Example Project

Generated by Doxygen 1.7.6.1

Tue Jan 21 2014 12:20:44

Contents

Chapter 1

Main Page

This short program contains some sample code illustrating how the doxygen comments must appear

- for documenting a file, and in particular
- for documenting functions

when we intend to use the doxygen tool for preparing HTML documentation of our code. Note that this file is intended *only* to illustrate a particular set of **commenting conventions** and how they show up when implemented with doxygen. Your actual requirements may not be the same as shown here. Im particular you may need less (or more) than is shown here.

Pay careful attention, in the source code,

to the distinction between the special doxygen comments,

which produce output here, and regular C++ comments which are ignored by doxygen, and look at the source code to see how line breaks in this paragraph are produced.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 vehicle Struct Reference

Public Attributes

- int type
- char orientation
- int rowPos
- int colPos

4.1.1 Member Data Documentation

4.1.1.1 int vehicle::colPos

This is the column number of the vehicle (for ID)

4.1.1.2 char vehicle::orientation

Determines whether vehicle is horizontal or vertical

4.1.1.3 int vehicle::rowPos

This is the row number of the vehicle (for ID)

4.1.1.4 int vehicle::type

Determines whether vehicle is a car or truck

The documentation for this struct was generated from the following file:

- example.cpp

Chapter 5

File Documentation

5.1 example.cpp File Reference

```
#include <iostream>
```

Classes

- struct vehicle

Functions

- void clearGrid (char grid[][COL])
- bool moveForward (char grid[][COL], vehicle &vType)
- int main ()

Variables

- const int ROW = 6
- const int COL = 6
- const int MAX_CAR = 10
- int carNums = 0

5.1.1 Function Documentation

5.1.1.1 void clearGrid (char *grid[][COL]*)

This function will initialize the grid so that each spot is vacant.

My program relies on a vehicle identifier. More specifically, the vehicle is either a truck (T) or a car (C). In my readData, moveForward, and moveBackward functions, I need

to distinguish between which vehicle is in which location on my grid. By initializing each location in my grid with a 'X', I am able to easily identify which element is empty.

Other than initialize the grid for valid operations, this function performs no other operations.

Parameters

grid[][]	the grid to initialize

Returns

This function is a void.

Precondition

The grid will be uninitialized

Postcondition

The grid will be initialized (will contain an 'X' in every position).

5.1.1.2 int main ( )

5.1.1.3 bool moveForward (char *grid[][COL],* vehicle & *vType*)

This function will move the vehicle forward (if it can even be moved forward).

This boolean function will check to see if the vehicle can move forward, and if it can, the vehicle will be updated. The function begins by checking the orientation of the vehicle. It does this because the operations for moving the vehicle are different if it is horizontal or vertical.

Depending on the orientation of the vehicle, the function will perform a series of tests to ensure that the vehicle can transition into a valid space on the grid. The first check it will perform (for both orientations) is to ensure that we are not moving to a space that is off of our grid. The second test (again, for both orientations) will ensure that we are not moving into a space that is already occupied by another vehicle. If these conditions are met, the function will then update the grid and the respective information associated with the vehicle we are operating on.

Parameters

grid[][]	the current traffic jam scenario
vType	a single vehicle to be updated

Returns

This function will return true if the vehicle can be updated. Also, if the vehicle can be updated, it will be updated. This all occurs if the function returns true. If the vehicle cannot be updated, it will return false.

Exceptions

None	

Precondition

The grid will reflect the current game state.

Postcondition

If the vehicle can be moved, the grid, along with the respective data members of the vehicle, will be updated.

5.1.2 Variable Documentation

5.1.2.1 int carNums = 0

Number of cars in grid

5.1.2.2 const int COL = 6

The columns of the grid

5.1.2.3 const int MAX_CAR = 10

The maximum number of cars allowed

5.1.2.4 const int ROW = 6

The rows of the grid