Programación básica con laberintos

3° Ciclo y Polimodal

Versión para imprimir

Didáctica de la actividad | Información de la actividad | Recursos

Es conveniente que el docente realice previamente todos los pasos solo, si nunca antes realizó una actividad de este tipo.

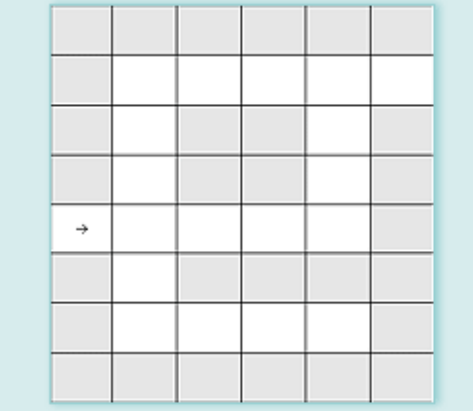
1. Explicar qué es un laberinto y qué es el ente que recorre un laberinto.

El **laberinto** es una matriz con celdas que pueden estar vacías (esto significa que se puede pasar a través de ellas) u ocupadas por una pared (lo cual significa que no se puede pasar a través de ellas). Hay una única entrada al laberinto, al menos una salida y al menos un camino de celdas vacías que conecta la entrada con la salida.

El **ente** tiene como atributos una posición en una celda vacía del laberinto (al comenzar está en la celda de entrada) y una orientación (Norte, Sur, Este u Oeste) hacia la cual puede avanzar. Este ente puede representar una persona, una computadora, una tortuga al estilo del lenguaje LOGO, etc.

2. Comentar el objetivo de la actividad: escribir un programa que lleve al ente desde la celda de entrada hasta una de las salidas del laberinto.

Ejemplo de laberinto con una entrada, una salida y, al menos, dos caminos entre ellas, con el ente posicionado en la celda de entrada y orientado hacia el Este:

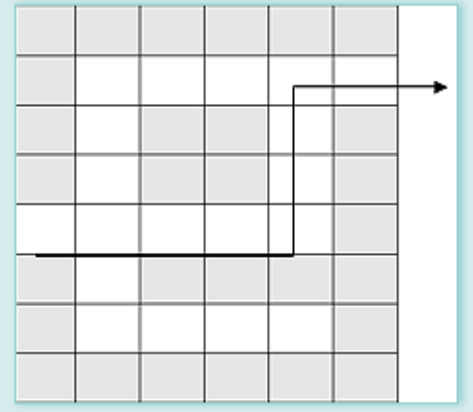


3. Explicar brevemente el lenguaje de programación.
Este lenguaje es simple y está basado en el lenguaje LOGO. Tiene dos instrucciones básicas, las cuales cambian el **estado** del ente. Es decir, pueden cambiar la posición u orientación del ente:

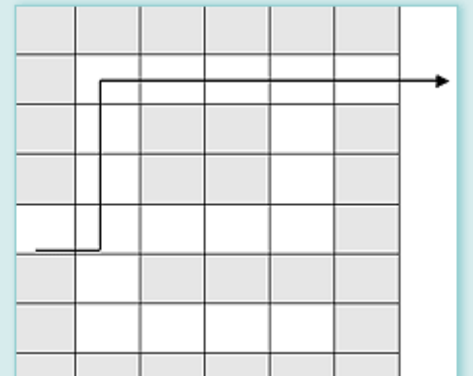
PASO: el ente avanza un paso hacia la orientación que tenga.
DER: el ente gira su orientación hacia su derecha.

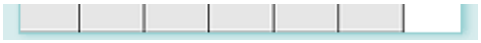
Estas instrucciones pueden utilizarse para escribir un programa que resuelva el laberinto del ejemplo anterior.

SOLUCIÓN 1:
PASO
PASO
PASO
PASO
DER
DER
DER
PASO
PASO
PASO
PASO
DER
PASO
PASO
FIN



Hay infinitas soluciones a este laberinto, algunas lógicas, otras no. La solución que dimos es la más simple. La siguiente figura muestra otra solución simple. El programa que obtiene esta solución queda como ejercicio para el lector.





4. Crear un subprograma
Para que el ente haga un giro hacia la izquierda, el programa debe repetir tres veces la instrucción DER. Podemos utilizar subprogramas para crear nuevas instrucciones a partir de las instrucciones ya existentes, y así poder escribir programas más cortos y legibles.

Un subprograma (al igual que un programa) se crea escribiendo un nombre, dos puntos, una serie de instrucciones y la directiva FIN. Una vez creado un subprograma, su nombre puede ser usado como una instrucción por cualquier otro (sub) programa.

Por ejemplo, crear la instrucción IZQ, que ordena al ente girar a la izquierda, utilizando la instrucción ya existente DER.

Crear una instrucción DOS_PASOS, basada en la instrucción PASO.

```
IZQ:    DOS_PASOS:
DER     PASO
DER     PASO
DER     FIN
FIN
```

5. Escribir entonces el programa de la primera solución en forma más concisa y clara:

```
SOLUCIÓN 1:
DOS_PASOS
DOS_PASOS
IZQ
DOS_PASOS
DOS_PASOS
DER
DOS_PASOS
FIN
```

Este programa va a ejecutar exactamente igual que la primera versión, ya que cuando la computadora lea la instrucción IZQ va a ejecutar el subprograma IZQ, es decir, va a ejecutar la instrucción DER tres veces. Igualmente al encontrar la nueva instrucción DOS_PASOS.

