Lab 05 - PC2

Generated by Doxygen 1.7.6.1

Tue Sep 30 2014 03:01:58

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 Board Class Reference

```
#include <classes.h>
```

Public Member Functions

- Board ()
- Board (Car[ ], int newNumCars)
- Board operator= (Board source)
- ∼Board ()
- void printBoard ()
- bool solveIt (int movesSoFar)
- bool forward (int carIndex)
- bool backward (int carIndex)
- bool amIDoneYet ()

Public Attributes

- int minMoves
- int numCars
- int boardArr [6][6]
- Car boardCars [10]

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Board::Board ( )

Default constructor for board class.

Default constructor for board class. Sets all values of the array to -1 to show emptyness. initialize variables

set all values of board to -1

3.1.1.2 Board::Board (Car *carList[ ]*, int *newNumCars*)

Parameterized constructor for board class.

Sets a board up with values given in car list with a given number of cars.

Parameters

carList	a list of cars
newNum-Cars	an int of how many cars in list

initialize variables

set all values of board to -1

loop through each car

save car info to list of cars in board class

for car length 2 and horizontal, place on board

for car length 3 and horizontal, place on board

for car length 2 and vertical, place on board

for car length 3 and vertical, place on board

3.1.1.3 Board::∼Board ( )

Default destructor for car class.

No memory was allocated dynamically.

3.1.2 Member Function Documentation

3.1.2.1 bool Board::amIDoneYet ( )

Function to check for completion.

Loops through each row and returns indication of value 0 at right edge (red car made it out).

Returns

bool which indicates red car's status

3.1.2.2 bool Board::backward (int *carIndex*)

Function to move car backward.

Finds appropriate type of car / orientation of car and attempts to move. Fails if data is in the way or is on edge.

Parameters

carIndex	int with given carIndex value

Returns

bool which indicates successful move backward

for length of 2 and horizontal

return false if at edge

return false if car in the way

move car

for length of 3 and horizontal

return false if at edge

return false if car in the way

move car

for length of 2 and vertical

return false if at edge

return false if car in the way

move car

for length of 3 and vertical

return false if at edge

return false if car in the way

move car

3.1.2.3 bool Board::forward (int *carIndex*)

Function to move car forward.

Finds appropriate type of car / orientation of car and attempts to move. Fails if data is in the way or is on edge.

Parameters

carIndex	int with given carIndex value

Returns

bool which indicates successful move forward

for length of 2 and horizontal

return false if at edge

return false if car in the way

move car

for length of 3 and horizontal

return false if at edge

return false if car in the way

move car

for length of 2 and vertical

return false if at edge

return false if car in the way

move car

for length of 3 and vertical

return false if at edge

return false if car in the way

move car

3.1.2.4 Board Board::operator= (Board *source*)

Overloaded assignment operator for board class.

This was not needed but I wrote it anyways. It works. Sets one board equal to another

Parameters

source	a Board to be copied from

Returns

Board new board with source's values

initialize variables

loop through cars

save car info to board's car list

set all values of this board array to equal source's board array

3.1.2.5 void **Board::printBoard** ( )

Print board function.

For testing purposes only. Prints values in board.

Returns

void

initialize variables

loop through array and print each value

3.1.2.6 bool **Board::solveIt** (int *movesSoFar*)

Function to solve board.

Begin's by checking if car 0 is at right edge, if it is, it sets minimum moves if nescessary. Next, it checks amount of moves so far to ensure under max value. Otherwise, it loops through cars and tries to move them forward or backward, recursively.

Parameters

movesSoFar	int that counts moves

Returns

bool which indicates solving completion

initialize variables

check if done yet

save number of moves

checks for too many moves tried

loops through cars

if is able to move car forward, call with new moves so far

if is able to move car backward, call with new moves so far

3.1.3 Member Data Documentation

3.1.3.1 int **Board::boardArr**[6][6]

3.1.3.2 **Car Board::boardCars**[10]

3.1.3.3 int **Board::minMoves**

3.1.3.4 int Board::numCars

The documentation for this class was generated from the following files:

- classes.h
- classes.cpp

3.2 Car Class Reference

```
#include <classes.h>
```

Public Member Functions

- Car ()
- ∼Car ()

Public Attributes

- int length
- bool isHorz
- int rowNum
- int colNum

3.2.1 Constructor & Destructor Documentation

3.2.1.1 Car::Car ( )

Default constructor for car class.

Default constructor for car class. Sets data members to zero and false. initialize variables

3.2.1.2 Car::∼Car ( )

Default destructor for car class.

No memory was allocated dynamically.

3.2.2 Member Data Documentation

3.2.2.1 int Car::colNum

3.2.2.2 bool Car::isHorz

3.2.2.3 int Car::length

3.2.2.4 int Car::rowNum

The documentation for this class was generated from the following files:

- classes.h
- classes.cpp

Chapter 4

File Documentation

4.1 classes.cpp File Reference

```
#include "classes.h" #include <iostream>
```

4.1.1 Detailed Description

Author

CatherinePollock

Date

9/30/14

This is the file that implements classes.h and the classes within that file.

4.2 classes.h File Reference

```
#include <iostream>
```

Classes

- class Car
- class Board

4.2.1 Detailed Description

Author

CatherinePollock

Date

9/30/14

This is the specification file for the classes implemented in classes.cpp. It includes Car and Board classes.

4.3 rush.cpp File Reference

`#include <iostream> #include "classes.h"`

Functions

- int saveCars (Car carArr[ ])

  function prototypes
- bool solve (int movesSoFar)
- int main ()

  main driver

Variables

- const int MAX_CARS = 10

  global constants
- const int BOARD_SIZE = 6

4.3.1 Detailed Description

Author

CatherinePollock

Date

9/30/14

This is the main driver file for the game Rush Hour. It saves given car information into a list of cars and a board array. It attempts to solve for the minimum moves required and prints results.

4.3.2 Function Documentation

4.3.2.1 int **main** ( )

main driver

initalize variables

save cars into car list

put cars onto board

solve for minimum moves

output results

return success

4.3.2.2 int **saveCars** (**Car** *carList[ ]*)

function prototypes

function implementation

saves cars to array

Gets the number of cars to be saved for first scenario. Saves the length, orientation, row and column numbers for each car. Repeats this until numCars is given 0 to end reading in of data. Returns the number of seperate scenarios saved.

Returns

scenarioNum int value of number of scenarios saved

Parameters

array	of cars

initialize variables

get number of cars to be saved

loop through and save car info

save car length

save car orientation

skip white space characters

save orientation correctly based on given letter

save car row number

save car column number

take in zero as end flag

return number of scenarios saved

4.3.2.3 bool solve (int *movesSoFar*)

4.3.3 Variable Documentation

4.3.3.1 const int BOARD_SIZE = 6

4.3.3.2 const int MAX_CARS = 10

global constants