



## Ejercicios examen sé programar

Lenguajes de Programación (Universidad de Buenos Aires)



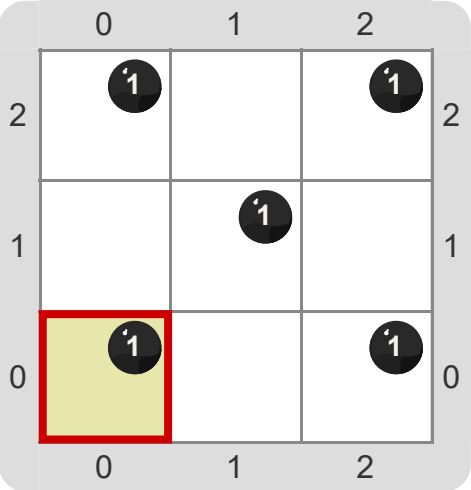
🕒 Te quedan 00:02:42

# Ejercicio 1: Ejercicio 1



Es bastante sabido que para recordar dónde se esconde un tesoro hay que marcar el lugar.

Una clásica opción para esto es utilizar una cruz , que en un tablero podría verse así:



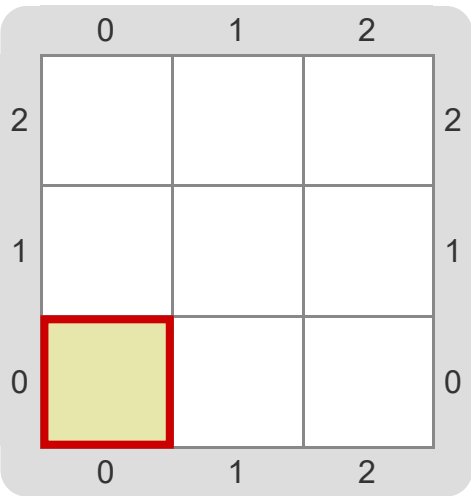
Creá un programa que dibuje una cruz de color Negro . El cabezal empieza en el origen (o sea, en el borde Sur-Oeste) pero no te preocupes por dónde finaliza.

```
1 program {
2   Poner(Negro)
3   repeat(2) {
4     Mover(Norte)
5   }
6   Poner(Negro)
7   repeat(2) {
8     Mover(Este)
9   }
10  Poner(Negro)
11  Mover(Sur)
12  Mover(Oeste)
13  Poner(Negro)
14  Mover(Este)
15  Mover(Sur)
16  Poner(Negro)
17 }
```

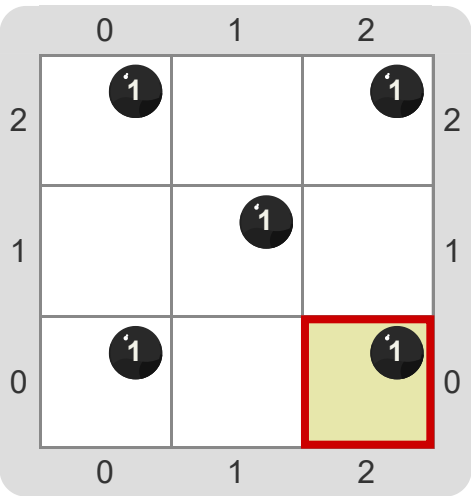
▶ Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



Siguiente Ejercicio: Ejercicio 7 >

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



🕒 Te quedan 00:02:47

## Ejercicio 2: Ejercicio 2



La verdad es que en el ejercicio anterior hicimos una cruz de un color específico porque es lo que solemos ver en películas o libros pero ¿qué nos impide que hagamos una cruz de cualquier color para marcar un lugar?

Definí el procedimiento `DibujarCruz` para que dibuje una cruz con el `color` que reciba por parámetro. No te preocupes por donde termina el cabezal.

```
1 procedure DibujarCruz(color) {
2   Poner(color)
3   repeat(2) {
4     Mover(Norte)
5   }
6   Poner(color)
7   repeat(2) {
8     Mover(Este)
9   }
10  Poner(color)
11  Mover(Sur)
12  Mover(Oeste)
13  Poner(color)
14  Mover(Este)
15  Mover(Sur)
16  Poner(color)
17 }
```



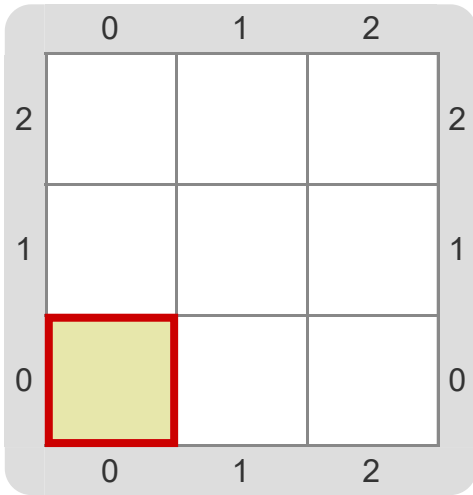
▶ Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

