Guía 5: variables.

Ejercicios iniciales con variables

1. Suponiendo que la celda actual refleja el resultado de un partido (bolitas azules: goles de Racing, bolitas rojas: goles de Independiente) hacer la función `quienGano()` que denote Azul si ganó Racing, ó Rojo si ganó Independiente.

2. Escribir las funciones `max(nro1, nro2)` y `min(nro1, nro2)` que denotan, respectivamente, el máximo y el mínimo entre dos números.

3. Escribir la función `diferenciaPositiva(col1,col2)`, que denota la diferencia entre la cantidad de bolitas de los colores indicados en la celda actual.
   P.ej. si hay 5 rojas y 2 verdes, o si hay 2 rojas y 5 verdes, en cualquiera de estos casos, `diferenciaPositiva(Verde,Rojo)` denota 3.

4. Escribir la función `direccionSegunCantDeColor(col)`, que devuelva la dirección a la que debe moverse el cabezal según la tabla del ejercicio 8 de Alternativa de la guía 3. O sea: si hay una bolita de color col denota Este, si hay 2 Norte, si hay 3 Oeste, si hay 4 o más, o no hay ninguna, Sur.

5. Escribir la función `ordenDeMagnitud()`, que si en la celda actual hay más de 10 bolitas denota 1, y si no denota 0.

6. Escribir la función `nroBolitasAlSiHay(dir,col)`, que denote el número de bolitas de color `col` en la celda vecina en dirección `dir`, y si no hay vecina, que denote 0.

7. Escribir la función `nroBolitasEnVecinas(col)`, que denote el número total de bolitas de color col en celdas vecinas a la actual. Usar `nroBolitasAlSiHay`.

8. Escribir la funcion `nroBolitasYoYMisVecinos(col)`, que denote el número total de bolitas de color `col`, considerando la celda actual más sus celdas vecinas. Usando los dos anteriores, sale fácil.

9. Escribir la función `direccionDeAtraccionHorizontal(col)`, que denote Este si hay más bolitas de color col en la vecina Este de la celda actual que en la Oeste, y Oeste en caso contrario. Precondición: la celda actual no está ni en el borde Este ni en el borde Oeste.

10. Escribir el procedimiento `MoverSegunAtraccionHorizontal(col)`, que “mueve” una bolita de color col de la celda actual a su vecina al Este o al Oeste, según en cuál haya más de ese color. Si en la celda actual no hay bolitas de color col, entonces no hace nada. Precondición: la celda actual no está ni en el borde Este ni en el borde Oeste. Consejo: usar `direccionDeAtraccionHorizontal`.

11. Escribir la función `colorDominante()`, que devuelve el color del que hay más bolitas en la celda actual. OJO es posible que se complique bastante sacarlo.

Recorridos con variables

12. Implementar la función `nroBolitasTotal(c)`, que denota la cantidad total de bolitas del color indicado que hay en el tablero.
13. Escribir la función `nroTotalTotalBolitas()`, que denote el número total de bolitas en el tablero. Usar `nroBolitasTotal`, no recorrer de nuevo.
14. Escribir la función `hayAlgunaBolita(col)`, que denote si hay al menos una bolita de color col en alguna celda del tablero.
15. Escribir la función `esColorDominanteGlobal(col1,col2)`, que indica si col1 es dominante sobre col2 (o sea, hay más bolitas de color col1 que de color col2) en cada celda del tablero.
16. Hacer los ejercicios 20, 21 y 22 de la Guía Común 5 (`nroFilas`, `nroColumnas`, `coordenadaX`, `coordenadaY`, `IrACoordenada`).
17. Escribir la función `nroCeldasConColor(col)`, que denota la cantidad de celdas que tienen al menos una bolita del color indicado.
18. Escribir la función `nroCeldasConAlMenos(cant,col)`, que denota la cantidad de celdas que tienen al menos `cant` bolitas del color indicado.

Ejercicios integradores de recorridos y funciones

19. Resolver el ejercicio 7 de la guía común 5. Es decir, la función `hayCeldaCromatica()`.
20. Resolver el ejercicio 12 de la guía común 5. Es decir, el procedimiento `Rellenar()`.

Más sobre apuestas

Retomando el modelo de casa de apuestas que desarrollamos en la guía 4 2da parte, hacer lo siguiente:

1. Escribir la función `cuantosApostaronA(nroApostado)`, que denota la cantidad de apuestas al número indicado que se encuentran en el tablero.
2. Escribir la función `cuantaPlataSeApostoA(nroApostado)`, que denota el monto total apostado al número indicado.

