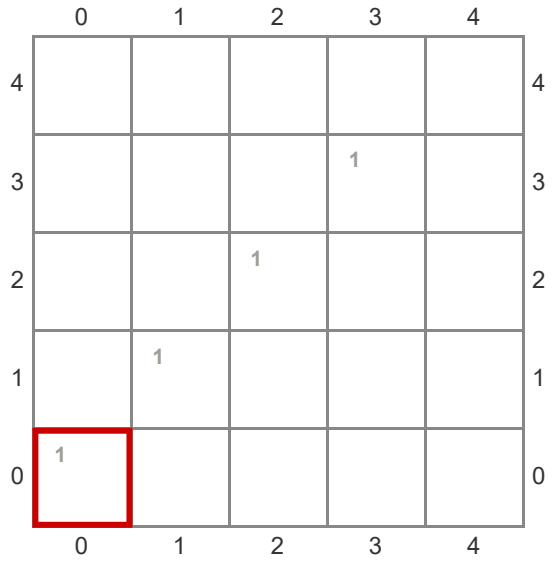
Entrando en calor... ¡Volviendo!

En la guía anterior, vimos que se podía usar `repeat` para hacer algo muchas veces. Hicimos un ejemplo que se llamaba `Diagonal4Azul`, que era más o menos así:

```
procedure Diagonal4Azul(){
    repeat(4){
        Poner(Azul)
        Mover(Este)
        Mover(Norte)
    }
}
```

> ¿Te animás a definir el procedimiento `Diagonal4AzulVolviendo`? Este procedimiento debería *hacer lo mismo* que `Diagonal4Azul`, pero tiene que dejar el cabezal **en la posición inicial**. Recordá que podés invocar todo lo que está en la Biblioteca sin necesidad de volver a definirlo.



Solución | Biblioteca

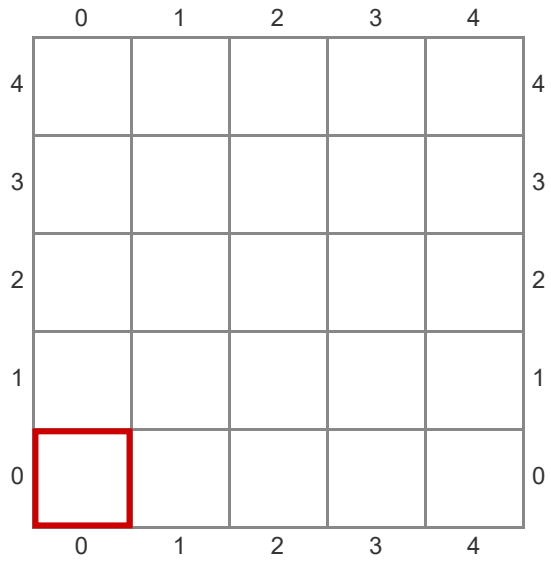
```
procedure Diagonal4AzulVolviendo(){
  Diagonal4Azul()
  repeat(4){
        Mover(Sur)
        Mover(Oeste)
```

```
    }
}
```

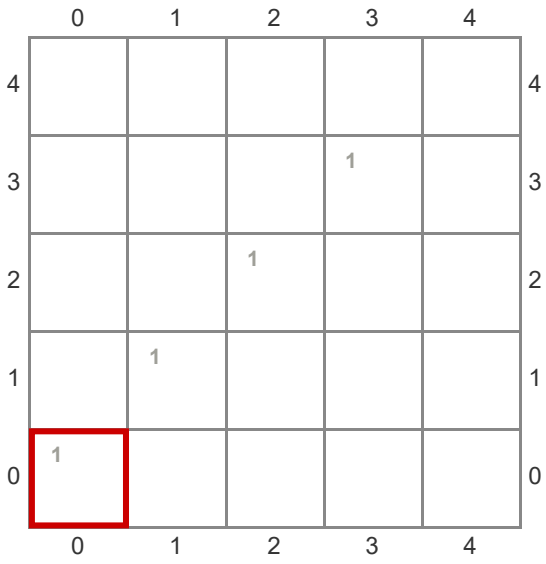
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



¡Bien!