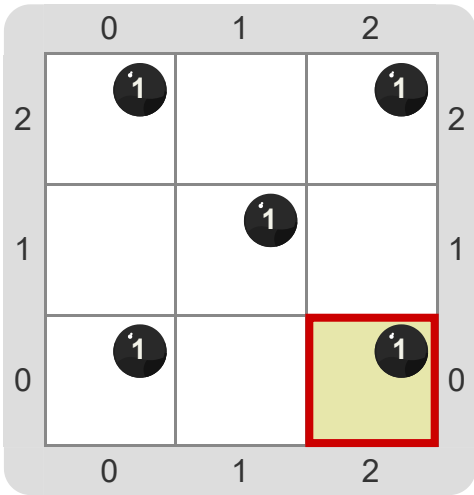
Resultados de las pruebas:



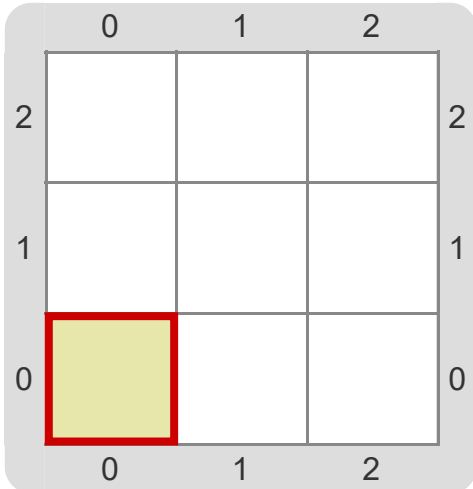
Tablero inicial



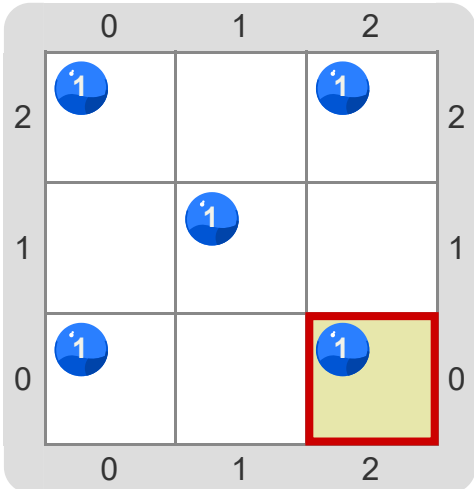
Tablero final



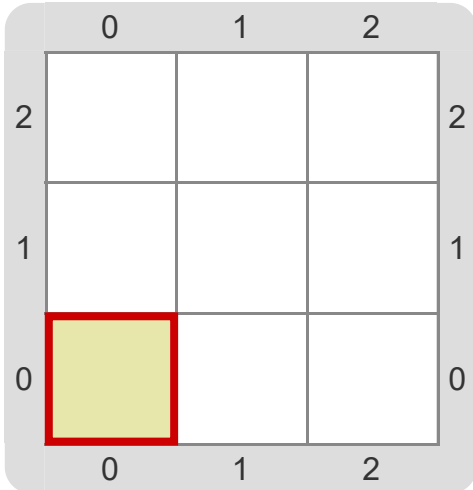
Tablero inicial



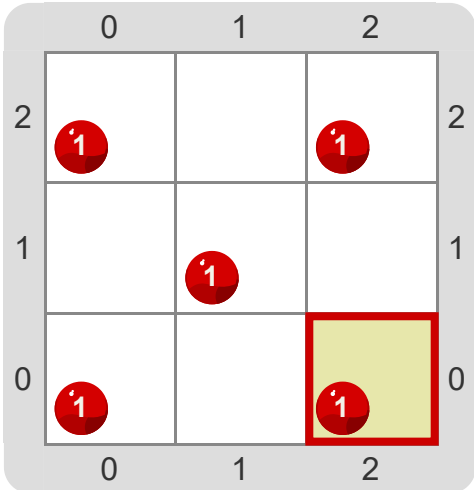
Tablero final



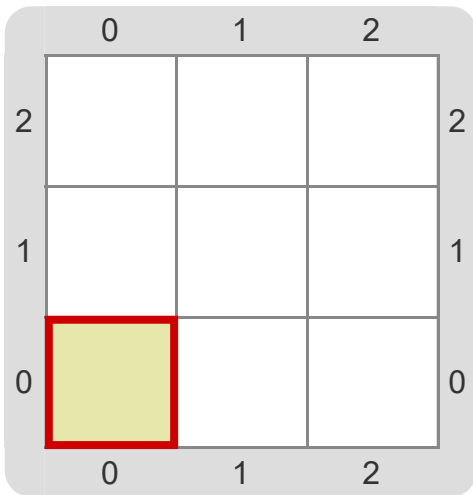
Tablero inicial



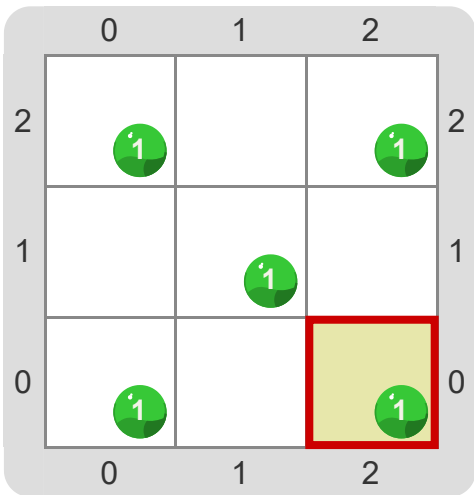
Tablero final



Tablero inicial



Tablero final



Siguiente Ejercicio: Ejercicio 7 >





🕒 Te quedan 00:02:53

# Ejercicio 3: Ejercicio 3

JS

Dejemos atrás los tableros y... ¡Pasemos a JavaScript!

A veces la matemática puede ser un poco tediosa . La buena noticia es que ahora podemos crear funciones que nos ayuden a resolver estos problemas.

Para eso vamos a crear una función que reciba 3 números y nos diga si la resta entre los 2 primeros es mayor al tercero. Por ejemplo:

```
> laRestaEsMayor(4, 2, 8)
false //Porque 4 menos 2 es 2 y es menor a 8

> laRestaEsMayor(12, 3, 5)
true //Porque 12 menos 3 es 9 y es mayor a 5
```

Definí la función `laRestaEsMayor`.

✎ Solución

>\_ Consola

```
1 function laRestaEsMayor(n1, n2, n3) {
2   return n1-n2 > n3;
3 }
```

▶ Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Siguiente Ejercicio: Ejercicio 7 >

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





⌚ Te quedan 00:02:58

# Ejercicio 4: Ejercicio 4

JS

Ahora vamos a hacer una función un poco particular.

Queremos crear un mezclador de palabras que reciba 2 palabras y un número. Si el número es menor o igual a 8 el mezclador concatena la primera palabra con la segunda. En cambio, si el número es mayor a 8, concatena la segunda con la primera:

```
> mezcladorDePalabras("planta", "naranja", 8)
"plantanaranja"

> mezcladorDePalabras("amor", "amarillo", 7)
"amoramarillo"

> mezcladorDePalabras("mate", "pato", 9)
"patomate"
```

Definí la función `mezcladorDePalabras`.

Solución

Consola

```
1 function mezcladorDePalabras(p1, p2, n) {
