Assignment 4 Design

Caitlin Smith

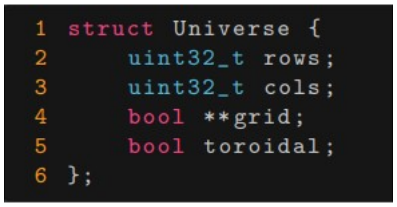
February 12, 2023

Program Description

In this assignment, an ADT called Universe is created in order to recreate the Game of Life. The universe.c file contains the implementation of the necessary functions to run the game. There is a program to then test and display the process using ncurses. It utilizes command-line arguments in the main function in order to set specifications. Input is read from an infile and the results are printed to an outfile.

1 universe.c

Purpose: This contains the struct definition and functions used to manipulate and get information from the Universe ADT. Below is the given struct definition.



```
struct Universe {
    uint32_t rows;
    uint32_t cols;
    bool **grid;
    bool toroidal;
};
```

Pseudocode:

```
Include standard .h files and universe.h.

Following the given code, define the Universe struct.

Create a function Universe *uv_create(uint32_t rows, uint32_t cols, bool toroidal) that create a Universe.
Allocate memory for a Universe pointer.
Set the Universe variables to the passed parameters.
Allocate memory for the grid double pointer using a for loop.
```

Create a function void uv_delete(Universe *u) that destroys a Universe.
Free all of the memory previously allocated to the Universe.
Set the pointer to null.

Create a function uint32_t uv_rows(Universe *u) that returns the number of rows.
Return u->rows.

Create a function uint32_t uv_cols(Universe *u) that returns the number of columns.
Return u->cols.

Create a function void uv_live_cell(Universe *u, uint32_t r, uint32_t c) that makes the specified cell live.
If the cell is within the Universe grid, set that cell to true. Use grid[r][c] because grid is a double pointer.

Create a function void uv_dead_cell(Universe *u, uint32_t r, uint32_t c) that makes the specified cell dead.
If the cell is within the Universe grid, set that cell to false. Use grid[r][c] because grid is a double pointer.

Create a function bool uv_get_cell(Universe *u, uint32_t r, uint32_t c) that returns the value of the specified cell.
If the cell is within the Universe grid, return the value of it.
If it is not in bounds, return false.

Create a function bool uv_populate(Universe *u, FILE *infile) that populates live cells into the Universe from the data in the infile.

```
Read line one to find the total number of rows and columns.
Using a while loop, read in the next lines using fscanf().
    If the cell is outside the bounds, return false.
    If not, set the specified cell to live.
Return true when the process has been completed successfully.
```

Create a function uint32_t uv_census(Universe *u, uint32_t r, uint32_t c) that returns the number of live adjacent cells to the one specified.

```
Set a uint32_t variable count to 0.
Calculate the location of adjacent cells. Account for toroidal moves. If it is not toroidal, don't consider out of bounds neighbors.
    If the neighbor is live, increase the count.
Return the count.
```

```
Create a function void uv_print(Universe *u, FILE *outfile) that prints a visual representation of the grid the specified outfile.
Use nested for loops to iterate through all of the rows and columns.
    If the cells is live, print “o” using fprintf().
    If the cell is dead, print “.” using fprintf().
    Print a new line at the end of every row.
```

2 life.c

Purpose: This file contains the main function. It utilizes command-line arguments in order to set specifications for running the game. The game is run, displaying the progression using ncurses if it is not silenced. The result is printed to the outfile.
Pseudocode:

```
Include the necessary standard and the universe.h file.
Define OPTIONS to be “tsn:i:o:h” and DELAY to be 50,000.

The following will all be within the main().
    Initialize int opt.
    Create variables for the number of generations, infile and outfile. Also create a toroidal and silence flag. The defaults are used if they are not specified in the command-line.

    Create a while loop with getopt.
    Case t will set the toroidal flag to true.
    Case s will set the silence flag to true.
    Case n will set the generations to the user input. The default is 100.
    Case i will set the infile to the user input. The default is stdin.
    Case o will set the outfile to the user input. The default is stdout.
    Case h will print the help message.
    The default prints the help message.

    Use fscanf() to read the first line of the infile to find find the size of the Universe.
    Create Universes A and B using the found sizes, making sure to set the Universe toroidal variable to true if that flag is set.
    Fill Universe A by calling the populate function.
    Set up the ncurses screen.
    Create a for loop for the number of generations.
        If the silence flag is not set, use ncurses to display Universe A.
        Using nested for loops, call the census function for each cell. According to the set of rules, set and clear the correct cells.
        Swap the two Universes, treating A as the current and B as the next.

    Close the screen with endwin().
    Print the result of A to the outfile by calling the print function.
    Close the files.
    Free any allocated memory by calling the delete function.
    Return 0.
```

3 Makefile

Purpose: This is a file that is used to clean, format, and compile the entire program. Make sure to include ncurses in the flags.