¿De qué se trata?

Tomate unos minutos (no más de 3) para tratar de descubrir qué es lo que hace el programa a continuación.

```
program {
  Poner(Negro)
  Mover(Este)
  Poner(Negro)
  Mover(Este)
  Poner(Negro)
  Mover(Norte)
  Poner(Negro)
  Mover(Oeste)
  Poner(Negro)
  Mover(Oeste)
  Poner(Negro)
  Mover(Norte)
  Poner(Negro)
  Mover(Este)
  Poner(Negro)
  Mover(Este)
  Poner(Negro)
}
```

¿Qué crees que hace?

- Llena cualquier tablero de bolitas negras.
- Dibuja una línea de bolitas negras.
- Dibuja una cruz de bolitas negras.
- Dibuja un cuadrado de bolitas negras.
- Pone 9 bolitas negras en una celda.

Cuando tengas tu respuesta pasá al siguiente ejercicio para ver si estás en lo correcto.

Un programa un poco largo

Ahora tenés la posibilidad de ver en acción el programa.

2. Un programa un poco largo

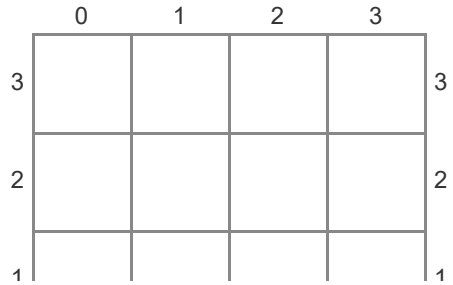
```
program {
  Poner(Negro)
  Mover(Este)
  Poner(Negro)
  Mover(Este)
  Poner(Negro)
  Mover(Norte)
  Poner(Negro)
  Mover(Oeste)
  Poner(Negro)
  Mover(Oeste)
  Poner(Negro)
  Mover(Norte)
  Poner(Negro)
  Mover(Este)
  Poner(Negro)
  Mover(Este)
  Poner(Negro)
}
```

¡Presioná Continuar para comprobar tu respuesta anterior!

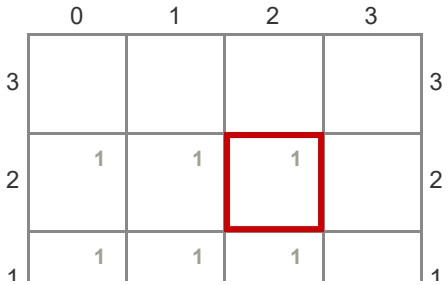
Continuar

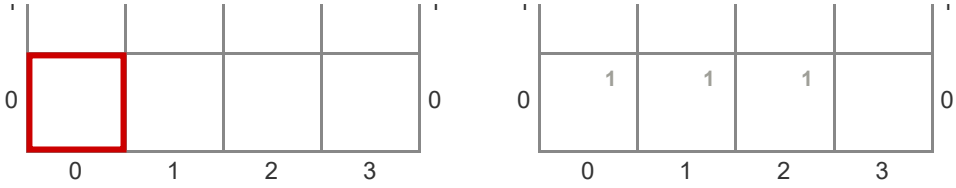
¡Muy bien!

Tablero inicial



Tablero final





Aunque ahora pudimos probarlo, sigue siendo un poco confuso saber de qué se trata el programa con solo leerlo ¿no sería mejor si **también** fuera fácil de entender para un humano?

Las cosas por su nombre

Mirá esta nueva versión del mismo programa. Aunque todavía hay elementos de la **sintaxis** que no conocés, confiamos en que vas a tardar mucho menos en descubrir qué hace.

> Enviá el código, así nos aseguramos de que **hace exactamente lo mismo** que el anterior.

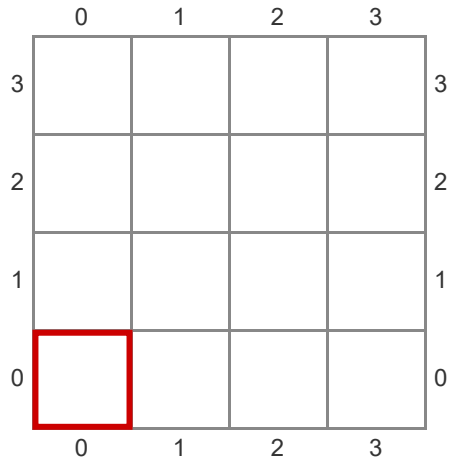
```
procedure DibujarCuadradoNegroDeLado3() {
  Poner(Negro)
  Mover(Este)
  Poner(Negro)
  Mover(Este)
  Poner(Negro)
  Mover(Norte)
  Poner(Negro)
  Mover(Oeste)
  Poner(Negro)
  Mover(Oeste)
  Poner(Negro)
  Mover(Norte)
  Poner(Negro)
  Mover(Este)
  Poner(Negro)
  Mover(Este)
  Poner(Negro)
}

program {
  DibujarCuadradoNegroDeLado3()
}
```

