PA11 - RushHour

Generated by Doxygen 1.8.11

Contents

1 Main Page

This program contains the necessary functions to implement

- Setup() that takes the inputs from the terminal and sets up the board according to the given specifications.
- moveFoward() takes in a car number and moves the car Forward if it is possible.
- moveBackward() takes in a car number and moves the car Backward if it is possible.
- isSolved() to check if car 0 is at the end of the board.
- Print() this was used for my own testing purposes to print out the board and its data.
- clear() used to clear the board , essentially the same as setup but used to make sure that the board is reset so the next board can be setup.
- SolveIt() which will be used to try to solve the board if possible.
- boardToString() which will convert our board into a string.

We implemented it with two classes, called Board and Car. The board contains all of the information about the board including the number of cars on the board and the number of current moves taken to solve the board. To prevent seg faults, We had to keep in mind the boundaries of the board when moving the cars inside the board. For my implementation, I created an int board, where 0 marked empty spaces with no cars that other cars could freely move to and (i+1) the car number in which it was entered in the terminal(so the first car will be 1, second car will be 2 and so on) on the board represented where the space was occupied by a car and another car could not move there. I used the board to see and decide if the cars could move forward or backward based on its current position.

2 Class Index

2.1 Class List

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

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 Board Class Reference

Classes

- class Car

Public Member Functions

- Board ()
- void Setup ()
- bool moveForward (int i)
- bool moveBackward (int i)
- bool isSolved ()
- void Print ()
- void clear ()
- string boardToString ()

Public Attributes

- int numberOfCars
- int numberOfMoves
- int BestMoves

Private Attributes

- int board [6][6]
- Car cars [18]

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Board::Board ( )

This function is the default constructor for the Board class.

This function will create a defaulted object of the Board class with a board that is empty, in other terms filled with 0s.The numberOfCars, numberOfMoves and BestMoves variables will be set to 0. The board will be empty and be entirely filled with 0s. The Car class will be intialized with the row being -1, the col being -1, the length being 0, and its orientiation members, vertical and horizontal set to false.

Parameters

none.	

Returns

This function does not return anything.

Precondition

none.

Postcondition

The Board class will be intialized and the private member Car class will also be initalized within this function.

4.1.2 Member Function Documentation

4.1.2.1 string Board::boardToString ( )

This function is the boardToString function of the Board class.

This function creates a string representing the board array in the Board class and returns it. It uses a nested for loop to add each element of the 2d array to the string.

Parameters

NONE	

Returns

This function returns a string which represents the board array in the Board class.

Precondition

Function takes the 2d board array in the current class.

Postcondition

A 2d board array is converted into a string and returned.

4.1.2.2 void Board::clear ( )

This function is the clear function for the Board class.

The function is intended to clear the board so that the next board can be setup properly. This This function will clear the Board class and set the board to empty, in other terms filled with 0s.The numberOfCars, numberOfMoves and BestMoves variables will be set to 0. The board will be empty and be entirely filled with 0s. The Car class will be cleared with the row being -1, the col being -1, the length being 0, and its orientiation members, vertical and horizontal set to false.

Parameters

none.	

Returns

This function does not return anything.

Precondition

Usually, it till be a class that is initialized already so that it may be reset.

Postcondition

The Board class will be cleared, and is identical to the default constructor.

4.1.2.3 bool Board::isSolved ( )

This function is the isSolved function of the Board class.

This function returns by checking to see if the first car's col + its length is equal to 6. If that is true, it returns true, otherwise it returns false.

Parameters

none.	

Returns

This function returns true if the first car's col + its length is equal to 6, else it will return false.

Precondition

Usually, it till be a class that is initialized and setup so we can test to see if the board is solved.

Postcondition

Nothing will be changed, but it will return true of false to check if the board is solved.

4.1.2.4 bool Board::moveBackward (int *i*)

This function is the moveBackward function of the Board class.

This function takes in a parameter of int i, which determines which car to check for. This function checks to see if the chosen car can move backwards. If the car can move backwards, the function will adjust the board properly to do so. The function will then return true if it had succeeding in moving backwards. If the function cannot do so, it will return false.

Parameters

int	i , which designates which car to check to move.

Returns

This function returns true if the car can move backwards and if the car cannot move backwards, it will return false.

Precondition

Usually, it till be a class that is initialized and setup so we can test to see if the car can move.

