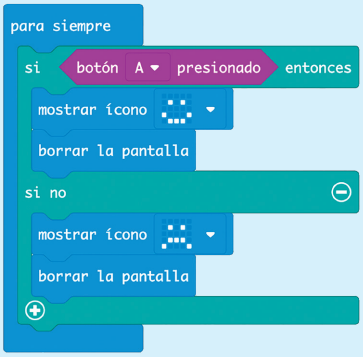
# FICHA 3
# EXPEDICIÓN ESPELEOLÓGICA

Sesión 1

## Aprendizajes

Al final de esta actividad se espera que puedas:

- Utilizar variables booleanas de entrada.
- Comunicar instrucciones utilizando la pantalla de LED y un código de flechas.
- Interpretar un diagrama de flujo para resolver problemas como el de un laberinto.
- Utilizar operaciones lógicas para decidir qué acción se ejecuta.
- Utilizar lazos que se repiten hasta terminar la tarea.

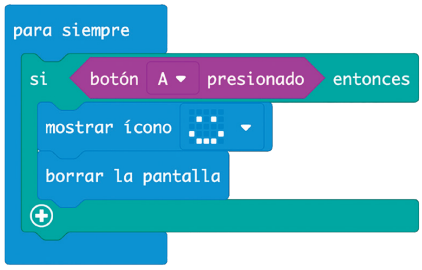



## Lo que sabemos, lo que debemos saber

En las fichas anteriores ya has trabajado con entradas como los botones los cuales pueden asumir dos valores: *presionado* y *no presionado*. Igualmente, hemos trabajado con los **LED**. Cada **LED** asume dos valores: *encendido* o *apagado*. Estas variables se denominan **booleanas** como verás en esta ficha. Igualmente, has utilizado bloques que representan algunas acciones o instrucciones que se deben repetir con **bucles** y **condicionales**.

Las ***variables booleanas*** pueden asumir dos valores solamente: ***verdadero*** o ***falso.***

Cuando el ***Botón A está oprimido,*** su valor es ***verdadero*** y cuando no, es ***falso.***




¿Con qué objetivo se incluye la instrucción **borrar pantalla**? ¿Qué sucede si no se incluye este bloque? Prueba en el editor tu predicción.

Examina ahora el diagrama de flujo que aparece abajo. ¿Qué hace? ¿En qué se diferencia del diagrama de bloques que acabas de observar?

En el ejemplo del programa en bloques que sigue, si presionas ***A*** verás una cara feliz. Esta es una forma de controlar la realización o no de ciertas instrucciones.


En este caso, al no estar oprimido el botón A, se verá una ***cara triste.*** Por el contrario, si lo oprimes verás una ***cara feliz.*** Una posible transcripción de este diagrama de flujo a bloques se muestra a la izquierda.

Para desplegar la opción **si no** basta oprimir el símbolo + (más), quedando el bloque como se muestra.

¿Se requiere en este caso incluir el bloque borrar pantalla? ¿Por qué sí o por qué no? Verifica en el editor.

¿Está oprimido el botón A?

Sí

¿Está oprimido el botón B?

Sí

¿La temperatura es mayor de 25°?

Sí

Acción si las tres condiciones son verdaderas

Recuerda que puedes acceder al editor Makecode en

*makecode.microbit.org*

En algunos casos se tienen dos o tres condiciones seguidas, como en el diagrama de la izquierda. En estos casos se pueden reemplazar varias condiciones por una sola verificación, que incluya las dos o tres condiciones:

¿Está oprimido el botón A
y
está oprimido el botón B
y
la temperatura es mayor de 25°?

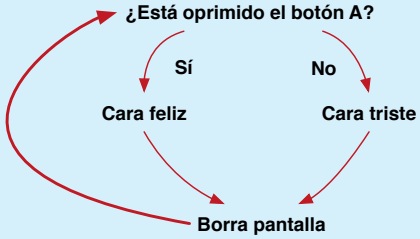
Sí

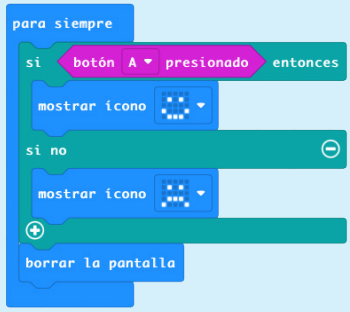
**Acción si las tres condiciones son verdaderas**

