|  |
| --- |
| Computer Programming Language |

【Fall, 2015】

Homework 5

**Program A： Game of life (50%)**

The game of life is a computer simulation that was created by a Cambridge Mathematician named John Conway. The idea is that in each generation life will populate, survive, or die based on certain rules. Read the Wikipedia article ( <http://en.wikipedia.org/wiki/Conway's_Game_of_Life> ) to learn more about this famous simulation.

The universe of the Game of Life is a two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, alive or dead. Every cell interacts with its eight neighbors, the immediately adjacent cells in orthogonal and diagonal direction. Cells on the border have less than eight neighbors. At each generation, the following transitions occur:

1. Any live cell with fewer than two live neighbors dies, as if by loneliness.

2. Any live cell with more than three live neighbors dies, as if by overcrowding.

3. Any live cell with two or three live neighbors lives, unchanged, to the next generation.

4. Any dead cell with exactly three live neighbors comes to life.

The following figures show the grid examples for the first generation, second generation, and third generation, respectively.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  | 1 |  | 1 |  |  |
|  |  |  | 1 |  |  |  |
|  |  |  | 1 |  |  |  |
|  |  | 1 |  | 1 |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

First generation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  | 1 |  |  |  |
|  |  | 1 | 1 | 1 |  |  |
|  |  | 1 | 1 | 1 |  |  |
|  |  |  | 1 |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Second generation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  | 1 | 1 | 1 |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  | 1 | 1 | 1 |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Third generation

Your task is to write a program to simulate the Game of Life for a 20x20 world. The 20x20 world needs to be initialized by reading in which cells are occupied from the user. Your program will generate and display the next generation iteratively based on the set of rules described above. The number of generation for the simulation is also input by the user.

**Bonus Points (10%~30%):**

You may obtain bonus points by implementing a better display of simulation results using the Console class for the standard input, output, and error streams for console applications. You may check the MSDN website for more detailed description of the usage of Console class (<http://msdn.microsoft.com/en-us/library/system.console(v=vs.110).aspx>).

An example program ConsoleOutputExample.cpp is also provided in the CEIBA course website for your reference on the commands you need for the control of screen output. The major commands you may use are as follows:

Console::SetCursorPosition(x, y); // Sets the position of the cursor

Console::Write(System::Char); // Writes the Unicode character to the screen

Console::Clear(); // Clear the console screen

Note: 1. You need to create a “CLR Console Application” project (not an Win32 Console Application) for your program in order to use the Console class.

2. You need to add the “using namespace System;” before main().

**Program B： Magic square (50%)**

A magic square is a square of numbers with *N* rows and *N* columns, in which each integer value from 1 to (*N*×*N*) appears exactly once, and the sum of each column, each row, and each diagonal is the same value. For example, the following figure shows a magic square in which *N* = 3, and the sum of the rows, columns, and diagonal is 15. Write a program that constructs and displays a magic square for any given odd number N. This is the algorithm:

*Insert the value 1 in the middle of the first row (element [0][N%2]).*

*After a value, x, has been placed, move up one row and to the right one column.*

*Place the next number, x+1, there, unless:*

1. *You move off the top (row = -1) in any column. Then move to the bottom row and place the next number, x+1, in the bottom row of that column.*
2. *You move off the right end (column = N) of a row. Then place the next number, x+1, in the first column of that row.*
3. *You move to a position that is already filled or out of the upper-right corner. Then place the next number, x+1, immediately below x.*

*Stop when you have placed as many elements as there are in the array.*

Column

Row

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | 8 | 1 | 6 |
| 1 | 3 | 5 | 7 |
| 2 | 4 | 9 | 2 |

A magic square

**Notes:**

1. Please submit your programs (source codes and execution files) to the CEIBA course website before **Dec. 4**. Hand in the hardcopies of your program codes in the class of **Dec. 4 (3:30PM)**.
2. Late submission will have a penalty of 10% discount per day of your grade toward a minimum score of 60. No late submission over a week will be accepted.
3. Criteria of grading include: (1) Program functionality; (2). User interface; (3). Structure of the program; (4). Suitable comments; (5). Programming style; (6). Creativity.