



MoverOeste10

Entremos en calor: definí un procedimiento `MoverOeste10` que mueva el cabezal 10 veces hacia el Oeste.

💡 ¡Dame una pista!

```
1 procedure MoverOeste10() {  
2   Mover(Oeste)  
3   Mover(Oeste)  
4   Mover(Oeste)  
5   Mover(Oeste)  
6   Mover(Oeste)  
7   Mover(Oeste)  
8   Mover(Oeste)  
9   Mover(Oeste)  
10  Mover(Oeste)  
11  Mover(Oeste)  
12 }
```

▶ Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial

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

Tablero final

	0	1	2	3	4	5	6	7	8	
1										1
0										0
	0	1	2	3	4	5	6	7	8	

¿Te imaginás cómo hacer lo mismo pero 20, 100 o 5000 veces? Sería bastante molesto, ¿no?

Evidentemente tiene que haber una forma mejor de hacerlo, si no eso de "automatizar tareas repetitivas" sería una mentira. Y bueno, de hecho la hay: ¡vayamos al siguiente ejercicio!

Esta guía fue desarrollada por Federico Aloí, Gustavo Trucco, Daniela Villani, Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)



La computadora repite por nosotros

Como te adelantamos en el ejercicio anterior, en Gobstones existe una forma de decir "quiero que **estos comandos se repitan esta cantidad de veces**".

Entonces, cuando es necesario repetir un comando (como `Mover`, `Poner`, `DibujarLineaNegra`, etc) un cierto número de veces, en lugar de copiar y pegar como veníamos haciendo hasta ahora, podemos utilizar la sentencia `repeat`.

Sabiendo esto, así es como quedaría `MoverOeste10` usando `repeat`:

```
procedure MoverOeste10() {
  repeat(10) {
    Mover(Oeste)
  }
}
```

Pero no tenés por qué creernos: ¡escribí este código en el editor y fijate si funciona!

```
1 procedure MoverOeste10() {
2   repeat(10) {
3     Mover(Oeste)
4   }
5 }
```

▶ Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial

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

Tablero final

	0	1	2	3	4	5	6	7	8	
1										
0										
	0	1	2	3	4	5	6	7	8	

Ahora sí se empieza a poner interesante esto de la programación.

Esta guía fue desarrollada por Federico Aloí, Gustavo Trucco, Daniela Villani, Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





MoverOeste5 usando repeat

Llegó tu turno de nuevo: definí un procedimiento `MoverOeste5` que se mueva 5 veces al Oeste.

Obvio, esta vez tenés que usar `repeat`.

💡 ¡Dame una pista!

```
1 procedure MoverOeste5() {  
2   repeat(5) {  
3     Mover(Oeste)  
4   }  
5 }
```

▶ Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial

	0	1	2	3	4	5	
1							1
0							0
	0	1	2	3	4	5	

Tablero final

	0	1	2	3	4	5	
1							1
0							0
	0	1	2	3	4	5	

Como ya descubriste, el comando `repeat` consta básicamente de dos elementos:

Un **número entero** (o sea, sin decimales), que indica cuántas veces hay que repetir. Este número va entre paréntesis (`()`) luego de la palabra `repeat` .

Y un **bloque de código**, que va encerrado entre llaves (`{ }`) y especifica qué comandos se quieren repetir. Es *MUY* importante que no te los olvides, porque sino la computadora no va a saber qué es lo que quisiste repetir (y fallará).

Esta guía fue desarrollada por Federico Aloï, Gustavo Trucco, Daniela Villani, Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





No todo es repetir

Los ejemplos que hiciste en los ejercicios anteriores se solucionaban simplemente repitiendo cosas. Pero no todo es repetir, también podemos poner comandos tanto antes como después del `repeat`, al igual que veníamos haciendo hasta ahora.

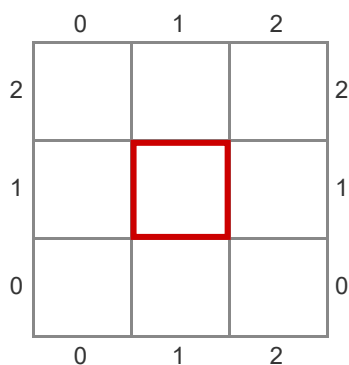
Por ejemplo, este es un programa que se mueve al `Sur`, luego pone 4 bolitas de color `Rojo` y por último vuelve a moverse al `Norte`:

```
program {
  Mover(Sur)
  repeat(4) {
    Poner(Rojo)
  }
  Mover(Norte)
}
```