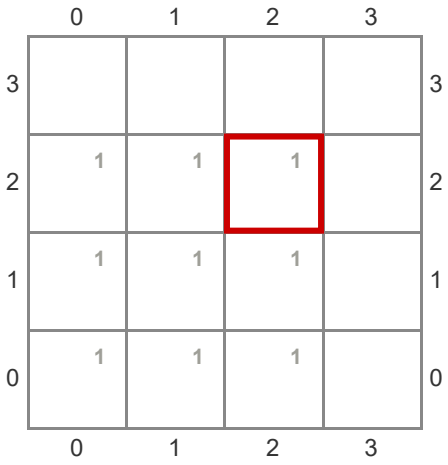
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



Mucho más fácil de entender, ¿no?

Probablemente te estés preguntando *¿cómo supo la computadora lo que tenía que hacer* `DibujarCuadradoNegroDeLado3` *? ¿Qué es eso de* `procedure` *?*

¡Vamos a averiguarlo!

Enseñándole tareas a la computadora

Como viste en el ejemplo del cuadrado, se puede empezar a diferenciar dos tipos de comandos dentro de un programa:

- los que **vienen definidos por el lenguaje** y nos sirven para expresar operaciones básicas, como `Mover`, `Poner` y `Sacar`. A estos los llamaremos **comandos primitivos**, o simplemente **primitivas**;
- y los que **definimos nosotros**, que nos sirven para expresar tareas más complejas. Como el nombre de esta lección sugiere, estos son los **procedimientos**.

Cuando *definimos* un procedimiento estamos "enseñándole" a la computadora a realizar una tarea nueva, que originalmente no estaba incluida en el lenguaje.

Prestale atención a la sintaxis del ejemplo para ver bien cómo definimos un procedimiento y cómo lo *invocamos* en un `program`.

```
procedure Poner3Rojas() {
    Poner(Rojo)
    Poner(Rojo)
    Poner(Rojo)
}

program {
    Poner3Rojas()
}
```

> ¿Qué te parece que hace el nuevo procedimiento? Copiá y enviá el código para ver qué pasa.

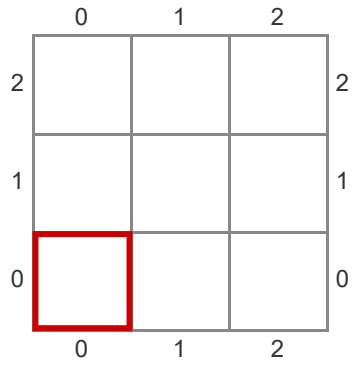
```
1 procedure Poner3Rojas() {
2     Poner(Rojo)
3     Poner(Rojo)
4     Poner(Rojo)
5 }
6
7 program {
```

```
8     Poner3Rojas()
9 }
```

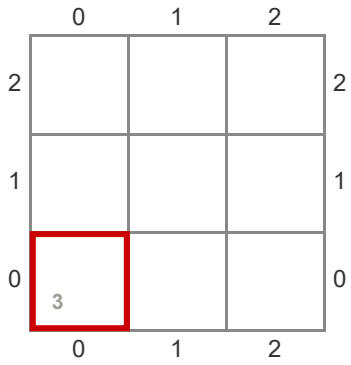
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



Ahora que ya probamos cómo funcionan, podemos ver las diferencias entre las sintaxis de **programas** y **procedimientos**.

El procedimiento se define con la palabra `procedure` seguida por un nombre y paréntesis `()`. Luego escribimos entre llaves `{}` todas las acciones que incluya. Para ver un procedimiento en acción hay que invocarlo dentro de un programa, si no sólo será una descripción que nunca se va a ejecutar.

El programa se crea con la palabra `program` seguida de llaves `{}`, y adentro de ellas lo que queremos que haga la computadora. ¡No lleva nombre ni paréntesis!

Información importante

Términos y Condiciones

Reglas del Espacio de Consultas

Procedimientos en acción

Si bien las palabras que utilizamos para crear programas (`program`) y definir procedimientos (`procedure`) se escriben parecido son cosas muy distintas.

Cuando creamos un programa nuevo le estamos diciendo a la computadora lo que queremos que suceda luego de tocar **Enviar**.

Pero si queremos crear una tarea nueva podemos agrupar las acciones que requiere en un procedimiento. Los procedimientos se definen con un **nombre** que describa lo que hace.