2   if (n <= 8) {
3     return p1+p2;
4   } if (n > 8) {
5     return p2+p1;
6   }
7 }
```

Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Siguiente Ejercicio: Ejercicio 7 >

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





⌚ Te quedan 00:03:03

# Ejercicio 5: Ejercicio 5

**JS**

Ale está haciendo un trabajo de investigación y nos pidió ayuda . Necesita poder sumar la cantidad de letras de las palabras cortas . Una palabra se considera corta si tiene 6 o menos letras. Veamos un ejemplo:

```
> sumaDeLetrasDePalabrasCortas(["hola", "murcielago", "caballo", "c  
hoclo", "poco", "luz", "sol"])  
20
```

Definí la función `sumaDeLetrasDePalabrasCortas` .

[Solución](#)[> Consola](#)

```
1 function sumaDeLetrasDePalabrasCortas(palabras) {  
2   let sumatoria = 0;  
3   for(let palabra of palabras) {  
4     if (longitud(palabra) <= 6) {  
5       sumatoria = sumatoria + longitud(palabra)  
6     }  
7   }  
8   return sumatoria;  
9 }
```

[▶ Enviar](#)

✓ ¡Muy bien! Tu solución pasó todas las pruebas

[Siguiente Ejercicio: Ejercicio 7 >](#)

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





🕒 Te quedan 00:03:09

# Ejercicio 6: Ejercicio 6



Los servicios de películas bajo demanda lograron despertar un interés renovado en la sociedad por el cine y las series . Es por ello que contamos registros de este estilo:

```
let gus = {
  nick: "Wuisti",
  promedioPeliculasMensuales: 5,
  plataforma: "NetFix"
};

let ariel = {
  nick: "Ari",
  promedioPeliculasMensuales: 10,
  plataforma: "Armazon"
};
```

Ahora debemos definir una función que permita obtener un resumen de la información registrada de manera simple. Por ejemplo:

```
> resumenInformacion(gus)
"Está estimado que Wuisti verá 60 películas en un año por la platafor
ma NetFix"

> resumenInformacion(ariel)
"Está estimado que Ari verá 120 películas en un año por la platafor
ma Armazon"
```

Definí la función `resumenInformacion` que nos permita obtener la información requerida.

