Gobstones

Introducción a la Programación - Práctica 4

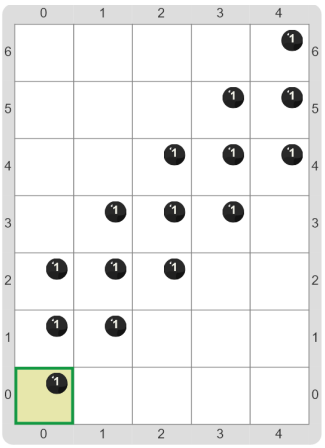
Expresiones y tipos

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.**

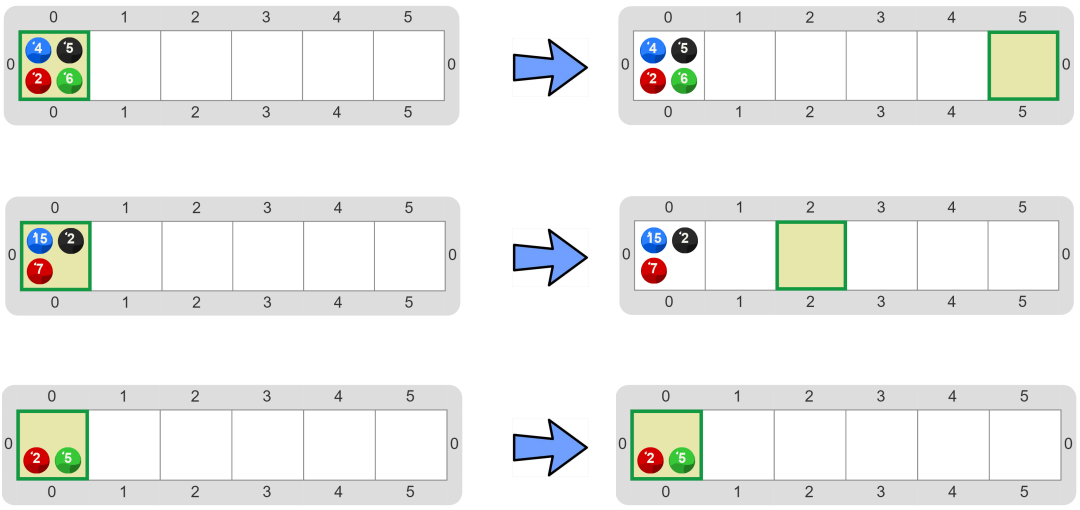
EJERCICIOS:

1. Escribir **`DibujarBanda_EnDiagonalNEDe_x_(color, alto, largo)`** que dibuja una banda diagonal ancha. Por ejemplo, en la figura se observa el resultado de ejecutar **`DibujarBanda_EnDiagonalNEDe_x_(Negro, 3, 5)`** en un tablero de 5x7, comenzando en la esquina SO del tablero.



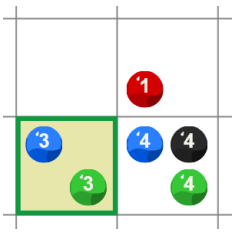
Observar que 3 es el alto de la banda, y 5 el largo.

2. Escribir el procedimiento **`Mover_SegúnColor_(dirección,color)`**, que mueve el cabezal en la dirección dada tantas celdas como bolitas de color dado hay en la celda actual. Como ejemplos se ofrecen los resultados de evaluar el comando **`Mover_SegúnColor_(Este, Negro)`**, en diferentes tableros iniciales.

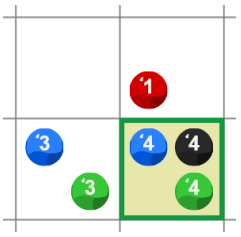


En el último caso, como la celda no tiene bolitas negras (o sea tiene 0 bolitas negras), entonces el cabezal no se mueve (o sea, se mueve 0 celdas hacia el Este).

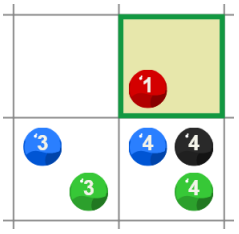
3. Indicar el valor y el tipo que representan las expresiones dadas en los ítems en cada uno de los tableros A , B y C, suponiendo definido un procedimiento con el contrato dado al final.



(A)



(B)



(C)

a. `nroBolitas(Negro) + nroBolitas(Azul)`
b. `opuesto(opuesto(Este))`
c. `nroBolitas(siguiente(Azul))`
d. `2*nroBolitas(colorAImitar)`
   suponiendo que esta expresión aparece dentro del cuerpo del procedimiento `PonerElDobleDe_QueDe_`, y que se lo invocó como
   `PonerElDobleDe_QueDe_(Rojo, Azul)`

El contrato del procedimiento dado es el siguiente:

```
procedure PonerElDobleDe_QueDe_(colorAPoner
                                , colorAImitar)
  /* PROPÓSITO: Poner bolitas de color **colorAPoner** en
                una cantidad que sea el doble de las que
                hay de color **colorAImitar** en la
                celda actual
     PRECONDICIONES: ninguna
     PARÁMETROS:
       * colorAPoner: Color. El color de las bolitas a
                      poner
       * colorAImitar: Color. El color de las bolitas cuya
                       cantidad se va a imitar en la celda
                       actual
  */
```

