My Project

Generated by Doxygen 1.8.11

Contents

Chapter 1

Main Page

This short program contains a recursive solution to solving a game of Rush Hour in the least possible amount of moves using breadth-first search.

The Board class manages all of the Vehicle objects within it, which does so by modifying an array and moving the car around by changing their row and column positions. The Vehicles inside of the Board class act as a Linked List, and whenever a function such as *moveForward()* is called, it will perform the method on the car that the cursor pointer is pointing to.

Chapter 2

README

#PA11-Rush-Hour-BFS

help us please

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Class Documentation

5.1 Board Class Reference

Public Member Functions

- Board ()
- **Board** (const Board &other)
- ~Board ()
- Board & **operator=** (const Board &other)
- void insert (int length_set, char orient_set, int row_set, int col_set)
- bool isEmpty () const
- void moveToBeginning ()
- void gotoNext ()
- void gotoPrior ()
- void remove ()
- void clear ()
- void setCursor (int i)
- void initBoard ()
- bool moveForward ()
- bool moveBackward ()
- bool isSolved () const
- void **printBoard** ()
- void readInput (int car_num)
- string boardToString ()

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Board::Board ( ) [inline]

Default constructor for board.

Simply initalizes the board. Sets grid elements to 0 and sets pointers.

Parameters

N/A	

Precondition

The board will not be initialized.

Postcondition

The board will be declared, and the grid be ready to have data to be written to it.

Exceptions

N/A	

Note

This method constructs a default board object

Returns

N/A

5.1.1.2 Board::~Board ( ) [inline]

Destructor for board.

Simply initalizes the board. Sets grid elements to 0 and sets pointers.

Precondition

The board will have a list of Vehicle objects.

Postcondition

All of the Vehicles will be deallocated from memory, and the List will be deleted from memory.

5.1.2 Member Function Documentation

5.1.2.1 string Board::boardToString ( ) [inline]

Parameters

int	(&board)[MAXBOARDSIZE][MAXBOARDSIZE]

Precondition

2d board is not converted to a string

Postcondition

2d board is converted to a string

Exceptions

N/A	

Note

the chars after N designate the numofmoves

5.1.2.2 void Board::clear ( ) [inline]

Removes all Vehicle objects from List.

Loops through the List by setting the cursor to the head, and deleting the head until the cursor is pointing to a nonexistent object.

Precondition

Cursor will be pointing at any Vehicle in the List.

Postcondition

The List will be empty.

5.1.2.3 void Board::gotoNext ( ) [inline]

Moves the cursor to the next Vehicle.

The cursor will be set equal to its next pointer, which is of type Vehicle.

Precondition

Cursor will be pointing at any Vehicle in the List.

Postcondition

Cursor will be pointing to the next Vehicle.

5.1.2.4 void Board::gotoPrior ( ) [inline]

Moves cursor to the previous Vehicle.

Sets a temp pointer that points to the head, and loops through the List until the temp pointer sees that the next Vehicle is the one that the cursor points to. The cursor is then assigned to the temp.

Precondition

Cursor will be pointing at any Vehicle in the List.

Postcondition

Cursor will be pointing to the previous Vehicle.

5.1.2.5 void Board::initBoard ( ) [inline]

Sets all elements of parking lot to 0, otherwise known as "empty."

Precondition

Lot will have cars occupying spaces.

Postcondition

The lot will be empty.

Set entire board to all zeros

5.1.2.6 void Board::insert (int *length_set,* char *orient_set,* int *row_set,* int *col_set*) [inline]

Inserts a Vehicle into the board, based on passed parameters.

Initalizes the board. Sets grid elements to 0 and sets pointers. Updates the board with the new Vehicle that was just inserted.

Parameters

length_set	The length of the car.
orient_set	The orientation of the car.
row_set	Row position of car.
col_set	Column position of car.

Returns

This is a void function.

Precondition

Car data will be obtained and be ready to be passed.

