Task

Implement a traffic simulator in the programming language C++. The program must be able to be compiled, executed and tested. Perform modularization correctly. Only standard libraries are allowed. Adhere to OOP principles, SOLID principles and conventions for the C++ programming language. Provide examples for testing implemented functionalities.

(12) Define a class for a directed graph such as a traffic network. Types of nodes are: 1) location node, e.g. city, parking etc. - everything that can serve as a starting point or destination when driving, 2) a road node, which represents a road or road segment, in one direction, between exactly two nodes in the graph, as well as 3) an intersection node, which can connect more roads. The following data are stored for each road node: 1) road length in meters, which determines the weight between the nodes connected by that road node, 2) maximum permitted speed of movement on the road and 3) maximum number of vehicles on the road. Road nodes are connected to the intersection node, so road nodes that have a directed connection to the intersection node are considered input nodes, while road nodes to which the intersection node has directed connections are considered output nodes. In this context, for each pair of input and output nodes, it is defined whether it is possible to go from that input node to that output node, the length in meters that needs to be covered, as well as the average speed of the vehicle, which depends on the current number of vehicles on the intersections. Also, it should be possible to specify the maximum allowed number of vehicles at the intersection. Enable serialization and deserialization of the traffic network graph.

(5) Define the class used to represent vehicles. The vehicle moves from one location to another. Each vehicle has a defined average speed at which it tends to move on the road, with the fact that the actual speed of the vehicle is, at some point, limited by the maximum speed of the road segment. Enable that the best path through the traffic network can be calculated for the vehicle, so that either the distance traveled or the time spent can be minimized, taking into account the lengths of the roads and speed limits on the road.

(8) Define a class that represents traffic simulation. The traffic simulation takes place in exactly one traffic network and a specified number of vehicles participate in it, so that each vehicle has its own starting point and destination (for a large number of vehicles, these parameters can be randomly selected at the beginning of the simulation). Each simulation has a defined duration of one simulation step. In each step, it is necessary to calculate and move each vehicle, taking into account its speed, the remaining length of the road, as well as restrictions on the number of vehicles on road sections. If the driver needs to move to a node that is already filled, the driver stays at the current node in the given step. The simulation class should have a public interface, which allows the user to perform a single simulation step, to check if the simulation has completed, to see the current state of individual vehicles, and to see the average travel time between two specified locations.

(5) Make it possible to specify the method of calculating the deceleration factor depending on the number of vehicles on the road for each road node, with the fact that this data should also be serializable and deserialized. The speed of the vehicle on the road should be scaled appropriately with the calculated deceleration factor. Implement a dynamic trajectory vehicle type. For such vehicles, the path is recalculated in each iteration of the simulation, taking into account other vehicles in traffic, road decelerations and restrictions on the maximum number of vehicles, so that the travel time is minimized. Illustrate with an example the situation in which the trajectory of a vehicle changes under these conditions.

Below is an illustration of the traffic graph. Nodes A, B, C and D represent locations. Empty nodes represent path nodes. Nodes marked with + represent intersections. Dashed lines represent possible turns at intersections. E.g. if the left intersection is approached from junction A, it is not possible to make a U-turn.