

## Array of Structs, Sorting, Code Instrumentation

For this project you will implement a relatively simple database application to manage information about trips. The application will initialize via a simple input file, and then respond to a variety of queries supplied via a second input file. The application will log all query processing to a file. Finally, the application will write a report file that lists the trips sorted in order of origin and summarizing the supplied trip data.

The trip data must be organized in a single array of `struct` variables, using a `struct` type with appropriate data members. The list must be sorted using the bubblesort algorithm.

In addition, you must instrument your code to aid debugging. The instrumentation should write to standard output, documenting the execution of certain sections of your command processing. In particular, you must instrument the array initialization, data acquisition, and all array searches. When instrumenting array traversals, indicate what operation is being carried out (e.g., a mileage search), and which array cells are rejected and accepted during the traversal.

The instrumentation should be switchable via a global Boolean setting.

### The trip data file:

The trip data file will be supplied in a file named "TripData.txt". A sample file is given below. The input file begins with two header lines that are meaningful to a human reader but must be ignored by the program. Lines of data, each pertaining to a particular trip that has been taken, follow the header section. Each of these lines specifies the name of the place where the trip began, the name of the destination where the trip ended, the number of miles traveled, and the time required for the trip.

Origin	Destination	Miles	Time
Blacksburg, VA	Knoxville, TN	244	3:25
Knoxville, TN	Nashville, TN	178	2:35
Nashville, TN	Memphis, TN	210	3:17
Memphis, TN	Little Rock, AR	137	2:05
Little Rock, AR	Texarkana, TX	141	2:10
Texarkana, TX	Dallas, TX	180	2:50
Dallas, TX	Ft Worth, TX	30	0:35
Ft Worth, TX	Abilene, TX	155	2:25
Abilene, TX	Pecos, TX	242	3:35
Pecos, TX	Carslbad, NM	75	1:07

The names are strings of characters, which may contain spaces; in order to make the data easier to read in, a single tab character follows each name. A single tab character also separates the mile and time values, and there is a newline character immediately after the time. Note that the appearance of tabs is somewhat unreliable. In the sample above the tabs are displayed according to the settings in Microsoft Word and everything is neatly aligned. Viewed in a text editor, such as the Visual C++ IDE, a tab is usually interpreted as having a fixed width (such as 4 spaces) and the alignment will not appear to be nearly as nice. The visual appearance of the data doesn't matter to the program that will read it, but the presence of tabs will make it easier to write code to read that data.

There will be data for an unknown number of trips, but there will never be more than 100. You also cannot assume that the given data will always be logically valid. The input file will always conform to the format above. However, it is possible that a given distance will be negative, or that a given time value will be zero or negative. If that happens, your program should ignore the data for that trip.

## What must be calculated?

For each trip, the program must calculate the average speed in miles per hour, MPH, and the time required for the trip in minutes. Also, the program must calculate summary data for the complete set of trips. Specifically, the program must calculate the number of trips reported, the total miles traveled, the total time required for all the trips, and the average speed in MPH for all the trips.

Note: the average MPH over all the trips is NOT just the sum of the individual MPHs divided by the number of trips. Think about it...

Warning: the necessary calculations are relatively simple, but they may require some thought. It would be a violation of the Honor Code to explain to another student how to do the calculations or to post that information to a course listserv, discussion board, or newsgroup.

In order to produce the most accurate results possible in C++, all the mileage and time values should be stored as integers, not as decimal numbers, and the average speed should be stored using variables of type `double`, not type `float`.

## The query file and log file:

The database queries are supplied in a file named "Queries.txt". A sample file is given below.

```
mileage Blacksburg, VA Knoxville, TN
mileage Blacksburg, VA Memphis, TN
time Blacksburg, VA Knoxville, TN
neighbors Memphis, TN
exit
```

The query contains lines corresponding to the following queries:

mileage <location> <location>

Report the distance for a trip between the given locations, in either direction. If the database contains more than one such trip, log the first one found. If the database does not contain such a trip, log an error message.

time <location> <location>

Report the time for a trip between the given locations, in either direction. If the database contains more than one such trip, log the first one found. If the database does not contain such a trip, log an error message.

neighbors <location>

Report all locations for which the database contains a trip to or from the given location. If the database does not contain such a trip, log an error message.

exit

Stop processing queries and write the report file, as specified below.

For each query line, the tokens will be separated by tabs, and the final token will be followed immediately by a newline character.

The application will write a log file, "Log.txt", specifying the results for each query. The log file will begin with the usual identification information. Each query must be echoed to the log file, along with a sequence number, starting at 0. The results for each query must be well formatted and clearly distinguished from other query results. Here is a sample log file:

```

Programmer: Bill McQuain
CS 1044 Notes Example

001-----
Command: mileage Blacksburg, VA      Knoxville, TN

Mileage between Blacksburg, VA and Knoxville, TN is 244.
002-----
Command: mileage Blacksburg, VA      Memphis, TN

No trip between Blacksburg, VA and Memphis, TN was found.
003-----
Command: time      Blacksburg, VA      Knoxville, TN

Time from Blacksburg, VA and Knoxville, TN is 3:25.
004-----
Command: neighbors Memphis, TN

        Little Rock, AR
        Nashville, TN
005-----
Exiting...

```

### The report file:

The report file is named "Summary.txt". A report file, which corresponds to the given input file, is shown below.

```

Programmer: Bill McQuain
CS 1044 Notes Example

Origin          Destination          Mileage  Minutes    MPH
-----
Abilene, TX     Pecos, TX           242      215       67.5
Blacksburg, VA  Knoxville, TN       244      205       71.4
Dallas, TX      Ft Worth, TX        30        35       51.4
Ft Worth, TX    Abilene, TX         155      145       64.1
Knoxville, TN   Nashville, TN       178      155       68.9
Little Rock, AR Texarkana, TX       141      130       65.1
Memphis, TN     Little Rock, AR     137      125       65.8
Nashville, TN   Memphis, TN        210      197       64.0
Pecos, TX       Carlsbad, NM        75        67       67.2
Texarkana, TX   Dallas, TX         180      170       63.5
-----

Number of trips:      10
Total miles:          1592
Total time:           24:04
Average MPH:          66.15

```

The first two lines specify the programmer and the assignment, and the third is blank. The fourth line specifies column headers for the table the program will write, and the fifth line separates the table data from the labels.

The table body contains one line of output for each given trip, displaying the trip origin and destination, the trip mileage, the length of the trip in minutes, and average speed for that trip. The trips are listed in ascending order according to their origins.

When a decimal number is written, the number of digits displayed after the decimal point is called the precision of the displayed value. The average speed for each must be written with precision 1, but the average speed over all the trips must

be written with precision 2 (as shown in the output sample). The total time for all the trips must be printed in HH:MM format (as shown in the input and output samples).