Argentina Programa

Ejercicio 5

Los lenguajes de programación son creados con algunas palabras que solo se pueden utilizar con un fin determinado. Se las llama **palabras reservadas** . En Gobstones, el lenguaje que estamos utilizando, program es una palabra reservada.

Como ya sabemos que nuestros programas son ejecutados por la máquina, de ahora en más diremos "creá un programa que haga ..." en vez de "creá un programa que ***provoque que la máquina*** haga ...".

Pero ojo: la máquina sigue estando ahí y no hay que olvidarla, sólo hacemos esto para escribir un poco menos.

Ejercicio 6

Notá que estos dos programas hacen lo mismo:

program {

Mover(Este)

Mover(Este)

Mover(Sur)

}

program {

Mover(Este)

Mover(Sur)

Mover(Este)

}

Moraleja: como te decíamos al principio ¡no hay una sola forma de resolver un problema!

Ejercicio 7: El orden de las cosas

Cuando trabajamos en Gobstones, hacemos las cosas en un cierto orden. Por ejemplo, si tenemos este programa:

program {

Mover(Norte)

Mover(Este)

}

una forma posible de leerlo (llamada **operacional**) es como lo haría una máquina, en orden, de arriba hacia abajo:

1. primero se mueve al norte: Mover(Norte)
2. luego se mueve al este: Mover(Este)

Y de hecho **se ejecuta de esa forma**. Esto es *cómo* lo hace.

Pero, los humanos, solemos pensar en función del resultado final, es decir, resaltamos el **objetivo** del programa. Nos importa más *qué* hace, y no cómo. Esta manera denotacional nos llevaría a decir que, simplemente, **mueve el cabezal al noreste**.

Por eso hay varias formas de resolver un mismo problema: podemos crear varios programas que hagan lo mismo (el *qué*), pero que lo hagan de forma diferente (el *cómo*).

# Ejercicio 8: Sí, esto también se puede romper

Hasta ahora veníamos tratando de esquivar el asunto, pero seguro ya habrás pensado que tiene que haber alguna forma de romper esto (incluso es posible que ya te haya pasado).

Si bien **nunca** vamos a querer que nuestro programa se rompa, es algo que definitivamente **te va a pasar muchas veces**. Pero no es motivo para frustrarse ni mucho menos, te aseguramos que a todo el mundo le pasó alguna vez (bueno, también 2, 3, 100, 800 veces...).

Dicho esto, te vamos a mostrar una forma de hacerlo, simplemente para que no te asustes tanto cuando te pase de verdad

**¡BOOOOOOOOOOOOOOOOOM!**

*Ey, ¿qué pasó?*  
Tu programa falló, se rompió, o como lo llamamos en el universo Gobstones: **hizo BOOM**.

*Y, ¿qué significa esto?*  
Que los comandos que le diste a la computadora no se pueden ejecutar, y hay algo que vas a tener que cambiar para que funcione. En este ejercicio eso no tiene sentido porque lo hicimos a propósito, pero tenelo en cuenta para cuando falles en el futuro.

Ejercicio 9: Nuestras primeras bolitas

Genial, ya entendiste cómo mover el cabezal del tablero usando la operación Mover y las direcciones (Sur, Oeste, etc). Vayamos un paso más allá: las **bolitas**.

En cualquier celda de nuestro tablero podemos poner bolitas. Las hay de distintos colores:

* rojas (Rojo);
* azules (Azul);
* negras (Negro);
* y verdes (Verde).

Por ejemplo, este es un tablero con una bolita roja y una negra:

Además de moverse, el cabezal también puede poner bolitas en la **celda actual**. Para eso contamos con la operación Poner, que le dice al cabezal que deposite una bolita del color dado:

program {

Poner(Rojo)

}

# Ejercicio 10: Más y más bolitas

Algo interesante de nuestros tableros es que en sus celdas podemos poner cualquier cantidad de bolitas de cualquier color.

y ejecutamos el siguiente programa:

program {

Poner(Rojo)

Poner(Rojo)

Poner(Azul)

Poner(Verde)

Poner(Rojo)

}

el cabezal colocará en la celda actual tres bolitas rojas, una azul y una verde.

Notá que en este problema, si cambiamos el orden en que *llamamos* (usamos a) Poner, el resultado no cambia: siempre nos terminará quedando un tablero con tres bolitas rojas, una azul y una verde.

Por ejemplo, los siguientes dos programas también resuelven este mismo problema:

program {

Poner(Rojo)

Poner(Rojo)

Poner(Rojo)

Poner(Verde)

Poner(Azul)

}

program {

Poner(Rojo)

Poner(Azul)

Poner(Rojo)

Poner(Verde)

Poner(Rojo)

}

# Ejercicio 11: Poné tus primeras bolitas

Como te habrás dado cuenta, estos tableros son un poco mágicos, podemos poner en una celda tantas bolitas como queramos: 2, 4, 12, 50, 1000. ¡No hay ningún límite!

Esto es algo muy interesante que ocurre al programar: podemos trabajar con cantidades tan grandes como queramos.

Ah, y ahora te toca a vos: creá un programa que ponga cuatro bolitas rojas y tres bolitas negras en la celda actual.

# Ejercicio 12: Sacar Bolitas

De la misma forma que hay un "poner bolita" (Poner), tenemos un "sacar bolita" (Sacar), que quita exactamente una bolita del color dado.

Por ejemplo, el siguiente programa saca dos bolitas de la posición inicial.

program {

Sacar(Rojo)

Sacar(Rojo)

}

Sabiendo esto, creá un programa que elimine **sólo** la bolita roja de este tablero. ¡Tené cuidado! Prestá atención a la posición del cabezal .

# Ejercicio 13: Cuando no hay bolitas

Cada vez que usamos Sacar, tenemos que tener más cuidado que con Poner, porque...

¿Querés saber por qué? Intentá sacar una bolita verde o azul de este tablero y descubrilo.

Claro, otra vez **BOOM**.

Esta vez lo que pasó fue que el cabezal intentó sacar una bolita de un color que no había, y como no sabía qué hacer se autodestruyó. Esto te va a pasar siempre que intentes sacar una bolita que no exista, así que ¡a prestar atención!

* [1. Fundamentos](https://mumuki.io/argentina-programa/chapters/530-fundamentos)
* [2. Práctica Primeros Programas](https://mumuki.io/argentina-programa/lessons/669-fundamentos-practica-primeros-programas)

# Ejercicio 1: Calentando motores

¡Veamos un primer ejemplo!

El siguiente programa coloca una bolita roja en la posición inicial y una negra al este.

program {

Poner(Rojo)

Mover(Este)

Poner(Negro)

}

Probá escribir y ejecutar este programa. Te mostraremos el resultado al ejecutarlo en un tablero de 2x2, y en otro de 3x2, ambos con el cabezal inicialmente en el origen.

Ahora que combinamos operaciones, la cosa se pone un poco mas complicada, porque hay que tener **más cuidado con el orden**.

Por ejemplo, mirá el programa que escribiste:

program {

Poner(Rojo)

Mover(Este)

Poner(Negro)

}

Operacionalmente:

1. pone una roja
2. luego se mueve al este
3. luego pone una negra

Es decir: pone una roja en la posicion inicial, y una negra al este

Y ahora mirá este otro:

program {

Mover(Este)

Poner(Rojo)

Poner(Negro)

}

Operacionalmente:

1. se mueve al este
2. luego pone una roja
3. luego pone una negra

Es decir: pone una roja y una negra al este de la posición inicial.

Moraleja: ¡no hacen lo mismo! Cambiar el orden nos cambió el *qué*.