Assignment Game of Life

CSIS-2530

Learning Objectives:

- Increase familiarity with Visual Studio and basic C# syntax
- Practice the use of two-dimensional arrays
- Position the cursor and add color to your console application
- Practice solving a problem by breaking it up into smaller tasks
- User private methods to improve the clarity and readability of your code

Description:

Create a console application that implements a simple cellular automaton called the *Game of Life*.

You start out with a grid of cells. Each cell can either be alive or dead. Depending on the number of neighbors, cells can get too lonely or overcrowded and die. Under perfect conditions new life emerges.

Here are the rules:

- Each living cell with 0 or 1 neighbors dies of loneliness
- Each living cell with 2 or 3 neighbors survives
- Each living cell with 4 or more neighbors gets overcrowded and dies
- Dead cells remain dead. However, under perfect conditions with exactly 3 neighbors a dead cell's place on the grid will be repopulated with a new living cell

It is important not to modify the grid before all cells of a given generation are updated. For that reason you will need a second grid, where you can store the new (updated) status of the cells.

One question that will arise is the following:
How should the number of neighbors be calculated if the cell is located on the border of the grid. There are a number of possible solutions. For this assignment we assume that the grid is like an oasis and everything outside the boundaries of the grid is dead.

Requirements:

- Implement the grid as a 2-dimensional int array with 25 rows and 40 columns.
- At the beginning all cells are dead except for a continuous horizontal block of 13 living cells in the 14th row (row index 13, column index 14 – 26 inclusive) and for 3 short vertical blocks (column index 12, 20, and 28, row index 8 - 10 inclusive)
  Tip: It might be easier to read and write if you only declare the grid as a field but initialize it with a private method from the constructor. This method would create a new 2-dimensional array of the appropriate size with the default initialization 0 (dead), set the cells specified above to 1 (living), and use it to initialize the field (i.e. assign it to the grid)
- Display the initial cells until the user presses enter
  Display a message below the grid that informs the user that he needs to press enter to start the game
- Use different colors for living cells and dead cells
  You can choose whether you want to use colors for foreground, background, or both. Just make sure that living cells are easily distinguishable from dead cells
- Update the cells in their place
  Position the cursor to draw at the right location

- Allow the user to stop the game by pressing a key.
  Change the original message to informs the user about this option
- Set the height and width of the Console window so that your grid is approximately centered
  Tip: the class Console has a static method called SetWindowSize
- A main part of this assignment is figuring out how to update the individual cells. Break up this task into sub-tasks. Use private methods with clearly defined responsibilities and descriptive names to perform these sub-tasks.

Extra Challenge (optional)

How can you make the program fast?

If you optimize your application for speed, write a comment on top and explain what you did to improve performance.

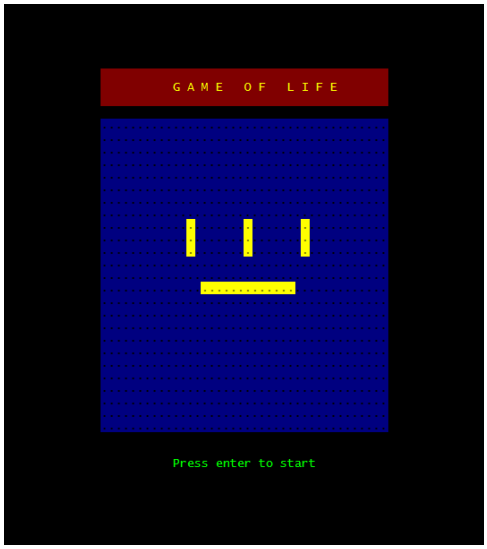
We can run and compare the fastest submissions in class

How can I verify that the cells are updated correctly?

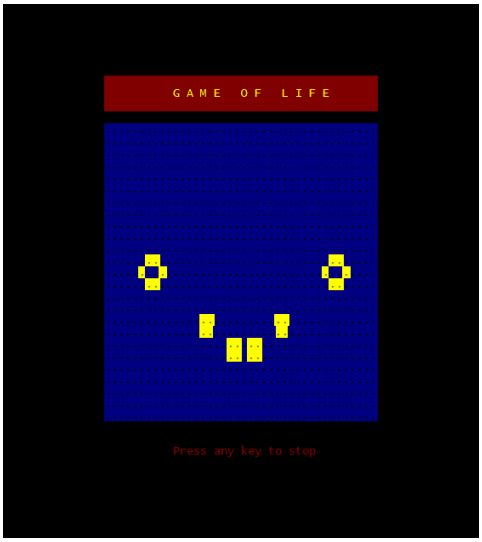
I include an exe file of a slow motion version. It displays each generation for 2 seconds. That should be long enough to compare.

Also: after a while the cells will reach a stable state that no longer changes (end position).

Start Position:



End Position:



Turning in:

Make sure to include a comment with your name on top of each source code file.

Zip up your solution (including the solution file) and turn it in via Canvas