Veamos cómo creamos un nuevo procedimiento llamado PonerVerdeYAzul.

```
procedure PonerVerdeYAzul() {
    Poner(Verde)
    Poner(Azul)
}
```

La computadora solo va a seguir las instrucciones dentro de un procedimiento cuando sea **invocado** dentro de un `program` . ¿Cómo lo invocamos? Escribimos su nombre seguido por paréntesis `()` .

```
program {
    PonerVerdeYAzul()
}
```

> Completá el código para que además de **definir** el procedimiento `PonerNegroYRojo` luego lo **invoque** en el `program` .

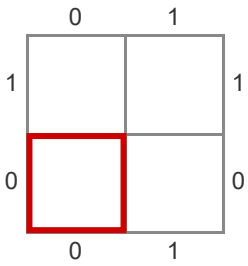
```
1 procedure PonerNegroYRojo() {
2   Poner(Negro)
3   Poner(Rojo)
4 }
5
6 program {
```

```
7   PonerNegroYRojo()
8 }
```

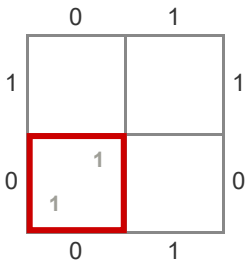
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



A la hora de **definir** e **invocar** procedimientos tenemos que prestar mucha atención a la sintaxis para no perder de vista el objetivo del problema por un error de escritura.

Escribiendo procedimientos

Llegó el momento de programar desde cero. ¡No te preocupes! Como recién estamos empezando, repasemos lo que aprendimos.

A esta altura ya sabemos que para programar siempre tenemos que tener en cuenta la sintaxis y que para definir nuevos procedimientos también tenemos reglas:

- empezamos con la palabra reservada `procedure` ;
- elegimos un nombre que lo describa y lo escribimos con mayúscula seguido de paréntesis `()` ;
- encerramos las acciones que queremos que haga entre llaves `{}` .

Entonces, un procedimiento que se mueve cuatro celdas al Norte se va a definir así:

```
procedure Mover4AlNorte() {
    Mover(Norte)
    Mover(Norte)
    Mover(Norte)
    Mover(Norte)
}
```

Y si lo queremos utilizar, tenemos que invocarlo dentro del `program` escribiendo su nombre tal cual y sin olvidar los paréntesis `()` ¡Prestá atención a la sintaxis!

```
program {
    Mover4AlNorte()
}
```

¡Ahora te toca a vos!

> **Definí** un procedimiento `Poner3Verdes` que ponga 3 bolitas verdes en la celda actual e **invocalo** en el `program` .

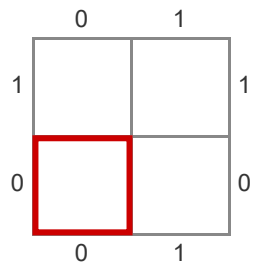
```
procedure Poner3Verdes() {
  Poner(Verde)
  Poner(Verde)
  Poner(Verde)
}

program {
  Poner3Verdes()
}
```

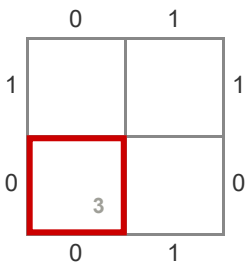
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



Resumiendo, en lo que a un procedimiento respecta, se pueden distinguir dos momentos:

- la **definición**, que es cuando ponemos `procedure Poner3Verdes()` y el bloque de código que especifica qué hace.
- el **uso** o **invocación**, que es cuando escribimos `Poner3Verdes()` en alguna parte de `program` (o de otro procedimiento).

Información importante

Términos y Condiciones

Reglas del Espacio de Consultas

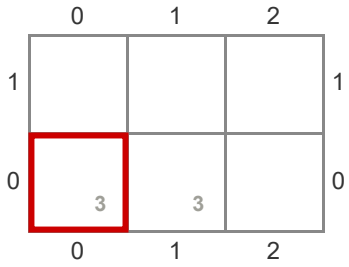
Procedimiento, ¡te invoco!

Algo *MUY* importante es que un procedimiento se define **una sola vez** y luego se puede usar **todas las veces que necesitemos**, como cualquier otro comando.

Por eso, su nombre debe ser **único** dentro de todo el programa. Recordá que la computadora no hace más que seguir órdenes y si existiera más de un procedimiento con el mismo nombre, no sabría cuál elegir.

Para facilitarte las cosas, a veces te ofreceremos partes del problema resuelto, para que sólo tengas que enfocarte en lo que falta resolver. ¿Y dónde están? Mirá el editor y fijate si encontrás la pestaña **Biblioteca** . Lo que aparece en la Biblioteca no hace falta que lo escribas en el código, ¡si está ahí podés invocarlo directamente!

¡Vamos a probarlo! Queremos poner 3 bolitas verdes en dos celdas consecutivas como muestra el tablero:



Creá un programa que lo haga invocando el procedimiento `Poner3Verdes` . Recordá que ya te lo damos definido ¡no tenés que volver a escribirlo!

