

2023

Manual del usuario plataforma.karelogic.net



Instituto Tecnológico de Nuevo León

1-5-2023

Instituto Tecnológico de Nuevo León

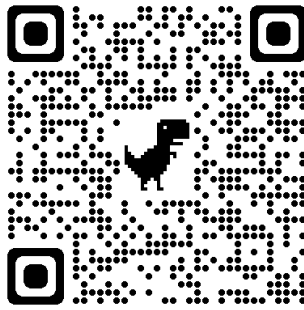
Desarrollo del talento y del pensamiento lógico-matemático

Guía instruccional para navegar en <https://plataforma.karelogic.net>

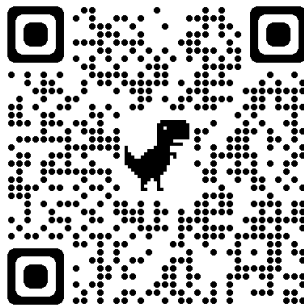
Índice

Prólogo	3
Antecedentes	4
Introducción	5
Objetivo de esta guía Institucional y Manual de usuario	6
Objetivo y características principales de la plataforma en la web	7
Primeros pasos para trabajar, navegar y avanzar en esta plataforma en la nube	8
Descripción del simulador y evaluador de Karel	12
El mundo del Robot, su simulador, sus componentes, elementos y características	13
ÁREA 1	14
ÁREA 2	16
ÁREA 3	16
ÁREA 4	18
ÁREA 5	18
Instrucciones básicas del Lenguaje de Karel	21
Herramientas de ayuda para una codificación sin errores	25
Estatuto de control de repetición, el "iterate(n)"	29
Los sensores virtuales de Karel	33
Sensores para los trompos y la mochila de Karel	33
Sensores para detectar bardas o paredes a media cuadra de donde esta Karel	34
Sensores para la orientación o dirección de Karel	36
Estatuto de control condicional simple y compuesto ("if" y "if-else")	39
Estatuto de control condicional simple "if"	39
Estatuto de control condicional compuesto "if-else"	41
Estatuto de control de repetición condicional "while"	43

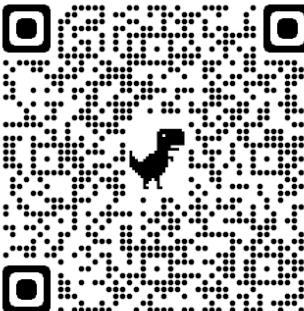
[Aquí puedes descargar este mismo manual](#)



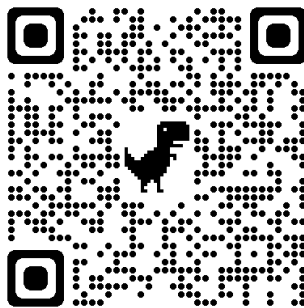
[Enlace del video tutorial de como navegar en plataforma.karelogic.net](#)



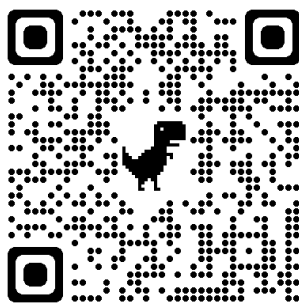
[Reflexiones de una maestra sobre plataforma.karelogic.net](#)



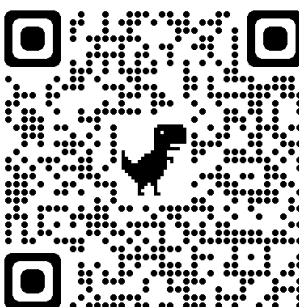
[Curso básico de Lenguaje C/C++; Prof. Gilberto Reyes B.](#)



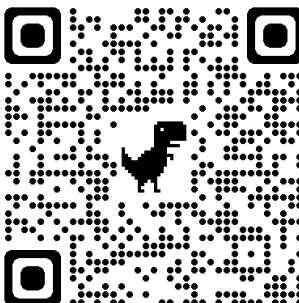
Tutorial de cómo utilizar el evaluador de programas en C/Cpp



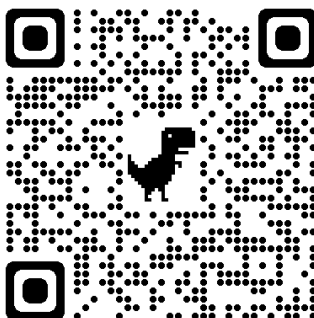
[Ver las Instrucciones básicas](#)



[Descarga el manual del Usuario de plataforma.karelogic.net](#)



[Todos los videos tutoriales](#)



Guía instruccional para navegar y avanzar en nuestra plataforma en la web; plataforma.karelogic.net

Prólogo.

En los últimos años las nuevas generaciones de jóvenes (Millennials y Generación Z) se caracterizan, entre otras cosas, por ser pioneras en el uso de las tecnologías de la Información y la Comunicación (TIC).

La tecnología digital se ha convertido para nosotros en una continuación real y virtual de nuestras vidas, otorgándoles un protagonismo especialmente importante.

El uso de dicha tecnología permite el desarrollo de una serie de habilidades, competencias y talentos nuevos y desconocidos hasta ahora, lo que hace pensar en planteamientos innovadores para afrontar el futuro (blockchain, conocimiento colaborativo, e-learning, etc.)

Para un proceso más eficaz en el desarrollo del talento de los estudiantes necesitamos herramientas, ámbitos educativos y entornos donde, además de las aptitudes, es fundamental desarrollar las actitudes y habilidades (digital skills) inherentes a la transformación digital, entre las que destacan la creatividad y la lógica para el entendimiento y solución de problemas a través del uso de lenguajes de programación por medio de la computadora.

Los talentos surgen de la habilidad general como una confluencia de las disposiciones genéticas, de las experiencias familiares y escolares, de los intereses específicos de los estudiantes y de los estilos de aprendizaje, Gagné (1985) bosqueja un modelo general para el desarrollo del talento en los jóvenes; y otros investigadores (Bloom, 1985; Gagné, 1985; Keating, 1979; Mackinnon, 1978; Tannenbaum, 1983; Taylor, 1978; Reis y Renzuilli, 1986) han explicado la naturaleza y desarrollo de los talentos.

Pero en todo este contexto es de mucha utilidad contar con una herramienta educativa ya probada y que granice el lograr identificar y desarrollar el talento de nuestro niños, niñas y jóvenes, sean o no ellos grandes talentos.

Es por lo anterior que hemos desarrollado una plataforma 24/7 en la nube, que permite identificar y darle seguimiento al desarrollo del talento en nuestros estudiantes de los niveles básico, medio y medio superior de nuestro país.

Antecedentes.

Karel el Robot es un robot virtual, al que se le puede dar órdenes a través de un programa (conjunto de ordenes o instrucciones), está dirigido a estudiantes que quieren desarrollar la habilidad de programar, dando al iniciado sólidas bases en cuanto al diseño en un ambiente de programación estructurado en un formato semejante al Java. Karel el Robot fue creado por Richard E. Pattis en su libro Karel The Robot: A Gentle Introduction to the Art of Programming ('Karel el Robot: Una agradable introducción al arte de la programación'). Pattis usó el lenguaje en sus clases en la Universidad Carnegie Mellon, con el objetivo de que sus estudiantes aprendiesen a pensar de manera ordenada y eficiente al programar a un robot virtual. El lenguaje toma su nombre del escritor checo Karel Čapek, quien creó la palabra robot (que significa esclavo) para su obra teatral R.U.R. (Robots Universales Rossum, 1920).

El Robot Karel fue introducido en 1981 por el profesor Richard Pattis, de la Universidad de Stanford, con el propósito de desarrollar en el estudiante la capacidad de abstracción, la noción de orden, de secuencia y la habilidad de programar en forma estructurada y modular. Su aplicación es tan sencilla y efectiva que actualmente es utilizado en las diferentes Olimpiadas de Informática de México a nivel Estatal y Nacional; con el propósito de incrementar el nivel de programación de los competidores. Utiliza un lenguaje semejante al formato de Java. Para que se efectué una tarea, programaremos al Robot Karel paso por paso con comandos básicos y sentencias de control, en donde comprobaremos en forma visual la realización de cada una de las órdenes interpretadas por el Robot Karel. Su lenguaje de programación es muy sencillo y fácil de emplear, esto permite que su aprendizaje sea más efectivo y ayuda a estructurar el pensamiento lógico de los estudiantes al momento de crear, aplicar y confirmar los procedimientos en base a los resultados obtenidos en cada una de las tareas efectuadas por el Robot Karel.

Introducción.

En el punto anterior explicamos algo de historia del Robot Karel, su creador y sus características principales, a continuación, les compartimos algo de los antecedentes de esta plataforma en la nube, cómo surgió la idea, principales logros y el porqué de la garantía de resultados en el desarrollo del talento a través de la Robótica virtual y la programación.

Breve semblanza del autor de los contenidos y de la idea original

El Profesor Gilberto Reyes Barrera es maestro de programación desde hace 50 años, en los niveles de primaria, secundaria, bachillerato y del nivel superior. Desde el 2003 es el delegado estatal de la Olimpiada de Informática en Nuevo León. Ha tenido la fortuna de entrenar a generaciones de niños y niñas para desarrollarles su talento a través de la Robótica Virtual y la programación, quienes han logrado éxitos importantes en las competencias de programación a nivel estatal, nacional e internacional, destacando el principal logro, el de Diego Alonso Roque Montoya, de Monterrey, NL, que consiguió la primer Medalla de ORO para México en la Olimpiada Internacional de Informática en el 2014 en Taipéi Taiwán. A Diego Alonso lo entreno el Profesor Gilberto desde el quinto año de primaria, precisamente con los dos lenguajes de programación que se utilizan en esta plataforma en la nube (plataforma.karelogic.net), estos dos lenguajes son “El Robot Karel” y el “Lenguaje C/Cpp”. Los materiales didácticos, videotutoriales, ejemplos y ejercicios de esta plataforma representan la experiencia de esos 50 años como maestro de programación del Profesor Gilberto Reyes.

Creador y desarrollador de la plataforma en la web plataforma.karelogic.net

Rubén Navarro Castillo, exolímpico Medallista nacional en la Olimpiada De Informática 2010

Muy bien, empecemos...

Objetivo de esta guía Instruccional y Manual de usuario.

Todo este material que vas a estudiar a continuación es un resumen de las características e instrucciones principales del Lenguaje el Robot Karel, así como una explicación detallada del entorno del simulador y editor de programas de Karel para la solución de ejemplos y problemas que serán probados y evaluados en vivo. **Te recomendamos estudiar muy bien todo este material antes de comenzar a utilizar la plataforma de plataforma.karelogic.net.**

El objetivo de este manual del usuario es que te sirva como guía y base para conocer y entender los aspectos básicos de la plataforma, de su entorno y cómo funcionan las instrucciones del Robot Karel, además de que conozcas como puedes hacer que Karel procese e interprete las instrucciones (código) que le darás en cada programa para cumplir con la tarea especificada en cada ejemplo o ejercicio a resolver. También te darás cuenta como el simulador te estará mostrando, paso a paso, la manera como Karel estará interpretando cada orden o instrucción de tu programa. Te darás cuenta como el simulador de Karel prueba, revisa y evalúa tu programa para confirmar si resuelve correctamente cada ejemplo y ejercicio, y te dará puntos por cada programa correcto, esto puntos los estarás acumulando para que veas tu avance de cada sesión en la plataforma.

