

Specification

Classes

Freeway

Method	Description
<code>Freeway(capacities, ready_percent, pass_limit)</code>	Prints Αυτοκινητόδρομος σε λειτουργία and constructs its data members
<code>Operate()</code>	void Calls the <code>Operate()</code> method of each <code>Segment</code> from end to start and prints the number of <code>Car</code> on the <code>Freeway</code>
<code>num_cars()</code>	size_t
<code>segments()</code>	vector<Segment*>

Segment

Variable Description

`kMaxCars` size_t Max number of `Cars` generated in each `Segment` ≥ 1

Method	Description
<code>Segment(capacity, prev, ready_percent, num_junctions, pass_limit)</code>	Creates a random number of <code>Cars</code>
<code>Enter()</code>	void Max possible <code>Cars</code> enter from <code>Tolls</code> and previous <code>Segment</code> . Required messages are printed
<code>Exit()</code>	void <code>Cars</code> whose destination is the next junction exit the <code>Freeway</code>
<code>Operate()</code>	void Calls <code>Exit()</code> , <code>Enter()</code> and randomly sets <code>ready_percent% Cars</code> as ready
<code>Pass(size_t)</code>	void Max possible <code>Cars</code> exit the <code>Segment</code> and enter the next one
<code>cars()</code>	vector<Car*>
<code>num_cars()</code>	size_t
<code>ready_cars()</code>	vector<Car*>
<code>capacity()</code>	size_t
<code>entrance()</code>	size_t Returns the id
<code>set_exit(exit)</code>	void
<code>exit()</code>	Junction*
<code>set_next(next)</code>	void Also sets exit

Junction

Variable Description

`kMaxTollsPerType` size_t Max number of tolls generated in each `Junction` ≥ 1

`kMaxCarsPerToll` size_t Max number of cars generated in each `Toll` ≥ 1

Method	Description
<code>Junction()</code>	Creates empty junction with correct id
<code>Junction(num_junctions, pass_limit)</code>	Creates a random number of <code>Tolls</code>
<code>Cars()</code>	vector<Car*>

Method	Description
NumCars()	size_t
Operate(max_allowed_cars)	vector<Car*> Returns max Cars respecting the Segment.capacity() and the pass_limit. If less than 3 * pass_limit Cars are allowed to enter, the pass_limit is decreased. If 3 * pass_limit Cars enter, then the pass_limit is increased. Finally, new Cars are added in each Toll
current_id()	static size_t Total number of Junctions initialized at 0
num_electronic()	size_t Returns electronic tolls number
id()	size_t
num_manned()	size_t Returns manned tolls number
pass_limit()	size_t

Toll

Variable Description

kMaxCars size_t Max number of cars generated in each Toll >= 1

Method	Description
Toll(current_junction, num_junctions)	Creates a random number of Cars
Add(car)	void
Remove()	vector<Car*> Removes all Cars
Remove(num_cars)	vector<Car*> Removes at most num_cars Cars
cars()	vector<Car*>
num_cars()	size_t

Car

Method	Description
Car(exit_junction, segment)	ready becomes false
exit()	size_t Returns the id
set_ready(ready)	void
ready()	bool
set_segment(segment)	void
segment()	Segment*

Usage

The executable file, e.g. build/project.out, receives from the command-line the following case-insensitive arguments with single, double, or no - prefix:

Argument Description

seed	uint Randomness seed
N	int Simulation steps number
NSegs	size_t Freeway segments number
K	size_t Initial max car number that can pass a manned toll station

Argument Description

Percent int Car percent on a segment that becomes `ready` in the next step

If any of these arguments is not provided, a default value **must** be used.

During the execution, `NSegs` numbers (`size_t`) are read from the standard input corresponding to the `capacity` of each `Segment`.

E.g:

```
oop-project git:master □ ./build/project.out -n 10 -nsegs 5 -k 10 -percent 30
Seed: 1454857303
N: 10
NSegs: 5
K: 10
Percent: 30
Enter the capacities: 10 15 5 12 17
```

An instance of `Freeway` should be constructed given the above data and then the `Operate()` method should be called `N` times.