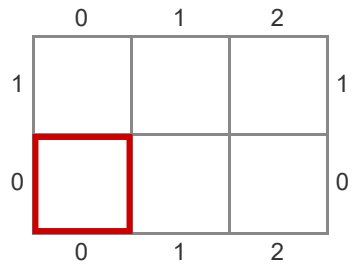
Solución | Biblioteca

```
program {
  Poner3Verdes()
  Mover(Este)
  Poner3Verdes()
  }
```

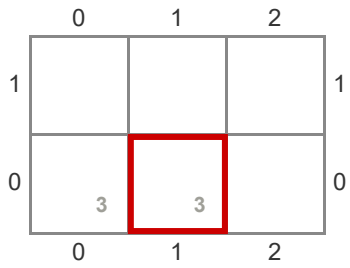
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



Repasemos:

- Para definir un procedimiento nuevo usamos `procedure` y le ponemos un nombre que describa lo que hace seguido por paréntesis `()`. A continuación y entre llaves `{}` lo que querramos que haga.
- Los procedimientos se definen **una sola vez**. Si ya están definidos en la Biblioteca o en alguna parte del código no hay que volver a hacerlo (no se pueden repetir los nombres, si se define un procedimiento más de una vez nuestro programa va a fallar).
- Para invocar un procedimiento escribimos su nombre (sin olvidar los paréntesis `()` al final). ¡Y podemos hacerlo todas las veces que sean necesarias!
- Aunque también se escribe entre llaves `{}`, el `program` **nunca** lleva nombre ni paréntesis.

Información importante

Términos y Condiciones

Reglas del Espacio de Consultas

Una definición, "infinitos" usos

Otro procedimiento que ya usamos antes es `Poner3Rojas` , que pone tres bolitas rojas en una celda. Te ahorramos el trabajo de escribirlo, si lo buscás vas a ver que está definido en la Biblioteca. ¡Ahora podés utilizarlo tantas veces como quieras!

10. Dibujamos con imagi

> Creá un programa que ponga 9 bolitas rojas en la celda actual invocando el procedimiento `Poner3Rojas` todas las veces que sea necesario.

¡Dame una pista!

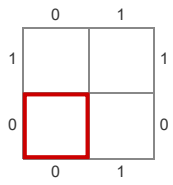
Solución | Biblioteca

```
program {
  Poner3Rojas()
  Poner3Rojas()
  Poner3Rojas()
  }
```

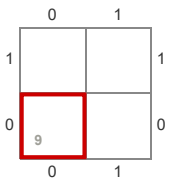
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



No te olvides de revisar las herramientas que nos ofrece la Biblioteca para saber cuáles podemos aprovechar cuando resolvamos nuestro problema.

Procedimientos dentro de otros

Cuando creamos procedimientos agrupamos varias acciones en una tarea que podemos reconocer y nombrar. Eso hace nuestros programas más claros, legibles y nos ahorra repeticiones innecesarias.

Ya vimos que un procedimiento puede ser invocado tantas veces como queramos dentro de un programa , pero como su objetivo es agrupar los pasos de una tarea para usarla cuando haga falta, también lo podemos invocar dentro de otros procedimientos. ¡Vamos a probarlo!

> **Definí** el procedimiento `Poner9Rojas` que, utilizando `Poner3Rojas` , ponga nueve bolitas rojas en una celda. Una vez definido, **invocá** el nuevo procedimiento en un `program` .

¡Dame una pista!

Solución | Biblioteca

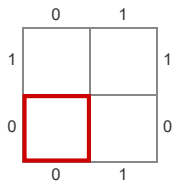
```
1 procedure Poner9Rojas() {
2   Poner3Rojas()
3   Poner3Rojas()
4   Poner3Rojas()
5 }
6 program {
7   Poner9Rojas()
8 }
```

Enviar

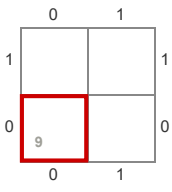
¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

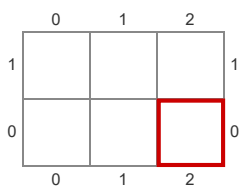
Tablero inicial



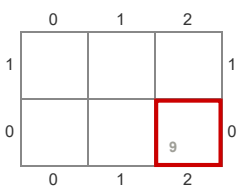
Tablero final



Tablero inicial



Tablero final



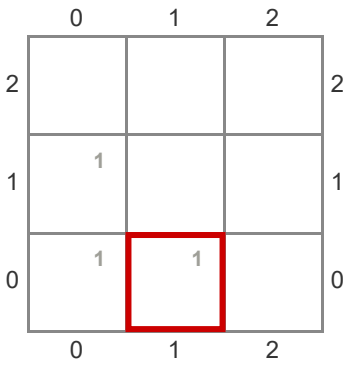
Bueno, ya sabemos cómo crear procedimientos, pero ¿por qué querríamos hacerlos?

