

Guía 4 segunda parte

Recorridos y funciones

Ejercicios iniciales sobre recorridos

1. Escribir el procedimiento `CambiarColorTotal(col1, col2)`, que “reemplaza” todas las bolitas de color `col1` del tablero por bolitas de color `col2`.
2. Resolver los ejercicios 5 y 6 de la guía común 5. Esto es, escribir los procedimientos `PonerEnVacias(c)` y `DuplicarBolitas()`.
3. Escribir el procedimiento `BuscarCeldaConAlMenos(cant, col)` que posiciona al cabezal en una celda que tenga al menos `cant` bolitas de color `col`.
P.ej. `BuscarCeldaConAlMenos(3, Verde)` posiciona el cabezal en una celda con 3 ó más bolitas verdes.
Vale suponer que hay al menos una celda que cumpla la condición.
4. Escribir el procedimiento `SacarTodasDelTablero(col)`, que saca todas las bolitas de color `col` del tablero.
5. Escribir el procedimiento `BuscarCeldaConIgualCantidad(col1, col2)` que posiciona al cabezal en una celda que tenga la misma cantidad de bolitas de color `col1` y `col2`, verificando que tiene que haber al menos una de cada.
P.ej. `BuscarCeldaConIgualCantidad(Verde, Azul)` posiciona al cabezal en una celda que tenga la misma cantidad de bolitas verdes y azules (2 verdes y 2 azules, 1 verde y 1 azul, 42 verdes y 42 azules, etc.).
Si no se encuentra ninguna celda que cumpla la condición, dejar el cabezal en el extremo noreste del tablero.
6. Escribir el procedimiento `BuscarCeldaVacía()`, que posiciona al cabezal en una celda sin bolitas. Si no se encuentra ninguna celda vacía, dejar el cabezal en el extremo noreste del tablero.
7. Escribir el procedimiento `MoverVerdesAlOeste()`, que en cada celda que tenga bolitas verdes y una celda vecina al oeste, “mueva” todas las bolitas verdes a la vecina al oeste.
Ayuda: recordar la idea de hacer un procedimiento que realice la acción para cada celda.

Ejercicios iniciales sobre funciones

1. Escribir la función `nroBolitasCelda()`, que denota el total de celdas de la celda actual, contando las de los cuatro colores.
2. Escribir la función `estoyEnUnBorde()`, que denota verdadero si el cabezal está en un borde del tablero. Ayuda: esto es así si no se puede mover en alguna dirección.
3. Escribir la función `rojoEsDominante()`, que denota verdadero si en la celda actual, hay más bolitas rojas que verdes, que negras y que azules.
4. Escribir la función `rojoEsUltraDominante()`, que denota verdadero si en la celda actual, hay más bolitas rojas que la suma de verdes, negras y azules.
5. Escribir la función `haySoloRojo()`, que denota verdadero si en la celda actual hay bolitas rojas, y no hay bolitas de ningún otro color.
6. Escribir la función `hayBolitasAlExistente(dir, col)`, que denota verdadero si en la celda vecina a la actual en la dirección indicada, hay al menos una bolita de color `col`. Vale tomar como precondition que hay una celda vecina en la dirección indicada.
7. Escribir la función `hayBolitasAl(dir, col)`, que denota verdadero si hay una celda vecina a la actual en la dirección indicada, y si en esa celda hay al menos una bolita de color `col`.
8. (difícil) Escribir la función `estoyRodeadoDe(col)`, que denota verdadero si en todos los vecinos de la celda actual hay bolitas del color indicado.
9. Escribir la función `estoyEnFilaDeColor(col)`, que denota verdadero si hay bolitas del color indicado en todas las celdas de la fila actual.


Farmville

En estos ejercicios, usamos el tablero para representar un conjunto de granjas que se dedican a la cría de animales. Cada celda es una granja, cuyos vecinos corresponden a las celdas vecinas en el tablero.


Vamos a usar bolitas para representar los animales en cada granja, una bolita corresponde a un animal, según esta codificación de colores:

- rojo: chancho
- negro: vaca
- azul: oveja



Por ejemplo, esta celda  representa una granja que tiene 5 chanchos, 34 vacas y 8



ovejas, mientras que esta  representa una granja con 6 vacas y 28 ovejas, y sin chanchos.