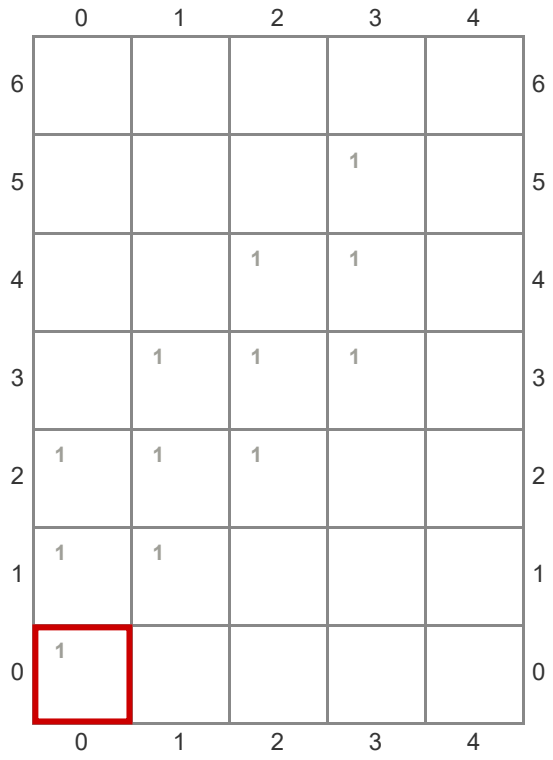
Tenés que acostumbrarte a pensar... *¿No podría **usar** algún procedimiento que **ya definí** antes?*

Una gran ventaja de los procedimientos es que, una vez que están escritos, podés guardártelos para volver a usarlos. Si ahora los necesitás en algún ejercicio vas a notar que **ya los guardamos por vos** y te dejamos la **Biblioteca** lista para usar, pero en la vida real ese trabajo vas a tener que hacerlo vos.

Una diagonal más ancha

Sigamos probando tus habilidades para reutilizar...

Ahora, tenés que hacer este dibujo:



> El procedimiento debe llamarse `BandaDiagonal4`. ¡Ojo! prestá atención a la **posición final** del cabezal.

¡Dame una pista!

Solución | Biblioteca

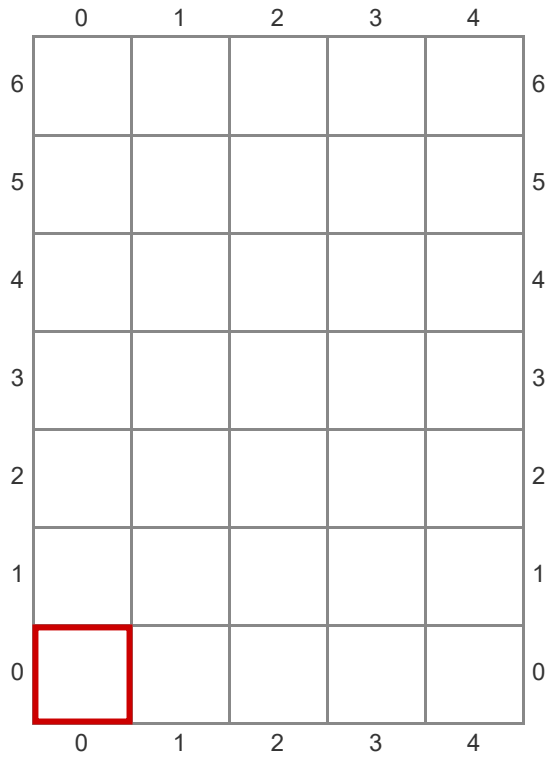
```
procedure BandaDiagonal4(){
 repeat(3){
  Diagonal4AzulVolviendo()
  Mover(Norte)
 }
 Mover(Sur)
 Mover(Sur)
```

```
 Mover(Sur)
}
```

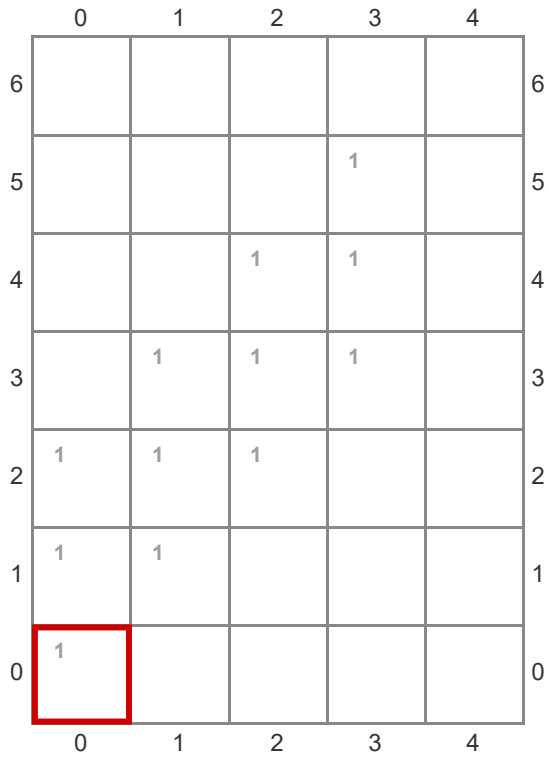
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