Objetivo y características principales de la plataforma en la web

El objetivo de la plataforma de plataforma.karelogic.net es ofrecer un sitio en la web, 24/7 como una herramienta eficaz para desarrollar el talento de niños, niñas y jóvenes a través de la Robótica virtual y la programación. Motiva al participante dado que le va mostrando su avance de cada sesión, también lo apoya por múltiples videotutoriales y ejemplos ya resueltos para explicarle el funcionamiento de las instrucciones al robot y de cómo programarlo para realizar tareas específicas de cada ejemplo y ejercicio. Es decir, que se trata de programar a un robot virtual en un mundo intuitivo a través de ordenes imperativas.

La plataforma está dividida en unidades, cada unidad contiene lecciones y cada lección contiene ejemplos resueltos y ejercicios para resolver.

Las unidades de la cero a la cuatro contienen todo lo relacionado con el mundo, el lenguaje del Robot Karel, y su simulador y evaluador automático.

La Unidad 5 es todo un curso básico del Lenguaje C/Cpp, que también está dividida en lecciones, y cada lección contiene ejemplos a probar y ejercicios a resolver.

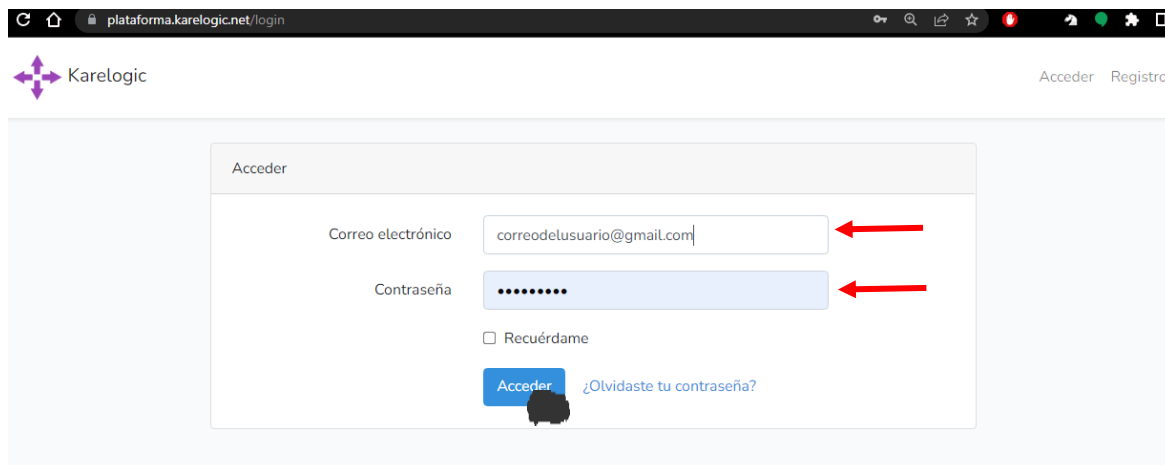
La Unidad 10, está integrada por lecciones con materiales didácticos, ejemplos y ejercicios a resolver sobre temas de Programación Competitiva (en Lenguaje C/Cpp).

En el siguiente enlace puedes ver el videotutorial donde se explica cómo utilizar esta plataforma:
<https://youtu.be/HC6PyiE17KE>

Primeros pasos para trabajar, navegar y avanzar en esta plataforma en la nube



1. Ingresas a la página principal de <https://plataforma.karelogic.net>
2. Das “click” en “Iniciar sesión”, y aparece la siguiente pantalla:

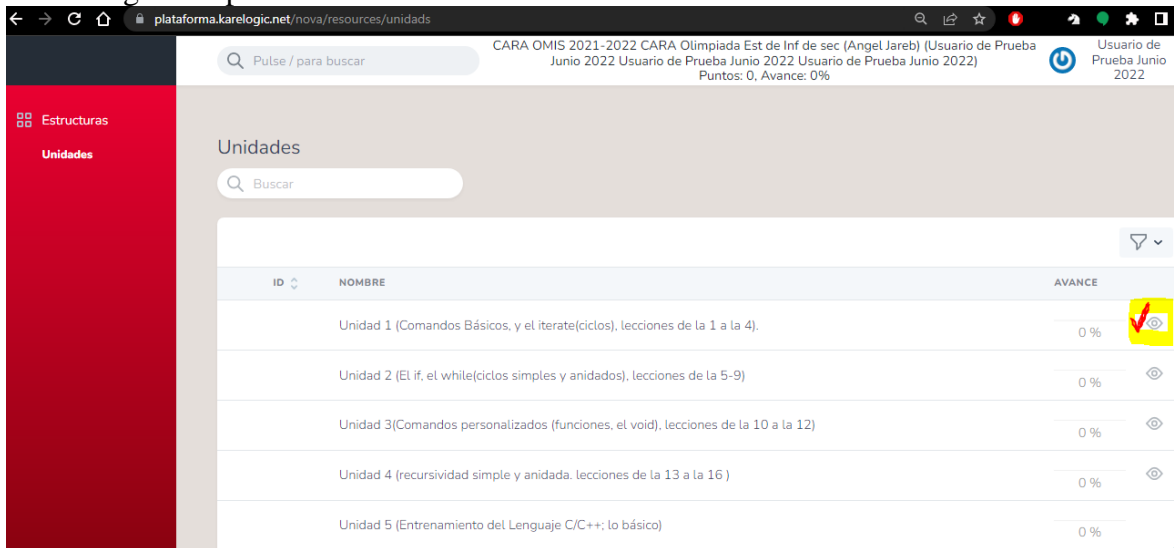


MANUAL DEL USUARIO PLATAFORMA.KARELOGIC.NET

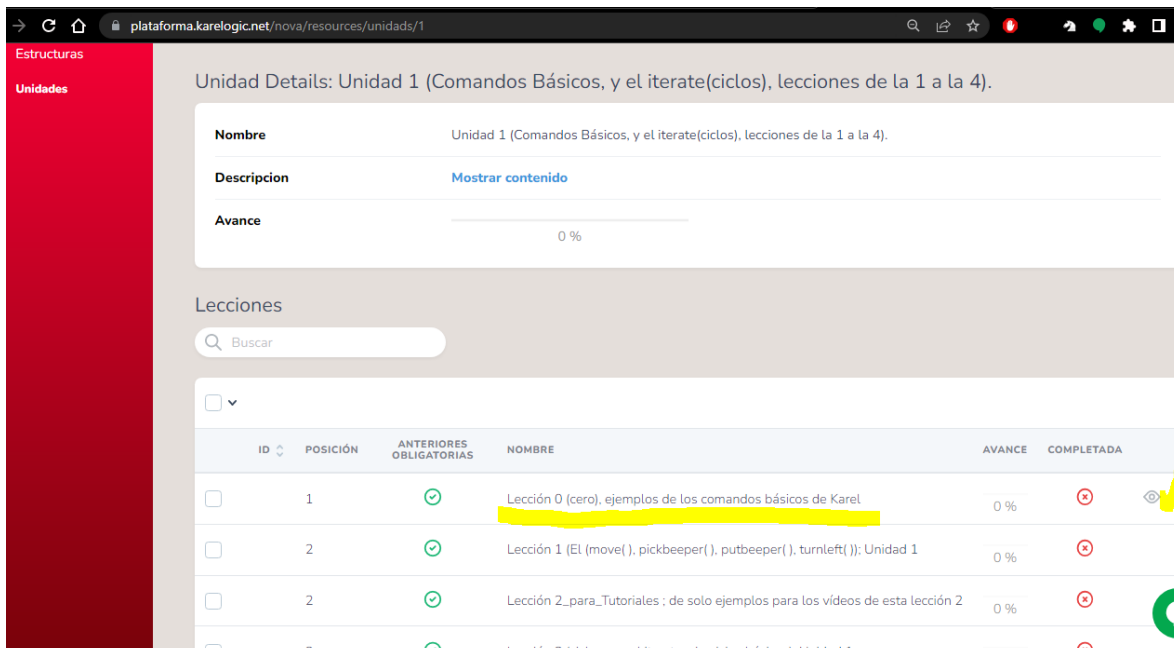


Aquí ingresas con el correo con el que estas registrado, así como tu contraseña. Das “click” en el botón “Acceder”, y se te mostrara la siguiente pantalla:

3. Das “click” en la opción de más a la izquierda, donde dice “Unidades”, se te mostrara la siguiente pantalla:



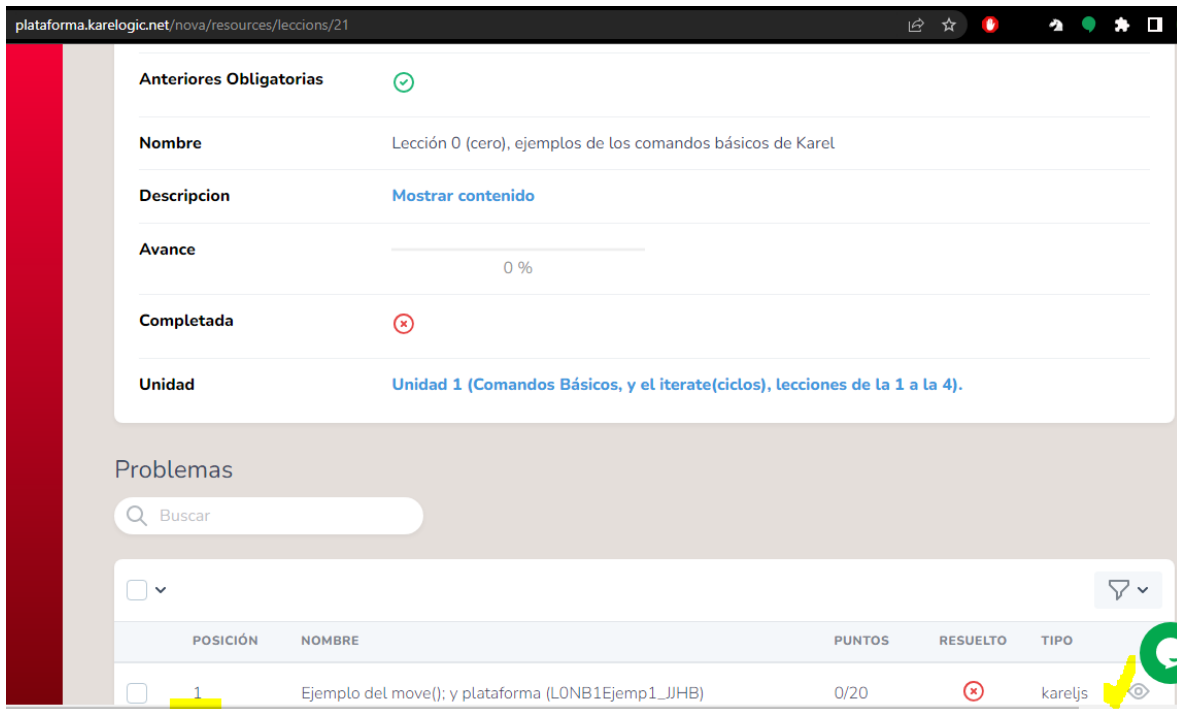
4. Das “click” en el “ojito” que se encuentra en el extremo derecho de donde dice “Unidad 1”, y



