Assignment 4: Design Doc

Madison Ormsby

February 3, 2022

1 Description

Using a c program called `universe.c`, Conway's Game of Life is implemented. Using an input of coordinates or coordinates in a file, live cells are spawned at certain points on the grid. This grid is defined by the player with `fscanf()` to find out how many rows and columns will be utilized in the grid. Also, the grid can be a torus (loops around like a bagel) and be "infinite" or it can be flat like a simple plane. The live cells as mentioned earlier are what controls whether or not the simulation will continue. For a set amount of "generations" retrieved from a `getopt` loop, if a "dead" cell has exactly three live neighbors next to it, it will become a live cell. If a live cell has less than two neighbors or more than three, it will die from loneliness or overcrowding. The grid will be portrayed on screen using `<ncurses.h>` to portray periods for dead cells and "o"s for live cells.

Two universes will be printed out using the dimensions from `fscanf()` (Universe A and Universe B). Using `uv_populate()`, Universe A will be populated with the coordinates from the input file. Using `uv_census()`, a loop will run through all cells and mark them as dead or alive in Universe B, performing one "generation". Finally, the two universes will be swapped to "update" the universe into the new current generation. This will continue until there are no more generations. The screen will then close and output the final Universe A into the specified output.

2 Files Included

universe.c A C program that implements the Universe ADT along with other commands for `universe.h`

universe.h A header file that contains the interface for `universe.c`

life.c The main C file that implements the Game of Life

Makefile A makefile used to build, clean, and format the C programs and header files shown above.

README.md A text file using markdown formatting to briefly explain the program and Makefile along with listing all possible parameters the program will accept.

DESIGN.pdf A pdf file written in LaTeXwith a detailed plan on coding the assignment along with a list of items to be included in the final submission.

3 Pseudocode — Universe.c

Create a structure Universe:
    uint32 rows
    uint32 columns
    boolean grid

bool toroidal

function uvcreate that returns Universe type and takes rows, columns, and toroidal
- allocate memory to a variable of type Universe
- allocate memory to the rows of the grid
- use a for loop to iterate from 0 to rows
  - allocate memory for columns
- set the struct toroidal to toroidal
- return the created Universe

create a function to delete the universe
- use the same for loop in create to free memory in columns using free()
- free the memory from the rows of the grid
- free memory from the variable universe

create a function uvrows to return the rows of the grid
- return the rows in the universe

create a function uvcols to return the columns of the grid
- return the columns in the universe

create a function uvlivecell taking args for a row and column
- set the cell located at column and row to true

create a function uvdeadcell taking args for a row and column
- set the cell located at the coords to false

create a function uvgetcell taking args for a row and column
- return the value of the cell located at row and column

create a function uvpopulate taking a universe and a file
- use a for loop to loop through the lines of the file using wordcount
  - if column of cell is in the range of 0 to uvcols:
    - if row of cell is in the range of 0 to uvrows:
      - use uvlivecell to set each cell at the coords in the file to live
- else return false
- return true if universe was populated

create a function uvcensus taking args of coordinates
- uint32 count
- if universe is NOT toroidal:
  - use a for loop to iterate through VALID neighbors
  - for every neighbor that is alive, add one to count
- else:
  - use a for loop to iterate through the neighbors
  - not entirely sure double check Euegenes section from friday!!
- return the amount of neighbors that are alive

create a function uvprint taking a universe and an output file
- use a loop to iterate through the rows:

```
        for each column in columns:
            if uvgetcell(row, column) == false
                fprintf(outfile, ".");
            else
                fprintf(outfile, "o");
        fprintf(outfile, "newline")
```

4 Pseudocode — Life.c

```
in the main function
    use a get opt loop to cycle through args "tsn:i:o:"
        in case t: set toroidal to true;
        in case s: set silence to true;
        in case n: set generations to the optarg
        in case i: fopen the optarg with read
        in case o: fopen the optarg with write
    fscan in the coordinates for the num of rows and columns in the input file
    create two universes, a and b and use uvcreate on them
    create a temp universe but dont fill it
    populate universe A with the infile
    if silence is false, initialize the ncurses
    for every generation:
        for every row in the universe:
            for every column in the universe:
                if a live cell has 2 or 3 neighbors
                    set the cell to live
                if a dead cell has 3 live neighbors:
                    set the cell to live
                else the cell is dead
        refresh the screen
        let it sleep for 50,000 miliseconds
        swap universe A and B
        clear the screen
    close the window
    print out the final array (universe b)
    close both filein and fileout
    finally delete the universes
```