3. Escribir la función `cuantaPlataAposto(nroJugador)`, que denota el monto total que apostó el jugador indicado.
4. Escribir la función `algunJugadorApostoMasDe(nroApostado,minimo)`, que indica si al menos un jugador apostó al número indicado, el importe mínimo indicado o más.
5. Escribir la función `algunJugadorApostoEnTotalMasDe(minimo)`, que indica si al menos un jugador apostó, en total, el importe mínimo indicado o más.
6. Escribir el procedimiento `CancelarApuestasANro(nroApostado)`, que borra todas las apuestas al número indicado.
7. Escribir el procedimiento `DuplicarApuestasANumerosPopulares()`, que duplica todas las apuestas a números a los que se apostó, en total, 20 pesos o más.
8. Escribir la función `montoApuestaMasAlta()`, que denote … exactamente eso.
9. Escribir la función `jugadorQueHizoLaApuestaMasAlta()`, que denote el número del jugador que hizo la apuesta de mayor monto.
10. Escribir la función `jugadorQueApostoMas()`, que denote el número del jugador que apostó más, considerando el total del monto apostado por cada uno.
11. Escribir el procedimiento `RegalarApuestaAlQueApostoMas(nroApostado)`, que agrega una apuesta por 5 pesos al número indicado, para el jugador que registre el mayor monto total de apuestas (usar `jugadorQueApostoMas`, claro).

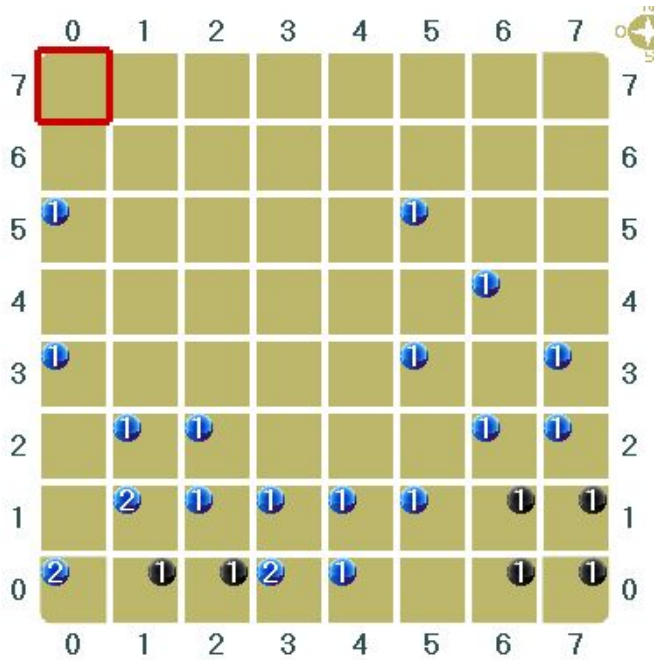
Tetris

Resolver el ejercicio 9 de la guía 4 común, que define un montón de funciones y procedimientos que pueden ayudar para modelar una versión del Tetris.
Vale consultar sobre el enunciado a su docente amigo.

Estalagmitas

En estos ejercicios, usamos el tablero para representar una cueva en la que se forman estalagmitas. Del techo de la cueva caen gotas de agua, que cuando se acumulan en la base, van formando las estalagmitas. De esta forma, las estalagmitas se forman desde el piso de la cueva, y van subiendo.
En este modelo, el techo de la cueva está al norte, cada gota de agua es representada por una bolita azul, y las celdas que tienen estalagmita con una bolita negra.
Cada gota de agua va bajando, hasta llegar o bien al piso de la cueva, o bien a una celda que tiene una estalagmita inmediatamente al sur. Una vez que la gota llega a una celda de la cual no puede seguir bajando, se queda ahí, no se escurre hacia otro lado ni se evapora. Si en una misma celda llega a haber tres gotas de agua, se genera una estalagmita. En el proceso, se consumen las tres gotas de agua.

Entendamos las reglas mirando este tablero.

