Rush Hour

AUTHOR

Version 3.44

10/01/2014

Table of Contents

Rush Hour

The goal of Rush Hour is to unstick a car from a traffic game. The puzzle involves a 6x6 grid of squares. Vehicles are scattered over the grid at integer locations. The goal is to move your car out of the 6x6 grid and escape the traffic jam.

Author:

Saharath Kleips

Version:

1.00

Vehicles are always 1 square wide, but cars are always 2 squares long and trucks are 3 squares long. Vehicles are orientated either horizontally or vectically relative to the grid. Vehicles cannot move through each other, cannot turn, and cannot move off the edge of the grid. They may move forwards and backwards with respect to their orientation only if they are not blocked by another vehicle. Only one vehicle may move at a time and may only move one square at a time. Move vehicles until your car arrives at the rightmost edge of the grid, where it has escaped the traffic jam.

This program will solve any solvable Rush Hour puzzle and output the minimum possible number of (1 square at a time) moves. **Todo:**

Implement multiple puzzles per program execution.

Todo List

page Rush Hour

Implement multiple puzzles per program execution.

Class Index

Class List

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

File Index

File List

Here is a list of all documented files with brief descriptions:

Class Documentation

vehicle Struct Reference

Public Attributes

char **id**
int **x**
int **y**
char **orientation**
int **length**

Detailed Description

Definition at line 44 of file RushHour.cpp.

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

1 RushHour.cpp

File Documentation

RushHour.cpp File Reference

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

Classes

struct vehicle

Functions

bool loadPuzzle (vehicle grid[6][6])

Loads grid with a Rush Hour puzzle scenario from the console.

bool canForward (vehicle grid[6][6], vehicle aVehicle)

Determines whether a vehicle can move forwards.

bool canBackward (vehicle grid[6][6], vehicle aVehicle)

Determines whether a vehicle can move backwards.

bool moveForward (vehicle grid[6][6], vehicle aVehicle)

Attempts to move the vehicle forward.

bool moveBackward (vehicle grid[6][6], vehicle aVehicle)

Attempts to move the vehicle backward.

void **bruteForce** (vehicle[6][6], int)

bool isSolved (vehicle grid[6][6])

Determines whether the state of the grid is solved by checking the last column for any 0s.

void printGrid (vehicle grid[6][6])

Prints a formatted representation of the grid.

int main ()

Main function that controls user input, console output, and program loops.

Variables

const bool _DEBUG_ = false

Debug console output and function tracing.

const bool _FULL_OUTPUT_ = false

Extended console output for puzzle diagram, moves used, etc.

int minSolution = 10

The lowest possible number of moves to solve the puzzle.

int totalVehicles = 0

The total amount of vehicles in this instnace of the puzzle.

Detailed Description

Definition in file RushHour.cpp.

Function Documentation

bool canBackward (vehicle *grid*[6][6], vehicle *aVehicle*)

Determines whether a vehicle can move backwards.

It can move backwards so long as it is not at the edge of the grid and there are no other vehicles blocking it.

Precondition:

The vehicle is the left most piece for horizontal and top most piece for vertical.

Parameters:

grid	is the current state of the grid to check.
aVehicle	is the character representation of the vehicle in the grid.

True if the vehicle can move backwards, Flase if the vehicle cannot.

Definition at line 222 of file RushHour.cpp.

bool canForward (vehicle *grid*[6][6], vehicle *aVehicle*)

Determines whether a vehicle can move forwards.

It can move forwards so long as it is not at the edge of the grid and there are no other vehicles blocking it.

Precondition:

The vehicle is the left most piece for horizontal and top most piece for vertical.

Parameters:

Returns:

grid	is the current state of the grid to check.
aVehicle	is the character representation of the vehicle in the grid.

True if the vehicle can move backwards, Flase if the vehicle cannot.

Definition at line 183 of file RushHour.cpp.

bool isSolved (vehicle *grid*[6][6])

Determines whether the state of the grid is solved by checking the last column for any 0s.

Parameters:

Returns:

grid	is the grid to check if it has been solved.

True if the grid is solved, False if it is not.

Definition at line 554 of file RushHour.cpp.

bool loadPuzzle (vehicle *grid*[6][6])

Loads grid with a Rush Hour puzzle scenario from the console.

The first integer indicates the number of vehicles (n) in the scenario between 0 <= n <= 10. The next n lines represent 1 vehicle where each line consists of a space separated list. Each

list contains a number (2 or 3) indicating length, a letter (H or V) indicating orientation, and a row number (0-5) of the up most square if orientated vertically or the column number (0-5) of the left most square if orientated horizontally. The first vehicle is treated as the escape vehicle and must be orientated horizontally.

Precondition:

The grid to be loaded must be empty.

Postcondition:

grid is solvable and meets the criteria of the above description.

Parameters:

Returns:

grid	is the grid the Rush Hour scenario will be loaded into.

True if a puzzle has been loaded, False if there is no puzzle.

The number of vehicles in the puzzle.

The length of the vehicle.

The orientation of the vehicle.

The starting row of the vehicle.

The starting column of the vehicle.

Definition at line 103 of file RushHour.cpp.

int main ()

Main function that controls user input, console output, and program loops.

The 2D array of characters that represents the puzzle state.

Definition at line 67 of file RushHour.cpp.

bool moveBackward (vehicle *grid*[6][6], vehicle *aVehicle*)

Attempts to move the vehicle backward.

If there is nothing blocking the vehicle, then it will move one space backward. If there is something blocking the vehicle, then it will stay still. Moving left for horizontal vehicles and moving up for vertical vehicles is considered to be moving backwards.

Precondition:

The vehicle is the left most piece for horizontal and top most piece for vertical.

Postcondition:

The grid will be updated with the new vehicle position.

Parameters:

Returns:

grid	is the grid that will be updated with the new vehicle position.
aVehicle	is the character representation of the vehicle in the grid.

True if the vehicle moves, False if the vehicle stays still.

The row of the piece to check against.

The column of the piece to check against.

Definition at line 340 of file RushHour.cpp.

bool moveForward (vehicle *grid*[6][6], vehicle *aVehicle*)

Attempts to move the vehicle forward.

If there is nothing blocking the vehicle, then it will move one space forward. If there is something blocking the vehicle, then it will stay still. Moving right for horizontal vehicles and moving down for vertical vehicles is considered to be moving forward.

Precondition:

The vehicle is the left most piece for horizontal and top most piece for vertical.

Postcondition:

The grid will be updated with the new vehicle position.

Parameters:

Returns:

grid	is the grid that will be updated with the new vehicle position.
aVehicle	is the character representation of the vehicle in the grid.

True if the vehicle moves, Flase if the vehicle stays still.

Placeholder variable for readability.

Placeholder variable for readability.

Definition at line 263 of file RushHour.cpp.

void printGrid (vehicle *grid*[6][6])

Prints a formatted representation of the grid.

Precondition:

The grid must not be empty.

Parameters:

Returns:

grid	is the grid to print out.

Variable Documentation

const bool _DEBUG_ = false

Debug console output and function tracing.

Definition at line 33 of file RushHour.cpp.

const bool _FULL_OUTPUT_ = false

Extended console output for puzzle diagram, moves used, etc.

Definition at line 35 of file RushHour.cpp.

int minSolution = 10

The lowest possible number of moves to solve the puzzle.

Definition at line 37 of file RushHour.cpp.

int totalVehicles = 0

The total amount of vehicles in this instnace of the puzzle.

Definition at line 39 of file RushHour.cpp.

Index

INDE