

Assignment 4, Part 1, Specification

SFWR ENG 2ME4

April 13, 2019

This Module Interface Specification (MIS) document contains modules, types and methods for implementing the state of a game of Life.

GameBoard Module

Module

GameBoard

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

GameBoard

Exported Access Programs

Routine name	In	Out	Exceptions
new GameBoard	string		badbit, failbit, invalid_argument
get		Seq of Seq of bool	
size_c		\mathbb{N}	
size_r		\mathbb{N}	
update			
clear			

Semantics

State Variables

raw: seq of bool board: seq of raw

State Invariant

None

Assumptions & Design Decisions

- The GameBoard constructor is called before any other access routine is called for that object. The constructor can only be called once. The constructor reads configuration from file. File should have format

***##**

##*****

Where # - cell with life, * - without each raw should have the same amount of cells

Access Routine Semantics

new GameBoard(*f*):

- transition: $board := while(line! = EOF), raw.clear() (\forall i \in line.size() : line[i] == \# ? raw.add(true) : raw.add(false)), board.add(raw)$
- output: none
- exception: $exc := (file.open(f) exceptions) || ((line[i]! = \# || line[i]! = *) || \exists i, j \in board.size() : board[i].size()! = board[j].size()) \implies invalid_argument$

get():

- output: $out := board$
- exception: none

size_c():

- output: $out := board.size()$

size_r():

- output: $out := board[0].size()$

update():

- transition: $board := updated_buffer()$

update():

- transition: $board := null$

Local Variables

temp_board : seq of seq of bool

Local Functions

updated_buffer : void \rightarrow seq of seq of bool

updated_buffer() $\equiv \forall i \in \text{board.size}() : \forall j \in \text{board}[i].\text{size}() : \text{temp_board} = \text{update_sell}(i, j)$

update_sell : $\mathbb{N} \times \mathbb{N} \rightarrow \text{bool}$

updated_sell(i, j) \equiv

board[i][j] == true	neighbors_number(i,j) < 2	false
	neighbors_number(i,j) > 3	false
	neighbors_number(i,j) > 1 and neighbors_number(i,j) < 4	false
board[i][j] == false	neighbors_number(i,j) == 3	true
	neighbors_number(i,j) != 3	false

- exception: $i < 0 \parallel i > \text{board.size}() \parallel j < 0 \parallel j > \text{board}[i].\text{size}() \rightarrow \text{out_of_range}$

neighbors_number : $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

neighbors_number(i, j) $\equiv (+\forall c \in i \pm 1 : \forall d \in j \pm 1 : (c \neq i \&\& d \neq j \&\& c < \text{board.size}() \&\& c > 0 \&\& d < \text{board}[c].\text{size}() \&\& d > 0) : \text{if}(\text{board}[c][d]) : 1)$

- exception: $i < 0 \parallel i > \text{board.size}() \parallel j < 0 \parallel j > \text{board}[i].\text{size}() \rightarrow \text{out_of_range}$

Gui Module

Module

Gui

Uses

Gameboard

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
action			

Semantics

State Variables

key: current input string board: GameBoard cmd: terminal

State Invariant

None

Assumptions & Design Decisions

- This is the main control and view Module, the main function "action" that always will take input from console and decide what to do depends on what user input is:
"u" - update and refresh the showing board
"o filename" - output the current state to the file

"start filename" - restart the game with the initial configuration specified in file

"q" - quit without saving

"HELP" - print help message

Initially it'll show the welcome message

Access Routine Semantics

action():

- transition: on first run: print_help_message()
if(key == "start filename"): board.clear(), board = new Gameboard(filename),
cmd.clear(), print_current_board()
else if(key == "u"): board.update(), cmd.clear(), print_current_board()
else if(key == "o filename"): output_to_file(filename)
else if(key == "q"): board.clear(), cmd.clear(), return
else if(key == "HELP"): print_help_message()
else: print_error_message()

Local Variables

Local Functions

cmd.clear : void \rightarrow void

cmd.clear() \equiv function from cstdlib: system("CLS")

print_help_message : void \rightarrow void

print_help_message() \equiv prints welcome message:

please type what you want and press enter:

"u" - update and refresh the showing board

"o filename" - output the current state to the file

"start filename" - restart the game with the initial configuration specified in file

"q" - quit without saving

"HELP" - print help message

print_current_board : void \rightarrow void

print_current_board() \equiv prints: "current board" + endl + board.get() + endl

`output_to_file : string → void`

`output_to_file(f) ≡ file = File.open(f); fstream board.get() to it`

- exception: *exc* := (*file.open(f)exceptions*)

`print_error_message : void → void`

`print_error_message() ≡ print: "try again, you can do it! If you have any questions, you can message me wengyunbing@gmail.com"`