Initial Input

The global input data such as number of vehicle types, speed limit, number of parking spaces etc. are stored in global variables accessible throughout the main source code. Vehicle details that include vehicle type, license plate format, speed and volume weights are going to be stored in an array of structs called Vehicles. Similarly, the statistics associated with each type of vehicle is stored in an array of structs called Stats. The number of vehicle with indicate the index for using the arrays later on.

Inconsistencies between Stats.txt and Vehicles.txt are possible. Vehicle types need to be present in both of the files and the both files should have the same number of vehicle types. Also, the order of the vehicle types in the files needs to be the same. The program does not check for any inconsistencies.

Activity Engine

The Activity Engine class uses the stats to generate counts for each vehicle type in a day (using normal_distribution).

For each of those vehicles, license plates, random arrival times are generated (using uniform_int_distribution) and stored in an array.

The simulation begins at 00:00. At each minute, it is analysed if vehicles that have arrived have finished travelling the length of the road. If yes, the event is logged. Similarly, the program checks if any vehicle is arriving at that particular time. If there is, the vehicle is added to the simulation. Cases for each event with different probabilities are formulated. Using discrete_distribution, a case for each vehicle is generated, according to which the vehicle either parks, stops parking, leaves via side road, changes speed or stays constant.

The probabilities for a vehicle to perform the events at any given time are:

Parks- 10%
Stops parking- 5%
Leaves via side road- 6%
Changes speed- 20%
Stays constant- 64%

For a parking event, it is checked if the vehicle has a parking flag and if there are parking spaces available. A vehicle is allowed to park only once.

For leaving via a side road, vehicles must have travelled at least 0.5km.

For changing speed, a new speed is generated using the speed mean and standard deviation of the vehicle type given.

The average speed is calculated using the arrival time of the vehicle and the time recorded at its departure. Any parking vehicles do during the simulation Is recorded and considered in the calculation of the average speed.

The format of the logs is:

Time of event Vehicle Type Event name Vehicle license Arrival Speed

The vehicle type and vehicle license identifies every unique vehicle. Arrival speeds of each vehicle are stored in the logs as it would be required to calculate the average speed in the later stages. The time of event is listed at the front for ease of reading. Parking/unparking events are recorded with a -1 index.

Analysis Engine

The analysis engine goes through the log files that have been created in the activity engine and records the count and speeds of each vehicle types. It stores the data read in from the logs into two different files; BaselineData.txt and SimData.txt.

The counts of each vehicle type and the number of speed breaches are recorded in the SimData.txt file. After these values have been calculated, the engine calculates the means and SDs for each vehicle type and stores the values into BaselineData.txt.

In case the analysis engine runs for the live data that is to be used for the alert engine, the calculated data are written in a different file so that the baseline data do not get overwritten.

Alert Engine

The alert engine processes simulation data obtained from the analysis engine, then compares them against previously calculated baseline means and standard deviations for both volume and speed to check for anomalies in the simulation.
On the first run through the engine, the engine obtains all the information it requires to perform data comparisons to check for anomalies, namely the information obtained from the baseline simulation and weights for each vehicle type. The data is read from a text file formatted as follows:

> Vehicle Type, Volume Mean, Volume Standard Deviation, Speed Mean, Speed Standard Deviation, Volume Weight, Speed Weight

The above information is stored into an array of structs, with array entries each representing a unique vehicle type. Volume and speed anomaly thresholds are also calculated based on the weights provided. All subsequent simulation data is compared against this information.

On subsequent runs through the engine, the engine reads simulation data from another text file which contains, for each day:

> The vehicle types, the number of entries for that vehicle type, and the individual speed for each entry for vehicle type.

For each day, a list of speed limit breaches is also provided, but is not relevant to the process of anomaly checking to be conducted by the alert engine. Anomaly checking is done as follows:

1. For each volume/speed entry, its difference from the volume/speed mean is calculated in terms of volume/speed standard deviations.
2. For each volume/speed difference, the volume/speed anomaly counter increases by the product of the volume/speed difference and its corresponding weightage.
3. For each day, the final volume/speed anomaly counter is compared against the volume/speed anomaly thresholds. An anomaly counter exceeding its corresponding threshold means that there is anomaly of that type for the given day.

The volume and speed thresholds and the daily results for anomaly checks are both outputted to the terminal and written into a text file for storage.