MANUAL DEL USUARIO PLATAFORMA.KARELOGIC.NET

se te mostrara la siguiente pantalla:

5. Das “click” en el “ojito” que se encuentra en el extremo derecho de la “Lección 0”, se te



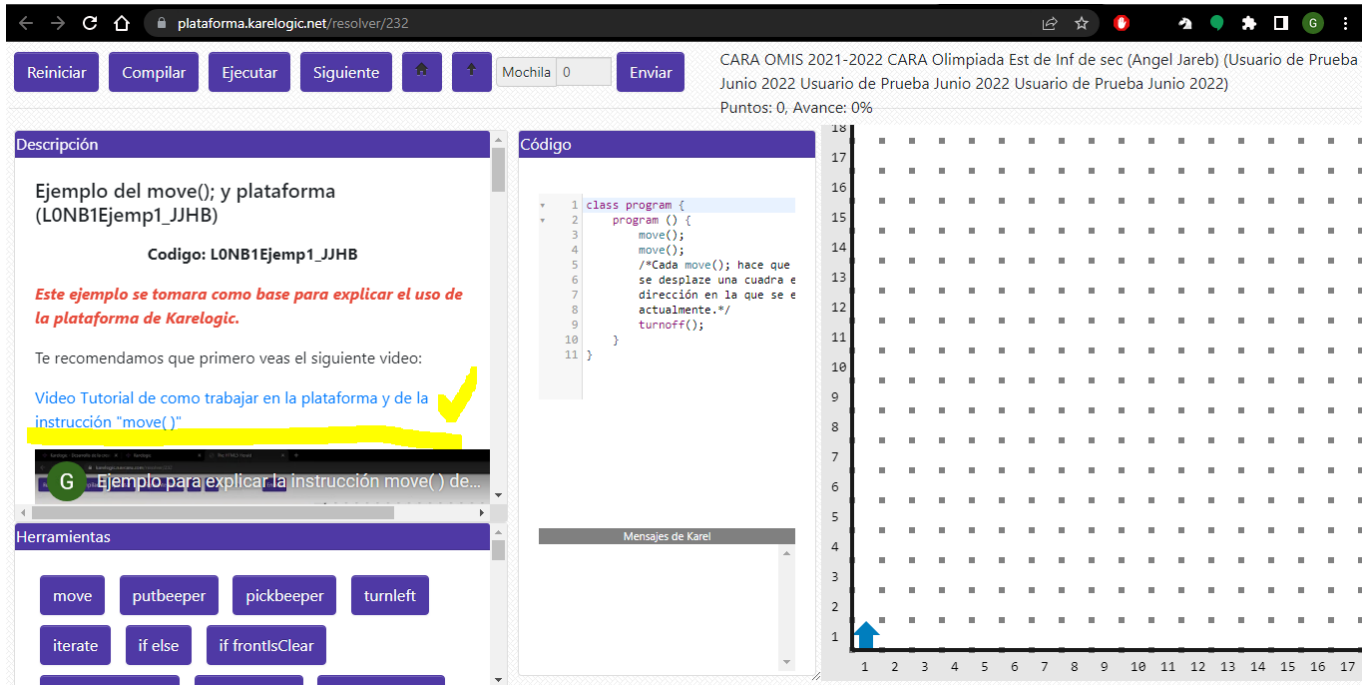
mostrara la siguiente pantalla:



6. Das “click” en el “ojito”, en el extremo derecho, del ejemplo #1, se te mostrara la siguiente pantalla:

MANUAL DEL USUARIO PLATAFORMA.KARELOGIC.NET

Das “click” en el enlace que dice “Ir a Evaluador”, y se mostrara la siguiente pantalla:



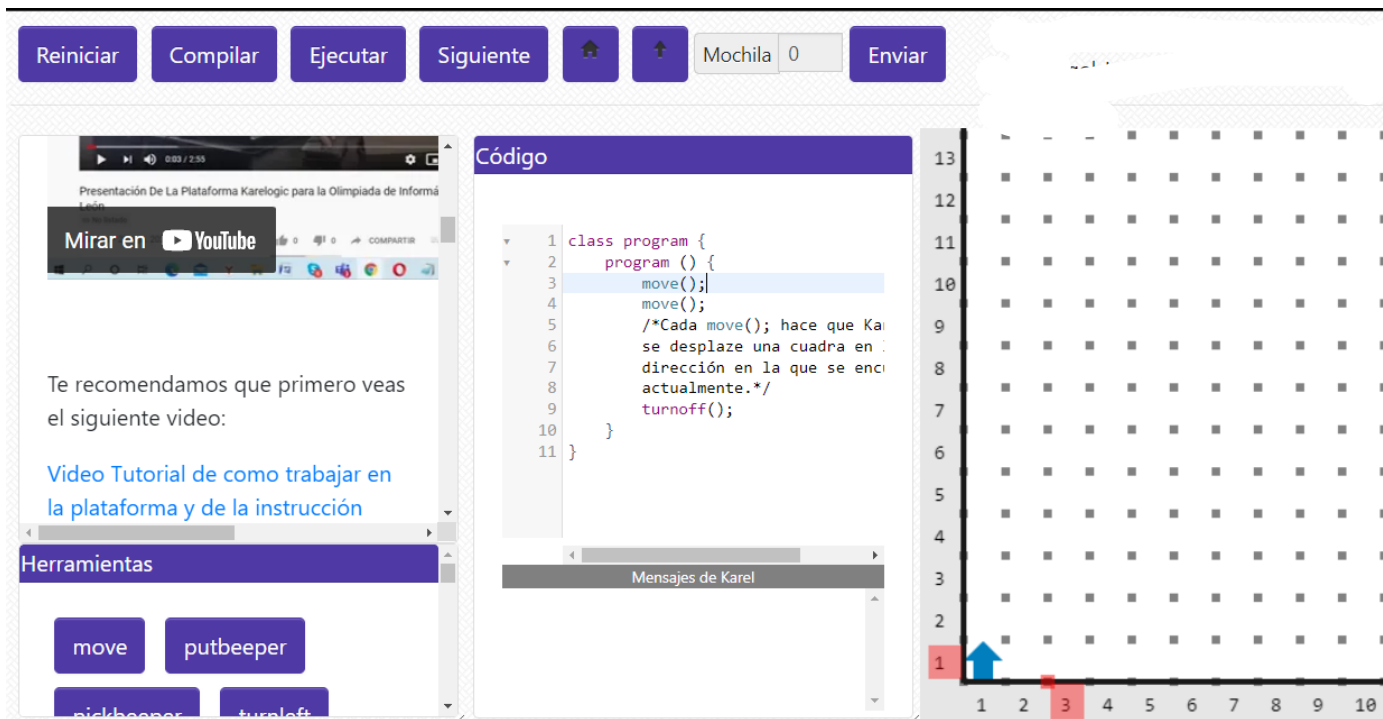
En la imagen anterior puedes ver el enlace de un videotutorial, al dar “click” en ese enlace se muestra el video donde se explica con más detalle como trabajar y avanzar en esta plataforma en la nube 24/7. Aquí también te comparto ese mismo enlace del videotutorial <https://youtu.be/HC6PyiE17KE>

Descripción del simulador y evaluador de Karel.

La plataforma donde manejamos a Karel el Robot es un simulador dirigido a principiantes en el estudio y aprendizaje de aprender a programar y de los lenguajes de programación, esta plataforma y este proceso les brinda sólidas bases en cuanto al diseño y estructuración del código para resolver problemas. En *Karelogic* utilizamos el formato lenguaje de programación Java.

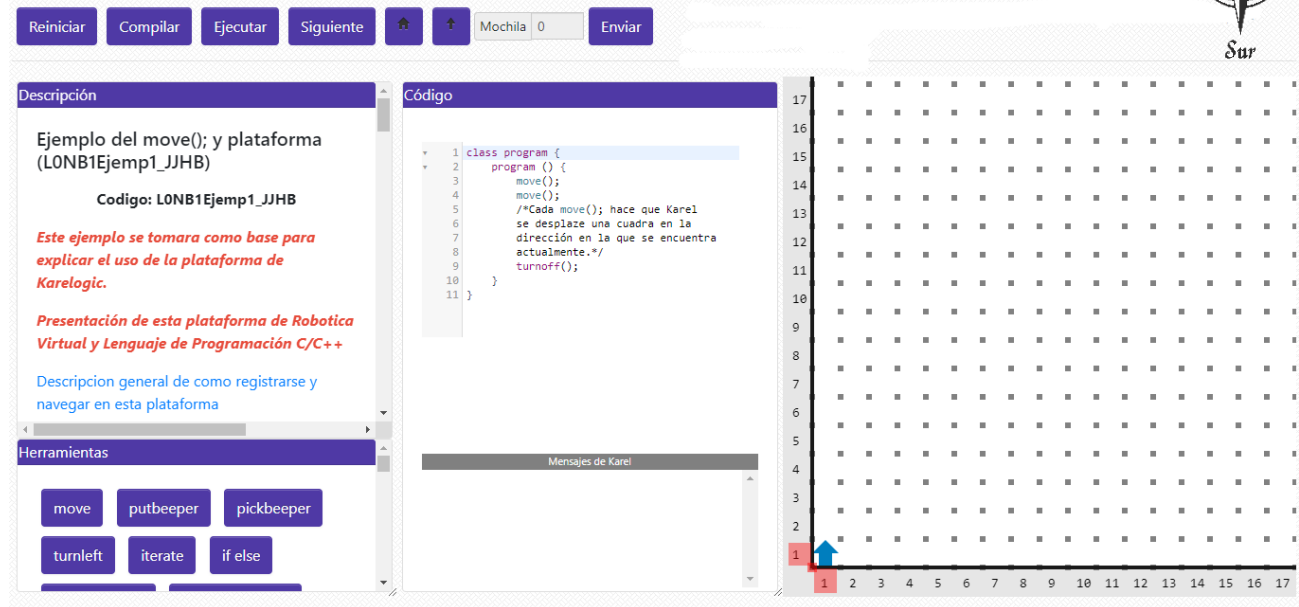
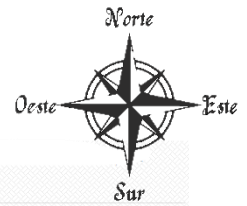
Hablando sin centrarnos mucho en cada elemento del simulador, podrás observar que hay diferentes áreas que te servirán para interactuar con este simulador Karel, resolver los ejercicios y entender los ejemplos que se te darán dentro de la plataforma.

El área de **descripción** que contiene material de ayuda y la redacción del ejercicio o ejemplo, el área de **código (o del programa)**, en donde podrás estudiar y visualizar los ejemplos resueltos, o bien, resolver los ejercicios dentro de la plataforma de *Karelogic*; también se encuentra el **área de herramientas** que te ayudará a tener una redacción impecable a la hora de estar escribiendo tu código-solución, y por último dentro del simulador, encontrarás el mundo de Karel.



