Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

Grado en Ingeniería Informática
Algoritmos y Estructuras de Datos Avanzadas

Curso 2019-2020

Práctica 2: El juego de la Vida

1. Objetivo

En esta práctica se implementan y usan tipos de datos definidos por el usuario en lenguaje C++.

2. Entrega

Se realizará en dos sesiones de laboratorio en las siguientes fechas:

Sesión tutorada: del 2 al 5 de marzo de 2020
Sesión de entrega: del 9 al 12 de marzo de 2020

3. Enunciado

El juego de la vida [1] es un ejemplo de autómata celular diseñado por el matemático británico John Horton Conway en 1970. Un autómata celular es un modelo matemático que modela un sistema dinámico que evoluciona en pasos discretos.

El juego de la vida es un juego de cero jugadores, lo que quiere decir que su evolución está determinada por el estado inicial y no necesita ninguna entrada de datos posterior. El **tablero** de juego es una malla formada por **células** que se extiende hasta el infinito en todas las direcciones.

Cada célula tiene 8 células vecinas, que son las que están más próximas a ella en el tablero. Una célula tiene un estado con dos valores posibles: “viva" o "muerta".

El estado de todas las células del tablero evolucionará en unidades de tiempo discretas (**turnos**). En un turno cada célula del tablero actualizará su estado en función de su valor de estado y del estado de sus 8 células vecinas en el turno anterior. El cambio de estado dependerá del número de células vecinas vivas según las siguientes **reglas de transición**:

- Una célula “muerta” con exactamente 3 células vecinas “viva” pasa al estado “viva” en el siguiente turno. En cualquier otro caso permanece “muerta”
- Una célula “viva” con 2 ó 3 células vecinas “viva” continúa “viva” en el siguiente turno. En cualquier otro caso pasa al estado “muerta”.

Durante las sesiones de laboratorio se podrán proponer modificaciones y mejoras en el enunciado.

4. Notas de implementación

Se distinguen dos tipos de objetos:

1. El objeto `Celula` que representa la presencia o ausencia de vida, mediante su atributo estado con los valores: “viva” o “muerta”, en una posición del tablero.
    a. El estado es un atributo privado de la célula al que se accede mediante los métodos:

```
int getEstado() const
int setEstado(int)
```

    b. La posición `(i,j)` que ocupa una célula en el tablero también es privada.

c. Cada célula es responsable de actualizar su estado, cuando le corresponda en el turno actual, siguiendo para ello las reglas de transición que definen el juego. Para ello implementa un método público:

```
int actualizarEstado()
```

d. Puesto que las reglas de transición se definen en función del número de células vecinas vivas, cada célula debe interactuar con sus 8 células vecinas para consultarles sus estados. Para ello dispone de un método público que recibe como parámetro el tablero:

```
int contarVecinas(const Tablero&)
```

e. Las 8 células vecinas de la célula `(i,j)` son las que ocupan las posiciones:

```
(i-1,j-1)|(i-1,j)|(i-1,j+1)
-------------------------
  (i,j-1)|  (i,j)|  (i,j+1)
-------------------------
(i+1,j-1)|(i+1,j)|(i+1,j+1)
```

f. Una célula es responsable de su visualización en pantalla, utilizando el carácter `'X'` para representar el estado "viva", y el carácter blanco `' '` para el estado "muerta". Para ello se sobrecarga el operador de inserción en flujo.

2. El objeto `Tablero` que contiene la malla de `NxM` células. Este objeto es responsable de crear y almacenar las células que constituyen el juego y de establecer su valor de estado inicial. También es responsable de contar las unidades de tiempo (turnos), de forma que en cada turno todas las células actualizan su estado.

   a. Las dimensiones de la malla se solicitan al usuario en tiempo de ejecución, y se reserva en memoria dinámica el espacio necesario para almacenar el array de punteros a células.

   b. Cada célula de la malla se crea en memoria dinámica. Se inicializa con la posición `(i,j)` que le corresponda en la malla y el valor de su estado en turno 0.

   c. Para garantizar que la actualización de las células en cada turno tiene en cuenta los valores del turno anterior, el tablero recorre dos veces la malla de células. En el primer recorrido cada célula cuenta sus vecinas; y en el segundo recorrido cada célula actualiza su estado.

   d. En cada turno se realiza un tercer recorrido de la malla para que cada célula se muestra en pantalla de la siguiente forma:

Turno 0:

```
XX  XX
XX  X X XXX
      X
```

e. Hay que tener en cuenta que las células del borde del tablero no cumplen la condición de tener 8 células vecinas. Para evitar la complicación en el cómputo de la transición el tablero creará una malla con las dimensiones `(N+2)x(M+2)` pero sólo se actualizan y visualizan las `NxM` células del interior de la malla, que cumplen la condición de tener las 8 células vecinas.

3. El funcionamiento del programa principal es el siguiente:

   a. Solicita las dimensiones de la malla, N y M, y el número máximo de turnos que tendrá el juego.

   b. Crea un objeto `Tablero` con todas sus células en estado “muerta”.

   c. Solicita las posiciones de las células que deben estar en estado “viva” en el turno 0 y actualiza el estado en dichas células del tablero.

   d. Muestra por pantalla el estado de la malla del tablero.

   e. Ejecuta un bucle donde cada iteración se corresponde a un turno en la evolución de juego. En cada turno se actualiza y se muestra por pantalla el estado de la malla del tablero.

5. Referencias

[1] Wikipedia: https://es.wikipedia.org/wiki/Juego_de_la_vida