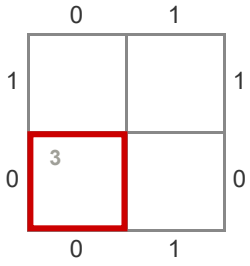
Pongamos... ¡Todo lo que queramos!

Ahora que tenemos una idea de *reutilización*, y practicamos *repetición*, vamos a definir un procedimiento **que nos va a servir de acá en adelante.**

3. Pongamos... ¡Todo lo que queramos!

Necesitamos un procedimiento que nos ayude a poner muchas bolitas. Sí, podríamos simplemente usar un `repeat` para lograrlo, pero como es una tarea re-común que vamos a hacer un montón de veces, vamos a preferir definir un `procedure` llamado `PonerN`. Nuestro procedimiento debe poner la cantidad de bolitas indicada de un color dado.

Por ejemplo, `PonerN(3, Azul)` haría esto:



Definí el procedimiento `PonerN(cantidad, color)`.

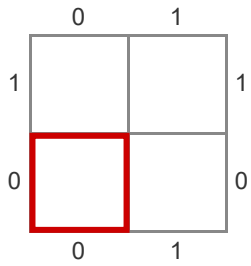
```
procedure PonerN(numero, color){
 repeat(numero){
  Poner(color)
  }
}

```

Enviar

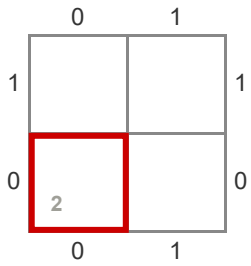
¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

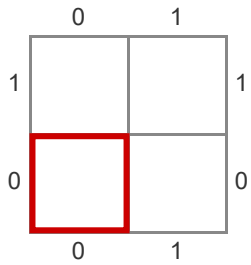
Tablero inicial



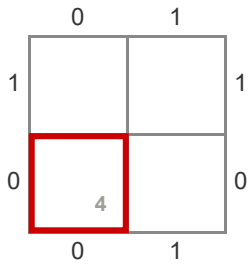
Tablero final



Tablero inicial



Tablero final



Aunque quizás no veas todavía la utilidad de este `procedure` que creamos, te contamos dos aspectos que es importante tener en cuenta al programar:

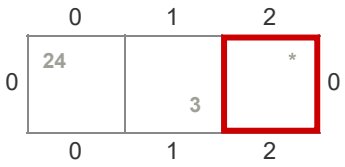
- **reutilización de código:** como poner muchas bolitas es una tarea común, está bueno tener un procedimiento que lo resuelva: lo escribimos una vez y lo usamos para siempre;
- **declaratividad:** cuando tengamos que resolver un problema más complejo, tener este procedimiento nos va a ayudar a pensar a más alto nivel, ya que no vamos a tener que preocuparnos por **cómo** poner muchas bolitas sino en **qué** queremos construir con ellas.

Día de la Memoria

Muchas veces vamos a usar el tablero de Gobstones como memoria, o sea, para recordar algo importante que vamos a necesitar más adelante.

¿Qué podríamos representar con bolitas? Por ejemplo una fecha. Una fecha que debemos recordar es el *24 de marzo de 1976*, hoy constituido Día de la Memoria por la Verdad y la Justicia en Argentina.

> El objetivo, entonces, es definir un procedimiento `DiaDeLaMemoria()` :
>
> - En la celda actual, poné 24 bolitas Azules, que **representan** el día.
> - En la celda inmediatamente al Este, poné 3 bolitas Verdes, que **representan** el mes.
> - En la celda a continuación, poné 1976 bolitas Negras, **representando** el año.



¡Dame una pista!