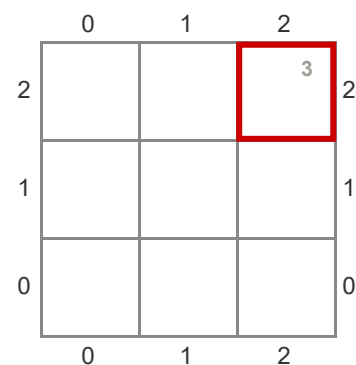
Fijate que `Mover(Sur)` lo pusimos **antes** del `repeat` y `Mover(Norte)` lo pusimos **después**. Por lo tanto cada movimiento se ejecuta solo una vez. Teniendo en cuenta esto:

Definí el procedimiento `Poner3AlNoreste()`, que ponga 3 bolitas negras en la primera celda al Noreste del cabezal.

Inicial



Final



💡 ¡Dame una pista!

```
1 procedure Poner3AlNoreste() {  
2   Mover(Norte)  
3   Mover(Este)  
4   repeat(3) {  
5     Poner(Negro)  
6   }  
7 }
```



▶ Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial

	0	1	2	3
3				
2				
1				
0				
	0	1	2	3

Tablero final

	0	1	2	3
3				
2				
1		3		
0				
	0	1	2	3

¿Viste qué importante es definir bien qué comandos hay que repetir y cuáles no?

Es muy común, al principio, olvidarse de colocar las llaves o incluso pensar que no son importantes. Pero tené mucho cuidado: poner las llaves en el lugar erróneo puede cambiar por completo lo que hace tu programa. Mirá qué distinto sería el resultado si hubieras puesto el `Mover(Este)` adentro del `repeat` :

```
procedure Poner3AlNoreste() {  
  Mover(Norte)  
  
  repeat(3) {
```



```
Mover(Este)
Poner(Negro)
}
}
```

	0	1	2	3	
3					3
2					2
1		1	1	1	1
0					0
	0	1	2	3	

Esta guía fue desarrollada por Federico Aloí, Gustavo Trucco, Daniela Villani, Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

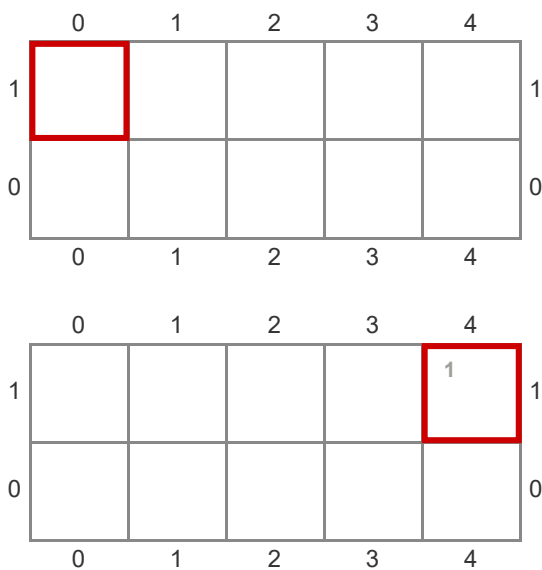
[Reglas del Espacio de Consultas](#)





También vale después

Definí el procedimiento `PonerAzulLejos`, que coloque una bolita Azul 4 celdas hacia el Este:



💡 ¡Dame una pista!

```
1 procedure PonerAzulLejos() {  
2   repeat(4) {  
3     Mover(Este)  
4   }  
5   Poner(Azul)  
6 }
```



▶ Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial

	0	1	2	3	4	
1						1
0						0
	0	1	2	3	4	

Tablero final

	0	1	2	3	4	
1					1	1
0						0
	0	1	2	3	4	

Como ya experimentaste, pueden ponerse comandos tanto antes como después del `repeat` .
En definitiva... ¡es sólo un comando más!

Esta guía fue desarrollada por Federico Aloj, Gustavo Trucco, Daniela Villani, Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





Repitiendo varios comandos

Hasta el momento los ejemplos que vimos sólo repetían un comando, pero como mencionamos al comenzar es posible repetir cualquier **secuencia de comandos** - en definitiva lo que se repite es un **bloque** y, como ya sabíamos, en un bloque puede haber tantos comandos como se nos ocurra.

