PA11-PC3 Catherine Pollock

Generated by Doxygen 1.7.6.1

Sun Nov 30 2014 19:46:54

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 Board Class Reference

board class

```
#include <rushClasses.h>
```

Public Member Functions

- Board ()
- Board (const Board &)
- Board & operator= (const Board &)
- bool operator== (const Board &)
- bool operator!= (const Board &)
- void saveData (int)
- bool forward (int)
- bool backward (int)
- bool isSolved () const
- bool canMove (int, int) const
- int getNumCars () const

Public Attributes

- int numCars
- Car boardCars [10]

3.1.1 Detailed Description

board class

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Board::Board ( )

Default constructor for board class.

Default constructor for board class. Sets the data members. initialize number of cars to zero

3.1.2.2 Board::Board (const Board & *other*)

Copy constructor for board class.

Sets a board up with values given in car list with a given number of cars. Initializes boardCars with other's boardCars values.

Parameters

other	source of boardCars to copy from

3.1.3 Member Function Documentation

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

backward

Attempts to move car backward. Checks that move is within dimensions of board and that the space to move to is empty. Changes car's row or column number if move was sucessful.

Parameters

carIndex	int of car's index in boardCars

Returns

bool if move was successful

for horizontal car

check that car is not at left edge of board and that it can move left

update car's column value and return success

for vertical car

check that car is not at top edge of board and that it can move up

update car's column value and return success

return that move was not successful

3.1.3.2 bool **Board::canMove** (int *row*, int *col*) const

canMove

Attempts to move a car to specified row and column space. Does this by iterating through each car and checking it's row and col values to see if space can be moved to. Returns whether move is possible ot not.

Parameters

row	int to check
col	int to check

Returns

bool if move is possible

intitialize

iterate through all cars

check entire length of car

if car is horizontal

if car is vertical

return that move is possible

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

forward

Attempts to move car forward. Checks that move is within dimensions of board and that the space to move to is empty. Changes car's row or column number if move was sucessful.

Parameters

carIndex	int of car's index in boardCars

Returns

bool if move was successful

for horizontal car

check that car is not at right edge of board and that it can move right

update car's column value and return success

for vertical car

check that car is not at bottom edge of board and that it can move down

update car's column value and return success return that move was not successful

3.1.3.4 int Board::getNumCars ( ) const

getNumCars

Returns number of cars in board.

Returns

int number of cars in board

3.1.3.5 bool Board::isSolved ( ) const

isSolved

Checks if car of index 0 (red car) is at the edge of the board.

Returns

bool if car is at right edge and board is solved.

3.1.3.6 bool Board::operator!= (const Board & *other*)

Overloaded inequality operator

Returns if boards are not identical

Parameters

other	source of boardCars to compare

Returns

bool if board is not equivalent

returns opposite of equality operator

3.1.3.7 Board & Board::operator= (const Board & *other*)

Overloaded assignment operator

Sets a board up with values given in car list with a given number of cars. Set boardCars with other's boardCars values.

Parameters

other	source of boardCars to copy from

Returns

Board for chain of assignments

initialization

if they are not the same board

set number of cars

loop through to set data values

return reference to this board

3.1.3.8 bool Board::operator== (const Board & *other*)

Overloaded equality operator

Returns if boards are identical or not

Parameters

other	source of boardCars to compare with

Returns

bool if board is identical

initialization

check for same number of cars

loop through and compare cars

return true if identical

3.1.3.9 void Board::saveData (int *cars*)

saveData

Reads in and saves data into boardCars until no more cars to get data for.

Parameters

cars	int of how many cars

initialize

loop through until no more cars

3.1.4 Member Data Documentation

3.1.4.1 Car Board::boardCars[10]

3.1.4.2 int Board::numCars

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

- rushClasses.h
- rushClasses.cpp

3.2 Car Class Reference

car class

```
#include <rushClasses.h>
```

Public Attributes

- int length
- int row
- int col
- char orientation

3.2.1 Detailed Description

car class

3.2.2 Member Data Documentation

3.2.2.1 int Car::col

3.2.2.2 int Car::length

3.2.2.3 char Car::orientation

3.2.2.4 int Car::row

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

- rushClasses.h

Chapter 4

File Documentation

4.1 rushClasses.cpp File Reference

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

4.1.1 Detailed Description

Author

Catherine Pollock

Date

11/29/14

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

4.2 rushClasses.h File Reference

```
#include <iostream>
```

Classes

- class Car

  car class
- class Board

  board class

4.2.1 Detailed Description

Author

CatherinePollock

Date

11/29/14

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

4.3 rushSTL.cpp File Reference

```
#include <iostream> #include <string> #include <sstream> ×
#include <map> #include <queue> #include "rushClasses.h"
```

Functions

- int solve (const int)

  function prototypes
- string boardToString (const Board &)
- Board stringToBoard (const string)
- int main ()

  main driver

Variables

- const int MAX_CARS = 10

  global constants
- const int BOARD_SIZE = 6

4.3.1 Detailed Description

Author

Catherine Pollock

Date

11/29/14

This is the main driver file for the game Rush Hour. It saves given car information into boards, which are saved into a queue and map with checked boards. It attempts to solve for the minimum moves required and prints results. This is done with a breath first search, that checks across each level before continuing search.

4.3.2 Function Documentation

4.3.2.1 string **boardToString** (const **Board** & *board*)

boardToString

Converts a board to a string containing car data.

Parameters

board	Board to get string data from

Returns

string containing board data

initialize

concatenate string with each car's data

return string with car data for board

4.3.2.2 int **main** ( )

main driver

initialization

read in number of cars in first scenario

continue until no more scenarios

solve the board

print results

increment scenario number

read in number of cars for next scenario

4.3.2.3 int **solve** (const int *numCars*)

function prototypes

function implementation

solve

Attempts to solve the board using a breath first search. Creates a queue of boards and adds boards as moves are made. Reads in data for initial board. Makes moves on board and saves boards. Checks if board is solved and returns numMoves for board that is solved soonest.

Parameters

numCars	int number of cars in board

Returns

number of moves required

initialization

save data to board

save board as a string

check if board is solved

adds board to queue

adds board to map of checked boards

continue until queue is empty

take board off queue

finds board on checked board and saves numMoves

check if board is solved

iterate through cars on board

move car forward

update board string

check map of boards

put board on queue

put board on checked boards map

move car backward

update board string

move car backward

update board string

check map of boards

put board on queue

put board on checked boards map

move car backward

update board string

return -1 if not solved

4.3.2.4 Board stringToBoard (const string *boardString*)

stringToBoard

Converts a string to a board containing car data.

Parameters

boardString	string to get board data from

Returns

board with car data

initialize

return Board with data from string

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