Gobstones

Introducción a la Programación - Práctica 6

Repetición condicional y recorridos

CONSEJOS:

- Leer el enunciado en su totalidad y pensar en la forma de resolverlo **ANTES** de empezar a escribir código.
- Si un ejercicio no sale, se puede dejar para después y continuar con los ejercicios que siguen.
- Los ejercicios están pensados para ser hechos después de haber mirado la teórica correspondiente, y esta después de las actividades de indagación.
- Los ejercicios están tomados de las guías prácticas utilizadas en la materia de Introducción a la Programación de la Universidad Nacional de Quilmes por Pablo Ernesto “Fidel” Martínez López y su equipo, Introducción a la Programación de la Universidad Nacional de Hurlingham de Alan Rodas Bonjour y su equipo, de ejercicios y actividades realizadas por Federico Aloi y Miguel Miloro, a su vez basada en las guías Ejercicios de Introducción a la Programación del CIU General Belgrano, elaboradas por Carlos Lombardi y Alfredo Sanzo, y Fundamentos de la Programación del Proyecto Mumuki. Agradecemos a todos los que nos ayudaron con su inspiración.
- **Realizar EN PAPEL los ejercicios que así lo indiquen.**
- **Si un ejercicio indica BIBLIOTECA significa que será útil para la realización de futuros ejercicios tanto en esta guía como en las siguientes, pudiendo ser utilizado sin tener que volver a definirlo. Es útil mantener registro de dichos procedimientos en su carpeta.**

EJERCICIOS:

1. Definir el procedimiento **`IrAlBorde_(dirección)`**, que lleva al cabezal al borde dado por el parámetro **`dirección`**, SIN utilizar el comando primitivo **`IrAlBorde`**. Dado que el único otro comando primitivo que permite mover el cabezal es **`Mover`**, debe *repetirse* su uso *hasta que* se haya cumplido el propósito. ¿Cuál es la condición que indica que el propósito se cumplió?

2. Volver a definir el procedimiento **`SacarTodasLasDeColor_(colorASacar)`**, que quita todas las bolitas del color dado por el parámetro **`color`** de la celda actual, pero esta vez SIN utilizar la expresión primitiva **`nroBolitas`**.

3. Considerar el procedimiento **`VaciarFilaDe_(color)`**, que debe quitar todas las bolitas del color dado por el parámetro **`color`** de cada una de las celdas de la fila actual[1]. El cabezal puede empezar en cualquier celda de la fila, y también puede terminar en cualquier celda de la fila (ya sea celda inicial o cualquier otra).

    a. Definir el procedimiento, como siempre, comenzando por establecer el contrato, y luego recién el código.
    b. ¿La solución dada funciona si el cabezal se encuentra en medio de una fila? Si no es así, corregir el programa para que funcione en este caso también.
    c. Al recorrer la fila, ¿en qué dirección se movió el cabezal? ¿Podría haberse movido en la dirección opuesta?
    d. A partir de la respuesta anterior, ¿de cuántas formas posibles se puede realizar el recorrido de una fila?

4. En cada uno de los casos siguientes, definir de la forma indicada el procedimiento **`VaciarTableroDe_(color)`**, que quite todas las bolitas del color dado por el parámetro **`color`** de cada una de las celdas del tablero. El cabezal puede empezar en cualquier celda del tablero, y también puede terminar en cualquier celda del tablero (ya sea la celda inicial o cualquier otra).

    a. Estructurar el procedimiento como un *recorrido sobre las filas*[2]. ¿Qué subtareas van a precisarse en este caso? ¿Es necesario volver a definirlas o se pueden encontrar en esta práctica?
    b. Estructurar el procedimiento como un *recorrido sobre las celdas* del tablero. Las subtareas necesarias serán diferentes, y puede ser que sea necesario definir alguna que aún no está disponible en esta práctica.
    c. Reflexionar sobre las diferencias de los recorridos dados en los puntos a. y b. ¿Qué subtareas son más complejas en cada caso? ¿Podrían considerarse otros recorridos que no fueran sobre celdas o filas?

5. Considerar los recorridos por filas y por celdas del tablero realizados en el ejercicio anterior, y en cada uno, las posibles direcciones en las que se realiza el procesamiento.