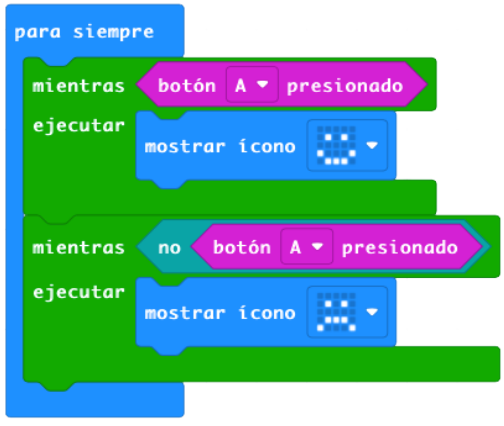
Lo mismo puede ser experesado por un solo bloque condicional con varias condiciones, como el que se muestra abajo. Para que se visualice la cara feliz se requiere que las tres condiciones sean verdaderas. Si alguna es falsa, no debe mostrarse la cara feliz:


**RESUMEN**

Las instrucciones ***si*** (condición) ***entonces*** (instrucciones) ***si no*** ***entonces*** (instrucciones) se representa por:



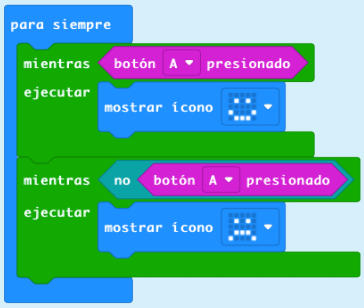




Compara los diagramas del resumen con el que está a la izquierda.

¿hacen lo mismo?

Si no, ¿en qué se diferencian en su operación?

Usa el editor para simular.

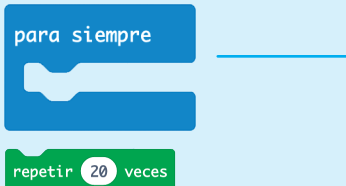


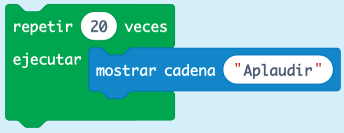

Si examinas el programa anterior, que copiamos a la izquierda de nuevo, encontrarás algunos bloques con los cuales ya hemos trabajado:

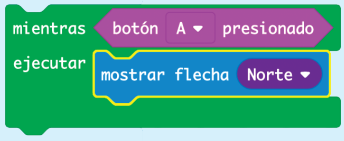
- ***Para siempre***: repite lo que se encuentra dentro indefinidamente.
- ***Mientras botón A presionado***: que ejecuta lo que se encuentra dentro del bloque mientras el botón A esté presionado.
- ***Mientras botón A no presionado***: que ejecutará lo que está dentro del bloque mientras el botón A **NO** esté presionado.

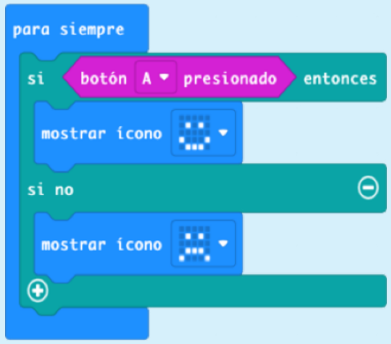
A este tipo de bloque se les denomina **bucle**. Su función es repetir un conjunto de instrucciones según alguna condición, o para siempre.

**Bucles** y **condicionales** son instrucciones poderosas de un lenguaje de programación y es lo que hace que un computador pueda resolver problemas repitiendo y seleccionado ciertas instrucciones según alguna condición.





