El Juego de la Vida

El Juego de la Vida de Conway es una simulación de un universo de células que interactúan entre sí con reglas bien definidas entre esta y su vecindad y donde el universo puede evolucionar en periodos de tiempo discretos, llamados ticks. Este modelo, conocido como autómata celular, fue propuesto por el matemático inglés John Horton Conway en 1970. Puede leer más y ver una simulación en Wikipedia (https://es.wikipedia.org/wiki/Juego_de_la_vida).

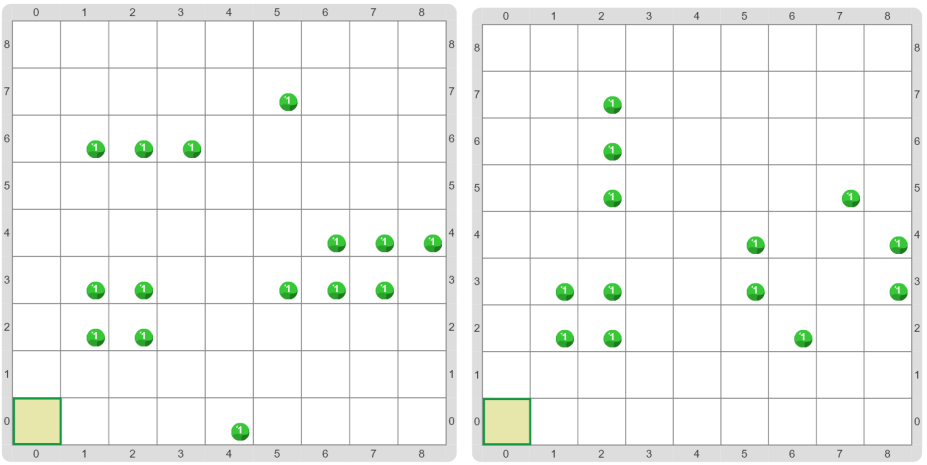
La simulación se desarrolla en una grilla rectangular cuadriculada (igual que el tablero de Gobstones) y en donde en cada celda hay una célula, que puede estar o bien viva o bien muerta. Las células (vivas o muertas) interactúan con las células de su vecindad (Todas las que se encuentran alrededor de la célula en las 4 direcciones ortogonales (en línea recta) y en las 4 direcciones diagonales. Es decir que una célula puede tener hasta 8 células vecinas (En direcciones N, NE, E, SE, S, SO, O, NO), aunque puede tener menos sí se encuentra en alguno de los bordes de la grilla. Al estado de la grilla con sus células se lo denomina, universo.

Como parte de la interacción con sus vecinas, una célula viva puede morir, o una muerta revivir. Las pautas del juego que determinan la interacción son las siguientes:

- Una célula viva con menos de dos vecinas vivas, muere (escasez de población).
- Una célula viva con más de tres vecinas vivas, muere (sobrepoblación).
- Una célula muerta con exactamente tres vecinas vivas, revive (reanimación).
- El resto de las células no sufren cambios de una generación a otra.

Esta actualización del estado de las células ocurre simultáneamente para todas las células del universo al mismo tiempo, dando lugar a un nuevo universo. La aplicación de estas pautas a todas las células del universo para dar lugar a uno nuevo se denomina “tick”.

Podemos modelar en Gobstones el Juego de la Vida con la siguiente representación. Cada celda del tablero representará una célula del universo. Aquellas celdas que contengan una bolita de color Verde serán células vivas, las que no contengan una bolita de dicho color son células muertas.



Se muestran en la imágen de arriba dos tableros. El de la izquierda es un posible tablero inicial (aunque el tamaño del tablero y las células que comienzan vivas podrían ser cualquiera). El tablero de la derecha es el resultado de aplicar un tick sobre el universo del tablero de la izquierda. Notar por ej. como la célula en 4,0 murió por escasez de población, o como la célula en 2,5 revivió.

Cuando se modela en Gobstones este problema hay que tener en cuenta que no se puede actualizar el estado de una célula de forma aislada, sí lo que se desea de fondo es actualizar la totalidad del universo, pues, de eliminar una célula al procesarla, e ir a procesar su vecina, al contar las vecinas vivas de esta, ahora habrá una menos de las que debería. Para evitar este inconveniente, se propone utilizar dos pasadas para procesar el universo. En la primera pasada se marcarán algunas células para su eliminación o reanimación, y en una segunda pasada, efectivamente se eliminarán o reanimaran dichas células. Se sugiere entonces bolitas rojas para las marcas de eliminación, y azules para las de reanimación.