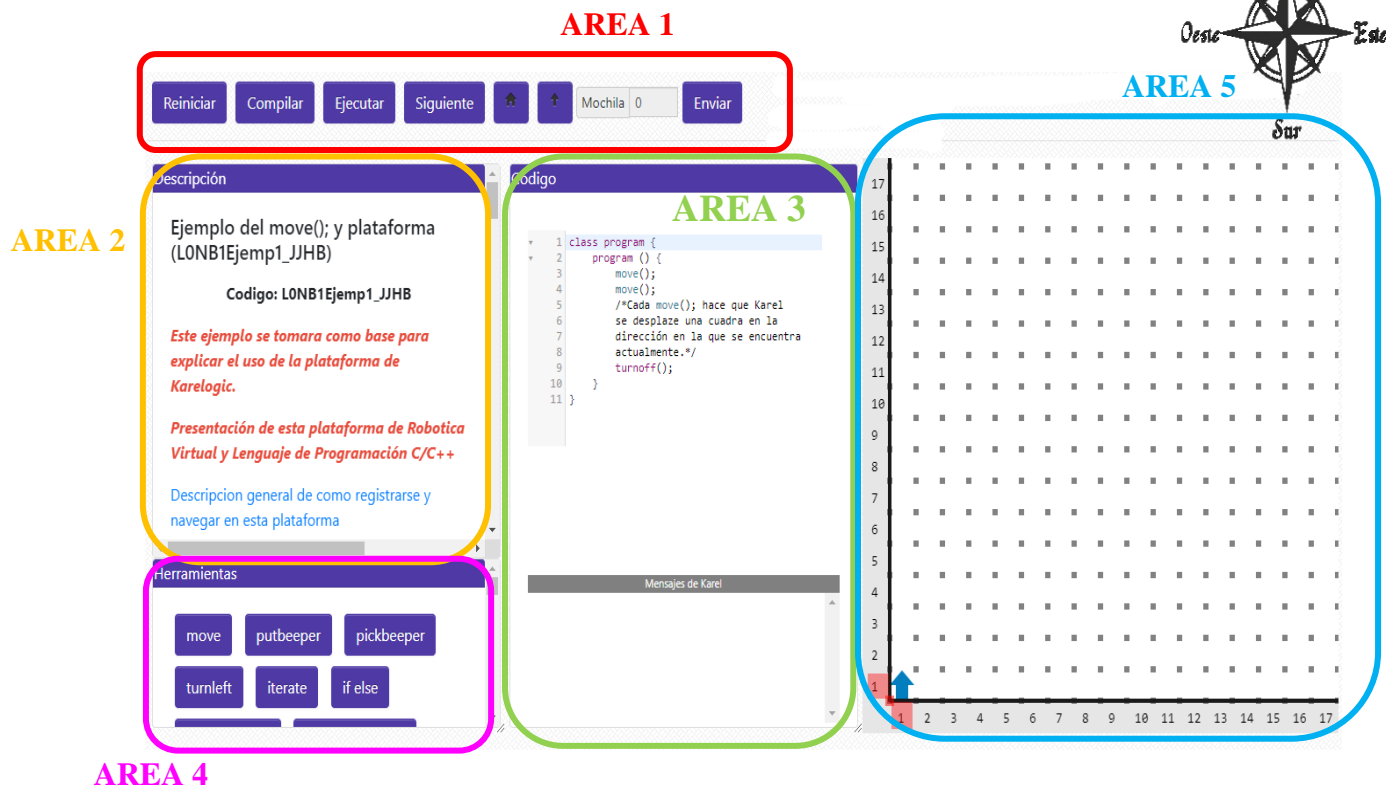
El mundo del Robot, su simulador, sus componentes, elementos y características

En la siguiente imagen podemos observar el escenario completo del **simulador** y **evaluador** de problemas y programas en **Lenguaje del Robot Karel**.



A continuación, veremos para que nos sirve cada uno de estos **elementos** que se encuentran en las **5 ÁREAS** del **simulador**.

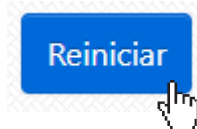
Comencemos reconociendo estas **5 ÁREAS** las cuales son **muy importantes** y **útiles** dentro del **evaluador** y **simulador**:



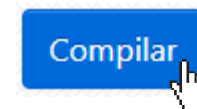
Una vez teniendo identificadas todas las cinco áreas, te explicaremos para que sirven, lo que verás y podrás hacer con cada una de ellas dentro del **simulador** y **evaluador** de problemas y programas en **Lenguaje del Robot Karel**.

AREA 1: En esta área se encuentran **7 botones** y la **mochila** del **Robot Karel**. Cada uno de estos botones te permitirá **interactuar** con el código-solución (**AREA 3**) o con el mundo de Karel (**AREA 5**). En seguida te los presentamos:

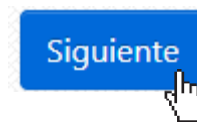
AREA 1



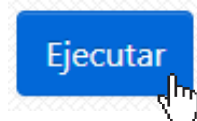
Botón Reiniciar: Sí colocas el cursor sobre el botón, dentro del simulador, podrás leer un pequeño recuadro **“Regresar el mundo a su estado original”** y para eso mismo funciona este botón, después de haber **ejecutado** tu código y que Karel se haya movido de su lugar original para realizar las tareas, puedes darle clic al botón Reiniciar y automáticamente el mundo **regresa a como se encontraba antes de iniciar su tarea**.



Botón Compilar: Este botón nos ayudará mucho una vez que vayamos escribiendo nuestro código-solución, pues al darle clic nos **revisará inmediatamente** cualquier **error** dentro de éste y nos los dirá en la parte inferior del **AREA 3** (más adelante hablaremos de esa parte) para poder corregirlo y seguir con la solución. Puedes usarlo cuantas veces quieras y sean necesarias, recuerda presionarlo **antes de ejecutar** tu código.



Botón Siguiente: Si lo que necesitas es **revisar paso por paso** las acciones de Karel y **línea por línea** tu código-solución, te sugerimos usar el botón Siguiente, con este podrás **visualizar detenidamente** cada movimiento que Karel realice. Muy recomendado para **realizar pruebas** porque además de que va paso por paso, este botón lo puedes usar cada que lo necesites; es muy útil para revisar si las instrucciones que le estas dando a Karel son las correctas.



Botón Ejecutar: Cuando tengas tu **código**, o una parte, **compilado y corregido** podrás usar el botón Ejecutar. Después de darle clic podrás **observar** como **en el mundo** (**AREA 5**) el robot **Karel comienza a seguir las ordenes** que le diste **mediante el código**. El botón Ejecutar funciona aunque no tengas el código terminado, pero tienes que ser muy observador, ya que puede ser algo **rápido**.

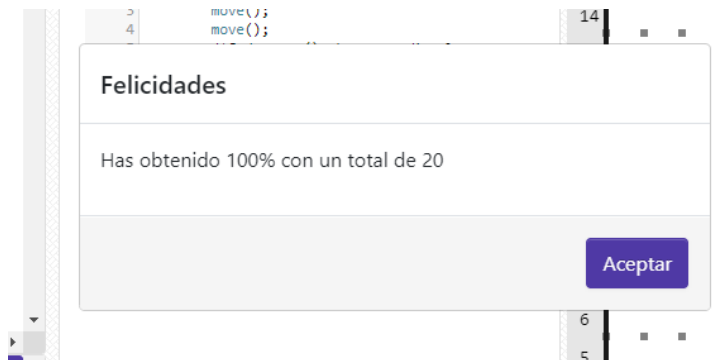
NOTA: Recuerda que antes de utilizar los botones **Ejecutar** y **Siguiente** tu código-solución debe estar **compilado** y no tener ni un solo error, y **no olvides** que, si quieres volver a utilizarlos, debes darle clic al botón **Reiniciar**, **de otra forma no podrás volver a ejecutar el programa con ninguno de los dos botones (botón Ejecutar y botón Siguiente).**



Recuadro “Mochila”: Más Adelante te explicaremos lo que es la mochila de Karel y que es lo que guarda en ella. **Por el momento:** En el recuadro de la izquierda **podrás observar cuantos trompos/beepers/zumbadores lleva Karel en la mochila.** Cuando se encuentra un -1 dentro del recuadro significa que hay infinitos zumbadores dentro de la mochila.



Botón Enviar: Este botón es muy importante, pues **una vez terminado y probado** tu código-solución de forma correcta **deberás presionar el botón Enviar para que tu trabajo sea calificado y que tu calificación sea subida a la plataforma de Karelogic.** Después de darle clic al botón Enviar, te saldrá este mensaje dentro de la ventana del simulador:



Deberás darle clic en aceptar y podrás cerrar la ventana del simulador para volver a Karelogic

AREA 2: Dentro de esta área podrás tener acceso a diferentes **videos explicativos** (o los enlaces a éstos), **sobre la plataforma** y los **comandos básicos** que necesitaras para **solucionar los ejercicios**. Los **videos te servirán como apoyo y ayuda para que no haya ninguna duda**. En este mismo apartado podrás encontrar **tablas y explicaciones más detalladas** que tendrás que leer para ir conociendo el mundo de Karel y cómo funciona. Y, por último, pero **muy importante**, aquí mismo estará la **descripción del ejemplo o ejercicio a resolver**, como también dos imágenes “Mundo inicial” y “Mundo Final”, y algunas consideraciones o notas que te ayudarán en los problemas.

Descripción

Ejemplo del move(); y plataforma (LONB1Ejemp1_JJHB)

Codigo: LONB1Ejemp1_JJHB

Este ejemplo se tomara como base para explicar el uso de la plataforma de Karelogic.

Presentación de esta plataforma de Robotica Virtual y Lenguaje de Programación C/C++

Descripción general de como registrarse y navegar en esta plataforma

Herramientas

AREA 2

Video Tutorial de como trabajar en la plataforma y de la instrucción "move()"

G Ejemplo para explicar la instrucción move()

Mirar en YouTube

Esto lo encontrarás dentro del área 2

Videos explicativos

Tablas de información

Explicaciones más detalladas sobre los elementos de la plataforma

Tabla de Instrucciones Básicas que Karel Puede Ejecutar

Instrucción	Descripción
move();	Karel avanza una cuadrada.
putbeeper();	Karel levanta un trompo o beeper.
putbeeper();	Karel deja un trompo o beeper.
turnleft();	Karel gira 90° hacia su izquierda.
turnoff();	Karel se detiene y se apaga.

Descripción de este ejemplo:

La tarea de Karel es avanzar dos cuadradas en la dirección en que inicia su tarea.

Karel comienza mirando al norte en la posición (4,4).

Sobre la Mochila de Karel:

Ya hemos hecho referencia a la mochila del Robot Karel, pero hace falta conocer cómo utiliza Karel su mochila de trompos o zumbadores al iniciar una tarea.

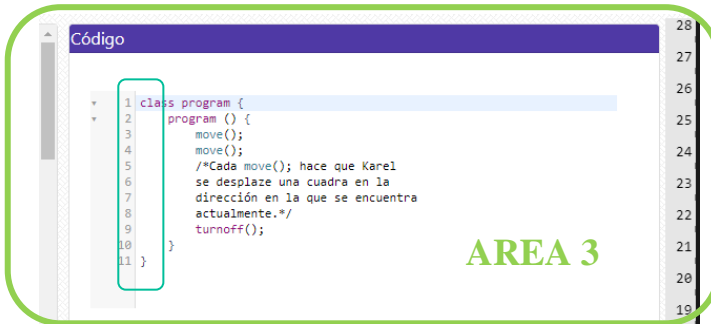
Mundo Inicial de ejemplo:

AREA 3: En el **AREA 3** es donde **escribirás o podrás ver el código-solución** del ejercicio o ejemplo que se encuentra en el **área de descripción (AREA 2)**, aquí mismo se encuentra el lugar de los “**Mensajes de Karel**”.

