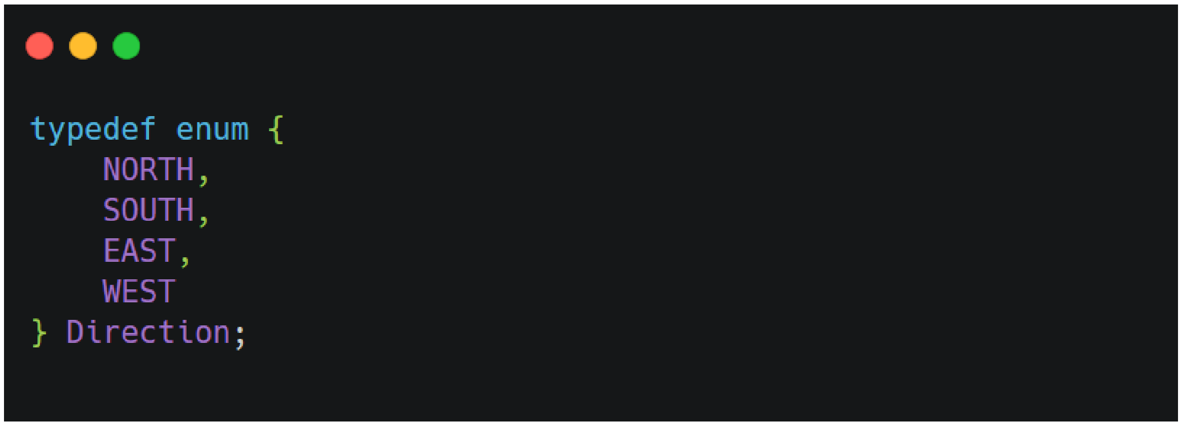
Directions

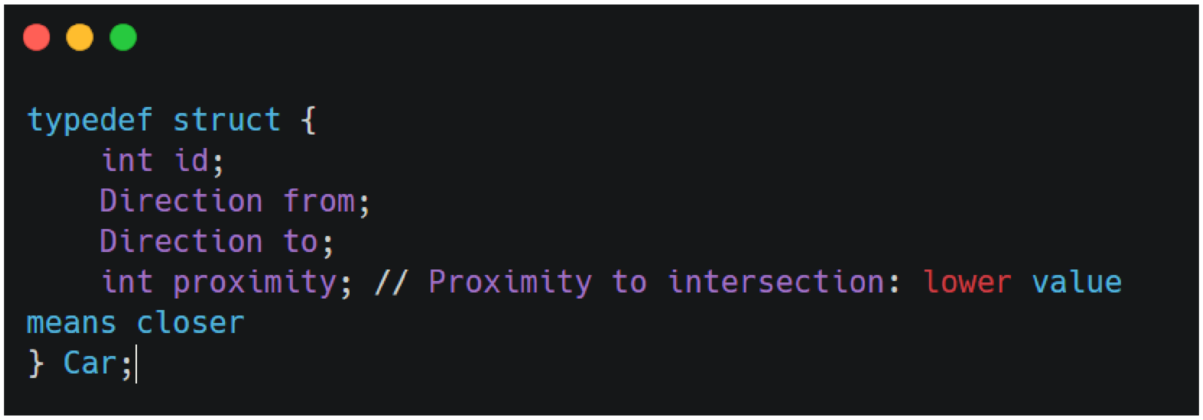
Define the possible directions for the cars.



```c
typedef enum {
    NORTH,
    SOUTH,
    EAST,
    WEST
} Direction;
```

Car Structure

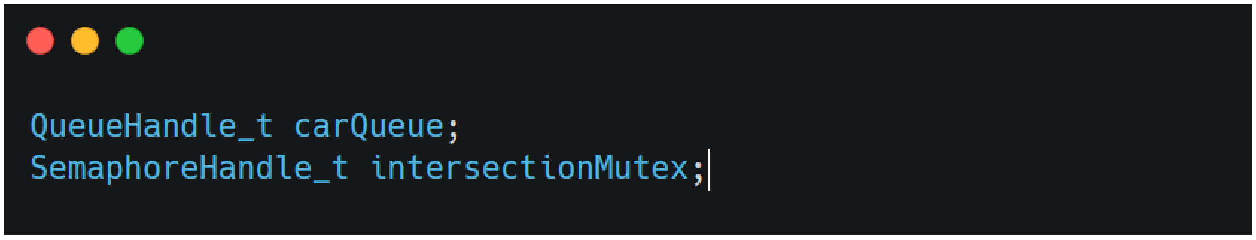
Structure to represent a car with ID, starting direction, destination direction, and proximity.



```c
typedef struct {
    int id;
    Direction from;
    Direction to;
    int proximity; // Proximity to intersection: lower value
means closer
} Car;
```

Queue and Mutex

Global variables for the car queue and the intersection mutex.