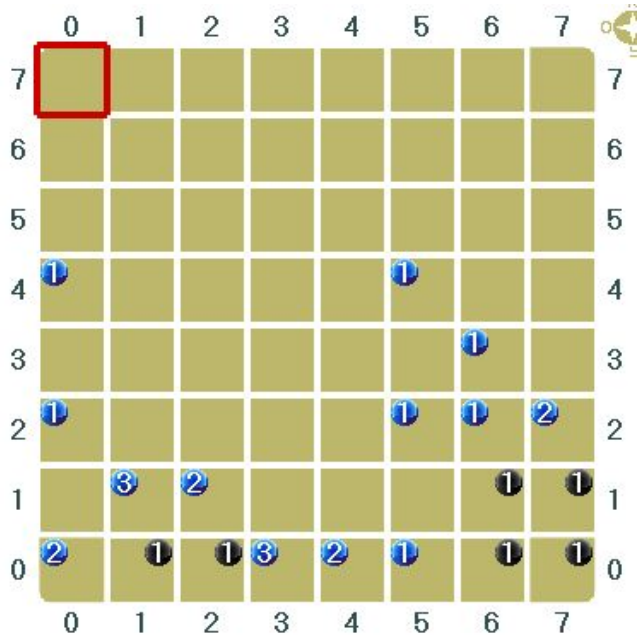


Veamos la situación de algunas de las gotas que aparecen:

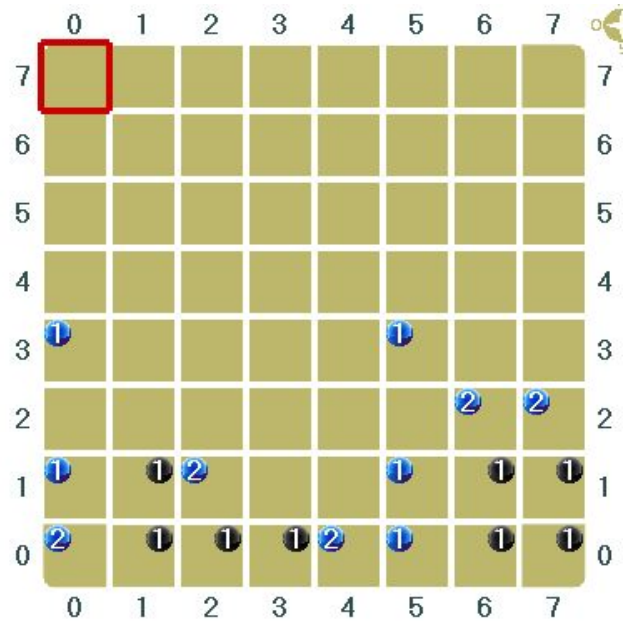
- las gotas de (0,3) y (0,5) bajan, en el siguiente paso, a (0,2) y (0,4) respectivamente.
- las dos gotas que están en (1,1) se quedan ahí, porque abajo tienen una estalagmita (la bolita negra).
- la gota en (1,2) baja a (1,1), juntándose con las dos gotas que hay ahí. Pasa a haber 3 gotas en (1,1). En el siguiente paso, esas tres gotas se transforman en una estalagmita.
- la gota en (2,1) se queda ahí, porque abajo tiene una estalagmita.
- la gota en (2,2) baja a (2,1), juntándose con la gota que hay ahí. Pasa a haber 2 gotas, que quedan ahí.
- las dos gotas en (0,3), y la gota en (0,4), quedan ahí, no tienen a dónde bajar.

- las gotas en (1,3) y (1,4) bajan, juntándose con las anteriores. Las tres gotas que quedan en (0,3) se transforman, en el siguiente paso, en una estalagmita. Las dos gotas que quedan en (0,4) se quedan ahí.
- las gotas en (5,1), (5,3) y (5,5) bajan una celda.
- las gotas en (6,2) y (7,2) se quedan ahí, tienen una estalagmita debajo.
- las gotas en (6,4) y (7,3) bajan, la de (7,3) se junta con la de (7,2) como explicamos antes en otros casos.