Además, en esta **AREA** es donde **podrás darle las instrucciones a realizar a Karel y Karel podrá leerlas e interpretarlas para realizarlas una por una dentro de su mundo**, por eso es importante prestar mucha atención a la hora de escribir tu código-solución.

A continuación, te **explicaremos** cada uno de los **elementos del AREA 3:**

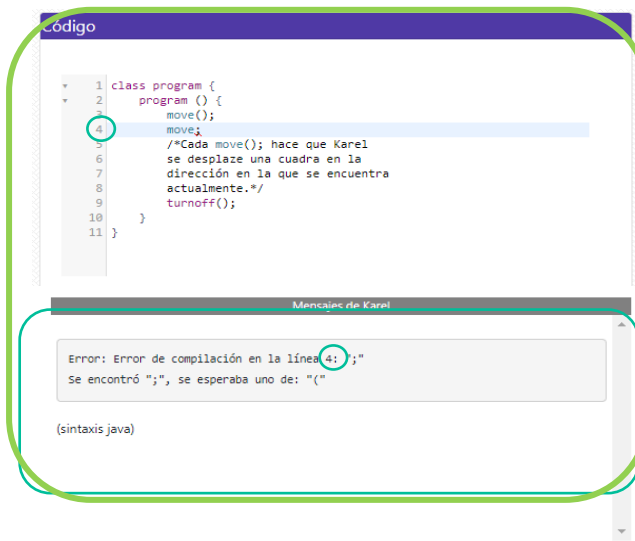
- **Código:** Cómo se mencionó anteriormente, este espacio **lo utilizarás para escribir, probar y corregir el código-solución que necesita Karel para completar la tarea** que se le asigna según la descripción del ejercicio o ejemplo.



Si bien, puedes observar que **cada línea** dentro del recuadro **se encuentra enumerada** al lado izquierdo del código, esto **te servirá para identificar el número de línea de los posibles errores**, y así, tener **orden** a la hora de estar ingresando la solución.

- **Mensajes de Karel:** Cuando te hablamos sobre el botón “**Compilar**” mencionamos que **después de compilar, Karel nos dirá el error o errores que cometimos al estar escribiendo el programa**, o bien, si es que está correcto nuestro programa no aparecerá ningún mensaje de error. Te mostraré un ejemplo:

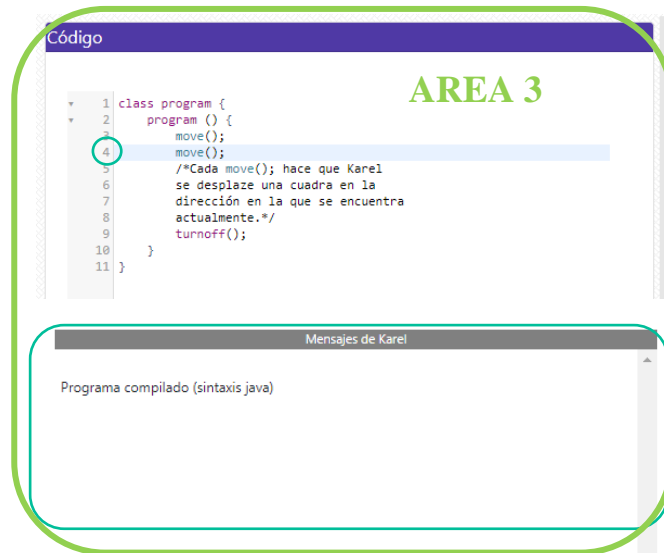
AREA 3



En la imagen se puede observar cómo **después de haber compilado mi código**, en la parte de abajo (“**Mensajes de Karel**”) me marca el error que cometí, en la línea 4 (la línea resaltada en azul) se encuentra escrito de manera equivocada la instrucción **move()** (más adelante te explicaremos a detalle para que sirve cada instrucción y como se debe escribir correctamente). En este ejemplo el error está en no haber colocado los paréntesis al final del “move” y en el mismo mensaje lo menciona: *Se encontró “;”, se esperaba uno de: “(”*

Ahora, corregiré mi error para mostrarte el **mensaje que debe enviar Karel cuando después de compilar, el código de nuestro programa no tiene ni un solo error:**

Después de haber **corregido el error en la línea 4** y haber agregado los paréntesis faltantes, el mensaje que ahora aparece en el recuadro **Programa compilado** significa **que nuestro programa está escrito correctamente y sin ningún error en la sintaxis** (error en la escritura).



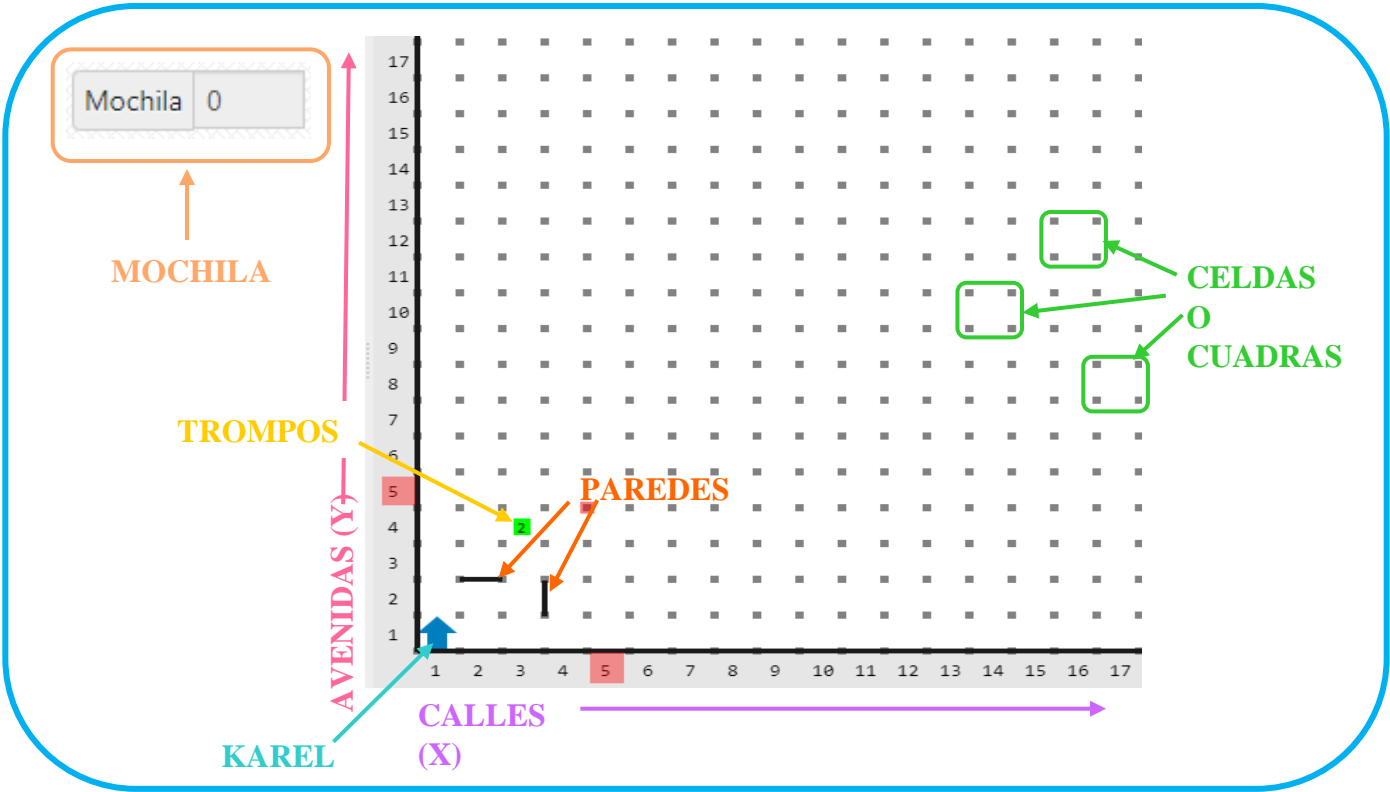
AREA 4: En el **AREA 4** encontrarás unos botones de color azul, que le llamamos "Herramientas". Estos botones te ayudaran **a escribir tu programa correctamente y así tendrás una codificación o código sin errores**. Solo necesitarás colocar el cursor en la línea del código (**AREA 3**) en la que quieras ingresar la orden para Karel, ir al recuadro de herramientas (**AREA 4**), buscar el botón de la herramienta que necesitas y presionarlo; una vez que lo presiones se escribirá correctamente dentro del código, ya solo deberás dar "enter" para continuar en la siguiente línea.



Más adelante te explicaremos la función de cada una de las instrucciones que contiene el cuadro de herramientas, cómo y cuándo utilizarlas.

AREA 5: El **AREA 5** es otra de las áreas **más importantes**, es el mundo y el simulador de Karel,

AREA 5



el lugar en donde observaremos si es que nuestro código-solución es el adecuado para realizar lo que nos pide (o muestra, en el caso de los ejemplos) la descripción.

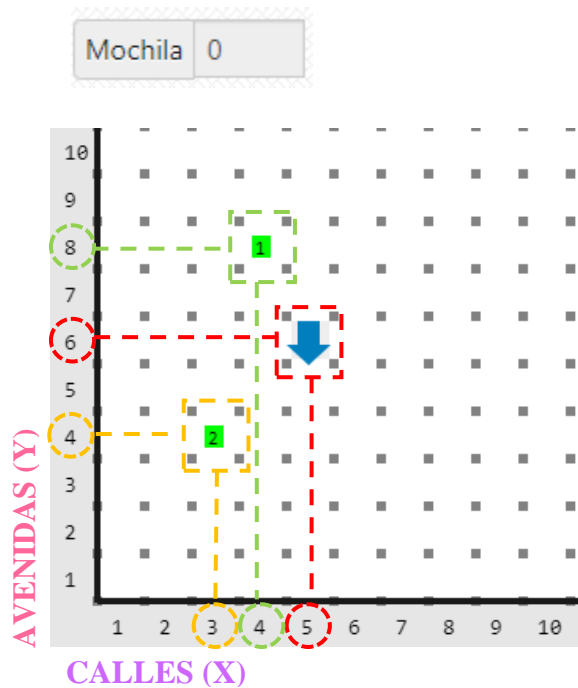
Como puedes observar, Karel vive en un mundo bastante simple. Ahora que tenemos identificados los elementos del **AREA 5** te los presentaremos junto a una breve descripción y cuál es su función dentro del mundo de Karel:

Elemento	Descripción
a) Robot KAREL	Karel se encuentra representado por la flecha azul que se encuentra en forma inicial, en algunos ejemplos, en la parte inferior derecha del mundo (1,1). Karel puede también iniciar en otros lugares, pero dentro de <i>Karelogic</i> su punto de partida, generalmente, será en las coordenadas (1,1).
b) MOCHILA de trompos	Durante la ejecución del programa, cada vez que se le ordena al Robot recoger o levantar un trompo, Karel lo coloca en su mochila de trompos . También se le puede ordenar al Robot que, en la esquina donde se encuentre parado, deje un trompo de los que se tiene en la mochila.