Postcondition

A new Vehicle object will be created, with the data that was passed into the method.

First, insert the car and its data into the linked list.

Then, update the board with the newly inserted car.

5.1.2.7 bool Board::isEmpty ( ) const [inline]

Checks if the List is empty or not.

Simply returns if head points to NULL or not.

Precondition

n/a.

Postcondition

The status of the head pointer will be returned.

5.1.2.8 bool Board::isSolved ( ) const [inline]

Checks if the board is solved.

Checks if the first Vehicle object's column position is 4, since that would indicate that the Vehicle object made it across the grid succesfully.

Precondition

n/a.

Postcondition

Status of solution will be returned.

5.1.2.9 bool Board::moveBackward ( ) [inline]

Moves the car backward (west/north).

Moves the car based on orientation. For both horizontal and vertical orientations, the car's head will check if the element behind the head of the car is empty or not. It also checks if the column position is greater than the boarders of the grid.

If conditions are met, the row/column position will be decremented, and have its new head spot in the grid to be set to 1.

Precondition

Vehicle will be sitting in a certain position.

Postcondition

Vehicle will be moved left/up one element.

Move head of car to the right

Erase previous tail position (clean up).

Finally, return successful move.

Move head of car down

Erase previous tail position (clean up).

Finally, return successful move.

5.1.2.10 bool Board::moveForward ( ) [inline]

Moves the car forward (east/south).

Moves the car based on orientation. For both horizontal and vertical orientations, the car's head will check if the head + length of car is equal to 0. This should work for any size car. It also checks if the column position is less than the size of the grid minus the length of the car.

If conditions are met, the row/column position will be iterated, and have its new head spot in the grid to be set to 1. A loop is run, that for as long as the iterator is less than the length, start filling ones to the right/bottom of the Vehicle.

Precondition

Vehicle will be sitting in a certain position.

Postcondition

Vehicle will be moved right/down one element.

Move head of car to the right

Finally, return successful move.

Move head of car down

Then, update the board with the newly inserted car.

Finally, return successful move.

5.1.2.11 void Board::moveToBeginning ( ) [inline]

Moves the cursor to the beginning.

Sets the cursor to the head.

Precondition

Cursor will be pointing at any Vehicle in the List.

Postcondition

Cursor will be pointing to the first Vehicle of the List.

5.1.2.12 void Board::readInput (int *car_num*) [inline]

Reads the input added by the user .

Reads the length orient, row and col of the new car used to create new car objects on the board

Precondition

n/a.

Postcondition

inserts the values into a new car object

Create variables that will read in data, which will then be inserted.

Insert car based on stats.

5.1.2.13 void Board::remove ( ) [inline]

Removes a Vehicle from the List.

First checks if the cursor is at the head, and if so, move the cursor to the next Vehicle and delete the head, as well as reassigning the head after. Otherwise, set a temp cursor pointing to the Vehicle to delete. Move the cursor backwards one, and then link the Vehicle to the Vehicle AFTER the temp pointer. Then delete the pointer. If the cursor not at the end of the List, move the cursor one Vehicle, otherwise, return the cursor to the head of the List.

Precondition

Cursor will be pointing at any Vehicle in the List.

Postcondition

Vehicle will be deleted and the cursor will be repositioned.

5.1.2.14 void Board::setCursor (int *i*) [inline]

Sets the cursor to the number of a car.

Takes in an integer, and then loops through the list until the counter is one less than the passed integer.

Precondition

Cursor will be pointing at any Vehicle in the List.

Postcondition

Cursor will be pointing at the Vehicle number passed.

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

- Game.h

Chapter 6

File Documentation

6.1 RushHour.cpp File Reference

This program will solve a traffic jam. The result will be the solution with the smallest number of moves.

```
#include <iostream>
#include <cstdlib>
#include <cstddef>
#include <map>
#include <queue>
#include <string>
#include "Game.h"
```

Include dependency graph for RushHour.cpp:

Index