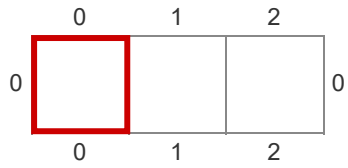
Solución | Biblioteca

```
1 procedure DiaDeLaMemoria(){
2   PonerN(24, Azul)
3   Mover(Este)
4   PonerN(3, Verde)
5   Mover(Este)
6   PonerN(1976, Negro)
7 }
8
```

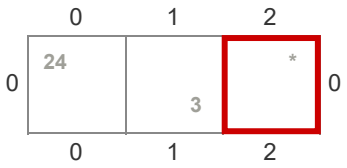
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



¿Sabías que `Azul` es una **expresión literal**? ¡También `1976` ! También son **expresiones literales**: `Verde` , `Negro` , `3` y `24` .

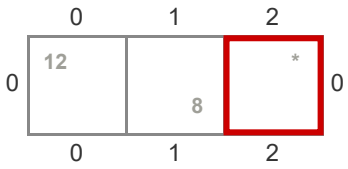
Cuando **usamos** un procedimiento que tiene parámetros como PonerN, `PonerN(56, Rojo)` tenemos que enviarle valores como argumento. ¡Y las expresiones sirven para eso!

Escribir cualquier fecha

Ahora que ya escribimos una fecha particular, hagamos un procedimiento que sirva para escribir cualquier fecha.

> Definí el procedimiento `Fecha(dia, mes, anio)`, que recibe los **tres valores** correspondientes, y escribe la fecha que representan, de esta manera:
>
> - En la celda actual, tantas bolitas azules para **representar** el día.
> - En la celda inmediatamente al Este, tantas bolitas Verdes para **representar** el mes.
> - En la celda a continuación, tantas bolitas Negras para **representar** el año.

Por ejemplo, `Fecha(12, 8, 1990)` produciría algo así:



¡Dame una pista!

Solución | Biblioteca

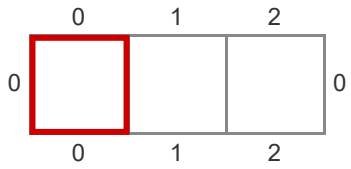
```
procedure Fecha(dia, mes, anio) {
  PonerN(dia, Azul)
  Mover(Este)
  PonerN(mes, Verde)
  Mover(Este)
  PonerN(anio, Negro)
}

```

Enviar

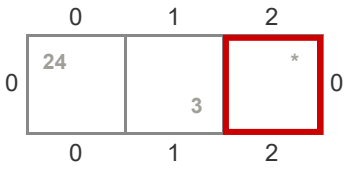
¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

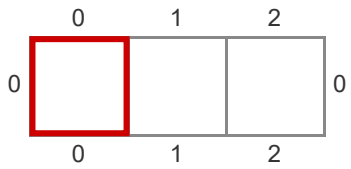
Tablero inicial



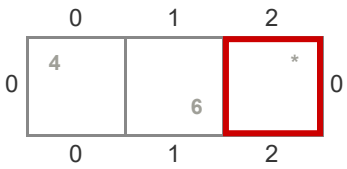
Tablero final



Tablero inicial



Tablero final



Dos cuestiones teóricas para pensar de este ejercicio:

1. Puede parecerte que estás **repitiendo código** con el ejercicio anterior. ¡Es cierto! Para arreglarlo, deberías volver al ejercicio anterior y allí usar el procedimiento `Fecha`.
2. Acá vemos que hay otro tipo de **expresiones**: ¡Los parámetros! Al **usar** un procedimiento, puedo enviarle tanto otros parámetros como literales: `PonerN(dia, Azul)`. Recordá en este caso que los nombes de los parámetros sólo nos sirven a los humanos, para la máquina sólo importa el orden.

Información importante

Términos y Condiciones

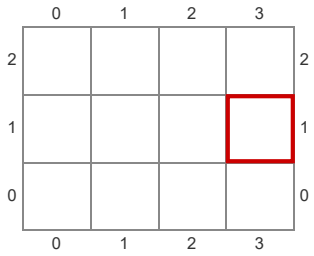
Reglas del Espacio de Consultas

Movamos... ¡Todo lo que queramos!