---

[1] La fila actual es aquella en la que se encuentra la celda actual.

[2] Recordar que un recorrido considera una secuencia de elementos de a uno por vez. Un *recorrido sobre filas* es uno que considera que los elementos a recorrer son las filas del tablero…

a. En el caso del recorrido por filas, ¿cuántas posibilidades hay para recorrer todas las filas (suponiendo que ya está hecho el procedimiento para procesar una fila)? ¿Qué dato es el que cambia en cada caso?
b. En el caso del recorrido por celdas, ¿cuántas posibilidades hay para recorrer todas las celdas?
c. Volver a definir los recorridos por filas y por celdas del ejercicio anterior, pero con diferentes direcciones de recorrido.

6. Escribir el procedimiento **`VaciarFilaDe_HaciaEl_(color, dirección)`**, que generalice el recorrido de la fila, recibiendo la dirección de movimiento como parámetro. ¿Cuáles son los valores posibles para el parámetro **`dirección`**? ¿Por qué no puede ser cualquiera, si lo que se desea es recorrer una *fila*?

7. **BIBLIOTECA** Escribir los procedimientos necesarios para generalizar la noción de recorrido por celdas de un tablero, para que las direcciones de recorrido no estén fijas. En particular, definir (como siempre, comenzando por los contratos):

   a. **`IrAPrimeraCeldaEnUnRecorridoAl_Y_(dirPrincipal, dirSecundaria)`**
   b. **`haySiguienteCeldaEnUnRecorridoAl_Y_(dirPrincipal, dirSecundaria)`**
   c. **`IrASiguienteCeldaEnUnRecorridoAl_Y_(dirPrincipal, dirSecundaria)`**

   Al escribir las precondiciones, tener en cuenta que las direcciones no pueden ser cualesquiera, sino que deben estar relacionadas… ¿Cuál es esa relación? ¿Cómo expresarla?

8. **EN PAPEL** El caminante

   Se puede modelar el paseo de un caminante por el tablero con las siguientes consideraciones para la representación.

   - El caminante está representado por una a cuatro bolitas azules. La dirección de su paseo es Norte si es una bolita, Este si son dos, Sur si son tres y Oeste si son cuatro.
   - Las indicaciones de cambio de dirección se representan con bolitas verdes. Si el caminante llega a una celda con una de estas indicaciones, debe cambiar de dirección. La cantidad de bolitas verdes indica la nueva dirección, con la misma representación de direcciones dadas para el caminante.
   - El caminante deja una huella de bolitas negras a su paso, una por cada paso.
   - La meta se representa con cualquier número de bolitas rojas. El paseo del caminante termina si llega a la meta.
   - La celda actual siempre se encuentra sobre el caminante.
   - La única celda con bolitas azules es la del caminante.
   - Todas las celdas tienen un máximo de 4 bolitas verdes.
   - Las indicaciones llevan al caminante a la meta.

   Como ayuda para guiar la división en subtareas, ya se realizó un análisis *top-down* de la estrategia, y se eligieron ciertas subtareas. Se pide, entonces, implementar los procedimientos y funciones que expresan dichas subtareas, que son los indicados a continuación. Observar que en su gran mayoría, las tareas están presentadas en

forma *top-down*, por lo que es interesante mirarlas todas antes de empezar a implementar, y definir todos los contratos antes de proceder a escribir el código de cada una, ya que las de niveles más alto se pueden servir de las de niveles más bajos. Además, puede tomarse la siguiente función como primitiva:

```
function direcciónDelCódigo_(código)
/* PROPÓSITO:
     describir la dirección correspondiente al
     código dado
   PRECONDICIONES:
     * el código está entre 1 y 4
   PARÁMETROS:
     * código: Número. El número que codifica la
       dirección descripta
*/
```

Al escribir los contratos, no olvidar establecer las precondiciones necesarias (ya que las mismas no siempre se explicitan en los enunciados).

a. **`caminante()`**, **`indicador()`**, **`huella()`** y **`meta()`**, que describen los colores con los que se representa cada uno de los elementos nombrados.

b. **`LlevarAlCaminanteALaMeta()`** que, suponiendo que en el tablero está representado un escenario válido para el caminante, lleva al caminante hasta la meta.

c. **`estáEnLaMeta()`** que indica si el caminante está o no en la meta.

d. **`DarUnPaso()`** que realiza un paso en el paseo del caminante, de acuerdo a las siguientes reglas.
   i. Si el caminante ya llegó a destino, no hay nada que hacer.
   ii. Si hay que cambiar la dirección, lo hace.
   iii. Finalmente, se mueve en la dirección correspondiente.