Miremos el código de `DibujarLineaNegra6` que podríamos haber hecho sin usar `repeat`, con algunos espacios en blanco para ayudarnos a reconocer la secuencia que se repite:

```
procedure DibujarLineaNegra6() {  
  Poner(Negro)  
  Mover(Este)  
  
  Poner(Negro)  
  Mover(Este)  
  
  Poner(Negro)  
  Mover(Este)  
  
  Poner(Negro)  
  Mover(Este)  
  
  Poner(Negro)  
  Mover(Este)  
  
  Poner(Negro)  
  Mover(Este)  
}
```

¿Notás qué es lo que se repite y cuántas veces? Bueno, eso es lo que tenés que poner en el `repeat`.

Definí una versión superadora de `DibujarLineaNegra6`, esta vez usando `repeat`.

💡 ¡Dame una pista!

```
1 procedure DibujarLineaNegra6() {  
2   repeat(6) {  
3     Poner(Negro)  
4     Mover(Este)  
5   }  
6 }
```

▶ Enviar

✔ ¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial

	0	1	2	3	4	5	6	
1								1
0								0
	0	1	2	3	4	5	6	

Tablero final

	0	1	2	3	4	5	6	
1								1
0	1	1	1	1	1	1		0
	0	1	2	3	4	5	6	

Esta guía fue desarrollada por Federico Aloï, Gustavo Trucco, Daniela Villani, Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)





¿Dónde está el error?



Esta solución para LineaRoja4 no resuelve el problema como esperábamos, mirá:

Tablero inicial

	0	1	
4			4
3			3
2			2
1			1
0			0
	0	1	

Lo que hace

	0	1	
4	1		4
3	1		3
2	1		2
1	1		1
0			0
	0	1	

Lo que esperábamos

	0	1	
4			4
3	1		3
2	1		2
1	1		1
0	1		0
	0	1	

¿Nos ayudás a corregirla? Te dejamos el código en el editor.

💡 ¡Dame una pista!

```
1 procedure LineaRoja4() {
2   repeat(4) {
3     Poner(Rojo)
4     Mover(Norte)
5   }
6 }
```



 Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial

	0	1	
4			4
3			3
2			2
1			1
0			0
	0	1	

Tablero final

	0	1	
4			4
3	1		3
2	1		2
1	1		1
0	1		0
	0	1	

¡Bien, corregiste el error! No fue tan fácil de encontrar, ¿no? Ahora ya sabemos que el orden dentro de un `repeat` también importa, y mucho.

Esta guía fue desarrollada por Federico Aloí, Gustavo Trucco, Daniela Villani, Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

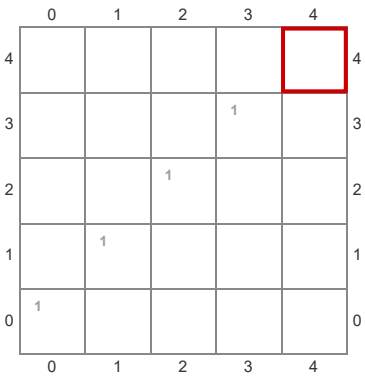
[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)



Diagonal con una bolita

Definí un procedimiento `Diagonal4Azul` que dibuje una diagonal de longitud 4 hacia el Noreste, donde cada celda tenga una bolita azul. El cabezal de la diagonal quedará donde muestra la imagen.

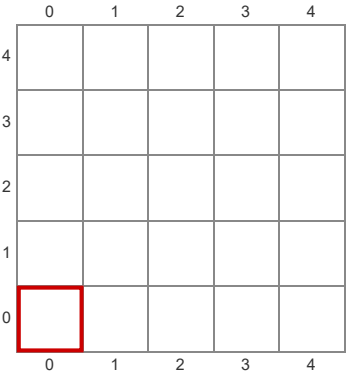


```
1 procedure Diagonal4Azul() {
2   repeat(4) {
3     Poner(Azul)
4     Mover(Norte)
5     Mover(Este)
6   }
7 }
```

▶ Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

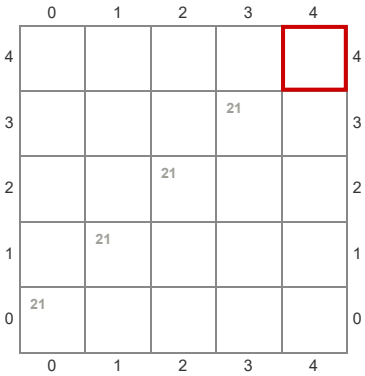
[Reglas del Espacio de Consultas](#)



Diagonal "pesada"

Ahora vamos a hacer lo mismo, pero en versión "pesada".