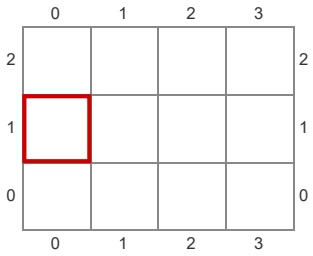
Definí un procedimiento `MoverN(cantidad, direccion)` que haga que el cabezal se desplace la cantidad especificada de veces en la dirección indicada.

Por ejemplo, `MoverN(3, Oeste)` provocaría:

Inicial



Final



¡Dame una pista!

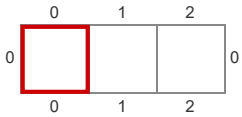
```
procedure MoverN(cantidad, direccion) {
  repeat(cantidad){
    Mover(direccion)
  }
}
```

Enviar

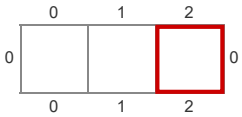
¡Muy bien! Tu solución pasó todas las pruebas

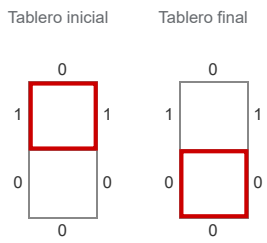
Resultados de las pruebas:

Tablero inicial



Tablero final





¡Perfecto!

Recordamos entonces que entre los paréntesis del `repeat` no sólo pueden ir números (como `7` o `42`), sino también otros tipos de **expresiones** que denoten valores numéricos (como `cantidad` en este caso).

Los números del reloj

¡Ya sabés Kung Fu!

Ahora, tenés que mostrarnos que podés *dibujar un reloj*. Lo que haremos por ahora es solamente poner los números que aparecen en un típico reloj de agujas:

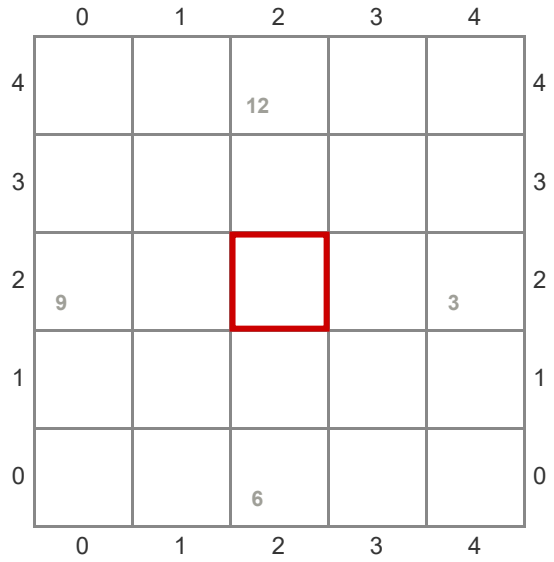
- El 12 arriba,
- El 3 a la derecha,
- El 9 a la izquierda, y
- el 6 abajo.

> Definí un procedimiento `DibujarReloj(radio)` que ponga los números del reloj como se indica arriba: **alrededor del casillero actual**. El tamaño del reloj se indica con el `radio` que recibís como parámetro: mientras más grande es el radio, más alejados están los números del centro.

Dado el siguiente program:

```
program {
  DibujarReloj(2)
}
```

El reloj resultante es así:



¡Dame una pista!

Solución | Biblioteca

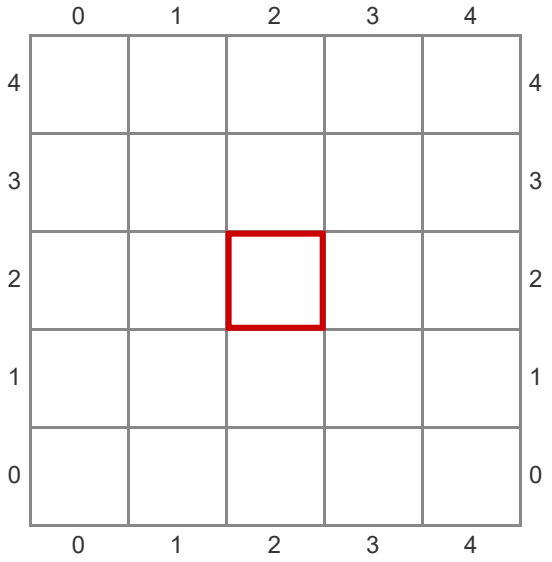
```
1  procedure DibujarReloj(radio){
2    MoverN(radio,Norte)
3    PonerN(12, Rojo)
4    MoverN(radio,Sur)
5    MoverN(radio,Este)
6    PonerN(3, Rojo)
7    MoverN(radio,Sur)
8    MoverN(radio,Oeste)
9    PonerN(6, Rojo)
10   MoverN(radio,Norte)
11   MoverN(radio,Oeste)
12   PonerN(9, Rojo)
13    MoverN(radio,Este)
14 }
```