Actividades (Castillo A. Programas, Contratos y Procedimientos):

Ejercicio 1)
Realice el procedimiento **`MorirCélula`** que deja la célula viva actual muerta. Puede asumir que hay una célula viva en la celda actual.

Ejercicio 2)
Realice el procedimiento **`PonerMarcaParaReanimación`** que marca para reanimar la célula muerta actual.

Ejercicio 3)
Escriba el procedimiento **ProcesarMarcaDeEliminación** que procesa una célula viva que fue marcada para ser eliminada, dejando la célula muerta y eliminando la marca de eliminación.

Actividades (Castillo B. Repeticiones Simples, Parámetros, Expresiones y Tipos):

Ejercicio 4)
Escriba el procedimiento **`Simular_Ticks`** que dado un número de ticks, simula dicha cantidad de ticks en el universo.

Para solucionar este problema se puede hacer uso de la siguiente primitiva:

```
SimularUnTick
PROPÓSITO: Simula un tick en el universo, aplicando las pautas del juego
a todas las células, y dejando el universo en la nueva generación.
PRECONDICIONES: Ninguna
```

Ejercicio 5)
Escribir el procedimiento **`QuitarMarca_`** que dado un color de marca, elimina la marca de dicho color en la célula actual. sí existe. Luego, utilice dicho procedimiento para programar **`QuitarMarcas`** que quita tanto las marcas de reanimación como de eliminación de la célula actual.

Para programar lo mencionado, se proveen las siguientes primitivas:

```
colorMarcaDeReanimación
PROPÓSITO: Describe el color de las marcas de reanimación.
TIPO: Color
PRECONDICIONES: Ninguna
```

```
colorMarcaDeEliminación
PROPÓSITO: Describe el color de las marcas de eliminación.
TIPO: Color
PRECONDICIONES: Ninguna
```

Actividades (Castillo C. Alternativas Condicionales y Funciones Simples):

Ejercicio 6)
Se pide escriba la función **`esCélulaViva`** que indica si la célula actual está viva.

Ejercicio 7)
Se pide escriba la función **`estáMarcadaParaSerEliminada`** que indica si la célula actual está marcada para ser eliminada.

Ejercicio 8)
Se pide escriba el procedimiento **`ProcesarMarcasDeCélula`** que procesa las marcas que tenga la célula actual, sí las tuviera.

Ejercicio 9)
Se pide escriba el procedimiento **`AplicarPautasDeMarcadoACélula`** que aplica las pautas del juego a la célula actual, marcandola para ser reanimada, eliminada, o no marcandola según corresponda.

Para solucionar el problema puede hacer uso de la función primitiva siguiente:

```
cantidadDeVecinasVivas
PROPÓSITO: Describe la cantidad de células vivas que hay en la vecindad
de la célula actual.
TIPO: Número
PRECONDICIONES: Ninguna
```

Actividades (Castillo D. Repeticiones Condicionales, Variables, Funciones con Procesamiento y Recorridos):

Ejercicio 10)
Se pide desarrolle la función **`hayVecinaVivaAl_`** que indica si existe una célula viva en la dirección dada. Nótese que describe Falso tanto si la célula existente en esa dirección está muerta, como si no hubiera una célula en dicha dirección (por encontrarse en el borde del universo)

Ejercicio 11)
Se pide desarrolle la función **`cantidadDeVecinasVivas`** dada previamente como primitiva, que determina la cantidad de vecinas vivas que tiene la célula actual.

Para resolver este problema puede hacer uso de la función primitiva siguiente:

```
hayVecinaVivaEnDiagonalAl_YAl_
PROPÓSITO: Indica si la célula en diagonal hacia las direcciones dadas
está viva.
TIPO: Booleano
PRECONDICIONES:
    * Las direcciones dadas no son opuestas ni iguales.
```

Ejercicio 12)
Escribir el procedimiento, dado antes como primitiva, **`ProcesarUnTick`** que procesa la totalidad de las células del universo, aplicando las pautas del juego a cada una de ellas, y procesando luego las marcas de reanimación o eliminación que se hayan generado.

Ejercicio 13)
Escribir la función **`cantidadDeCélulasVivasEnGeneración_`** que dado un número que representa una cantidad de ticks a simular, describe la cantidad de células vivas que quedan en el universo tras simular esa cantidad de ticks.