps_data function works in a very similar way to the abovementioned function find_wrong_smartwatch_data.

erase_wrong function takes advantage of some vector functionality and enables me to delete (erase) the data points that do not accurately display the reality of the vehicle. I do this before I enable search on the datasets as I believe that users should not be able to search datasets with incorrect data points in them. This method first deletes data that is less than 1 or more than 99 as per requirement #1 of my project and then deletes the outliers that are there due to interference and other issues. Then, it simply prints out a message mentioning the deleted data point.

my_search is the last function implemented and used within my program. It enables the user to search each one of the datasets within this program using binary search from the algorithm library. It first converts the passed-in double vector into an integer vector for ease of use, then calculates the correct index using *distance, begin, and lower bound* and lastly prints out a message with the either found or not found target speed.

Limitations

The program could be more accurate in computing the final speed as well as in finding data that is likely inaccurate. Moreover, the my_search function is not designed in the best way as it needs to convert the whole vector into a different data type before searching it; this would be a problem with larger dataset.

Results

```
Computing final speeds at all times...

Looking for outliers in Smartwatch dataset:
The data 0.13977 at time 33 may be wrong.
The data 30.9663 at time 35 may be wrong.
The data 38.0201 at time 37 may be wrong.
The data 54.9687 at time 54 may be wrong.
The data 61.033 at time 65 may be wrong.
The data 68.2057 at time 67 may be wrong.
The data 75.9473 at time 75 may be wrong.

Looking for outliers in RTK_GPS dataset:
The data 29.9029 at time 22 may be wrong.
The data 120.244 at time 79 may be wrong.


This is the speed at each time:
-----
Speed: 0.924 m/s at time: 0
Speed: 1.9777 m/s at time: 1
Speed: 2.9529 m/s at time: 2
Speed: 4.0105 m/s at time: 3
Speed: 4.895 m/s at time: 4
Speed: 5.9414 m/s at time: 5
Speed: 6.9925 m/s at time: 6
Speed: 7.9733 m/s at time: 7
Speed: 8.9955 m/s at time: 8
Speed: 9.8681 m/s at time: 9
Speed: 10.8466 m/s at time: 10
Speed: 12.0509 m/s at time: 11
Speed: 13.0373 m/s at time: 12
Speed: 14.0128 m/s at time: 13
Speed: 14.9044 m/s at time: 14
Speed: 16.0442 m/s at time: 15
Speed: 16.8712 m/s at time: 16
Speed: 17.9498 m/s at time: 17
Speed: 19.0098 m/s at time: 18
Speed: 19.8841 m/s at time: 19
Speed: 20.9985 m/s at time: 20
Speed: 22.0984 m/s at time: 21
Speed: 26.3766 m/s at time: 22
Speed: 24.1152 m/s at time: 23
Speed: 25.0859 m/s at time: 24
Speed: 26.0069 m/s at time: 25
Speed: 26.8769 m/s at time: 26
```

```
Speed: 75.9011 m/s at time: 75
Speed: 77.0148 m/s at time: 76
Speed: 78.0619 m/s at time: 77
Speed: 79.0151 m/s at time: 78
Speed: 100.121 m/s at time: 79
Speed: 80.9623 m/s at time: 80
Speed: 82.04 m/s at time: 81
Speed: 82.9643 m/s at time: 82
Speed: 84.0379 m/s at time: 83
Speed: 85.003 m/s at time: 84
Speed: 86.0573 m/s at time: 85
Speed: 87.116 m/s at time: 86
Speed: 87.9681 m/s at time: 87
Speed: 89.0613 m/s at time: 88
Speed: 90.0316 m/s at time: 89
Speed: 90.9591 m/s at time: 90
Speed: 91.9645 m/s at time: 91
Speed: 92.9195 m/s at time: 92
Speed: 94.0419 m/s at time: 93
Speed: 95.0327 m/s at time: 94
Speed: 96.0339 m/s at time: 95
Speed: 97.0273 m/s at time: 96
Speed: 97.9581 m/s at time: 97
Speed: 98.9131 m/s at time: 98

Erasing wrong data from all 3 datasets before enabling search on them...
Deleted 0.960361
Deleted 99.0341
Deleted 0.13977
Deleted 30.9663
Deleted 79.1124
Deleted 61.033
Deleted 104.894
Deleted 0.88755
Deleted 29.9029
Deleted 120.244
Deleted 0.923955
Deleted 24.1152
Deleted 17.0103
Deleted 33.5305
Deleted 66.6146
Deleted 63.559
Deleted 89.9422
Deleted 100.121

Search for speeds from any of the following datasets:
Input 1 for the SmartWatch dataset
Input 2 for the RTK_GPS dataset
Input 3 for the Final Speed dataset
Input 0 to exit
```

```
Search for speeds from any of the following datasets:
Input 1 for the SmartWatch dataset
Input 2 for the RTK_GPS dataset
Input 3 for the Final Speed dataset
Input 0 to exit
Input: 2
Search in integers
Input: 34
Found it! 34 The corresponding index is 31

Search for speeds from any of the following datasets:
Input 1 for the SmartWatch dataset
Input 2 for the RTK_GPS dataset
Input 3 for the Final Speed dataset
Input 0 to exit
Input: 1
Search in integers
Input: 34
Could not find 34 in the data.

Search for speeds from any of the following datasets:
Input 1 for the SmartWatch dataset
Input 2 for the RTK_GPS dataset
Input 3 for the Final Speed dataset
Input 0 to exit
Input: 1
Search in integers
Input: 23
Found it! 23 The corresponding index is 21

Search for speeds from any of the following datasets:
Input 1 for the SmartWatch dataset
Input 2 for the RTK_GPS dataset
Input 3 for the Final Speed dataset
Input 0 to exit
Input:
```