Enviar

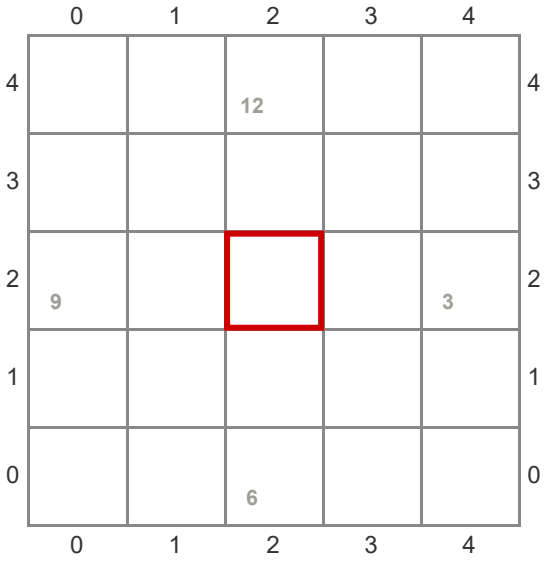
¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

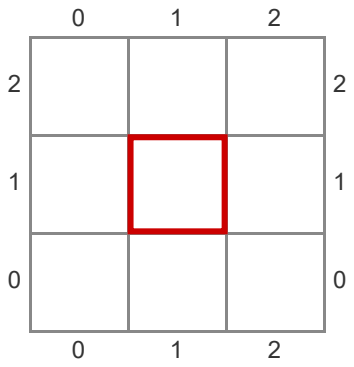
Tablero inicial



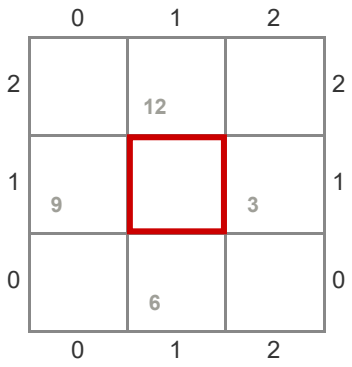
Tablero final



Tablero inicial



Tablero final



¿Te fijaste? Estamos usando bolitas para representar la hora de un reloj. Al programar, usamos las *abstracciones* que tenemos para *modelar* cosas del mundo real.

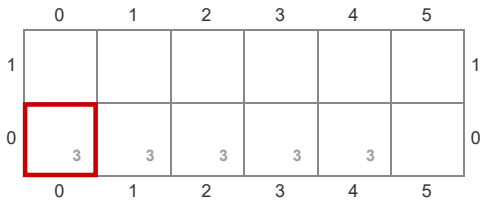
Y como siempre, es **muy importante** dividir el problema en subtareas. Y si puedo **no repetir código**, ¡Aún mejor!

Antes de pasar al siguiente ejercicio preguntate: *¿repetí código?*

Una línea heavy

El procedimiento `LineaEstePesada(peso, color, longitud)` debe dibujar hacia el `Este` una línea del `color` dado, poniendo en cada celda tantas bolitas como indique el `peso`. La linea debe ser tan larga como la `longitud`.

A modo de ejemplo, `LineaEstePesada(3, Verde, 5)` debería dibujar una línea verde, ocupando cinco celdas hacia el Este y poniendo tres bolitas en cada una de ellas:



Definí el procedimiento `LineaEstePesada(peso, color, longitud)`. Tené en cuenta que el cabezal **debe regresar a la posición inicial**. Para eso vas a tener que invocar `MoverN`.

