Rush Hour

1

Generated by Doxygen 1.8.8

Wed Oct 1 2014 07:52:54

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 car Struct Reference

Public Attributes

- int size
- char direction
- int ycoor
- int xcoor

3.1.1 Detailed Description

Car Struct These will be the cars of the program. Will store the size, direction, y coordinate and x coordiante

3.1.2 Member Data Documentation

3.1.2.1 char car::direction

3.1.2.2 int car::size

3.1.2.3 int car::xcoor

3.1.2.4 int car::ycoor

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

- rush.cpp

Chapter 4

File Documentation

4.1 rush.cpp File Reference

```
#include <iostream>
#include <cstdlib>
#include <fstream>
```

Classes

- struct car

Functions

- void printGrid (int grid[][GRID_SIZE])
- void fillTable (int grid[][GRID_SIZE], car theCars[], int carSize)
- bool moveForward (int grid[][GRID_SIZE], car &theCar)
- bool moveBack (int grid[][GRID_SIZE], car &theCar)
- bool solved (int grid[][GRID_SIZE], car theCars[])
- bool solveIt (int grid[][GRID_SIZE], car theCars[], int &moveCounter, int &minMoves, int carSize)
- int main ()

Variables

- const int GRID_SIZE = 8

4.1.1 Function Documentation

4.1.1.1 void fillTable (int *grid[][GRID_SIZE],* car *theCars[],* int *carSize*)

This function is going to edit the grid to have all the cars in their proper place

Parameters

the	playing grid which is an integer 2D array

the	cars which are in an array of type Car

Returns

void function

Precondition

a grid and cars have been created

Postcondition

the playing grid will be loaded

empty grid

fill border of game with flag

insert cars

add first part

add rest of each car

4.1.1.2 int main ( )

4.1.1.3 bool moveBack (int *grid[ ][GRID_SIZE],* car & *theCar*)

This function is going to move the car in the parameter back If the car is veritcal it will move it up and if it is horizontal it will move it left

Parameters

the	playing grid which is an integer 2D array
the	car desired to move

Returns

if the car was able to move (true) or not (false)

Precondition

a grid and cars have been filled properly

Postcondition

the playing grid will be good with the car moved one spot backwards

check for direction

check for size

update

4.1.1.4 bool moveForward (int *grid[ ][GRID_SIZE],* car & *theCar*)

This function is going to move the car in the parameter forward If the car is veritcal it will move it down and if it is horizontal it will move it right

Parameters

the	playing grid which is an integer 2D array
the	car desired to move

Returns

if the car was able to move (true) or not (false)

Precondition

a grid and cars have been filled properly

Postcondition

the playing grid will be good with the car moved one spot forward

check for direction

check size

check space in front

4.1.1.5 void printGrid (int *grid[ ][GRID_SIZE]*)

4.1.1.6 bool solved (int *grid[ ][GRID_SIZE],* car *theCars[ ]*)

This function is going to check if our main car has reached the edge

Parameters

the	playing grid which is an integer 2D array
the	car array

Returns

if the car is at the edge (true) and if it is not (false)

Precondition

a grid and cars have been filled properly

Postcondition

everything will be the same

4.1.1.7 bool solveIt (int *grid[ ][GRID_SIZE],* car *theCars[ ],* int & *moveCounter,* int & *minMoves,* int *carSize*)

This function is going to solve the puzzle recursively

Parameters

the	playing grid which is an integer 2D array

the	car array
the	number of moves

Returns

if the game is solved true if not false

Precondition

a grid and cars have been filled properly

Postcondition

our car will be at the end of the board

check if solved

4.1.2 Variable Documentation

4.1.2.1 const int GRID_SIZE = 8

Index