## EN RESUMEN

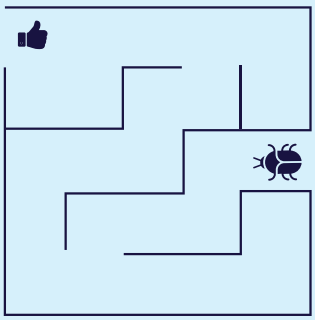
Según el objetivo que tengas, puedes utilizar diferentes tipos de ***bucles***:

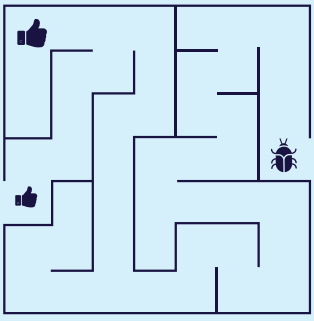
- ***Bucles indefinidos*** o para siempre que repiten el conjunto de instrucciones indefinidamente, o hasta que se detenga el ***procesador***. Verás en la siguiente sección que la ***micro:bit*** tiene un bucle ***para siempre***.
- ***Bucles*** que se repiten un número exacto de veces, por ejemplo, en este caso 20 veces.
- ***Bucles*** que se repiten mientras una condición sea cierta, por ejemplo, mientras el botón B esté oprimido o mientras la temperatura sea muy baja.

Los ***bucles*** en general requieren condicionales. En el bloque anterior, la condición es tener el botón A presionado para mostrar la flecha. Si no está presionado, dejará de mostrarse la flecha y el programa seguirá adelante.

- Sin embargo, los condicionales que se encuentran en **Lógica**, se usan también para decidir qué instrucciones realizar y cuáles no. Por ejemplo, en el bloque a la izquierda si el botón A está presionado se verá una cara feliz, si no lo presionamos, una cara triste. En este caso este pequeño programa se repetirá indefinidamente por estar en un ***bucle para siempre***.
- Todas las acciones que están dentro de este espacio se ejecutarán si la condición se cumple, es decir, es VERDADERA.
- Todas las acciones que están dentro de este espacio se ejecutarán si la condición no se cumple, es decir, es FALSA

## Desconectadas



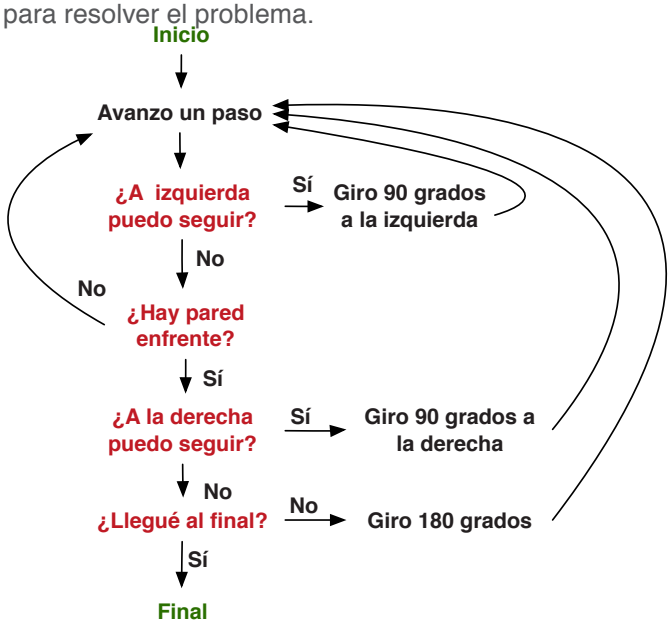


Te has inscrito al grupo de espeleología del municipio en el que se entretienen explorando cavernas. En una de las salidas se encuentran con una cueva que no conocen y parece tener muchos túneles. El grupo se pregunta si al entrar podrán salir sin tener un mapa. Tú sabes que usando la lógica de la computación puedes lograrlo y les cuentas de un algoritmo llamado "**seguir la pared**" (se muestra abajo de esta página y en el anexo). El resto del grupo está escéptico y para demostrarles que el algoritmo funciona con cualquier tipo de configuración les pides que te den dos laberintos en papel, como los que se muestran a la izquierda (se encuentran en los anexos también).

Puedes hacer el laberinto con cinta en el piso para que te puedas desplazar por él o usar una hoja con el dibujo de los laberintos y una ficha para representar tu desplazamiento. Si estas trabajando en grupo pueden asignar los siguientes roles:

- **Depurador**: sigue el diagrama de flujo poniendo una ficha en la instrucción que se está ejecutando e indicándola en voz alta.
- ***Procesador:*** se ubica a la entrada del laberinto, si está hecho con cinta en el piso, o coloca un objeto o ficha en la entrada sobre el papel. En el caso de usar un objeto, debe tener claro cuál es el frente para saber cuál es la izquierda o la derecha.
- ***Medidor de complejidad:*** va contando los pasos requeridos para salir del laberinto.
- ***Verificador:*** si hay alguien más en el grupo, esta persona debe verificar que se siga la secuencia de instrucciones y cuenta cuántos pasos se dan.

