PA05 Rush Hour

Generated by Doxygen 1.8.11

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 Vehicle Struct Reference

Public Attributes

- int length
- char orientation
- int row
- int column

3.1.1 Detailed Description

this structore holds all of the vehicle data

3.1.2 Member Data Documentation

3.1.2.1 int Vehicle::column

3.1.2.2 int Vehicle::length

3.1.2.3 char Vehicle::orientation

3.1.2.4 int Vehicle::row

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

- rushhour.cpp

Chapter 4

File Documentation

4.1 rushhour.cpp File Reference

```
#include <iostream>
#include <unistd.h>
```

Classes

- struct Vehicle

Functions

- void read (int board[ ][MAX_ARR], int &numCars, Vehicle cars[ ])
- void setBoard (int board[ ][MAX_ARR], const Vehicle &v, const int car)
- bool isCar (const Vehicle &v)
- void print (const int board[ ][MAX_ARR])
- void fillArray (int board[ ][MAX_ARR])
- bool moveForward (Vehicle &v, int board[ ][MAX_ARR])
- bool moveBackward (Vehicle &v, int board[ ][MAX_ARR])
- bool isComplete (const Vehicle &v, const int board[ ][MAX_ARR])
- bool isHorizontal (const Vehicle &v)
- void solve (int numMoves, Vehicle cars[ ], int board[ ][MAX_ARR], int &best, const int &numCars, bool &result)
- bool isCollisionForward (const Vehicle &v, const int board[ ][MAX_ARR])
- bool isCollisionBackward (const Vehicle &v, const int board[ ][MAX_ARR])
- int main ()

Variables

- const int CAR = 2
- const int TRUCK = 3
- const char HORIZONTAL = 'H'
- const int MAX_ARR = 6
- const int MAX_VEHICLE = 10

4.1.1 Detailed Description

Author

Aaron Mcanerney

Version

Revision 1.0 solves the rush hour game using DFS

Uses DFS to solve the rush hour puzzle game. uses a struct to hold vehicles and does everything else with pass by reference methods.

Date

10/3/2017

4.1.2 Function Documentation

4.1.2.1 void fillArray (int *board[ ][MAX_ARR]*)

fillArray method that populates the board with 0's

Returns

void

Parameters

board	board that the game is played on

Precondition

and empty board

Postcondition

a 2d array filled with zeros

4.1.2.2 bool isCar (const Vehicle & *v*)

IsCar method that indcates whether a vehicle is a car or a truck

Returns

bool is a a car?

Parameters

v	a vehicle

Precondition

vehicle v

Postcondition

whether or not they vehicle is a car

4.1.2.3 bool isCollisionBackward (const Vehicle & *v,* const int *board[ ][MAX_ARR]*)

isCollissionForward method that indcates whether or not moving a vehicle backwards results in a collision

Returns

bool indicating collision course

Parameters

board	board that the game is played on
v	a vehicle

Precondition

vehicle v, 2d board

Postcondition

a boolean value indicating collision

4.1.2.4 bool isCollisionForward (const Vehicle & *v,* const int *board[ ][MAX_ARR]*)

isCollissionForward method that indcates whether or not moving a vehicle forward results in a collision

Returns

bool indicating collision course

Parameters

board	board that the game is played on
v	a vehicle

Precondition

vehicle v, 2d board

Postcondition

a boolean value indicating collision

4.1.2.5 bool isComplete (const Vehicle & *v,* const int *board[ ][MAX_ARR]*)

isComplete used as base case. Determines whether or not to still play the game.

Returns

boolean Whether or not the first car is at the far right position

Parameters

board	board that the game is played on
v	a vehicle

Precondition

vehicle v, 2d board

Postcondition

a boolean value indicating if the game is complete.

4.1.2.6 bool isHorizontal (const Vehicle & *v*)

IsHorizontal method that indcates whether a vehicle is horizontal

Returns

bool is a horizontal?

Parameters

v	a vehicle

Precondition

vehicle v

Postcondition

whether or not they vehicle is horizontal

4.1.2.7 int main ( )

Main method

Returns

int indicating success

Precondition

unsolved rush hour

Postcondition

solved rush hour

4.1.2.8 bool moveBackward (Vehicle & *v*, int *board[ ][MAX_ARR]*)

MoveBackward method that indcates whether or not moving a vehicle backward is legal and moves the car forward if so

Returns

bool indicating if the vehicle was moved backward

Parameters

board	board that the game is played on
v	a vehicle

Precondition

vehicle v, 2d board

Postcondition

a boolean value indicating if the car was moved. A car in a new position on the board.

4.1.2.9 bool moveForward (Vehicle & *v*, int *board[ ][MAX_ARR]*)

MoveForward method that indcates whether or not moving a vehicle forward is legal and moves the car forward if so

Returns

bool indicating if the vehicle was moved forward

Parameters

board	board that the game is played on
v	a vehicle

Precondition

vehicle v, 2d board

Postcondition

a boolean value indicating if the car was moved. A car in a new position on the board.

4.1.2.10 void print (const int *board[ ][MAX_ARR]*)

print method that prints the board

Returns

void

Parameters

board	board that the game is played on

Precondition

a const board

Postcondition

a printed 2d array

4.1.2.11 void read (int *board[ ][MAX_ARR],* int & *numCars,* Vehicle *cars[ ]*)

read method that populates the board and vehicles array. uses helper function set board.

Returns

void

Parameters

board	board that the game is played on
v	a vehicle
numCars	the number of cars on the board

Precondition

unfilled board and car array

Postcondition

filled board and car array

4.1.2.12 void setBoard (int *board[ ][MAX_ARR],* const Vehicle & *v,* const int *car*)

Set board method that populates the two dimensional array with cars

Returns

void

Parameters

v	a vehicle
board	board that the game is played on

Precondition

array and vehile with car number to be displayed

Postcondition

a board with a new car in postion x,y

4.1.2.13 void solve (int *numMoves,* Vehicle *cars[ ],* int *board[ ][MAX_ARR],* int & *best,* const int & *numCars,* bool & *result*)

solve method that recursivley checks every possible move and calculates the minimum possible moves it requires to complete the game (if such moves exist)

Returns

void

Parameters

board	board that the game is played on
cars	an array containing every car on the board
numMoves	the number of moves currently used
best	the best score thus far
result	indicates whether or not the puzzle is solvable
numCars	number of cars currently on the board

Precondition

vehicle v, 2d board

Postcondition

a boolean value indicating if the car was moved. A car in a new position on the board.

4.1.3 Variable Documentation

4.1.3.1 const int CAR = 2

4.1.3.2 const char HORIZONTAL = 'H'

4.1.3.3 const int MAX_ARR = 6

4.1.3.4 const int MAX_VEHICLE = 10

4.1.3.5 const int TRUCK = 3

Index