Algunas posibles respuestas:

- para **simplificar código**, escribiendo una sola vez y en un solo lugar cosas que vamos a hacer muchas veces;
- para **escribir menos**, por qué querríamos hacer cosas de más;
- para que el propósito de nuestro programa **sea más entendible para los humanos**, como vimos en el ejemplo de `DibujarCuadradoNegroDeLado3` . Para esto es fundamental **pensar buenos nombres**, que no sean muy largos (`DibujarCuadradoNegroDeLado3FormadoPor9BolistasDeArribaAAbajo`), ni demasiado cortos (`DibCuaNeg`), y sobre todo que **dejen en claro qué hace nuestro procedimiento**;
- para comunicar la **estrategia** que pensamos para resolver nuestro **problema**;
- y como consecuencia de todo esto: para **poder escribir programas más poderosos**.

Dibujamos con imaginación

Vamos a usar un poco la imaginación y vamos a hacer un procedimiento que dibuje una "punta" negra en la esquina inferior izquierda de esta forma:



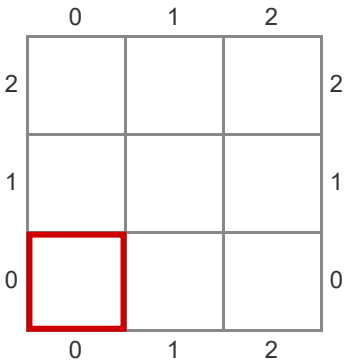
> Definí el procedimiento `DibujarPuntaNegra` e invocalo dentro de un `program`. El cabezal comienza en el origen y debe terminar en el extremo inferior derecho de la punta.

```
procedure DibujarPuntaNegra() {
  Poner(Negro)
  Mover(Norte)
  Poner(Negro)
  Mover(Este)
  Mover(Sur)
  Poner(Negro)
}
program {
  DibujarPuntaNegra()
}
```

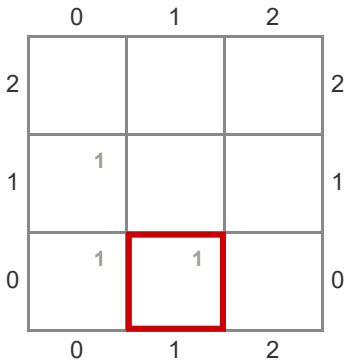
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial

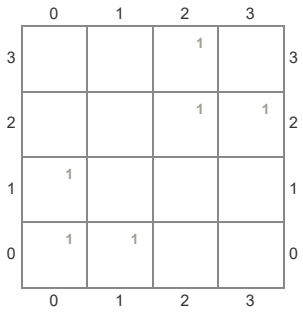


Tablero final



De punta a punta

En el ejercicio anterior ya dibujamos una punta, ahora vamos a pensar cómo aprovechar el procedimiento que hicimos para crear un tablero como este:



Definí el procedimiento `DibujarDosPuntas` e invocalo dentro un `program`. Acordate de utilizar `DibujarPuntaNegra`.

¡Dame una pista!

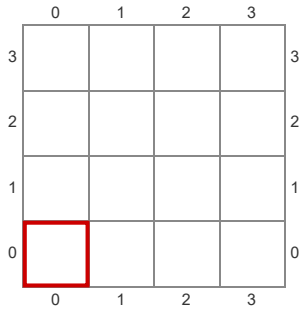
Solución | Biblioteca

```
procedure DibujarDosPuntas() {
  DibujarPuntaNegra()
  Mover(Norte)
  Mover(Norte)
  Mover(Este)
  DibujarPuntaNegra()
}
program {
  DibujarDosPuntas()
}
```

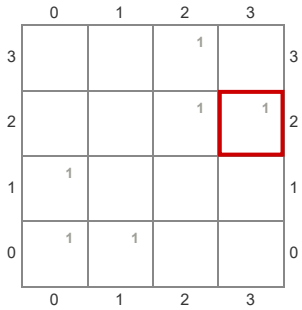
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



Para resolver este problema lo que hicimos fue separarlo en partes, identificando las tareas más pequeñas que ya teníamos resueltas.

Los procedimientos son muy útiles para esto, se ocupan de resolver una *subtarea* y nos permiten repetirla o combinarla para solucionar un problema mayor que la incluya.

Rojo al borde

Ya vimos que los comandos que vienen definidos por el lenguaje se llaman **primitivas**. Hay una primitiva que no usaste hasta ahora que queremos presentarte.

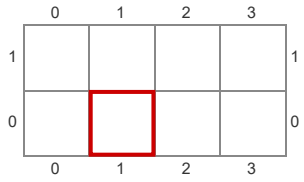
Imaginate que no sabés ni dónde está el cabezal ni qué tamaño tiene el tablero pero querés llegar a una esquina: La primitiva Mover no te va a ser de mucha ayuda.