Si aplicamos un paso de evolución a todo el tablero, queda:



Mostramos un paso de evolución más, ver en particular que se forman estalagmitas en (1,1) y (3,0):



Para hacer andar una versión básica de este modelo, se pide desarrollar:

1. la función `hayUnaGota()`, que indica si en la celda actual hay exactamente una gota. Si no hay gotas, o hay más de una gota, entonces debe devolver falso.
2. la función `puedeCaerGota()`, que indica si de haber una gota en la celda actual, esta gota puede moverse una celda hacia el sur. Para esto, tiene que haber una celda al sur, y además esta celda no puede tener estalagmita.
3. el procedimiento `CaidaGota()`, que hace caer la gota en la celda actual, o sea, "moverse" una celda hacia el sur. Se supone que en la celda actual hay exactamente una gota, y que esta gota se puede caer.
4. el procedimiento `EvolucionarCelda()`, que
    a. si hay exactamente una gota en la celda actual, y esta gota se puede caer, entonces la hace caer (ver los ítems anteriores)
    b. si se puede generar una estalagmita, se genera, o sea, se "transforman" tres gotas de agua en un estalagmita, para lo cual hay que borrar las gotas de agua y agregar la estalagmita.
    c. en cualquier otro caso, no hace nada.
5. los procedimientos `EmpezarRecorridoTablero()` y `IrAlSiguienteEnRecorridoTablero()`, más la función `haySiguienteEnRecorridoTablero()`, que implementan el recorrido del tablero, en el orden que requiere esta simulación. Seguro hay que ir de sur a norte.
6. el procedimiento `EvolucionarTablero()`, que recorre las celdas en el orden del ítem anterior, y hace evolucionar cada una.

Puede ser interesante armar un programa interactivo para probar EvolucionarTablero(). Podemos hacer que con Enter dé un paso de evolución. También podemos agregar la capacidad de agregar gotas que vayan cayendo desde el techo, dejando el cabezal en la fila más al norte (eso se puede forzar en el tablero inicial), haciendo que se mueva con las flechas izquierda y derecha, y poniendo una tecla para agregar una gota.
Para armar el programa interactivo, desarrollar:

7. la función `celdaVacia()`. Una celda está vacía si no tiene ni agua ni estalagmita.
8. el procedimiento `CrearGota()`, que pone una gota en la celda actual, si está vacía. En caso contrario, que no haga nada.
9. el procedimiento `MoverConTeleport(dir)`, que mueve el cabezal una celda en la dirección indicada. Si no puede por estar al borde, que pase al borde opuesto.

Campeonato de Piedra, Papel ó Tijeras

Supongamos que un casillero representa una mano de piedra, papel ó tijeras. Las bolitas azules representan la jugadande un jugador, y las bolitas negras la jugada del otro jugador. Las jugadas son:

- 1 bolita: Piedra
- 2 bolitas: Papel
- 3 bolitas: Tijeras

Jugando con expresiones y funciones:

1. `leGana(jug1, jug2)`. Las jugadas son números, y leGana denota si la jugada `jug1` le gana a la jugada `jug2`, teniendo en cuenta que:
   a. Piedra le gana a tijeras.
   b. Tijeras le gana a papel.
   c. Papel le gana a piedras
   d. (en cualquier otro caso es False)

   P.ej. `leGana(1,3)` denota `True` porque piedra le gana a tijeras, `leGana(1,2)` denota `False` porque piedra no le gana a papel, y `leGana(1,1)` denota `False` porque es empate, la jugada 1 no le gana a la 2.
2. `oponente(color)`. Denota Azul cuando recibe Negro, y viceversa.
3. `ganoLaMano(color)` denota True si el color indicado es el ganador de la mano.
   Precondición: en el casillero actual están las dos jugadas correspondientes.

   Por ejemplo, esta mano el Azul ganó:  (papel contra piedra).
4. `hayEmpate()` denota si hay empate en la celda actual (o sea si nadie ganó).

El torneo:

5. `cuantasManosGano(color)` denota la cantidad de manos que el color indicado ganó en la fila actual.
   Precondición: En todas las casillas de la fila hay una jugada, y el cabezal está en el borde Oeste de la fila.
6. `ganador()` denota el jugador ganador del torneo.
   La información sobre el torneo está organizada así: cada fila del tablero representa un jugador diferente, que es el que corresponde al color Azul en cada jugada de la fila. El jugador ganador es el que haya ganado más manos. El número de jugador de cada fila está representado por el número de bolitas verdes en el borde Oeste de la fila.
   Por ejemplo, en el tablero siguiente, el jugador ganador es el 2: (porque ganó 3 veces)

7. Más difícil: suponer que no están las bolitas verdes, y resolver de todas formas el punto anterior. Suponer que en la fila 0 está el jugador 1, en la fila 1 está el jugador 2, y así sucesivamente.