**Name: init**

Description: will initialize the following two global variables:

- Struct Map\* baseMap: contains the map showing the delivery range and buildings.

- Struct Truck\* trucks: has the 3 trucks with each truck assigned to the Blue, Green or Yellow route.

- This function is called in main.c to initialize these variables so that they can be further used by the other functions.

Parameters

- This function does not receive any parameters

Returns

- has return type void.

**Name: read**

Description:

- This function is used to get the user’s input from the input buffer, and create a shipment struct based on what the user enters.

- It reads the user-input into 3 variables: weight, size, and destination (4 characters), with each of the input separated by a space as the delimiter.

- The destination variable is then formatted into a row and column variable and converted from a human readable format to a computer-readable format (0, 0).

- This function acts as the main menu, which processes user input, sends input into validation, and then creates a shipment struct if input is valid and sends it to a function

[AssignTrucks function]to get it assigned to the correct truck.

- If user enters ‘0 0 x’ then the program will terminate.

Parameters

- This func does not receive any parameters

Returns

- has return type void.

**Name: validate**

Description:

- This function is used to verify that the parameters received are valid parameters for the Shipment struct.

- This is to make sure that all the shipments ordered have valid weight and size as these values will be passed on to the AssignTruck function

- This function uses the populateMap() function to create a test map to check for valid destinations.

Parameters

- double size: Denotes the package size which can only be 0.25 / 0.50 / 1.00

- int weight: The package weight between 1 to 1000 kg.

- struct Point destination: The shipment’s delivery destination from {0, 0} to {24, 24} and all destinations should be inside of a building.

Returns

- 1: If the package is valid.

- -1: If the size is invalid.

- 0: If the exit code is entered.

- -2: If the weight is invalid.

- -3: If the destination is invalid.

**Name: getAllTruckPaths**

Description:

- Gets all the valid routes from the shortestPath function. This is imp as there are points of a truck’s route that cannot reach the destination due to traveling backwards etc

- It populates the "routes" array with all valid routes that reach the destination of the shipment. [valid routed have destination point at array end]

- The "size" parameter is a pointer to an integer that holds the size of the "routes" array. It is used to keep track of the number of valid routes found.

- It uses the Truck and the Map structs to determine all the valid routes.

- This function

- goes to each point of the Trucks route

- calculate the shortest route at each point

- checks if the route is valid

- adds route to the routes array

- size is used to keep track of the number of valid routes found

Parameters

- s: a Shipment struct

- truck: a Truck struct

- map: a const Map pointer to a struct containing all points in the map

- routes: an array of Route structs that holds all valid routes

- size: a pointer to an integer that holds the size of the “routes” array

Returns

- return type is void. It just populates array routes

**Name: getBestRoute**

Description:

- Gets the shortest route from the array of valid Routes. The shortest route will be assigned to deliver a package.

- This function loops through all valid routes in the "routes" array and gets the routes length stored in “numPoints”

- The "s0" parameter is an integer that holds the size of the "routes" array. It is used to ensure that the function only loops through valid routes.

- The s parameter is a Shipment struct

- The function keeps track of the index of the route with the shortest distance found so far

Parameters:

- routes: an array of Route structs that holds all valid routes

- s0: an integer that holds the size of the "routes" array

- s: a Shipment struct containing the destination of the shipment

Returns

- int as an index of the Route in "routes" of the shortest distance

**Name: containsDestination**

Description:

- This function checks if the given route contains the shipment destination.

- In all valid routes the last element will be the destination point

Parameters:

- route: a Route struct

- s: a Shipment struct containing the destination of the shipment

Returns:

- 1: if the destination is found in the route

- 0: if the destination is not found in the route

**Name: printRoute**

Description:

- Loops through all the points in the route and displays them in coordinate format

Parameters:

- route: a Route struct

- s: a Shipment struct containing the destination of the shipment

Returns:

- return type is void

**Name: calculateUtilizationScore**

Description:

- Assign a “utilization” score to a truck. A utilization score is a metric to measure how full a truck is.

- utilization score = (cargoWeight / MAX\_CARGO\_WEIGHT) / (cargoVol / MAX\_CARGO\_VOL);

- This function calculates the utilization score for a given truck

- This function calls getBestRoute using which it changes the best route acc to the score between trucks that are equidistant to package destination.

Parameters:

- weight of truck

- volume of truck

Returns:

- float: the score from 0 to 1, where 0 means the truck is empty and 1 means the truck is full

**Name: AssignTruck**

Description:

- Calculates which truck is best to deliver the shipment and assigns the shortest route to the respective truck

Parameters:

- struct Shipment

- struct Truck trucks[3]: list of trucks

- struct Map \*map:

Returns:

- 2: blue truck

- 4: green truck

- 8: yellow truck

--1: cannot be delivered