Por suerte existe una primitiva llamada `IrAlBorde`, que toma una dirección, y se mueve todo lo que pueda en esa dirección, **hasta llegar al borde**.

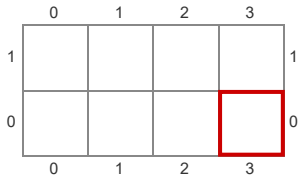
¿Cómo? Mirá el resultado del siguiente programa:

```
program {
    IrAlBorde(Este)
}
```

Inicial



Final



¡Vamos a aprovecharlo!

> Definí el procedimiento `RojoAlBorde` que ponga una bolita roja en la esquina superior izquierda del tablero e invocalo en el `program`.

¡Dame una pista!

```
1 procedure RojoAlBorde() {
2   IrAlBorde(Norte)
3   IrAlBorde(Oeste)
4   Poner(Rojo)
5 }
6 program {
7   RojoAlBorde()
8 }
```

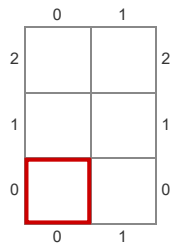
Enviar

¡Muy bien! Tu solución pasó todas las pruebas

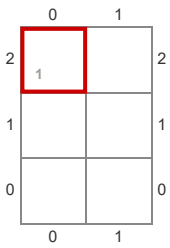
Resultados de las pruebas:

- Tablero de 2x3 con el cabezal abajo a la izquierda

Tablero inicial

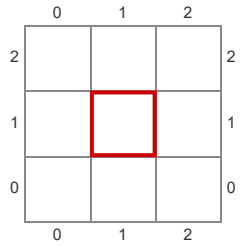


Tablero final

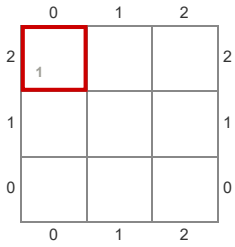


✅ Tablero de 3x3 con el cabezal en el medio

Tablero inicial



Tablero final



¡Excelente!

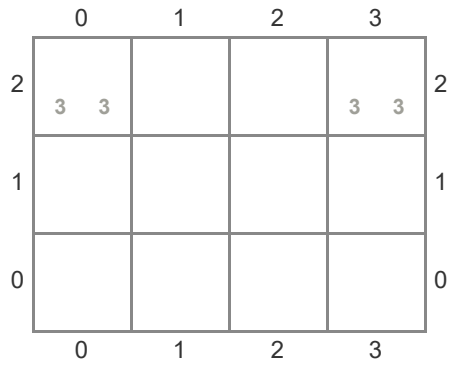
`IrAlBorde` es una primitiva muy útil para cuando no conocemos las condiciones de nuestro tablero.

Adornando el tablero

Para resolver un problema nos conviene comprender bien de qué se trata para elegir una estrategia. Es el momento de empezar a hacerlo aprovechando los procedimientos.

Uno de los objetivos al usar procedimientos es identificar y nombrar las subtareas que conforman un problema y combinar sus soluciones para poder resolverlo. Veamos un ejemplo:

Queremos decorar con guirnaldas las dos esquinas superiores de *cualquier* tablero como muestra la imagen.



Pensemos una estrategia distinguiendo subtareas:

Cada guirnalda se compone de 3 bolitas rojas y 3 bolitas verdes. Ya resolvimos cómo hacerlo en otros ejercicios , hacer una guirnalda solo requerirá combinar esas soluciones. Y ponerla donde corresponda, claro.

¿Y que más? el procedimiento que decore el tablero debería poder aprovechar la creación de una guirnalda para usarla varias veces en las posiciones que querramos decorar. Nos vendría muy bien alguna primitiva que nos ayude a llegar a los bordes.

¡Manos a la obra!

> Definí dos procedimientos: el procedimiento `PonerGuirnalda` que coloque 3 bolitas rojas y 3 bolitas verdes en una celda y el procedimiento `DecorarTablero` que lo utilice y ponga una

guirnalda en cada esquina superior. Invocá `DecorarTablero` en el `program`. Tené en cuenta que no sabemos la posición inicial donde se encontrará el cabezal.

¡Dame una pista!