Implementar los siguientes procedimientos.

1. `MigrarUnaOveja(dir)`: mover una oveja de la granja actual a la vecina en dirección `dir`. Se puede suponer que la granja en la celda actual tiene al menos una oveja, y que hay una vecina en la dirección indicada.
2. `CambiarOvejaPorChancho(dir)`: la granja correspondiente a la celda actual le da una oveja a su vecina en dirección `dir`, obteniendo a cambio un chancho. Se puede suponer que la granja en la celda actual tiene al menos una oveja, que hay una vecina en la dirección indicada, y que esa vecina tiene al menos un chancho.
3. `CambiarOvejaPorChanchoSiPuede(dir)`: lo mismo que antes, pero no se puede suponer nada. Si no hay oveja y/o chancho para intercambiar, o no hay celda vecina en la dirección indicada, no se hace nada.
4. `BuscarGranja()`: pone el cabezal en una celda que represente una granja, o sea, en la que haya al menos un animal. Se puede suponer que en el tablero hay al menos una granja.
5. `CobrarImpuesto()`: en la celda actual, quitar una oveja, si no hay ovejas quitar un chancho, si no hay ni ovejas ni chanchos, quitar una vaca. Suponer que la celda actual es una granja.
6. `CobrarImpuestoTotal()`: cobrar el impuesto del ítem anterior en cada celda que tenga una granja.
7. `ParirHijosDeParejasUnicas()`: en cada celda en que haya una granja, si hay exactamente dos ovejas agregar una nueva, lo mismo con chanchos, lo mismo con vacas.
8. `MigrarUnaOvejaAlSurTotal()`: mover una oveja de la granja actual a la vecina hacia el sur, para cada granja que tenga al menos una oveja y tenga vecina al sur.

9. FomentarActividadPorcina(): en cada granja que tenga 3 chanchos o menos, agregar un chancho más.

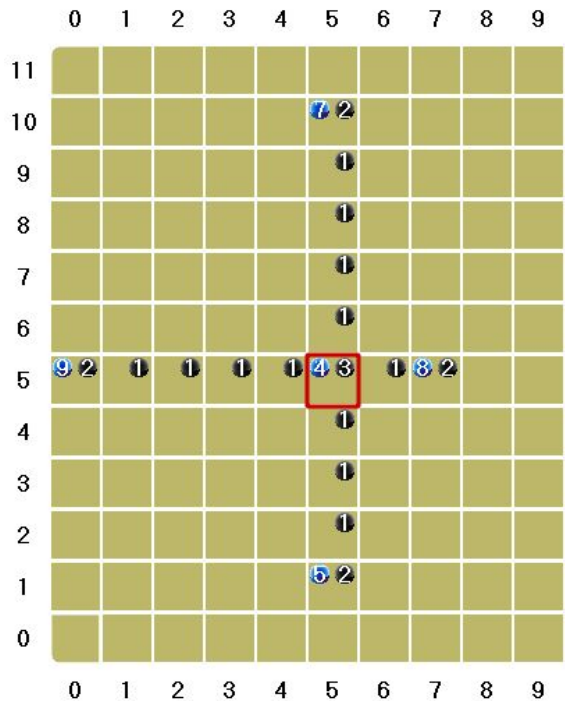
Después, hacer lo siguiente.

10. Implementar la función esGranja(), que indica si la celda actual es una granja. Modificar el procedimiento BuscarGranja para que use esta función.
11. Implementar la función cantidadAnimales(), que devuelve cuántos animales tiene la granja actual. Que devuelva 0 si no tiene ningún animal.
12. Implementar la función hayChanchoAl(dir), que indica si hay o no un chancho en la granja vecina en la dirección indicada. Vale tomar como precondition que hay una granja vecina en esa dirección. Se puede usar esta función para mejorar la implementación de CambiarOvejaPorChanchoSiPuede.
13. Implementar la función tengoVecinoGrosoAl(dir), que es verdadera si hay una granja vecina en la dirección indicada, y esa granja tiene más de 8 animales en total. Si no tengo vecino en la dirección indicada, o si tengo vecino pero no es groso, entonces esta función es falsa.
14. Implementar la función estoyMuyBienRodeado(), que es verdadera si tengo vecinos grosos en las cuatro direcciones.

