Assignment 4

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

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

boardUniverse
Represents the state of a Game of life . . . . . **??**

Chapter 2

File Index

2.1 File List

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

include/DisplayGen.h
Printing the current grid universe and cell states. to the console . . . . . **??**
include/Gameboard.h
Provides an ADT representing a model for the game of life by Conway including game board state and its transitions . . . . . **??**
include/GameboardTypes.h
Provides type definition and enumerations for a game grid and cells . . . . . **??**
include/ReadWrite.h
Reading text file to build game grid and state, and outputting current state of game grid to text fie **??**

Chapter 3

Class Documentation

3.1 boardUniverse Class Reference

Represents the state of a Game of life.

`#include <Gameboard.h>`

Public Member Functions

- boardUniverse (std::string f)

  Constructs a new Grid with n rows and n columns intialized each cell to be of type CellT(dead or alive).
- bool overSize (int rows, int cols)

  Checks if grid's rows or columns exceed the maximum limit set.
- bool rowInBound (int row)

  Checks if grid's row number is a positive number and doesn't exceed the gird's total number of rows.
- bool colInBound (int col)

  Checks if grid's column number is a positive number and doesn't exceed the gird's total number of columns.
- int getRows ()

  Gets total number of rows to define game grid.
- int getCols ()

  Gets total number of columns to define game grid.
- std::vector< std::vector< CellT > > getUniverse ()

  Gets game grid with cell states.
- CellT cellState (int row, int col)

  Gets cell state of specific cell on game grid.
- int numAliveNeighbors (int row, int col)

  Gets number of neighbors in state Alive of current cell.
- void NextUniverse ()

  Uses the state of the cells of the current grid to create a new grid state.
- bool gameOver ()

  Checks to see if game is over by checking if all cells of the grid are in state Dead.

3.1.1 Detailed Description

Represents the state of a Game of life.

The board consists of a n by n grid with cells having state alive or dead from which the current grid creates the next state of the cells.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 boardUniverse()

```
boardUniverse::boardUniverse (
            std::string f )
```

Constructs a new Grid with n rows and n columns intialized each cell to be of type CellT(dead or alive).

Parameters

f	File name containing initial gameboard state

Exceptions

out_of_range	If the number of rows of columns exceed the max rows or max columns defined for the game.

3.1.3 Member Function Documentation

3.1.3.1 cellState()

```
CellT boardUniverse::cellState (
            int row,
            int col )
```

Gets cell state of specific cell on game grid.

Parameters

row	Represents specific row of the game grid.
col	Represents specific column of the game grid.

Exceptions

out_of_range	If column and row are out of bounds of game grid.

Returns

CellT representing state of cell at that row and column.

3.1.3.2 colInBound()

```
bool boardUniverse::colInBound (
            int col )
```

Checks if grid's column number is a positive number and doesn't exceed the gird's total number of columns.

Parameters

col	Represents specific column of the game grid.

Returns

Boolean representing if column is in bound of the game grid.

3.1.3.3 gameOver()

```
bool boardUniverse::gameOver ( )
```

Checks to see if game is over by checking if all cells of the grid are in state Dead.

Returns

Boolean representing if all cells are Dead and game is over.

3.1.3.4 getCols()

```
int boardUniverse::getCols ( )
```

Gets total number of columns to define game grid.

Returns

Integer representing total number of columns in game grid.

3.1.3.5 getRows()

```
int boardUniverse::getRows ( )
```

Gets total number of rows to define game grid.

Returns

Integer representing total number of rows in game grid.

3.1.3.6 getUniverse()

```
std::vector<std::vector<CellT> > boardUniverse::getUniverse ( )
```

Gets game grid with cell states.

Returns

Vector of CellT representing game grid with cell states.

3.1.3.7 NextUniverse()

```
void boardUniverse::NextUniverse ( )
```

Uses the state of the cells of the current grid to create a new grid state.