Solución | Biblioteca

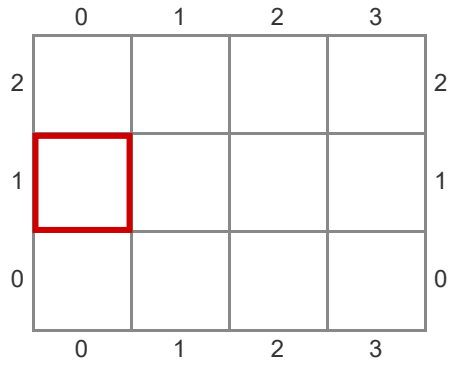
```
1  procedure PonerGuirnalda() {
2    Poner3Verdes()
3    Poner3Rojas()
4  }
5
6  procedure DecorarTablero() {
7    IrAlBorde(Norte)
8    IrAlBorde(Este)
9    PonerGuirnalda()
10   IrAlBorde(Oeste)
11   PonerGuirnalda()
12 }
13
14 program {
15   DecorarTablero()
16 }
```

Enviar

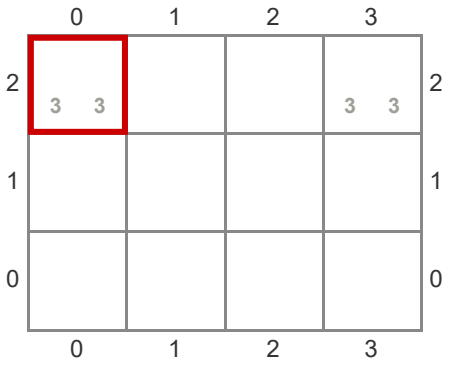
¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

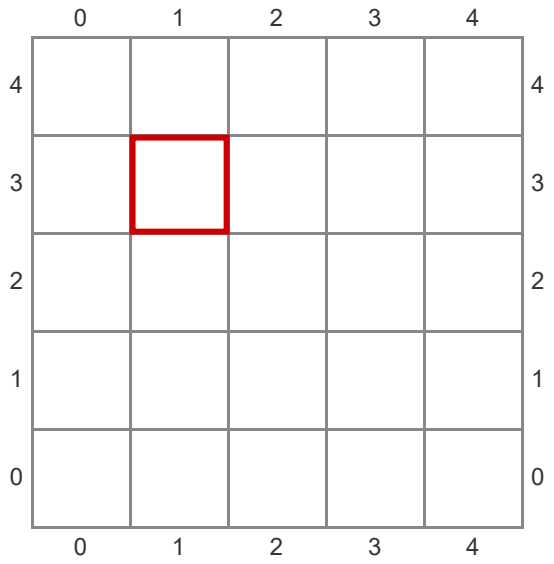
Tablero inicial



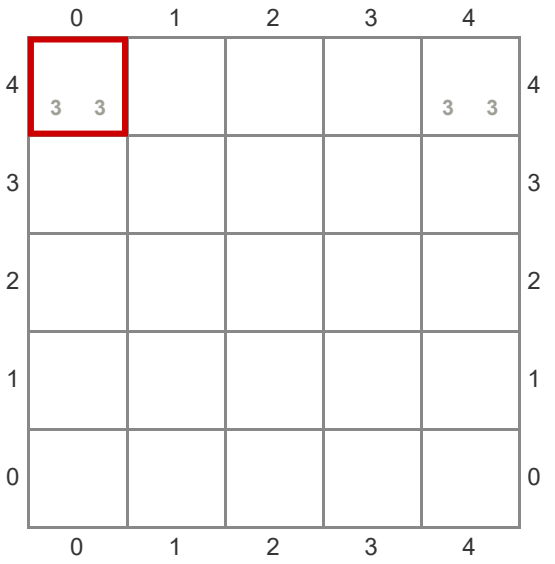
Tablero final



Tablero inicial



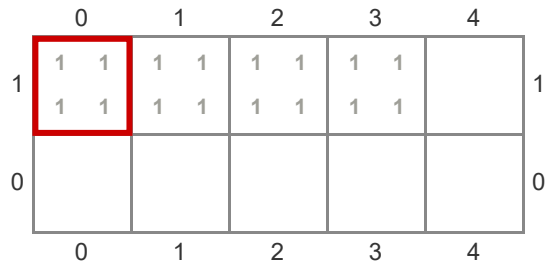
Tablero final



Cuanto más complejo sea el problema, más útil nos va a ser pensar una estrategia y organizar la solución en subtareas ¡y los procedimientos están para ayudarnos!

Colores, colores, colores

Vamos a darle un poco más de color a todo esto haciendo líneas multicolores como esta:



Como se ve en la imagen, cada celda de la línea debe tener una bolita de cada color (una roja, una negra, una verde y una azul).

¿Cómo podemos dibujarla? ¿Cuál es la tarea que se repite? ¿Se puede definir un nuevo procedimiento para resolverla y aprovecharlo para construir nuestra solución?

> Definí un procedimiento `DibujarLineaColorida` que **dibuje una línea multicolor** de cuatro celdas hacia el `Este` y al finalizarla ubique el cabezal en la celda inicial. Tené en cuenta que siempre partimos del extremo `Oeste`. Invocá el nuevo procedimiento en un `program`.

