Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

Grado en Ingeniería Informática
Algoritmos y Estructuras de Datos Avanzadas

Curso 2021-2022

Práctica 3: La vida en la frontera

---

1. Objetivo

En esta práctica se trabaja el polimorfismo dinámico en lenguaje C++.

2. Entrega

Se realizará en una sesión de entrega en el laboratorio entre el 21 y el 25 de marzo de 2022.

3. Enunciado

En esta práctica retornamos al Juego de la Vida [3] de Conway realizando una versión basada en la implementación del estado generalizado desarrollado en la práctica anterior [1]. Además se quiere modificar la implementación de la rejilla del autómata celular [2] para simular distintas **condiciones de frontera**.

En las implementaciones realizadas en las prácticas anteriores el tratamiento de las células que se encuentran en el borde de la rejilla ha simulado la condiciones de frontera que se denominan **frontera abierta** [2]. Además se ha añadido un "marco" de células en estado "muerta", que no se actualizan para simular una **frontera fría** [2].

Se pide implementar nuevas versiones de la rejilla que simulan otras condiciones de frontera [2], como son:

- **Frontera periódica**. En una rejilla bidimensional se considera que las células del borde inferior son adyacentes con las células del borde superior, y viceversa. Y las células del borde izquierdo son adyacentes a las células del borde derecho, y viceversa. Así por ejemplo, en una rejilla de tamaño 5x7, las células vecinas (vecindad de Moore) a la célula en la posición (0,0) son:

```
(4,6)|(4,0)|(4,1)
-------------------------
(0,6)|(0,0)|(0,1)|...|(0,6)
-------------------------
(1,6)|(1,0)|(1,1)|...|(1,6)
     ---------------------
     | ... | ... |...|...
     ---------------------
     |(4,0)|(4,1)|...|(4,6)
```

- **Frontera reflectora**. Se considera que las células fuera de la rejilla reflejan los valores de las células dentro de la rejilla. Así por ejemplo, en una rejilla bidimensional las células vecinas (vecindad de Moore) a la célula en la posición (0,0) son:

```
(0,0)|(0,0)|(0,1)
-------------------------
(0,0)|(0,0)|(0,1)|...|(0,6)
-------------------------
(1,0)|(1,0)|(1,1)|...|(1,6)
     ---------------------
     | ... | ... |...|...
```

- **Sin fronteras**. Se utiliza una estructura de datos que pueda crecer dinámicamente para aumentar el tamaño de la rejilla según se requiera, esto es, cuando las células de la rejilla deben interactuar con las células fuera de la rejilla.

4. Notas de implementación

1. A partir de la clase base abstracta `State` se derivan las clases que representan los siguientes estados concretos: `StateAlive` y `StateDead`.

    a. Una célula que se encuentra en estado `StateDead` se corresponde con una célula "muerta" de la versión clásica. Esto es, la vecindad incluye a las ocho vecinas más próximas (vecindad de Moore) y la célula transita a estado `StateAlive` cuando en su vecindad existan exactamente 3 células vivas. En cualquier otro caso permanece en estado `StateDead`.

    b. Una célula que se encuentra en estado `StateAlive` se corresponde con una célula "viva" de la versión clásica. Esto es, la vecindad incluye a las ocho vecinas más próximas (vecindad de Moore) y la célula permanece en el mismo estado cuando en su vecindad existan 2 o 3 células en estado `StateAlive`. En cualquier otro caso la célula transita a estado `StateDead`.

2. Para permitir que el programa puede trabajar con cualquiera de las versiones de la rejilla se define una jerarquía de clases rejilla, siendo `Grid`, la clase base abstracta donde sólo se definen los siguientes métodos:

    a. La rejilla da acceso a cualquiera de las células contenidas, y también da acceso a las células fuera del borde aplicando el tratamiento de las condiciones de frontera que corresponda.

    ```
    virtual Cell& Grid::getCell(int, int) =0;
    virtual const Cell& Grid::getCell(int, int) const =0;
    ```

    b. La rejilla es responsable de contar las unidades de tiempo, **turnos**, de forma que en cada turno todas las células actualizan su estado.

    ```
    void Grid::nextGeneration();
    ```

    c. Destructor virtual que permita a cada clase derivada liberar adecuadamente la memoria que utilicen.

    ```
    virtual Grid::~Grid() {};
    ```

3. Sobrecargar el operador de inserción en flujo para visualizar en pantalla una rejilla.

4. La responsabilidad de inicializar las células vivas en el turno 0 se pasa al programa principal, de forma que las clases rejilla no dependen de los estados de la célula.

5. Se renombra a `GridWithOpenBorder` la implementación de la clase `Grid` utilizada en las prácticas anteriores; y se convierte en un caso particular de implementación de rejilla con frontera abierta haciendo que derive de la nueva clase abstracta `Grid`.

6. A partir de la clase base `Grid` se derivan las clases: `GridWithPeriodicBorder`, que implementa una rejilla con las condiciones de frontera periódica; y la clase `GridWithReflectiveBorder`, que implementa las condiciones de frontera reflectora.

7. De forma opcional, se implementará una rejilla con las condiciones sin frontera.
8. El funcionamiento del programa principal es el siguiente:
    a. Solicita al usuario el tipo de rejilla y el tamaño inicial de la misma.
    b. Inicia un puntero a la clase base `Grid` con la dirección del objeto rejilla elegido que se crea en memoria dinámica con todas sus células en estado `StateDead`.
    c. Solicita al usuario las posiciones de las células vivas en el turno 0, y actualiza el estado en dichas células a `StateAlive`.
    d. Muestra por pantalla la rejilla en el turno 0.
    e. Ejecuta un bucle mientras el usuario no detenga la ejecución, y en cada iteración se invoca al método `Grid::nextGeneration()` para avanzar un turno en el juego, y se visualiza en pantalla la rejilla.
9. Durante las sesiones de laboratorio se podrán solicitar cambios y/o ampliaciones en la especificación de este enunciado.

5. Referencias

[1] Moodle: Enunciado de la Práctica 2.
[2] Wikipedia: Autómata celular. https://es.m.wikipedia.org/wiki/Autómata_celular
[3] Wikipedia: Juego de la vida. https://es.m.wikipedia.org/wiki/Juego_de_la_vida