c) CALLES (X)	El mundo donde trabaja Karel está compuesto por líneas horizontales y verticales que sirven de referencia para saber su ubicación. Las líneas verticales , paralelas al eje Y, son llamadas CALLES o bien “posición de Karel en X”
d) AVENIDAS (Y)	El mundo donde trabaja Karel está compuesto por líneas horizontales y verticales que sirven de referencia para saber su ubicación. Las líneas horizontales , paralelas al eje X, son llamadas AVENIDAS o bien “posición de Karel en Y”
e) CELDAS O CUADRAS	Las CELDAS O CUADRAS están representadas por cuatro pequeños cuadritos grises en forma de un cuadrado, y se encuentran ubicadas en las intersecciones o uniones entre cada CALLE(X) y AVENIDA(Y) . Nos ayudan a colocar las PAREDES dentro del mundo de Karel.
f) PAREDES	Otro componente importante dentro del mundo de Karel son las PAREDES estas nos sirven como obstáculos para Karel , ya que, si Karel encuentra una pared en su camino Karel tendrá que rodearla, no puede atravesarlas o bien también nos sirven para simular casas, laberintos, etc. todo depende de la descripción del problema.
g) TROMPOS	Los TROMPOS , zumbadores o beepers están representados por números arábigos dentro de recuadros verdes , se encuentran dentro de una celda o cuadra del mundo de Karel . El Robot Karel puede interactuar con ellos, puede levantarlos o colocarlos, según la instrucción que le des ; por otro lado, también pueden representar lugares a los que Karel tiene que llegar, depende también de la descripción del problema. Estos TROMPOS al levantarlos se guardarán en la mochila de Karel y al dejarlos, Karel los toma de su mochila.

Consideraciones importantes acerca del mundo de Karel:

- Para Karel es **muy importante su orientación o dirección** para poder desplazarse por su mundo, entre las calles y avenidas, encontrar los zumbadores y no perderse en el intento, por lo tanto, tú también deberás tener **los puntos cardinales (Norte, Sur, Este y Oeste) muy bien aprendidos** para que no te pierdas y puedas ayudar a Karel a completar sus tareas. Pero no te preocupes, **dentro del recuadro de Descripción o AREA 2** podrás encontrar una imagen de ayuda.
- Los números que se encuentran por fuera del mundo de Karel indican el número de calle o número de avenida, y te ayudarán muchísimo; como bien lo mencionamos antes, **las líneas horizontales y verticales nos ayudarán en la ubicación de elementos como trompos, paredes o incluso para saber en donde se encuentra o donde debe apagarse Karel** y estas ubicaciones las podemos escribir como **coordenadas** gracias a estos números. Te mostraremos algunos ejemplos.



***NOTA:** En todas las coordenadas primero se escribirá el número de **CALLE (X)**, una coma y luego el número de **AVENIDA (Y)**.

Como puedes observar cada elemento tiene un número en **X (CALLE)** y un número en **Y (AVENIDA)**.

Iremos primero con **Karel**, Karel se encuentra en la **CALLE 5** y **AVENIDA 6** (orientado hacia el sur). Entonces sus coordenadas serían **(5, 6)**.

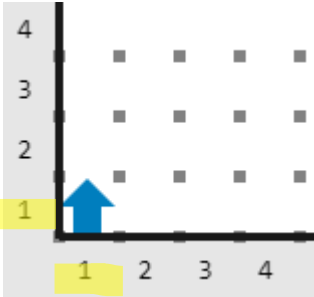
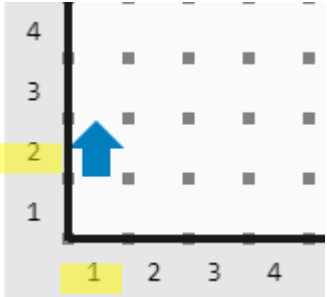
Ahora, es el turno del **trompo 1**, el trompo 1 está posicionado en la **CALLE 4** y **AVENIDA 8**. Eso significa que sus coordenadas serían **(4, 8)**.

Por último, el **trompo 2**, el trompo 2 está sobre la **CALLE 3** y la **AVENIDA 4**. Así que sus coordenadas son **(3, 4)**.

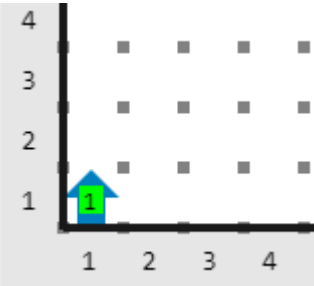
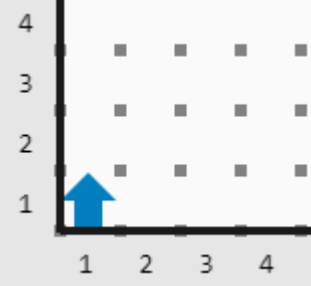
Instrucciones básicas del Lenguaje de Karel.

Ahora, te explicaremos el uso de las instrucciones básicas del lenguaje de Karel y las funciones de cada una:

- **move()** Con la instrucción de “move()” Karel se desplazará una sola cuadra en la dirección en la que se encuentra. A continuación, te mostraremos un ejemplo:

move()		
Código	Posición de Karel antes de ejecutar la instrucción	Posición de Karel después de ejecutar la instrucción
<div>Código</div> <pre> 1 class program { 2 program () { 3 move(); 4 turnoff(); 5 } 6 }</pre>		

- **pickbeeper()** Con la instrucción “pickbeeper()” Karel levantará el trompo de la esquina en la que se encuentra, y lo guardará en su mochila (pero si donde Karel está parado no hay trompo, se genera un error, y se detiene el proceso).

pickbeeper()		
Código	Situación de Karel antes de ejecutar la instrucción	Situación de Karel después de ejecutar la instrucción
<div>Código</div> <pre> 1 class program { 2 program () { 3 pickbeeper(); 4 turnoff(); 5 } 6 }</pre>	<div>Mochila 0</div> 	<div>Mochila 1</div> 

- **putbeeper()** Al ejecutar la instrucción de “putbeeper()” Karel deberá dejar un trompo en la esquina donde está parado que, necesariamente, lleva en su mochila (para que no se genere un error, debe haber, al menos, un trompo en su mochila de trompos, de no ser así el código marcará un error y se detendrá el proceso).

putbeeper()		
Código	Situación de Karel antes de ejecutar la instrucción	Situación de Karel después de ejecutar la instrucción
<div>Código</div> <pre> 1 class program { 2 program () { 3 putbeeper(); 4 turnoff(); 5 } 6 } </pre>		

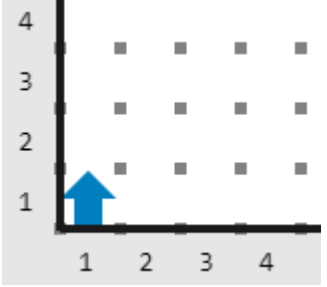
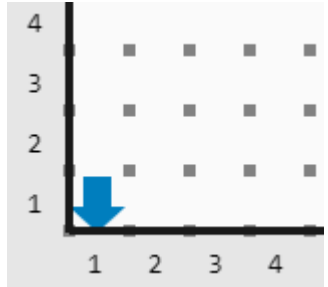
- **turnleft()** Con esta instrucción Karel girará 90° a su izquierda una sola vez, y no avanza, se queda en la misma esquina o cuadra.

turnleft()		
Código	Orientación de Karel antes de ejecutar la instrucción	Orientación de Karel después de ejecutar la instrucción
<div>Código</div> <pre> 1 class program { 2 program () { 3 turnleft(); 4 turnoff(); 5 } 6 } </pre>		

***Nota:** Karel no cuenta con una instrucción para girar a la derecha, por lo tanto, si Karel se encuentra en dirección al Norte y quieres que gire a la derecha, es decir al Este deberás utilizar tres “turnleft()”. Todo depende de a donde se encuentre viendo Karel.

Enseguida te mostraremos más detalles del “turnleft()”:

turnleft()

Código	Orientación de Karel antes de ejecutar la instrucción	Orientación de Karel después de ejecutar la instrucción
<div><div>Código</div><pre>1 class program { 2 program () { 3 turnleft(); 4 turnleft(); 5 turnoff(); 6 } 7 }</pre></div>		

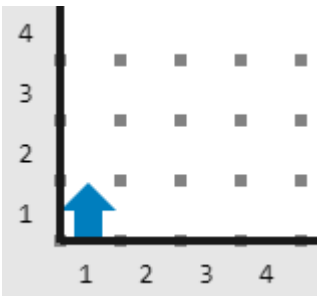
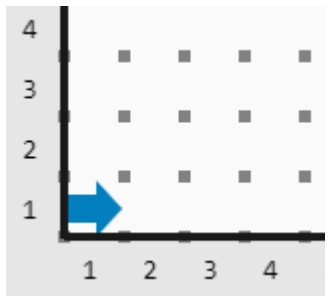


Como puedes observar, si Karel está viendo hacia el Norte, después de utilizar dos “turnleft()” terminará viendo hacia el Sur.



Como puedes observar, si Karel inicia viendo hacia el Norte, después de utilizar tres “turnleft()” terminará viendo

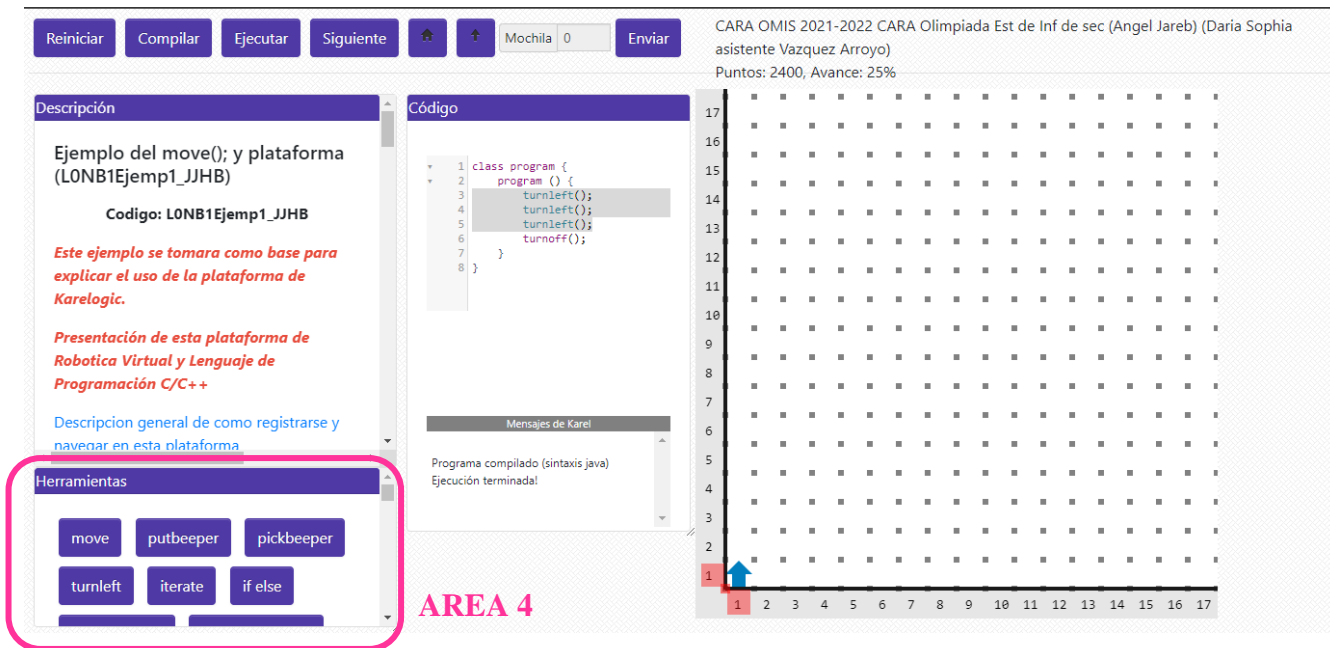
turnleft()