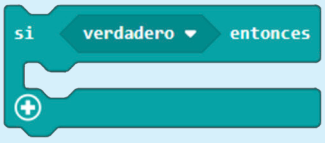
**Al pasar al segundo laberinto cambiar los roles.** Terminada la labor, compara el número de pasos dados en los dos laberintos. También podrás buscar otros laberintos más complejos para probar el algoritmo y verificar qué tantos pasos debes dar para resolver el problema.



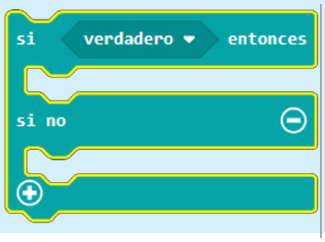
**Recuerda que:**

Todo programa en la ***micro:bit*** debe estar en un bucle de repetir general como el siguiente (hay otros):


Los bloques condicionales encajan perfectamente en los bloques de bucle. El hexágono que está al lado de la palabra clave “si” representa una variable ***booleana,*** por lo tanto, asume dos posibles valores: verdadero o falso. Los bloques que insertes en el bloque condicional “si”, solo se ejecutarán cuando el valor del hexágono sea “verdadero”.




El bloque condicional “si, si no” tiene un espacio adicional para agregar bloques que se ejecutarán cuando el valor del hexágono sea “falso”. Puedes convertir el bloque "si" en uno "si, si no" presionando el signo “+” de la parte inferior.

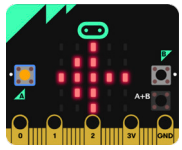



## *Conectadas: manos a la micro:bit*

Es hora de seguir profundizando tus conocimientos sobre la ***micro:bit***. Hasta ahora has explorado el entorno de programación, el simulador y los bloques para crear bucles. En esta ocasión, revisarás los bloques de lógica condicional. Recuerda que para trabajar con la ***micro:bit*** necesitarás entrar a ***MakeCode*** en tu computador o al editor en línea si tienes acceso a Internet.

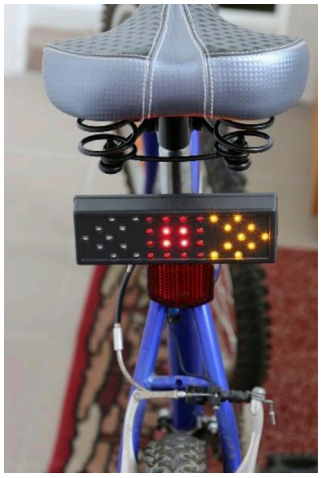
Para poner a prueba los nuevos bloques aprendidos, crearás un dispositivo que te permita ayudar a las personas que se desplazan en bicicleta a transitar de forma más segura por las vías permitidas.

El dispositivo permitirá tener luces informativas mientras se monta en bicicleta. Es posible extender los botones de la ***micro:bit*** para que sean presionados desde los manubrios, donde se colocan las manos. Sin embargo, con fines ilustrativos y a modo de prueba de concepto de tu diseño, usarás los botones A y B que ya conoces.



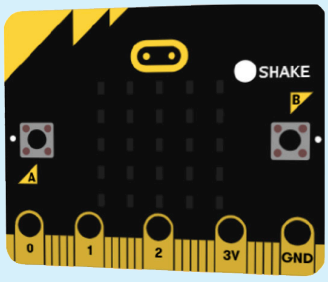

El dispositivo funcionará de la siguiente manera:

1. Cuando se presione el botón A, la ***micro:bit*** deberá mostrar una luz direccional a la izquierda parpadeando.
2. Cuando se presione el botón B, la ***micro:bit*** deberá mostrar una luz direccional a la derecha parpadeando.
3. Cuando se presionen los dos botones, la ***micro:bit*** deberá mostrar un indicativo de frenar para alertar a quien va detrás. ¿Cuál te parece más apropiado?