   No hay que olvidar que el caminante debe dejar una huella. En la Figura 1 se ofrece un ejemplo del uso de **`DarUnPaso()`** en medio de la ejecución del programa.

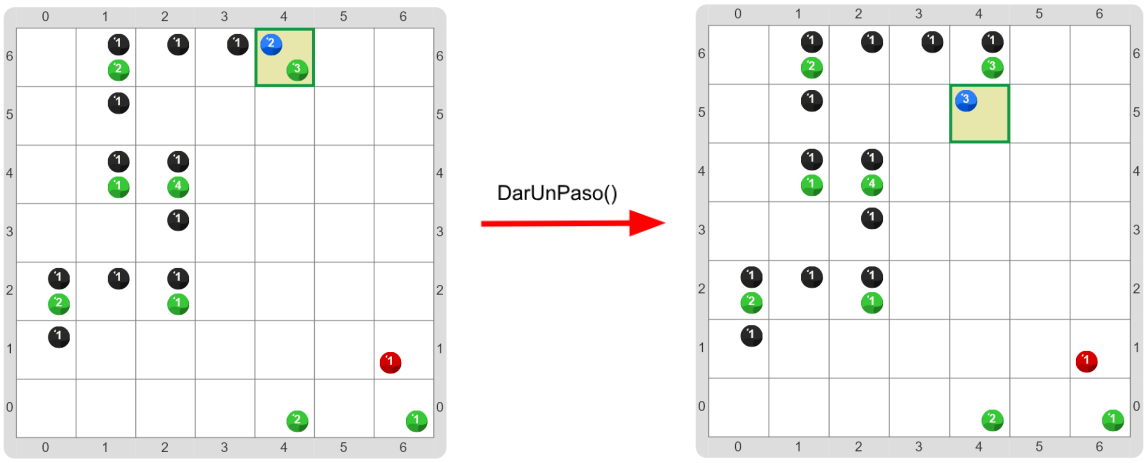


Figura 1: ejemplo del funcionamiento de **`DarUnPaso()`** en medio de la ejecución del programa

e. **`CambiarDeDirecciónSiHayIndicador()`** que cambia la dirección del caminante cuando se encuentra con un indicador.

f. **`DejarHuella()`** que deja una huella en la celda actual.

g. **`MoverAlCaminanteAl_(dirección)`** que mueve al caminante un paso en la dirección dada.

h. **`hayIndicadorDeCambioDeDirección()`** que describe verdadero cuando en la celda actual hay un indicador de dirección.

i. **`direcciónIndicada()`** que describe la dirección en la que está mirando el caminante.

j. **`Cambiar_ParaImitar_(colorACambiar, colorAImitar)`** que cambia la cantidad de bolitas de **`colorACambiar`** según la cantidad de bolitas de **`colorAImitar`** que haya en la celda actual.

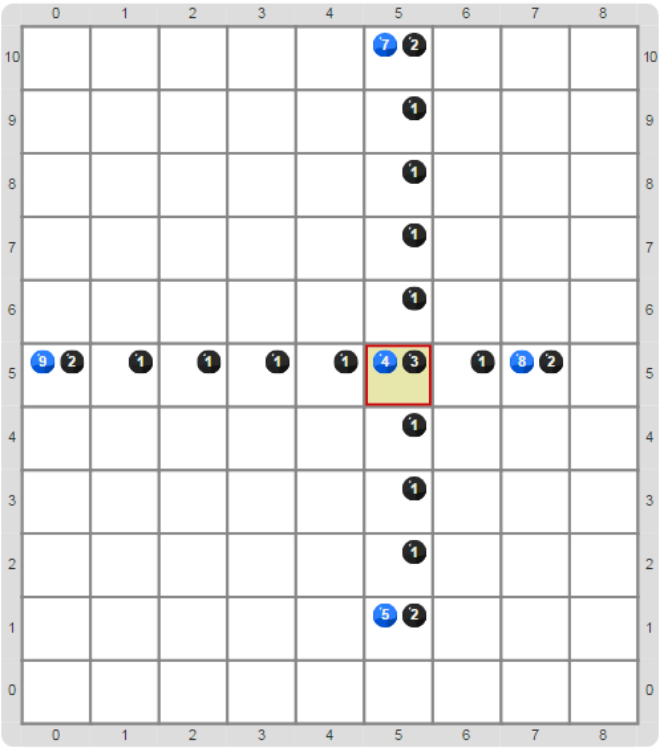
k. **`Mover_Bolitas_Al_(cantidad, color, dirección)`** que "mueve" (es decir, quita de una celda para llevar a la otra) la **`cantidad`** indicada de bolitas de **`color`** a la celda lindante en la **`dirección`** dada, y deja el cabezal en esa celda. Suponer que hay una celda lindante en esa dirección.