Código	Posición de Karel antes de ejecutar la instrucción	Posición de Karel después de ejecutar la instrucción
<div><div>Código</div><pre>1 class program { 2 program () { 3 turnleft(); 4 turnleft(); 5 turnleft(); 6 turnoff(); 7 } 8 }</pre></div>		

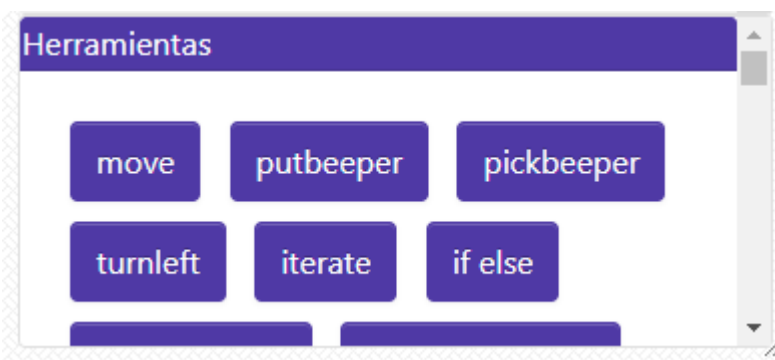
- **turnoff()** La instrucción “turnoff()” sirve para apagar a Karel, si bien, no es una instrucción que se pueda visualizar (como las anteriores). En cuanto se ejecuta, Karel inmediatamente se apaga en donde se encuentra parado y no recibe más instrucciones.

Herramientas (botones azules) de ayuda para una codificación sin errores.

Las **herramientas** de ayuda se encuentran dentro del **AREA 4** del simulador del Robot Karel, en la parte inferior izquierda:



Dentro de estas **herramientas** podrás encontrar botones con las instrucciones básicas del lenguaje del Robot Karel, también los sensores (con las condiciones que Karel puede evaluar), estatutos de control (como el “if”, “if-else”, “iterate” y el “while”), entre otras instrucciones que te serán útiles conforme vayas avanzando en la codificación de tu programa, y que también te explicaremos, más adelante, dentro de este mismo manual.



Como ya te hemos mencionado con anterioridad, estas **herramientas** te ayudarán a tener una redacción impecable y sin errores a la hora de estar escribiendo tu código-solución dentro del simulador y evaluador de Karel que utilizamos en nuestra plataforma.

Ahora, te explicaré como se utilizan y que sucede en tu código al darle clic a alguno de estos botones azules:

Para este ejemplo haremos que Karel se mueva tres cuadras hacia el Norte y deje un trompo al llegar.

Una vez que hayas entrado al simulador (a resolver un ejercicio), el area del codigo como la imagen de la derecha, algunas veces podrán contener pequeños textos explicativos, pero la mayoría de las veces se encontrará como se ve aquí a la derecha:

Código

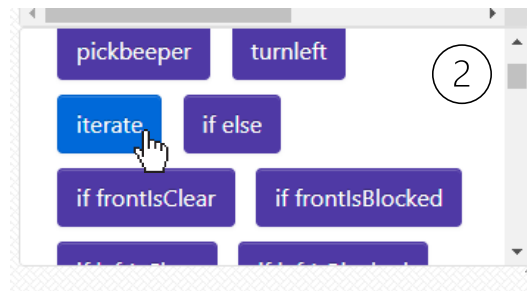
```
1 class program {  
2   program() {  
3     // Escribir código solución aquí  
4     turnoff();  
5   }  
6 }  
7  
8
```

Ahora veamos que sucede cuando le des clic al boton de “iterate”. Observa como, en la imagen de abajo, el cursor se encuentra en la linea 3 dentro del código de tu programa. Al momento de darle clic a este botón se insertará la instrucción “iterate” en la linea 3 de tu programa

Código

1

```
1 class program {  
2   program() {  
3     |  
4     turnoff();  
5   }  
6 }  
7  
8
```



Código

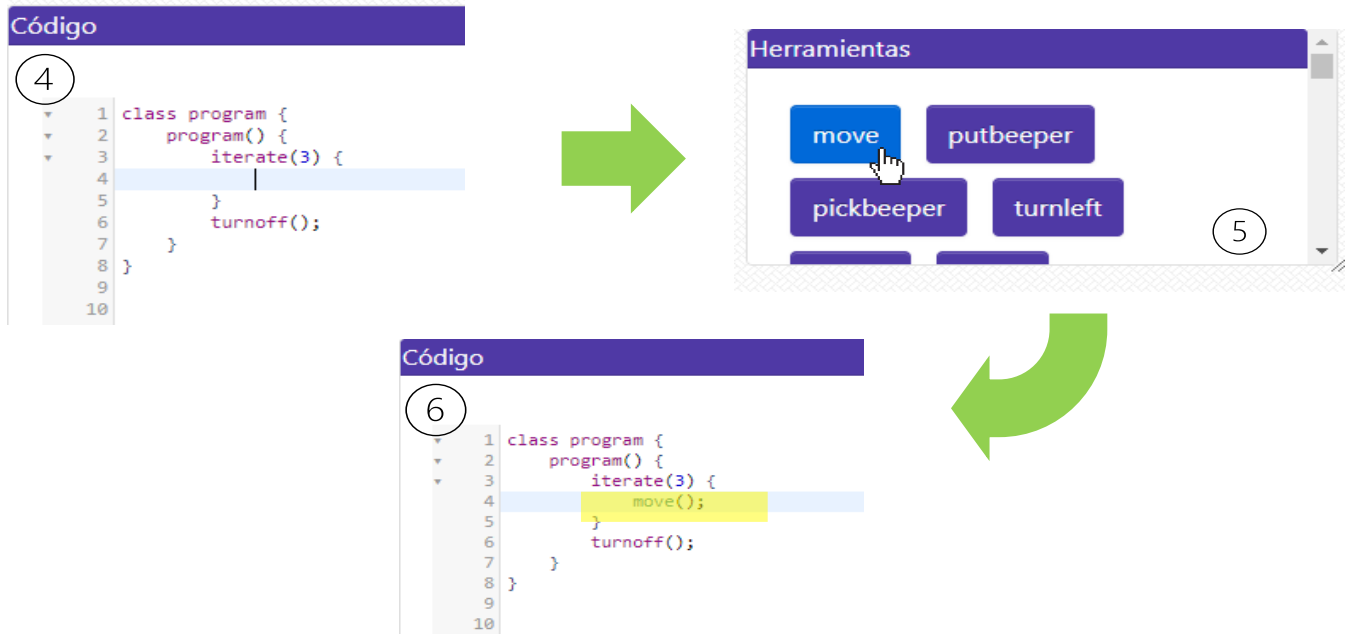
```
1 class program {  
2   program() {  
3     iterate() {  
4       |  
5     }  
6     turnoff();  
7   }  
8 }  
9  
10
```

3

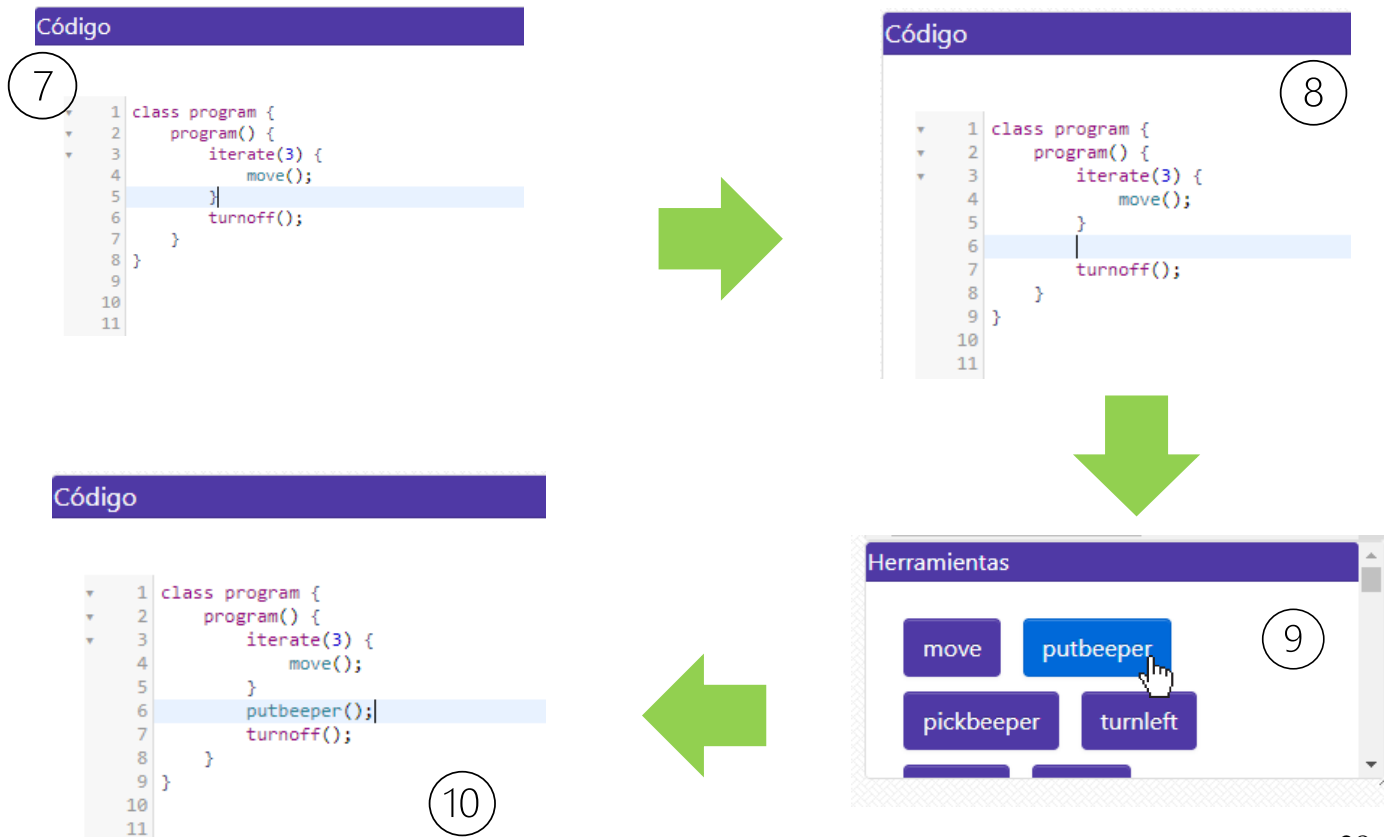


MANUAL DEL USUARIO PLATAFORMA.KARELOGIC.NET

Una vez teniendo el “iterate” (que más adelante te explicaremos la manera correcta de usarlo y para que nos sirve) y escribiendo el “3” el cual indica las veces que quiero que repita la instrucción, posicionaré el cursor en la línea 4 y observa que sucede al presionar el botón “move” de la sección de herramientas:



Para finalizar con el código-solución, colocaré el cursor en la línea 5, justo después de la llave que cierra el “iterate”, dare enter y presionaré el botón “putbeeper”:



Y así es como damos por finalizado nuestro código-solucion para este ejemplo.