¿Qué quiere decir esto? Que en vez de poner 1 bolita en cada celda, ahora hay que poner 21. Mirá la imagen:



Definí un procedimiento `DiagonalPesada4Azul` que resuelva el problema.

💡 ¡Dame una pista!

```
1 procedure Poner21Azul() {
2   repeat(21) {
3     Poner(Azul)
4   }
5 }
6
7 procedure DiagonalPesada4Azul() {
8   repeat(4) {
9     Poner21Azul()
10    Mover(Norte)
11    Mover(Este)
12  }
13 }
```

▶ Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

- ✓ Con el cabezal en el origen

Tablero inicial

	0	1	2	3	4
4					
3					
2					
1					
0					
	0	1	2	3	4

Tablero final

	0	1	2	3	4
4					
3				21	
2			21		
1		21			
0	21				
	0	1	2	3	4

Con el cabezal desplazado

Tablero inicial

	0	1	2	3	4	5
4						
3						
2						
1						
0						
	0	1	2	3	4	5

Tablero final

	0	1	2	3	4	5
4						
3					21	
2				21		
1			21			
0		21				
	0	1	2	3	4	5

Muy bien. Aunque el `repeat` es poderoso y nos ayuda a escribir menos código, sigue siendo igual de importante la división en subtareas.

¡No vale olvidarse de lo aprendido hasta ahora!

Esta guía fue desarrollada por Federico Aloí, Gustavo Trucco, Daniela Villani, Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL
[Información importante](#)
[Términos y Condiciones](#)
[Reglas del Espacio de Consultas](#)





El caso borde

Muchas veces cuando usamos un `repeat` nos encontramos con que el último caso es levemente distinto a los anteriores, situación que solemos llamar **caso borde**. Pero mejor, veamos un ejemplo.

El procedimiento `LineaNegra4Este` que te presentamos dibuja una línea negra hacia el Este dejando el cabezal **fuera de la línea**, una celda hacia el Este.

```
procedure LineaNegra4Este() {  
  repeat(4) {  
    Poner(Negro)  
    Mover(Este)  
  }  
}
```

Si ahora queremos hacer que deje el cabezal en la última celda de la línea, tenemos dos opciones:

- **Mover el cabezal al Oeste luego de dibujar la línea.** Un truco medio feo, porque para funcionar necesita que haya al menos 5 espacios al Este de la posición inicial, cuando nuestra línea sólo ocupará 4.
- **Tratar el último caso de manera especial.** Esta opción es más interesante y más fiel a lo que queremos hacer: la última vez no queremos que el cabezal se mueva, simplemente nos basta con poner la bolita negra.

	0	1	2	3	
1					1
0	1	1	1	1	0
	0	1	2	3	

Teniendo en cuenta esto último, definí una nueva versión de `LineaNegra4Este` que deje el cabezal en la última celda de la línea.

💡 ¡Dame una pista!

```
1 procedure LineaNegra4Este() {  
2   repeat(3) {  
3     Poner(Negro)  
4     Mover(Este)  
5   }  
6   Poner(Negro)  
7 }
```

▶ Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

✅ Con lugares de sobra

Tablero inicial

	0	1	2	3	4	
1						1
0						0
	0	1	2	3	4	

Tablero final

	0	1	2	3	4	
1						1
0	1	1	1	1		0
	0	1	2	3	4	

✅ Con el espacio justo

Tablero inicial

	0	1	2	3	
1					1
0					0
	0	1	2	3	

Tablero final

	0	1	2	3	
1					1
0	1	1	1	1	0
	0	1	2	3	

Siempre que tengas problemas como este vas a poder solucionarlos de la misma manera: **procesando el último caso por separado.**

Otra variante menos común, y tal vez más difícil de construir también, es la de procesar el **primer** caso aparte:

```
procedure LineaNegra4Este() {
  Poner(Negro)
  repeat(3) {
    Mover(Este)
    Poner(Negro)
  }
}
```

Por convención, vamos a preferir la forma que procesa distinto al último caso, aunque a menudo ambas sean equivalentes (es decir, produzcan el mismo resultado).

