Assignment 1 – Program Design and Testing

Mike Peters

petermi3@oregonstate.edu

10/01/2015

Design

Simple **cin** and **cout** will ask user to select a pattern (blinker, glider, or glider gun). Then ask user to pick a starting location. I will have the user input an X and Y coordinate. After that, I will use a loop to build my array depending on the pattern selected. The loops will help from having to type out every dead or alive cell. At that point the main algorithm can be run. This will check the 8 surrounding cells to determine if the current cell dies, lives, stays the same, or is born. I think a loop will again save a lot of typing.

Main algorithm will keep track of living cells touching it. For any [x][y] cell the loop will test:
[x][y+1], [x][y-1], [x+1][y], [x-1][y], [x+1][y+1], [x+1][y-1], [x-1][y+1], and [x-1][y-1].
For each living cell encountered a count variable will increase by 1. Then manage outcome of each cell according to the Game of Life rules.

Once the new array is made from loops above it will be printed using another loop. Then the main algorithm process is repeated along with the printing of the new array. All of this can continue for however many iterations/generations as we want.

The array size can easily be set to something larger than our 40 x 20 viewing/printing size. Doing so will help make the edges easier. Without making it larger, the algorithm won't work once edges come into play since it won't have a full 8 neighbors to test.

Mike Peters
petermi3@oregonstate.edu
10/04/2015

Testing and Reflections

I think my design was not completely thought out. I ran into several problems I should have seen coming. I ran into problems printing the array and changing it. After some more research I found the solution to be copying the array and changing the 2nd array, then copying it to the main array after it was printed. I also was unable to build the glider and glider gun with loops. I had to manually type these out which was slightly time consuming.

Surprisingly making the array bigger than the display area seemed to eliminate edge problems, as I didn't seem to have many. Overall it probably didn't so much solve the edge problem as it just pushed it outside the viewable area. I did have to limit the available starting coordinates for the gosper gun to make sure it would fit on the viewable grid.

I tested my progress throughout the process. First I made sure I had my print dimensions right by just printing an empty array. From there I moved on to building the blinker. I tested just the initial pattern with a single loop iteration. Then I moved on to adding the algorithm loop and copying of arrays. I tested this with preset x,y coordinates. Once that was working the rest was fairly easy. I had to type out the glider, and glider gun patterns. I was then able to add them in conjunction with the algorithm to make them work.

Some troubles I had to seek outside help for, were the system("clear") and usleep() commands. Both were brought up on the discussion board and were easy to implement once I knew the #include needed and the actual command. During some research I did notice that quite a few people said it was bad habit or "evil" to use system commands. I am not sure why. I ended up adding another input allowing the user to pick how generations they wanted the simulation to run for.

Sources:

www.cplusplus.com

our discussion board (many different individuals posted useful information).

http://code.runnable.com/ (there were several GoL codes on there) it was really helpful to be able to both see and run the code simutaniously.

https://en.wikipedia.org/wiki/Conway's_Game_of_Life