Es hora de programar y simular en el editor ***MakeCode*** el programa para verificar su funcionamiento.


1. ¿A qué parte del funcionamiento corresponde el programa que se muestra a la derecha?
2. Al programarlo en el editor ***Makecode*** ¿qué sucede si retiramos el bloque “borrar la pantalla”?
3. Ahora debes programar el resto del funcionamiento. En este mismo bloque “para siempre”, agrega las condiciones restantes.
4. Cuando tu código incluye la condición de A+B, en el simulador aparece un tercer botón para probar tu código.




## *Trabajando con otras entradas*

Ya conoces el sensor de temperatura, recuerda que es otra variable de entrada. ¡La ***micro:bit*** posee más sensores! En esta ocasión usarás el sensor llamado ***acelerómetro.*** Este sensor mide de cierta forma el movimiento del dispositivo.

Imagina que tienes una botella llena de agua y que al taparla queda una burbuja de aire atrapada en su interior. A medida que cambias de posición la botella, la burbuja se desplaza para quedar siempre lo más arriba posible. La burbuja se mueve tan rápido como muevas la botella. Así, el acelerómetro también puede saber en qué posición se encuentra la ***micro:bit***: logotipo arriba, logotipo abajo, inclinado, etc.

También puede saber si se agita el dispositivo y qué tan rápido se hace: 3 g, 6 g, 8 g. Esta agitación se mide con respecto a la gravedad, así como cuando un vehículo acelera y sientes que algo te presiona contra el asiento, 3 g significa una aceleración equivalente al triple de la gravedad.


### *Aplicando lo aprendido*

Puedes usar el acelerómetro para hacerte más visible mientras estés montando bici.

Teniendo en cuenta que el dispositivo se estará agitando a medida que te mueves, podrías mostrar flechas que indiquen que te estás desplazando hacia adelante.

**Para ilustrar un desplazamiento hacia adelante**

1. El bucle presentado en el diagrama de la izquierda, es un bloque general como el bucle “para siempre”. Dentro de este bucle hay dos bloques que se repetirán uno tras otro hasta que se termine la tarea.
2. Antes de implementar el diagrama mencionado, intenta predecir lo que ocurrirá al ejecutarlo. Recuerda que el bloque “para siempre” se está ejecutando en todo momento. En este caso la entrada “agitado” del acelerómetro, es una variable booleana que se encuentra en la sección de entradas.
3. Al agregar el sensor acelerómetro, el simulador presenta un botón que dice “SHAKE” (agitar) para representar que la tarjeta está siendo agitada.
4. ¿Ves la flecha desplazarse? ¿Podrías mejorar este desplazamiento? Considera que son como dos fotos que se muestran una tras otra y dan la sensación de movimiento; podrías agregar fotos intermedias para hacer el movimiento más fluido.
5. Complementa nuestro sistema de luces agregando animaciones para todos los indicadores.


## Para ir más lejos

El programa que creaste en la sección anterior puede ser muy útil cuando montes en bici para indicar a quienes estén a tu alrededor si vas a girar a la izquierda o a la derecha, si estás en movimiento o si vas a parar. Haciendo algunos cambios puedes hacer el programa aún mejor. ¿Has notado que al montar en bicicleta te inclinas ligeramente hacia el mismo lado hacia el que estás girando? ¿Cómo crees que puedes usar este fenómeno para mejorar tu programa? Como se mencionó anteriormente en esta ficha, la ***micro:bit*** cuenta con un acelerómetro. Además de medir si hay cambios en el movimiento, el acelerómetro puede indicar si la **micro:bit** está completamente horizontal o inclinada hacia la izquierda o hacia la derecha. Si fijas la ***micro:bit*** a tu cuerpo no necesitarías presionar los botones A o B para indicar un giro, la ***micro:bit*** podría usar su acelerómetro para determinar si estás girando y en qué dirección. Usa el bloque que se encuentra en el menú "Entrada" que se muestra a continuación:


## Lo que hemos aprendido

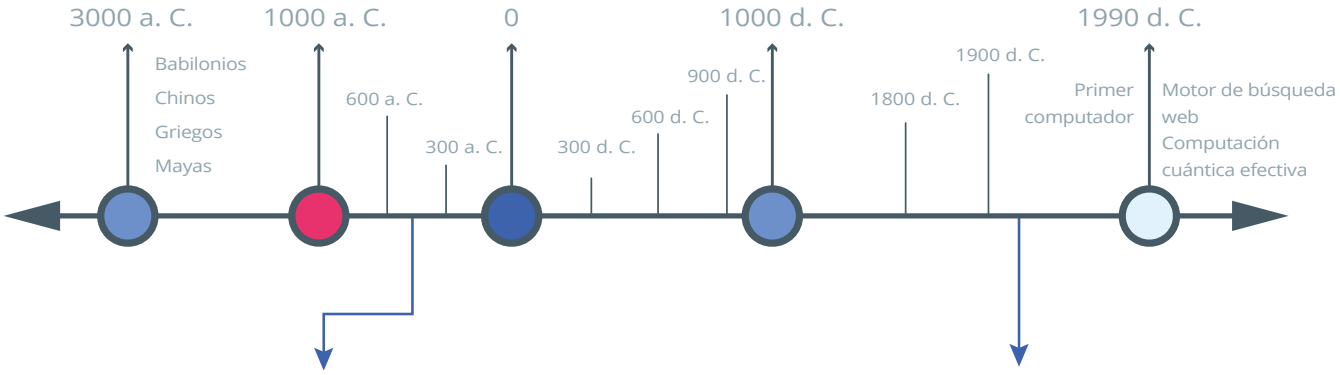
Revisa y completa la siguiente tabla marcando una X en la columna que mejor represente tu aprendizaje:

| Verifica los aprendizajes logrados | Sí | Algo | No |
|---|---|---|---|
| Utilizo variables booleanas de entrada. | | | |
| Comunico instrucciones utilizando la pantalla de LED y un código de flechas. | | | |
| Interpreto un diagrama de flujo para resolver problemas como el de un laberinto. | | | |
| Utilizo operaciones lógicas para decidir qué acción se ejecuta. | | | |
| Utilizo lazos que se repiten hasta terminar la tarea. | | | |

Selecciona la opción que mejor represente tu opinión:

| Contesta las siguientes preguntas | Sí | Algo | No |
|---|---|---|---|
| Las actividades realizadas fueron difíciles. | | | |
| Las actividades me motivaron. | | | |
| Siento que aprendí muchas cosas. | | | |
| Aún me quedan muchas dudas sobre lo que hice. | | | |

## *Un poco de historia*




**Euclides (325 a. C. - 265 a. C.)**

Matemático y geómetra griego.

Se le reconoce como el fundador de la geometría. Vivió en Alejandría, hoy norte de Egipto. En su escrito "Elementos de Euclides", se recopila todo el conocimiento de la época de manera formal y a partir de tan solo cinco postulados. En este documento también se presentó un algoritmo para calcular el máximo común divisor de dos números. Dicho algoritmo de más de 2000 años de antigüedad es conocido como el algoritmo de Euclides, y es incluso usado en la actualidad.

**John Backus (1924 - 2007)**

Científico computacional estadounidense. Conformó y lideró el grupo que inventó y desarrolló FORTRAN: el primer lenguaje de programación de alto nivel que fue ampliamente usado por la comunidad científica. Aun en la actualidad, versiones mejoradas de FORTRAN son preferidas para cálculos matemáticos por la velocidad de ejecución que logran en tareas complejas. Se le concedió el premio Turing en 1977, por su importante contribución a los lenguajes de programación, tanto por FORTRAN como por la contribución en la definición de un lenguaje que servía para definir lenguajes de programación: el metalenguaje Bakus-Naur.

El premio Turing es la máxima distinción en las ciencias de la computación.