Distribución de mercadería

Queremos modelar el movimiento de mercadería en una sencilla red de depósitos, que tiene un depósito central, más un depósito local para cada punto cardinal.

Para esto, vamos a dibujar un mapa muy simplificado en el tablero. Tres bolitas negras marcan el depósito central, dos bolitas negras marcan un depósito local, una bolita negra marca el camino de central a local, cada bolita azul marca una unidad de mercadería. Los depósitos locales forman una cruz, donde el centro es el depósito central. No se sabe a qué distancia están los depósitos locales del depósito central. Este es un ejemplo de modelo:



Se pide escribir

1. Las funciones `esDepositoCentral()` y `esDepositoLocal()` que indican si el cabezal está, respectivamente, en el depósito central o en un depósito local.
2. El procedimiento `IrDeCentralALocal(dir)`, que mueve el cabezal del depósito central al depósito local en dirección `dir`.
3. El procedimiento `IrDeLocalACentral(dir)`, que mueve el cabezal del depósito local en dirección `dir`, al depósito central. Vale asumir que el cabezal está en el depósito local correspondiente. OJO si se pide p.ej. `IrDeLocalACentral(Sur)`, quiere decir que el cabezal está en el depósito Sur, por lo tanto, debe moverse ***hacia el norte***.

Antes de seguir, un ejemplo de uso de estos dos procedimientos. A partir del tablero que se mostró, IrDeCentralALocal(Sur) deja el cabezal en el depósito local Sur, o sea:

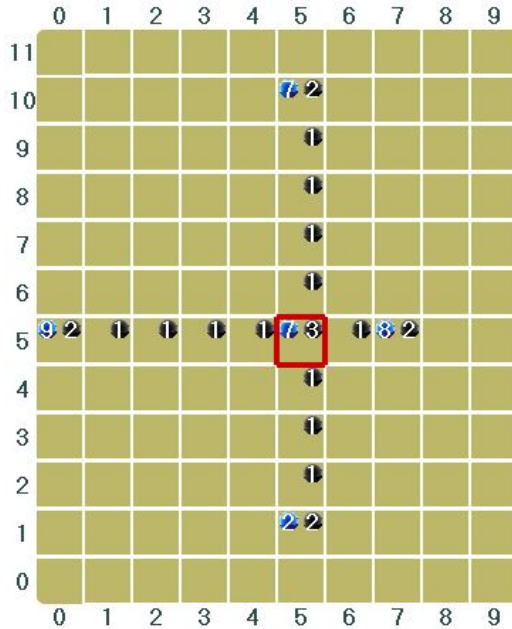
	0	1	2	3	4	5	6	7	8	9
11										
10						7 2				
9						1				
8						1				
7						1				
6						1				
5	9 2	1	1	1	1	4 3	1	8 2		
4						1				
3						1				
2						1				
1						5 2				
0										
	0	1	2	3	4	5	6	7	8	9

A partir de este tablero, IrDeLocalACentral(Sur) “vuelve” al tablero inicial, o sea, el cabezal va a al depósito central.

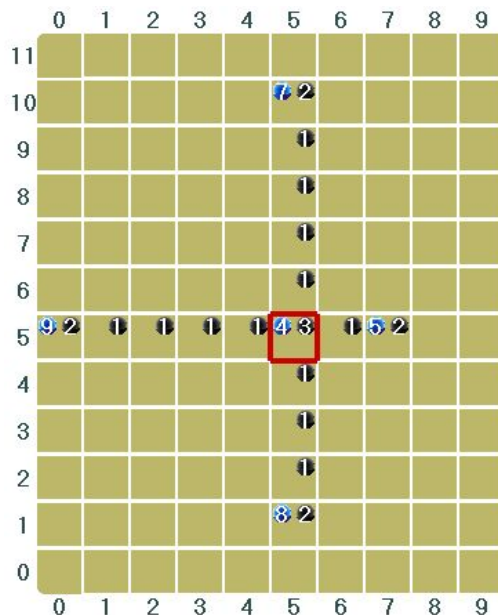
- El procedimiento LlevarMercaderia(cant, dir), que lleva la cantidad de mercadería indicada del depósito central al depósito local en la dirección indicada. Si en el depósito central no hay suficiente cantidad de mercadería, no se hace nada. Se puede suponer que el cabezal está en el depósito central, y debe dejarse en el mismo lugar. P.ej. a partir del tablero inicial, LlevarMercaderia(3, Sur) tiene este efecto:

	0	1	2	3	4	5	6	7	8	9
11										
10						7 2				
9						1				
8						1				
7						1				
6						1				
5	9 2	1	1	1	1	1 3	1	8 2		
4						1				
3						1				
2						1				
1						8 2				
0										
	0	1	2	3	4	5	6	7	8	9

5. El procedimiento `TraerMercaderia(cant, dir)`, que lleva la cantidad de mercadería indicada del depósito local en la dirección indicada, al depósito central. Si en el depósito local indicado no hay suficiente cantidad de mercadería, no se hace nada. Se puede suponer que el cabezal está en el depósito central, y debe dejarse en el mismo lugar. P.ej. a partir del tablero inicial, `TraerMercaderia(3, Sur)` tiene este efecto:



6. El procedimiento `MoverMercaderia(cant, dirOrigen, dirDestino)`, que mueve la cantidad indicada de mercadería del depósito local en dirección `dirOrigen` al que está en dirección `dirDestino`. Vale suponer que en el depósito origen hay la mercadería necesaria. y que el cabezal está en el depósito central, donde debe dejarse. P.ej. a partir del tablero inicial, `MoverMercaderia(3, Este, Sur)` tiene este efecto:



7. El procedimiento `IrACentralDesdeCualquierLado()`, que ubica el cabezal en el depósito central, independientemente de donde esté, fuera o dentro de la cruz que marca el “mapa” de depósitos.
8. La función `cantidadDeMercaderiaEnDepLocal(dir)`, que denota ... exactamente eso. Vale tomar como precondition que el cabezal está en el depósito central.
9. La función `cantidadTotalDeMercaderia()`, que denota el total de mercadería contando los cuatro depósitos locales más el central. Vale tomar como precondition que el cabezal está en el depósito central.

Apuestas

En esta serie de ejercicios, el tablero tiene información sobre apuestas, para un juego de extracción de números. En la casa de apuestas hay una cantidad de jugadores, cada jugador está identificado por un número.

Cada celda representa una apuesta, de esta forma:

- Bolitas rojas: el número de jugador.
- Bolitas azules: el número apostado.
- Bolitas verdes: el monto apostado, cada bolita verde es un peso.

OJO puede haber varias apuestas del mismo apostador, a distintos números.

Se pide escribir:

1. El procedimiento `AgregarApuesta(nroJugador, nroApostado, monto)` que agrega la apuesta indicada en alguna celda vacía.
Precondición: hay al menos una celda vacía.
2. El procedimiento `PagarYCobrar(nroQueSalio)`, que pague las apuestas que acertaron, y retire el dinero de las que no acertaron. Se paga 5 veces el monto de la apuesta, p.ej. para una apuesta de 3 pesos se pagan 15, quedando 18 pesos en la celda.
3. El procedimiento `RecogerPropinas()`, que extrae un peso (o sea una bolita verde) de cada celda donde haya al menos 15 pesos.
4. El procedimiento `SeVaJugador(nroJugador)`, que borra todas las celdas que registran una apuesta del jugador indicado.
5. El procedimiento `DuplicarApuestasAl(nroApostado)`, que duplica el monto de las apuestas al número indicado.
6. El procedimiento `DuplicarApuestasDelJugador(nroJugador)`, que duplica el monto de las apuestas que hizo el jugador indicado.
7. La función `estaElJugador(nroJugador)`, que denota verdadero si el jugador indicado tiene registrada, al menos, una apuesta.

8. La función `alguienJugoMasDe(nroApostado, cantMinima)`, que denota verdadero si algún jugador apostó al número indicado por un monto mayor a `cantMinima`.
9. El procedimiento `CambiarNroApostado(nroAnterior, nroNuevo)`, que cambia todas las apuestas de `nroAnterior` para que pasen a apostar a `nroNuevo`.
10. El procedimiento `BuscarApuesta(nroJugador, nroApostado)`, que posiciona el cabezal en una celda que sea el registro de una apuesta del jugador y número apostado indicados. Si no se ha registrado una apuesta con estas características, que el cabezal se ubique en el extremo noreste.