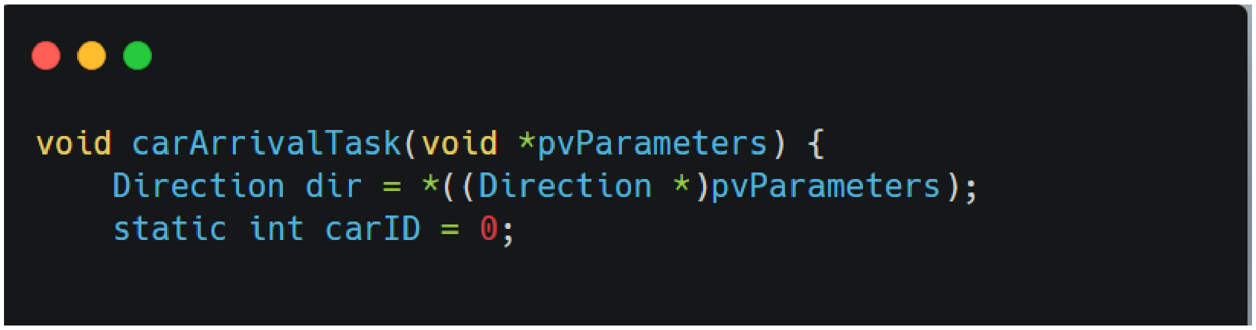


```c
QueueHandle_t carQueue;
SemaphoreHandle_t intersectionMutex;
```

Task Functions

Car Arrival Task

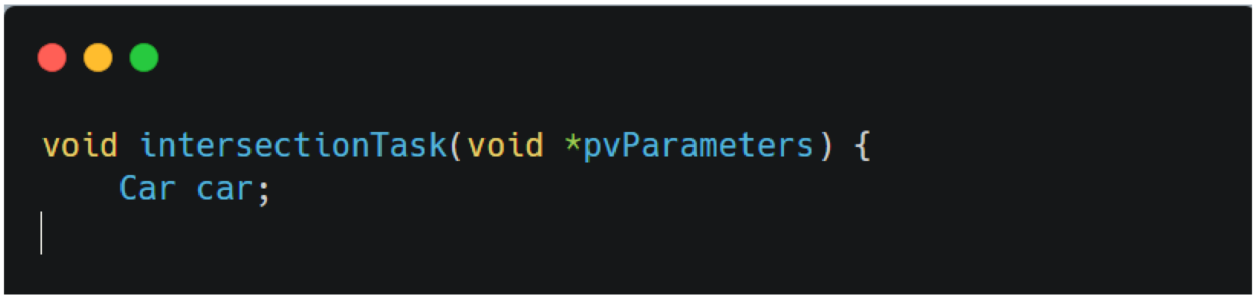
Simulates cars arriving from a specific direction.



```c
void carArrivalTask(void *pvParameters) {
    Direction dir = *((Direction *)pvParameters);
    static int carID = 0;
```

Intersection Task

Manages the intersection, allowing the car closest to pass through.



```c
void intersectionTask(void *pvParameters) {
    Car car;
```

Results

```
Car from WEST with ID 4 wants to go to NORTH with proximity 3
Car from EAST with ID 5 wants to go to WEST with proximity 8
Car with ID 2 has passed through the intersection.
Car from SOUTH with ID 6 wants to go to NORTH with proximity 3
Priority given to car from WEST with ID 4 to go to NORTH due to higher proximity.
Car from NORTH with ID 7 wants to go to SOUTH with proximity 10
Car from WEST with ID 4 is passing through to NORTH.
Car from SOUTH with ID 8 wants to go to NORTH with proximity 4
Car from NORTH with ID 9 wants to go to EAST with proximity 7
Car from WEST with ID 10 wants to go to SOUTH with proximity 7
Car from EAST with ID 11 wants to go to WEST with proximity 1
Car with ID 4 has passed through the intersection.
Priority given to car from EAST with ID 11 to go to WEST due to higher proximity.
```