4. El bosque, parte 3

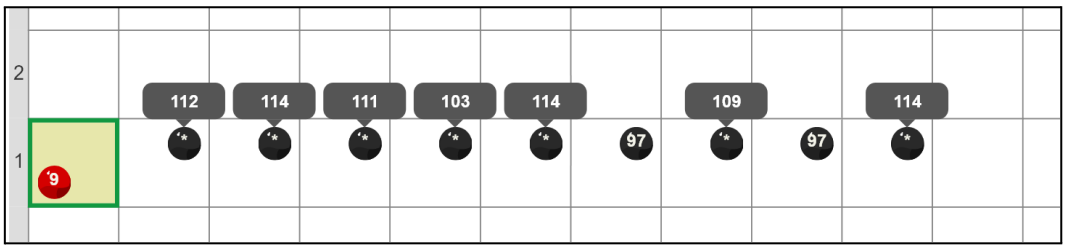
   En este ejercicio continuaremos con nuestro bosque, esta vez colocando semillas y árboles en la celda lindante hacia alguna dirección, y dejando el cabezal en la celda inicial.

   a. **Poner_SemillasAl_(cantidadDeSemillas, direcciónAPoner)**
      // deja el cabezal en la celda inicial
   b. **Sacar_ÁrbolesAl_(cantidadDeÁrboles, direcciónASacar)**
      // deja el cabezal en la celda inicial
   c. **Sacar_SemillasEnDiagonalAl_Y_(cantidadDeSemillas, primeraDirDiagonal, segundaDirDiagonal)**
      // deja el cabezal en la celda inicial
   d. **Sacar_ÁrbolesEnDiagonalHorariaAl_(cantidadDeÁrboles, direcciónDiagonal)**
      // la diagonal horaria de una dirección es aquella dada por la dirección y su dirección siguiente.
      // Ej. la diagonal horaria de Norte es Norte-Este, la de Sur es Sur-Oeste.
      // deja el cabezal en la celda inicial

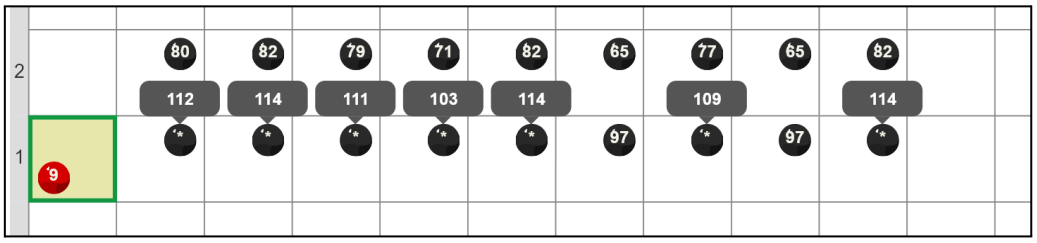
5. Hacer el procedimiento **`PasarPalabraActualAMayúsculas()`** que suponiendo que en la fila actual se codifica una palabra en minúsculas usando bolitas, ponga la misma palabra en mayúsculas en la fila al Norte.

   - Cada letra se representa con una cantidad diferente de bolitas negras, según un código numérico llamado ASCII.
   - En la celda más al Oeste de la fila actual se codifica la cantidad de letras usando bolitas rojas.
   - La primera letra de la palabra está en la celda lindante al Este de la que contiene la cantidad de letras.
   - En el código ASCII si las letras mayúsculas se codifican con un número N entonces la misma letra minúscula se representa con N+32 (pej. la 'a' minúsculas se representa con el número 97 y la 'A' mayúsculas, con el 65).
   - El cabezal se encuentra en la celda más al Oeste de una fila donde hay una palabra representada.

   Ejemplo de (fragmento de) un tablero inicial posible:



   Ejemplo del (fragmento de) tablero final correspondiente al anterior:



   ¿Cómo comenzar la resolución? En cada procedimiento, ¿qué parte debe escribirse primero?

6. **BIBLIOTECA** Escribir un procedimiento **`SacarTodasLasDeColor_(colorASacar)`**, que quite de la celda actual todas las bolitas del color indicado por el parámetro.
   Ayuda: considerar utilizar el procedimiento **`Sacar_DeColor_`**, definido en la práctica anterior. ¿Qué argumentos se le deberían pasar?

7. **BIBLIOTECA** Escribir un procedimiento **`VaciarCelda()`** que quite de la celda actual todas las bolitas de todos los colores, dejando la celda vacía.

8. ¡A la batalla!, parte 1

Suponiendo que se está programando un juego donde en las celdas del tablero se representan Soldados (los aliados con una bolita de color Negro y los enemigos con una bolita de color Rojo por cada soldado), escribir los siguientes procedimientos:

a. **`EnviarAliadosParaDuplicarEnemigos()`**, que agrega soldados aliados en la celda actual en cantidad suficiente para que haya el doble de aliados que de soldados enemigos.

b. **`PelearLaBatalla()`**, que simula una batalla, suponiendo que hay suficiente cantidad de soldados aliados como para ganar la batalla. Durante una batalla, 2 soldados enemigos pelean contra 3 soldados aliados y todos mueren. Por ejemplo, si hay 6 enemigos y 10 aliados, mueren los 6 enemigos y 9 de los aliados; si hay 11 enemigos y 21 aliados, mueren los 11 enemigos y 15 soldados aliados.
Ayuda: ¿qué cuenta hay que hacer para saber cuántos soldados aliados morirán?
Ayuda 2: para la división en Gobstones se usa el operador **`div`**.

9. Realizar el ejercicio “Soporte técnico” que se encuentra en el capítulo “Ejercicios de práctica” del curso online

(https://gobstones.github.io/gobstones-sr/?course=gobstones/curso-InPr-UNQ&github=gobstones/curso-InPr-UNQ&path=Proyectos/12.Ejercicios%20de%20pr%C3%A1ctica/2.Soporte%20T%C3%A9cnico)