Las bases del pensamiento computacional vienen desde la antigüedad

## *Mujeres y hombres que se destacan en la computación en Colombia y en el mundo*

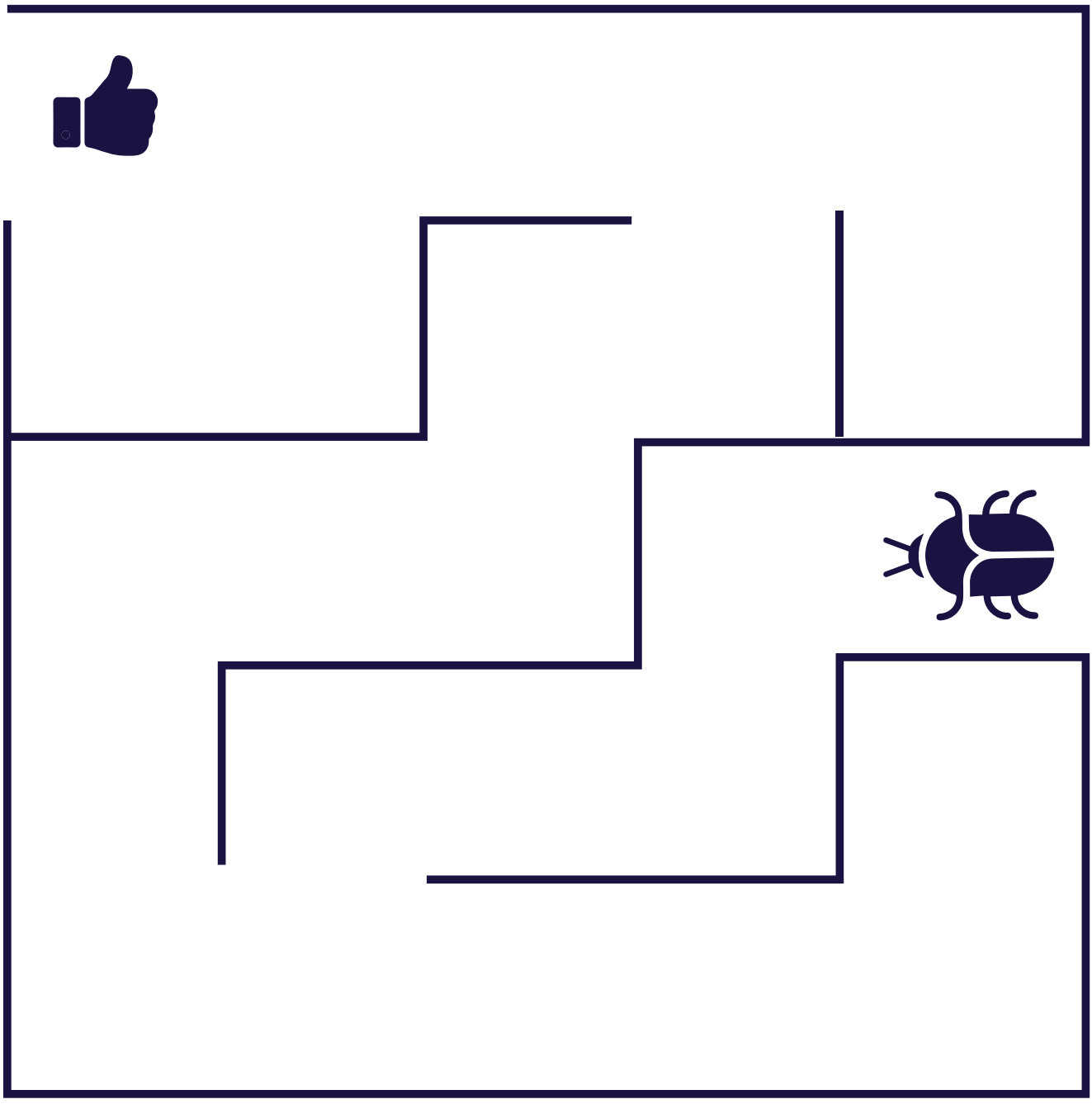
**Fernanda Moya**

Adriana Fernanda Moya estudió primaria y secundaria en el Colegio de la Presentación en Ubaté, Cundinamarca, muy cerca de Bogotá. Luego estudió Ingeniería de Sistemas en la Universidad de Cundinamarca y realizó una especialización en Ingeniería de Software en la Universidad Distrital Francisco José de Caldas.

Adriana es Ingeniera de Servicios en la Nube (Cloud Engineer) en Globant, una empresa que transforma organizaciones preparándolas para un futuro digital y cognitivo, y es una de las organizadoras de Women Techmakers Bogotá. Women Techmakers es una iniciativa mundial patrocinada por Google que busca hacer visibles, crear redes de apoyo y proveer recursos a mujeres y otras poblaciones con baja representación en las áreas de tecnologías computacionales. Si quieres ver la entrevista de Adriana Fernanda visita el enlace o escanea el código QR que está al comienzo de esta sección.