9. Distribución de mercadería

Se desea 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, se va a representar en el tablero un mapa muy simplificado.

- 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.

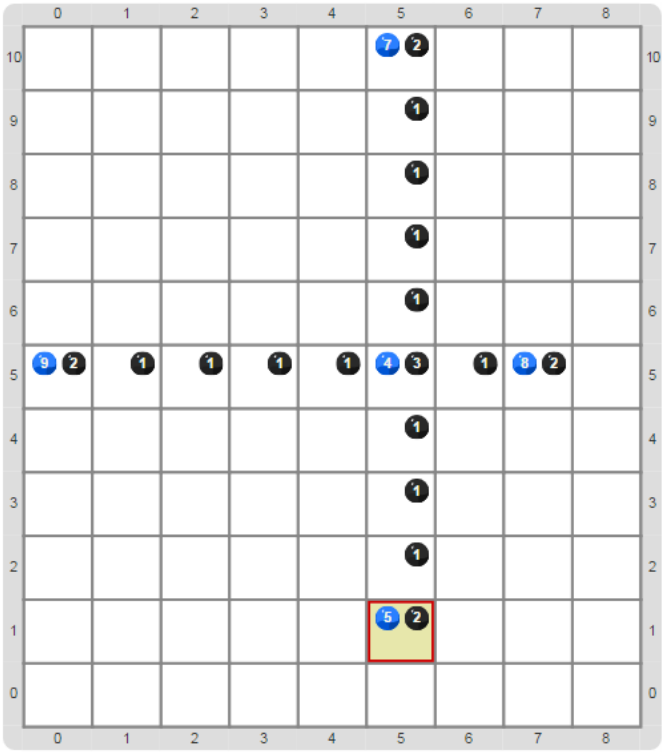
Los depósitos locales forman una cruz, donde el centro es el depósito central. No se sabe a qué distancia están los depósitos locales del depósito central. Este es un ejemplo de modelo:



Escribir

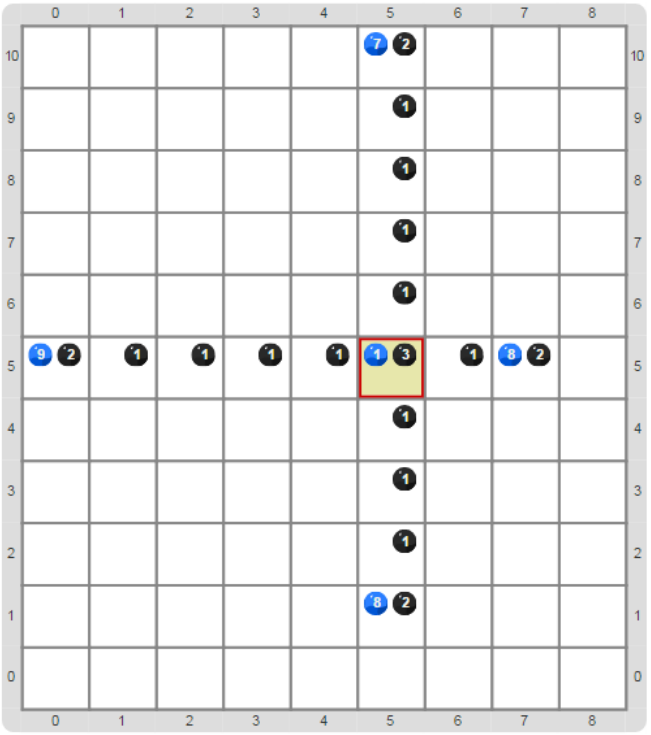
a. **`esDepósitoCentral()`** y **`esDepósitoLocal()`** que indican si el cabezal está, respectivamente, en el depósito central o en un depósito local.
b. **`IrDeCentralAlLocal_(dirección)`**, que mueve el cabezal del depósito central al depósito local que está en la `dirección` dada, suponiendo que el cabezal comience en el depósito central.
c. **`IrDelLocal_ACentral(dirección)`**, que mueve el cabezal al depósito central, suponiendo que el cabezal está en el depósito local que está en la **`dirección`** dada.
   Aclaración: si se pide **`IrDelLocal_ACentral(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, **`IrDeCentralAlLocal_(Sur)`** deja el cabezal en el depósito local **`Sur`**, o sea:

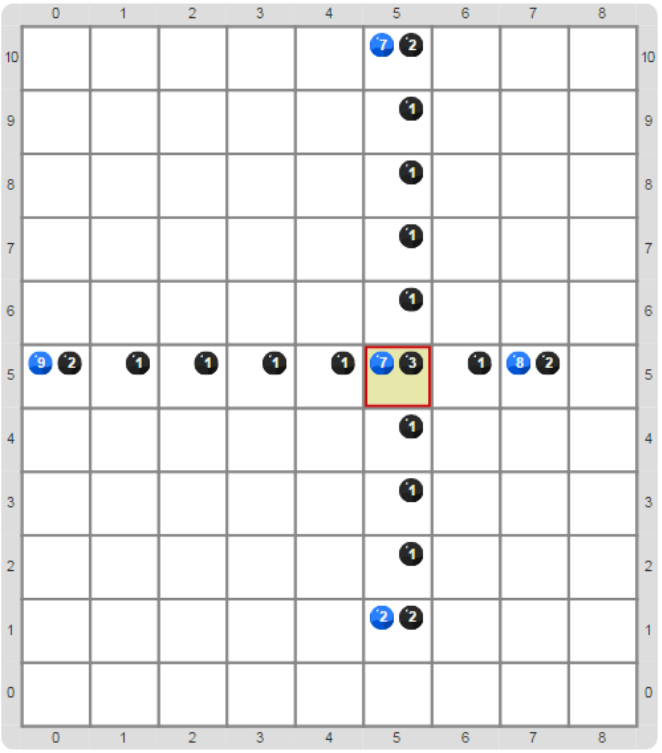


A partir de este tablero, **`IrDelLocal_ACentral(Sur)`** “vuelve” al tablero inicial, o sea, el cabezal va al depósito central.

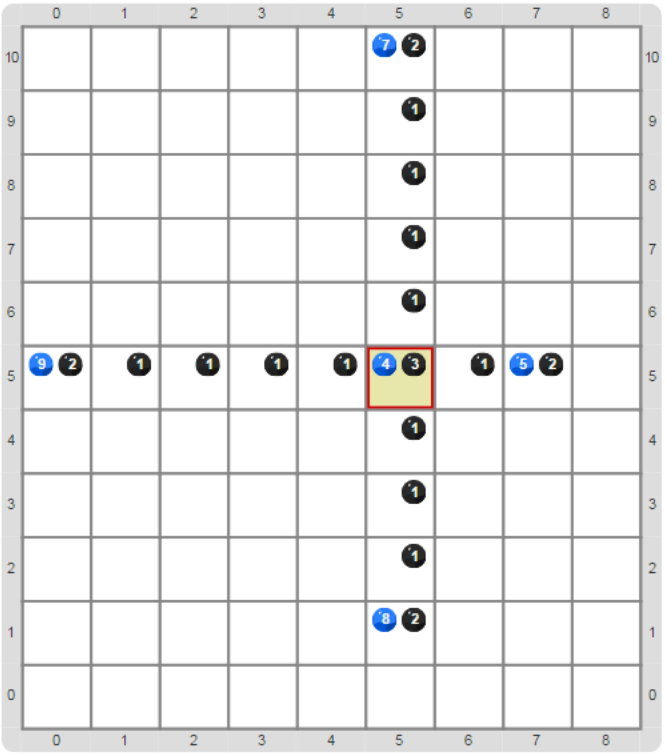
d. **`Llevar_MercaderíasAlLocal_(cantidad, dirección)`**, que lleva la **`cantidad`** de mercadería indicada del depósito central al depósito local que está en la **`dirección`** indicada. Si en el depósito central no hay suficiente cantidad de mercadería, no se hace nada. Se puede suponer que el cabezal está en el depósito central, y debe dejarse en el mismo lugar.
Por ejemplo a partir del tablero inicial dado como ejemplo, **`Llevar_MercaderíasAlLocal_(3, Sur)`** tiene este efecto:



e. **`Traer_MercaderíasDelLocal_(cantidad, dirección)`**, 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. Por ejemplo, a partir del tablero inicial, **`Traer_MercaderíasDelLocal_(3, Sur)`** tiene este efecto:



f. **`Mover_MercaderíasDelLocal_AlLocal_(cantidad, origen, destino)`**, que mueve la **`cantidad`** indicada de mercadería del depósito local que está en dirección **`origen`** al que está en dirección **`destino`**. Si en el depósito origen no hay la cantidad de mercadería necesaria, no se hace nada. Por ejemplo, a partir del tablero inicial, **`Mover_MercaderíasDelLocal_AlLocal_(3, Este, Sur)`** tiene este efecto:



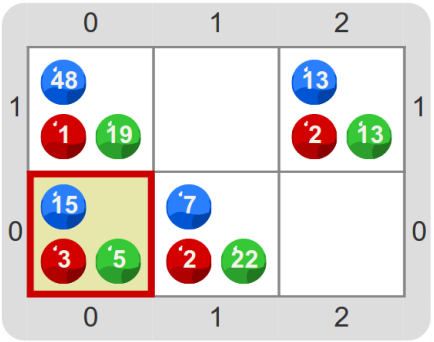
10. El casino, parte 1

En este ejercicio el tablero tiene información sobre las apuestas de diferentes jugadores en un casino, para un juego de extracción de números. En la mesa del juego hay una cantidad de jugadores, cada uno identificado por un número. Cada celda del tablero representa una apuesta, de la siguiente forma.

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

Puede haber varias apuestas del mismo apostador, a distintos números.
A modo de ejemplo, se muestra uno de los posibles tableros que modela la siguiente situación:

- El jugador 1 apostó 19 pesos al 48.
- El jugador 2 apostó 22 pesos al 7 y 13 pesos al 13.
- El jugador 3 apostó 5 pesos al 15.
- Queda espacio para registrar dos apuestas más.



Escribir las siguientes operaciones. En cada caso, recordar escribir primero el contrato, y expresar la estrategia usando subtareas (preferentemente con la metodología *top-down*).

a. **`AgregarApuestaDe_Al_Para_(monto, nroApostado, nroJugador)`** que agrega la apuesta indicada en alguna celda vacía, la cual debe existir.

b. **`PagarYCobrarAl_(nroQueSalió)`**, que pague las apuestas que acertaron, y retire el total del dinero de las que no acertaron (solamente el dinero). Se paga 5 veces el monto de la apuesta; por ejemplo, para una apuesta de 3 pesos se pagan 15, quedando 18 pesos en la celda.

c. **`RecogerPropinas()`**, que extrae un peso de cada celda donde haya al menos 15 pesos.

d. **`SeVaElJugador_(nroJugador)`**, que borra todas las celdas que registran una apuesta del jugador indicado.

e. **`DuplicarApuestasAl_(nroApostado)`**, que duplica el monto de las apuestas al número indicado.

f. **`DuplicarApuestasDelJugador_(nroJugador)`**, que duplica el monto de las apuestas que hizo el jugador indicado.

g. **`CambiarNroApostadoDel_Al_(nroAnterior, nroNuevo)`**, que cambia todas las apuestas de **`nroAnterior`** para que pasen a apostar a **`nroNuevo`**.

h. **`AumentarEn_LaApuestaDelJugador_Al_(monto, nroJugador, nroApostado)`**, que agrega el monto indicado a la apuesta de este jugador al número dado.

i. **`BuscarApuestaDelJugador_Al_(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, el cabezal debe ubicarse en el extremo noreste.