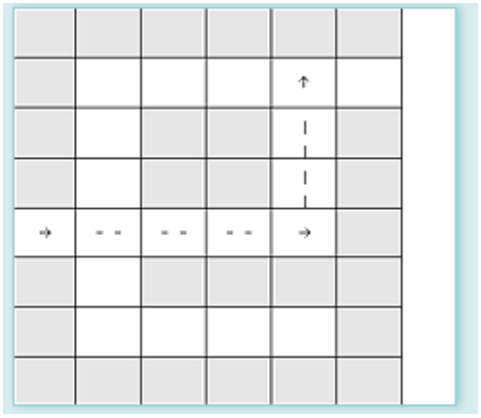
6. Introducir el concepto de repetición de una serie de instrucciones.

Retomar la primera solución. En vez de indicarle al ente que avance cuatro pasos, escribir un programa que haga avanzar al ente hasta encontrar una pared. Es decir, que le indique al ente ejecutar un PASO repetidamente, hasta que cierta condición deje de cumplirse. Para ello introducir condiciones o predicados lógicos en nuestro lenguaje, e instrucciones que controlen si esas condiciones se verifican o no (son verdaderas o falsas).

a. PARED? es una función lógica (o condición) que retorna VERDADERO sólo si en la siguiente casilla apuntada por la orientación del ente hay una celda ocupada por una pared.
b. NO condición: es la operación lógica de negación, que retorna VERDADERO sólo si la "condición" retorna FALSO.
c. MIENTRAS condición HACER instrucciones: es una instrucción (en realidad, un constructor de instrucciones) que permite escribir programas que repiten las "instrucciones" mientras la "condición" sea verdadera.
d. Por ejemplo, podemos crear un subprograma que haga avanzar el ente hasta que encuentre una pared:

```
AVANZAR:
MIENTRAS NO PARED? HACER
PASO
FIN
```

7. Utilizar este subprograma para reescribir la primera solución de manera que el ente avance hasta la pared, gire y luego avance hasta la otra pared.



```
SOLUCIÓN 1:
AVANZAR
IZQ
AVANZAR
DER
DOS_PASOS
FIN
```

Otra función lógica (o condición) muy útil es ADENTRO?, la cual retorna VERDADERO sólo si el ente está aún adentro del laberinto.

8. Para evitar tener que escribir un programa para cada laberinto escribir uno que resuelva cualquiera con las características de los presentados aquí: dos dimensiones, una entrada, al menos una salida y al menos un camino entre ellas.

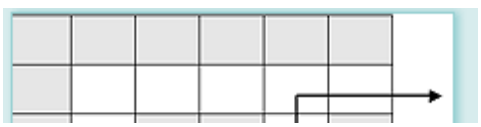
a. Escribir un subprograma que permita al ente "tantear" la pared en una celda a la que recién ingresó, comenzando por su derecha y siguiendo hacia la izquierda hasta encontrar una celda a la que pueda acceder.

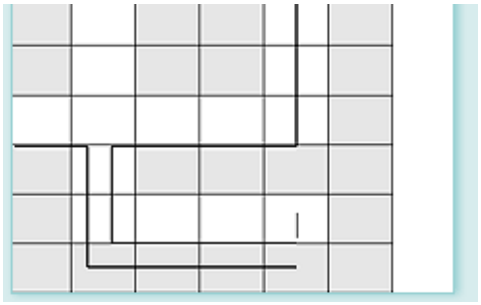
```
TANTEAR:
DER
MIENTRAS PARED? HACER
IZQ
END
```

b. Escribir un programa que permita al ente ir tanteando por la próxima celda más a su derecha y, cuando encuentra una celda libre, avanzar.

```
TANTEAR:
DER
MIENTRAS PARED? HACER
IZQ
END
```

c. Al ejecutar este programa en el laberinto del ejemplo, el ente realizará el siguiente recorrido, que no es el más corto, pero resuelve el problema.





Notas y sugerencias para realizar la actividad.

Durante la primera etapa -pasos 1 a 8- se crean distintos laberintos de diferente complejidad y se escriben programas que los resuelvan. Esta actividad puede hacerse sin computadoras, dibujando los laberintos en hojas cuadriculadas, en el pizarrón o en el suelo, y haciendo que un alumno del grupo responda ciegamente a las instrucciones del programa. Sin embargo, el lenguaje LOGO puede usarse para crear los laberintos y ejecutar los programas. Para ello se puede utilizar el programa MSWLogo y escribir los programas en forma similar a como se han escrito en el archivo laberinto.txt.

Se puede coordinar con otras áreas curriculares el estudio de laberintos: en la historia y la mitología, en la literatura, en la arquitectura, en los juegos (incluso en los videojuegos) y su simbología. Análisis, construcción y resolución de otros tipos de laberintos (3D, por ejemplo). Recomendamos la lectura de "La casa de Asterión" y "Abenjacán el Bojarí, muerto en su laberinto", cuentos ambos de Jorge Luis Borges.

Hemos visto que la computadora es una máquina electrónica simple que solamente puede obedecer órdenes muy simples. Pero esas órdenes combinadas en un programa pueden resolver problemas complicados. La segunda etapa permite ver cómo una combinación inteligente de estas instrucciones básicas puede hacer que la computadora se comporte de una forma que parezca inteligente.