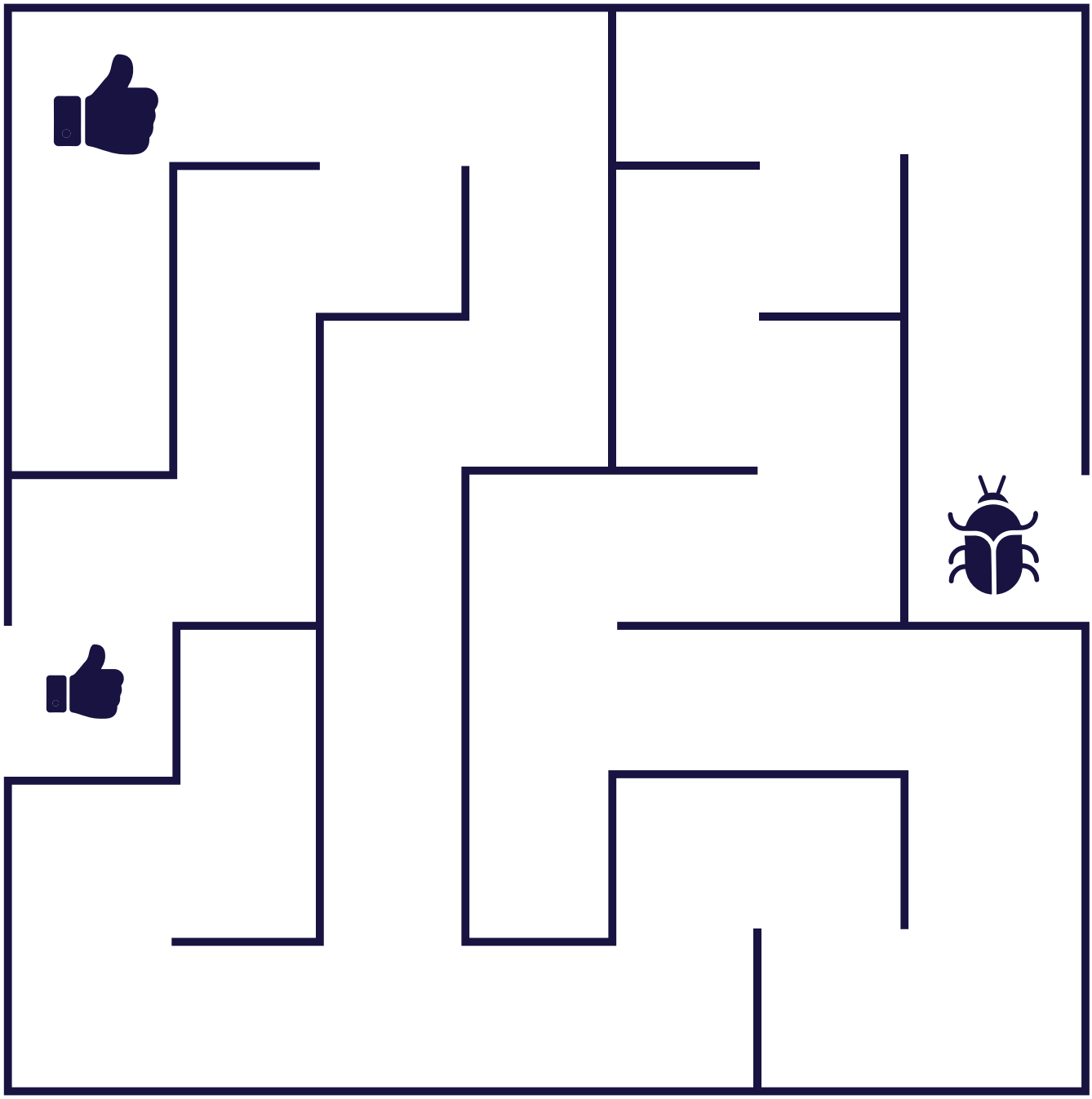
## Anexo 1

### Laberinto No. 1



## Anexo 2

### Laberinto No. 2



## Anexo 3

### Diagrama de flujo "seguir la pared"