¡Dame una pista!

```
1  procedure PonerColores() {
2    Poner(Verde)
3    Poner(Rojo)
4    Poner(Negro)
5    Poner(Azul)
6  }
7
8  procedure DibujarLineaColorida() {
9    IrAlBorde(Oeste)
10   PonerColores()
11   Mover(Este)
12   PonerColores()
13   Mover(Este)
14   PonerColores()
```

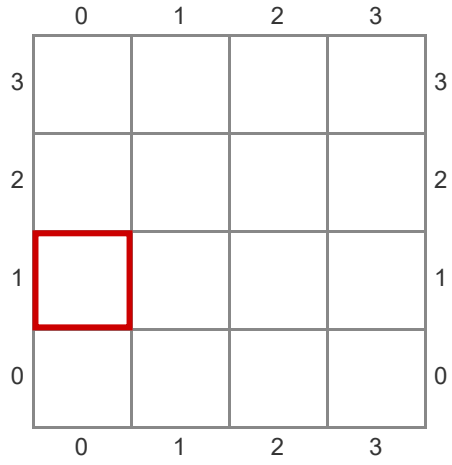
```
15    Mover(Este)
16    PonerColores()
17    IrAlBorde(Oeste)
18  }
19
20  program {
21    DibujarLineaColorida()
22  }
```

Enviar

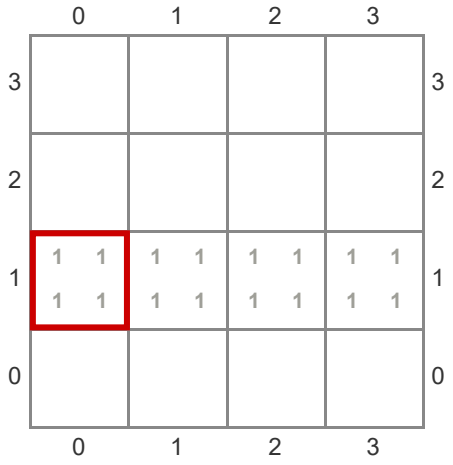
¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

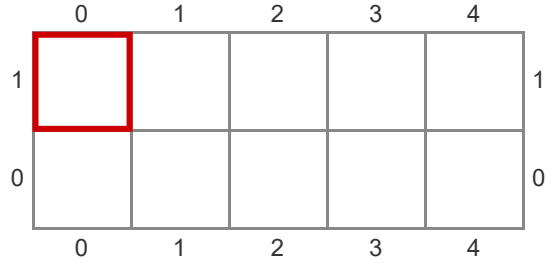
Tablero inicial



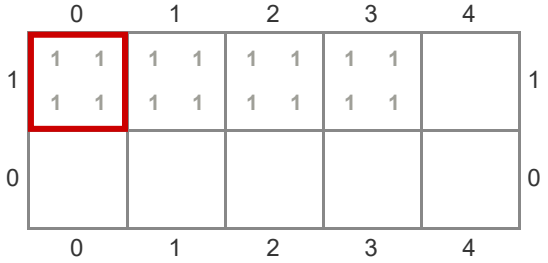
Tablero final



Tablero inicial

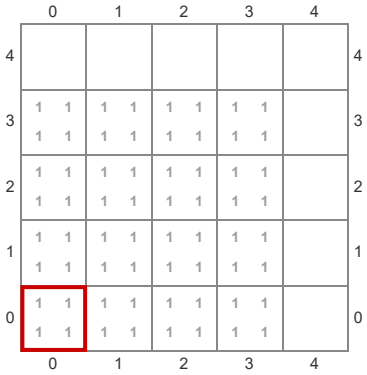


Tablero final



Cuadrado de colores

Vamos a crear un procedimiento que nos permita dibujar un tablero como este:



Definí un procedimiento `DibujarCuadradoColorido` que dibuje un cuadrado de 4×4 celdas en el que cada celda tenga una bolita de cada color e invocalo en el `program`. El cabezal debe quedar en la celda inicial.

¡Dame una pista!

Solución | Biblioteca

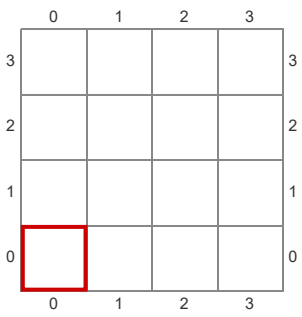
```
procedure DibujarCuadradoColorido() {
  DibujarLineaColorida()
  Mover(Norte)
  DibujarLineaColorida()
  Mover(Norte)
  DibujarLineaColorida()
  Mover(Norte)
  DibujarLineaColorida()
  Mover(Sur)
  Mover(Sur)
  Mover(Sur)
}

program {
  DibujarCuadradoColorido()
}
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final