Postcondition

The car will be moved backwards 1 slot from its current position. The board will then be updated to represented to show where the cars are.

4.1.2.5 bool Board::moveForward (int *i*)

This function is the moveForward function of the Board class.

This function takes in a parameter of int i, which determines which car to check for. This function checks to see if the chosen car can move forward. If the car can move forward, the function will adjust the board properly to do so. The function will then return true if it had succeeding in moving forward. If the function cannot do so, it will return false.

Parameters

int	i , which designates which car to check to move.

Returns

This function returns true if the car can move forward and if the car cannot move forwards, it will return false.

Precondition

Usually, it till be a class that is initialized and setup so we can test to see if the car can move.

Postcondition

The car will be moved forwards 1 slot from its current position. The board will then be updated to represented to show where the cars are.

4.1.2.6 void Board::Print ( )

This function is the print function for the Board class.

This function prints the data in the board class and the data of each car in the Car class for debugging and testing purposes.

Parameters

none.	

Returns

This function does not return anything.

Precondition

Usually, it till be a class that is initialized and setup, so i can print out its data.

Postcondition

Nothing will change, but the data of the board and cars will be outputted to the terminal.

4.1.2.7 void Board::Setup ( )

This function is the setup function for the Board class.

This function takes in the input of the terminal and properly arranges the board from the specified inputs. This function also takes the input of the cars and properly fills the data members of the Car class based on the inputs. The board will be filled with (i+1) the car number in which it was entered in the terminal(so the first car will be 1, second car will be 2 and so on), where a car exists and 0s where a car does not exist. We did this so that it was easier to convert the board into strings and see if it was seen before since using all ones would cause confusion with orientation for some cars. The class will contain the numberOfCars in the board. The Car class will input the size of the cars, its orientation will be set to true and its row and col will be set for its starting position.

Parameters

none.	

Returns

This function does not return anything.

Precondition

Usually, it till be a class that is initialized already so that it may be properly setup.

Postcondition

The Board class and Car class will be setup to contain the information that is taken from the terminal as an input.

4.1.3 Member Data Documentation

4.1.3.1 int Board::BestMoves

4.1.3.2 int Board::board[6][6] [private]

4.1.3.3 Car Board::cars[18] [private]

4.1.3.4 int Board::numberOfCars

4.1.3.5 int Board::numberOfMoves

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

- RushHour.cpp

4.2 Board::Car Class Reference

Public Attributes

- int row
- int col
- int length
- bool vertical
- bool horizontal

4.2.1 Member Data Documentation

4.2.1.1 int Board::Car::col

4.2.1.2 bool Board::Car::horizontal

4.2.1.3 int Board::Car::length

4.2.1.4 int Board::Car::row

4.2.1.5 bool Board::Car::vertical

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

- RushHour.cpp

5 File Documentation

5.1 okay.cpp File Reference

5.2 RushHour.cpp File Reference

```
#include <iostream>
#include <map>
#include <queue>
#include <string>
```

Classes

- class Board
- class Board::Car

Functions

- int SolveIt ()
- int main ()

5.2.1 Function Documentation

5.2.1.1 int main ( )

5.2.1.2 int SolveIt ( )

This is the free function SolveIt.

This creates a board based on user setup and solves the board using the least amount of move possible, returning this value to the main function. It uses a queue to iterate through board configurations and a map to check if specific configurations have already been seen. Once the board is set up, it is pushed onto the queue. While the queue is not empty it will continue to find a solution. Once in the while loop it first sets the rushHour board to the first board on the queue, then pops the queue If the board is solved it returns the number of moves to main. Otherwise it begins a for loop that will iterate though every vehical on the board. In the for loop it first checks if the vehical indicated by the index can move forward, moving forward if it can. If it moves forward, it then checks the map to see if the board has been seen before. If not, it increments the number of moves, adds the board to the map, adds the board to the queue, then decrements the number of moves. It then checks if the vehical indicated by the index can move backward, moving backward if it can. If it moves backward, it then checks the map to see if the board has been seen before. If not, it increments the number of moves, adds the board to the map, adds the board to the queue, then decrements the number of moves. The for loop then reapeats. Once the for loop finishes, the while loop loops again.

Parameters

NONE	

Returns

This function returns an integer which indicates the minimumn number of moves to solve a board.

Precondition

Function is called to solve a board the user will input.

Postcondition

A board is set up and the minimum number of moves to solve the board is found and returned.

Index