Esta guía fue desarrollada por Federico Aloí, Gustavo Trucco, Daniela Villani, Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](https://creativecommons.org/licenses/by/4.0/).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

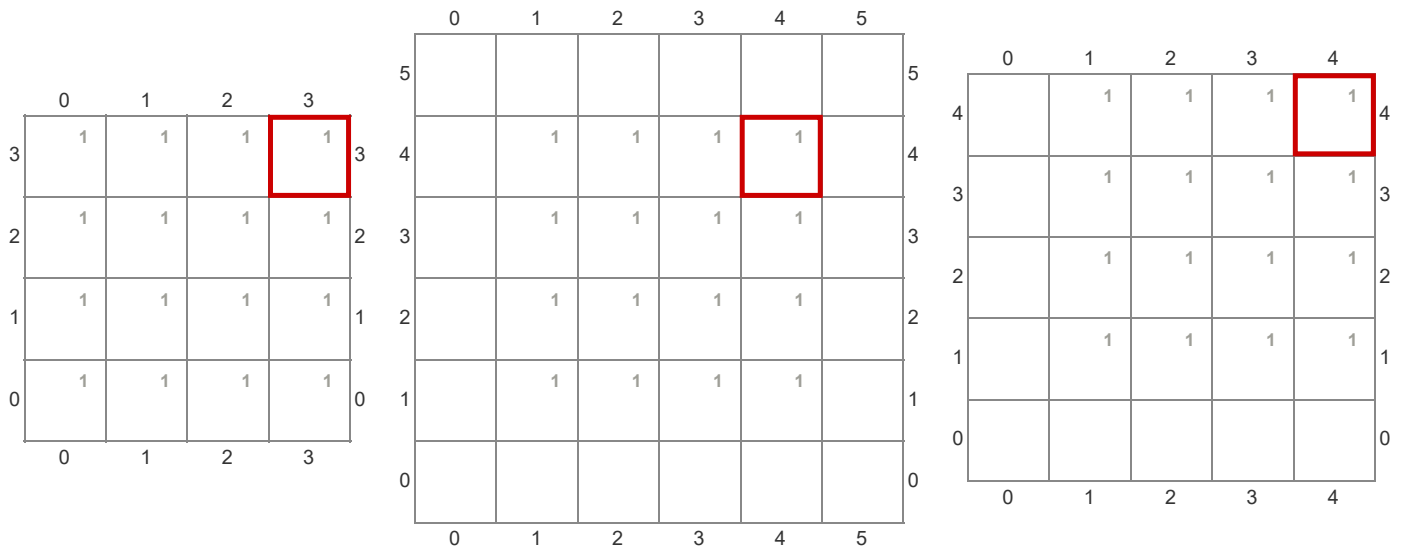
[Reglas del Espacio de Consultas](#)





De lado a lado, dibujamos un cuadrado

En el ejercicio anterior definimos el procedimiento [LineaNegra4Este](#). Ahora vamos a utilizarlo para dibujar un cuadrado negro igualito a los de los tableros de ejemplo:



Definí el procedimiento [CuadradoNegro4](#) para dibujar un cuadrado de 4x4 con bolitas negras. Al empezar, el cabezal se encuentra en la esquina inferior izquierda del cuadrado (no necesariamente del tablero) y cuando termine el programa el cabezal deberá quedar en el extremo superior derecho del cuadrado. No te olvides de invocar [LineaNegra4Este](#).

Tené en cuenta lo que hablamos en el ejercicio anterior sobre el **caso borde**.

💡 ¡Dame una pista!

Solución

Biblioteca

```

1 procedure CuadradoNegro4() {
2   repeat(3) {
3     LineaNegra4Este()
4     Mover(Norte)
5     repeat(3) {
6       Mover(Oeste)
7     }
8   }
9   LineaNegra4Este()
10 }
11 }
12
13
```



Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

✓ Con lugares de sobra

Tablero inicial

	0	1	2	3	
4					4
3					3
2					2
1					1
0					0
	0	1	2	3	

Tablero final

	0	1	2	3	
4					4
3	1	1	1	1	3
2	1	1	1	1	2
1	1	1	1	1	1
0	1	1	1	1	0
	0	1	2	3	

✓ Con el espacio justo

Tablero inicial

	0	1	2	3	4	
3						3
2						2
1						1
0						0
	0	1	2	3	4	

Tablero final

	0	1	2	3	4	
3		1	1	1	1	3
2		1	1	1	1	2
1		1	1	1	1	1
0		1	1	1	1	0
	0	1	2	3	4	

Esta guía fue desarrollada por Federico Aloï, Gustavo Trucco, Daniela Villani, Franco Bulgarelli bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).

© 2015-2022 Ikumi SRL

[Información importante](#)

[Términos y Condiciones](#)

[Reglas del Espacio de Consultas](#)