Solución >\_ Consola

```
1 function resumenInformacion(persona) {
2   return "Está estimado que " + persona.nick + "
   verá " + persona.promedioPeliculasMensuales*12 + "
   películas en un año por la plataforma " +
   persona.plataforma;
3 }
```

▶ Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Siguiente Ejercicio: Ejercicio 7 >







🕒 Te quedan 00:03:15

# Ejercicio 7: Ejercicio 7



¡Dejemos atrás a JavaScript para pasar a Ruby!

Vamos a modelar `Camioneta` s para poder:

- cargarle una cantidad de nafta determinada;
- ver si tiene carga suficiente, es decir, si tiene más de 39 litros de nafta.

Definí en Ruby, la clase `Camioneta` que tenga un atributo `@nafta` con su getter. Los autos entienden los mensajes `cargar_nafta!` (que recibe la cantidad a cargar por parámetro) y `nafta_suficiente?`. No te olvides de definir un `initialize` que reciba a la nafta inicial como parámetro.

Solución

Consola

```
1 class Camioneta
2   def initialize
3     @nafta = litros
4   end
5   def self.nafta
6     @nafta
7   end
8   def self.cargar_nafta!(litros)
9     @nafta += litros
10  end
11  def self.nafta_suficiente?
12    @nafta > 39
13  end
14 end
15
```

Enviar

## ✖ Tu solución no pasó las pruebas

Resultados de las pruebas:

- ✖ Si creo una instancia de Camioneta le puedo especificar su nafta inicial [Ver detalles](#)
- ✖ Una instancia de Camioneta suma 20 litros de nafta al enviarle cargar\_nafta!(20) [Ver detalles](#)
- ✖ Una instancia de Camioneta suma 5 litros de nafta al enviarle cargar\_nafta!(5) [Ver detalles](#)
- ✖ Una instancia de Camioneta suma 100 litros de nafta al enviarle cargar\_nafta!(100) [Ver detalles](#)
- ✖ Una instancia de Camioneta no tiene carga suficiente si tiene 39 litros de nafta [Ver detalles](#)
- ✖ Una instancia de Camioneta no tiene carga suficiente si tiene menos de 39 litros de nafta [Ver detalles](#)
- ✖ Una instancia de Camioneta tiene carga suficiente si tiene más de 39 litros de nafta [Ver detalles](#)

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







⌚ Te quedan 00:03:21

# Ejercicio 8: Ejercicio 8



Los compilados son discos que tienen la característica de recopilar canciones que comparten alguna característica, por ejemplo artista, época o género. Algunas de ellas con mayor duración que otras.

Teniendo en cuenta que las canciones saben responder al mensaje `titulo` ...

Definí en Ruby el método `nombres_de_canciones` que responda el nombre de las canciones del `Compilado`.

Solución

Consola

```
1 module Compilado
2   @canciones = [AmorAusente, Eco, Agujas, ElBalcon,
3   GuitarrasDeCarton]
4   def self.compilado
5     @canciones
6   end
7   def self.nombres_de_canciones
8     compilado.map { |canciones| canciones.titulo }
9   end
end
```

Enviar

✓ ¡Muy bien! Tu solución pasó todas las pruebas

Siguiente pendiente: Ejercicio 7 >

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





🕒 Te quedan 00:03:28

# Ejercicio 9: Ejercicio 9



Como bien sabemos, una `Banda` tiene integrantes. Cuando la banda toca, toca cada integrante:

- `Bajista` pierde una de sus `cuerdas` ;
- `Saxofonista` sube su `indice_de_coordinacion` en 34;
- `Triangulista` no hace nada.

Definí el método `tocar!` tanto en la `Banda` como en los distintos tipos de integrantes. Definí los getters necesarios en cada integrante.

Solución Consola

```
1 class Banda
2   def initialize(integrantes)
3     @integrantes = integrantes
4   end
5   def tocar!
6     @integrantes.each {|integrante|
7       integrante.tocar!
8     }
9   end
10
11 class Bajista
12   def initialize(cuerdas)
13     @cuerdas = cuerdas
14   end
15   def cuerdas
16     @cuerdas
17   end
18   def tocar!
19     @cuerdas -= 1
20   end
21
22 class Saxofonista
23   def initialize(indice_de_coordinacion)
24     @indice_de_coordinacion =
25       indice_de_coordinacion
26   end
27   def indice_de_coordinacion
28     @indice_de_coordinacion
29   end
30   def tocar!
31     @indice_de_coordinacion += 34
32   end
33
34 class Triangulista
35   def tocar!
36   end
37 end
```

Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Siguiente pendiente: Ejercicio 7 >