Solución | Biblioteca

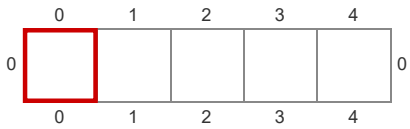
```
procedure LineaEstePesada(peso, color, longitud){
  repeat(longitud){
    PonerN(peso, color)
    Mover(Este)
  }
  MoverN(longitud, Oeste)
}
```

Enviar

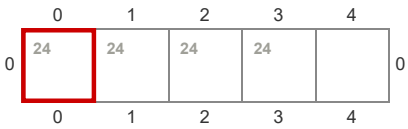
¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

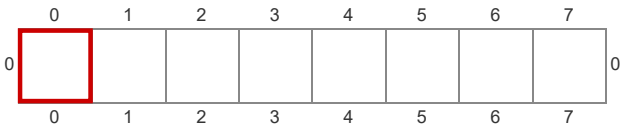
Tablero inicial



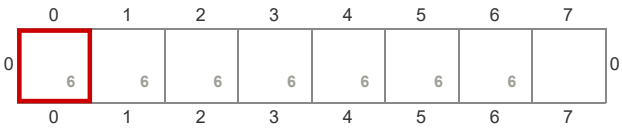
Tablero final



Tablero inicial



Tablero final



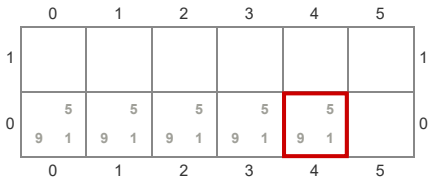
¿Viste que se pueden reusar `MoverN` y `PonerN` en varios lugares?

¡Son muy útiles!

Guarda con la guarda

Bueno, estamos en tiempo para algún ejercicio integrador...

> Definí un procedimiento `GuardaDe5()` , que haga una "guarda" de 5 azulejos (como las que decoran las paredes). Cada **azulejo** está conformado por 1 bolita verde, 5 negras y 9 rojas.



¡Dame una pista!

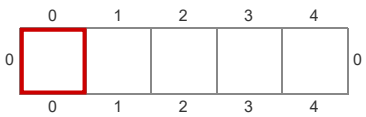
Solución | Biblioteca

```
procedure Guarda(){
  PonerN(9, Rojo)
  PonerN(5, Negro)
  Poner(Verde)
}


procedure GuardaDe5(){
  repeat(4){
    Guarda()
    Mover(Este)
  }
  Guarda()
}
```

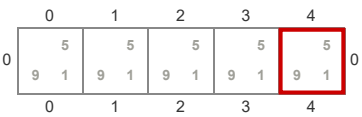
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



¡Bien! Recordaste cómo considerar el caso borde.

Además, en este ejercicio hay que dividir en subtareas para **evitar la repetición de código**. Esto es muy importante a la hora de programar. ¡Asegurate que tu solución no repita código!

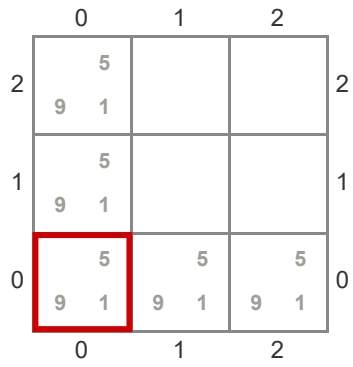
Información importante

Términos y Condiciones

Reglas del Espacio de Consultas

Una guarda en L

> Definí un procedimiento `GuardaEnL()` que haga una guarda en L como muestra la figura, pero **dejando el cabezal en la posición inicial**.



La ventaja ahora, es que **te regalamos** el procedimiento `PonerAzulejo`. ¡Pero **ojo** que necesitás dividir en más subtareas!

¡Dame una pista!

Solución | Biblioteca

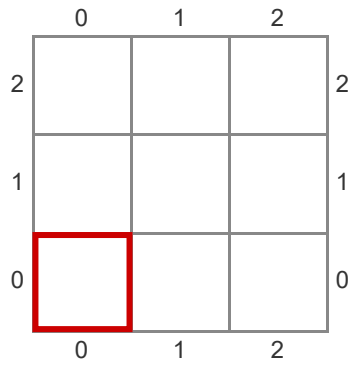
```
1  procedure AzulejoMover(direccion){
2    PonerAzulejo()
3    Mover(direccion)
4  }
5
6  procedure GuardaEnL(){
7    repeat(2){
8      AzulejoMover(Norte)
9    }
10   PonerAzulejo()
11   Mover(Este)
12   Mover(Sur)
13   Mover(Sur)
14   AzulejoMover(Este)
15   PonerAzulejo()
16   MoverN(2,Oeste)
```

```
17 }
18
19
20
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final