Uses the state of the cells of the current grid to create a new grid with required changes to the state of certain cells and assigns the new grid as the current grid of the game.

3.1.3.8 numAliveNeighbors()

```
int boardUniverse::numAliveNeighbors (
            int row,
            int col )
```

Gets number of neighbors in state Alive of current cell.

Parameters

row	Represents specific row of the game grid.
col	Represents specific column of the game grid.

Returns

Integer representing number of cells in state Alive that are neighbors to cell at row and column.

3.1.3.9 overSize()

```
bool boardUniverse::overSize (
            int rows,
            int cols )
```

Checks if grid's rows or columns exceed the maximum limit set.

Parameters

rows	Represents number of rows of the game grid.
cols	Represents number of columns of the game grid.

Returns

Boolean representing if row or column exceed maximum limit set.

3.1.3.10 rowInBound()

```
bool boardUniverse::rowInBound (
            int row )
```

Checks if grid's row number is a positive number and doesn't exceed the gird's total number of rows.

Parameters

row	Represents specific row of the game grid.

Returns

Boolean representing if row is in bound of the game grid.

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

- include/Gameboard.h

Chapter 4

File Documentation

4.1 include/DisplayGen.h File Reference

Printing the current grid universe and cell states. to the console.

```
#include <iostream>
#include <vector>
#include "GameboardTypes.h"
```

Functions

- void showUniverse (std::vector< std::vector< CellT >> Universe)

  Printing the current grid universe and cell states to the console.

4.1.1 Detailed Description

Printing the current grid universe and cell states. to the console.

Author

Harsh Patel

4.1.2 Function Documentation

4.1.2.1 showUniverse()

```
void showUniverse (
            std::vector< std::vector< CellT >> Universe )
```

Printing the current grid universe and cell states to the console.

Parameters

Universe	Current grid of the game of life.

4.2 include/Gameboard.h File Reference

Provides an ADT representing a model for the game of life by Conway including game board state and its transitions.

```
#include <iostream>
#include "GameboardTypes.h"
#include "ReadWrite.h"
#include <vector>
```

Classes

- class boardUniverse

  Represents the state of a Game of life.

4.2.1 Detailed Description

Provides an ADT representing a model for the game of life by Conway including game board state and its transitions.

Author

Harsh Patel

4.3 include/GameboardTypes.h File Reference

Provides type definition and enumerations for a game grid and cells.

```
#include <iostream>
```

Macros

- #define MAX_ROWS 20

  Defines max rows of game grid.
- #define MAX_COLS 20

  Defines max columns of game grid.

Enumerations

- enum CellT { **Dead**, **Alive** }

  Describes state of cell on game grid.

4.3.1 Detailed Description

Provides type definition and enumerations for a game grid and cells.

Author

Harsh Patel

4.4 include/ReadWrite.h File Reference

Reading text file to build game grid and state, and outputting current state of game grid to text fie.

```
#include <iostream>
#include <vector>
#include <string>
#include "GameboardTypes.h"
```

Functions

- std::vector< char > readFile (std::string f)

  Reading game grid and cells from text file.
- void writeFile (std::string f, std::vector< std::vector< CellT >> s)

  Output game grid and cells to text file.

4.4.1 Detailed Description

Reading text file to build game grid and state, and outputting current state of game grid to text fie.

Author

Harsh Patel

4.4.2 Function Documentation

4.4.2.1 readFile()

```
std::vector<char> readFile (
            std::string f )
```

Reading game grid and cells from text file.

Parameters

f	File name to read.

Exceptions

Runtime_error	If file not found.

Returns

Vector Representing state of cell at each column and row.

4.4.2.2 writeFile()

```
void writeFile (
            std::string f,
            std::vector< std::vector< CellT >> s )
```

Output game grid and cells to text file.

Parameters

f	File name to write to.
s	Grid to be outputted to the text file.

Exceptions

Runtime_error	If file not found.