Como habrás notado, las herramientas nos ayudan a no tener que estar escribiendo la instrucción y correr el riesgo de hacerlo de forma errónea, pues con solo presionar el botón correspondiente, de la sección de herramientas, nos escribe la instrucción, los parentesis y el punto y coma que le siguen.

Estatuto de control de repetición “iterate(n)”

El estatuto de control de repeticion “iterate(n)” lo puedes utilizar cada vez que quieras **ordenarle a Karel que repita una o varias veces las instrucciones que estan dentro del “iterate”**. Y a continuacion esta la **forma correcta** de su sintaxis:

Código

```

1 class program {
2   program() {
3     iterate(n) {
4       Instrucciones que se repetirán
5       "n" veces.
6     }
7   }
8   turnoff();
9 }
10 }
11 }
12 }

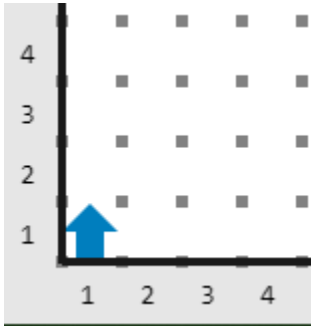
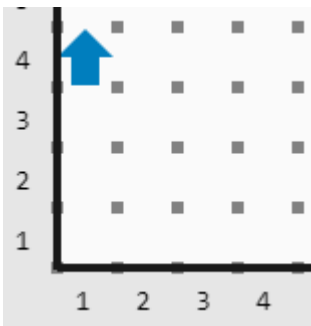
```

Al decir “n” veces nos referimos a **que puede repetirse las veces que se necesiten**.

Puedes utilizarlo cuando necesites que Karel recoja o levante grandes cantidades de trompos, que recorra muchas calles, etc.

A continuación, veamos unos ejemplos de como puedes utilizar el estatuto de control de repetición “iterate(n)”:

1. Imaginate Karel necesita **avanzar 3 cuadras hacia el Norte**.

iterate(n)			
Consideraciones	Código	Situación de Karel antes de ejecutar la instrucción	Situación de Karel después de ejecutar la instrucción
<ul style="list-style-type: none"> • Karel inicia en la coordenada (1, 1). • Karel está orientado hacia el Norte. • Karel debe terminar viendo hacia el Norte y en la coordenada (1, 4) 	<h3>Código</h3> <pre> 1 class program{ 2 program() { 3 iterate(3) { 4 move(); 5 } 6 } 7 turnoff(); 8 } 9 } 10 } </pre>		

2. Karel **debe levantar los 3 trompos** que se encuentran debajo de él.

iterate(n)			
Consideraciones	Código	Situación de Karel antes de ejecutar la instrucción	Situación de Karel después de ejecutar la instrucción
<ul style="list-style-type: none"> Karel inicia en la coordenada (1, 1). Karel está orientado hacia el Norte. Karel debe terminar (sin moverse) justo donde inició. Karel terminará con 3 trompos en la mochila. 	<p>Código</p> <pre> 1 class program{ 2 program() { 3 iterate(3) { 4 pickbeeper(); 5 } 6 } 7 } 8 } 9 10 </pre>		

3. Karel **debe colocar 4 trompos** debajo de él.

iterate(n)			
Consideraciones	Código	Situación de Karel antes de ejecutar la instrucción	Situación de Karel después de ejecutar la instrucción
<ul style="list-style-type: none"> Karel inicia en la coordenada (1, 1). Karel está orientado hacia el Norte. Karel debe terminar (sin moverse) justo donde inició Karel inicia con 4 trompos en la mochila. 	<p>Código</p> <pre> 1 class program{ 2 program() { 3 iterate(4) { 4 putbeeper(); 5 } 6 } 7 } 8 } 9 10 </pre>		

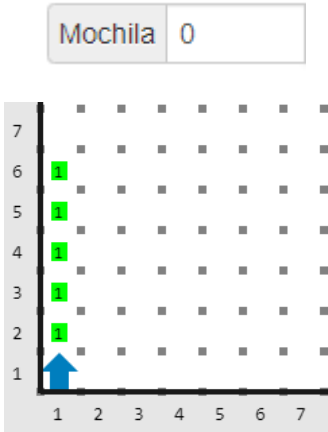
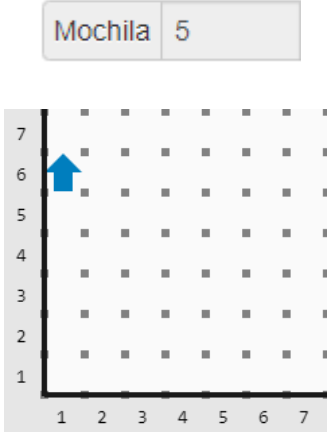
Karel, quien esta orientado hacia el Norte, **debe girarse hasta estar orientado hacia el Este.**

iterate(n)			
Consideraciones	Código	Situación de Karel antes de ejecutar la instrucción	Situación de Karel después de ejecutar la instrucción
<ul style="list-style-type: none"> Karel inicia en la coordenada (1, 1). Karel está orientado hacia el Norte. Karel no debe moverse, solo girar y terminar orientado hacia el Este 	<div>Código</div> <pre> 1 class program{ 2 program() { 3 iterate(3) { 4 turnleft(); 5 } 6 turnoff(); 7 } 8 } 9 10 </pre>		

4. Karel debe **avanzar 5 cuadras hacia el Norte** y en cada cuadra debe ir dejando un **trompo** que tomará de su mochila.

iterate(n)			
Consideraciones	Código	Situación de Karel antes de ejecutar la instrucción	Situación de Karel después de ejecutar la instrucción
<ul style="list-style-type: none"> Karel inicia en la coordenada (1, 1). Karel está orientado hacia el Norte. Karel debe terminar en la coordenada (1, 6) Karel inicia con 5 trompos en su mochila. 	<div>Código</div> <pre> 1 class program{ 2 program() { 3 iterate(5) { 4 putbeeper(); 5 move(); 6 } 7 turnoff(); 8 } 9 } 10 </pre>	<div>Mochila 5</div>	<div>Mochila 0</div>

5. Karel **deberá ir avanzando hacia el norte y recoger los 5 trompos** que se encontrará en la calle por la que va caminando.

iterate(n)			
Consideraciones	Código	Situación de Karel antes de ejecutar la instrucción	Situación de Karel después de ejecutar la instrucción
<ul style="list-style-type: none"> Karel inicia en la coordenada (1, 1). Karel está orientado hacia el Norte. Karel debe terminar en la coordenada (1, 6) Karel deberá tener 5 trompos en su mochila al terminar. 	<p>Código</p> <pre> 1 class program{ 2 program() { 3 iterate(5) { 4 move(); 5 pickbeeper(); 6 } 7 turnoff(); 8 } 9 } 10 </pre>		

Los sensores virtuales de Karel.

Como se sabe Karel es un robot virtual y, por lo tanto, tiene **sensores virtuales**, estos **sensores virtuales** le permiten a Karel saber cosas específicas con respecto a las condiciones o situaciones que se le presentarán en su mundo al estar resolviendo una tarea. Con cada uno de estos sensores, le puedes ordenar a Karel que revise si se cumple o no se cumple una determinada situación o condición, de las cuales te explicare a continuación. Cada pregunta se la hacemos a Karel mediante una función Booleana específica con respecto a cada sensor, la cual es denominada Booleana porque solo puede ser verdadera o falsa.

Ahora, te explicaré cada uno de los sensores y como le son de gran ayuda a Karel.

- Sensores para los trompos y la mochila de Karel.**

Estas 4 condiciones ayudarán a que Karel pueda saber si está sobre un trompo o si tiene algún trompo en su mochila:

Condición	Traducción
nextToABeeper()	Estoy sobre un trompo
notNextToABeeper()	No estoy sobre un trompo
anyBeepersInBeeperBag()	Hay algún trompo en la mochila
notAnyBeepersInBeeperBag()	No hay trompos en la mochila

A continuación, te mostraré algunas situaciones y las respuestas a las condiciones de los sensores para los trompos y la mochila de Karel.

En el siguiente ejemplo, **Karel se encuentra sobre 3 trompos y sin trompos en su mochila.**



Sensores para los trompos y la mochila de Karel		
Condición	Traducción	Respuesta
nextToABeeper()	Estoy sobre un trompo	Verdadero
notNextToABeeper()	No estoy sobre un trompo	Falso
anyBeepersInBeeperBag()	Hay algún trompo en la mochila	Falso

notAnyBeepersInBeepersBag()	No hay trompos en la mochila	Verdadero
------------------------------	------------------------------	-----------

En este segundo ejemplo que te muestro a continuación, **no hay un solo trompo debajo de Karel, pero si lleva 10 trompos en su mochila.**



Sensores para los trompos y la mochila de Karel		
Condición	Traducción	Respuesta
nextToABeeper()	Estoy sobre un trompo	Falso
notNextToABeeper()	No estoy sobre un trompo	Verdadero
anyBeepersInBeeperBag()	Hay algún trompo en la mochila	Verdadero
notAnyBeepersInBeepersBag()	No hay trompos en la mochila	Falso

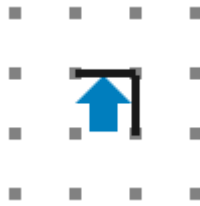
- **Sensores para detectar bardas o paredes a media cuadra de donde esta Karel.**

Para **saber si Karel tiene, a media cuadra, alguna pared frente a él o a sus lados**, se pueden utilizar 6 condiciones o funciones booleanas:

Condición	Traducción
frontIsClear()	Frente despejado
frontIsBlocked()	Frente bloqueado
leftIsClear()	Izquierda despejado
leftIsBlocked()	Izquierda bloqueado
rightIsClear()	Derecha despejado
rightIsBlocked()	Derecha bloqueado

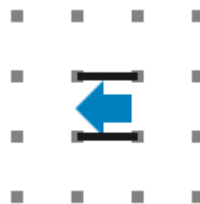
Bien, ahora que conoces las condiciones que Karel puede evaluar con respecto a bardas que esten a media cuadra de donde se encuentre Karel, te mostraré algunas situaciones dentro del mundo de Karel y las respuestas a las condiciones de los sensores con respecto a las paredes que esten a media cuadra de donde se encuentre Karel.

En el siguiente ejemplo, como puedes observar, **Karel se encuentra orientado hacia el Norte, a media cuadra, frente a él hay una pared y a su derecha hay otra.**



Sensores para las paredes dentro del mundo de Karel		
Condición	Traducción	Respuesta
frontIsClear()	Frente despejado	Falso
frontIsBlocked()	Frente bloqueado	Verdadero
leftIsClear()	Izquierda despejado	Verdadero
leftIsBlocked()	Izquierda bloqueado	Falso
rightIsClear()	Derecha despejado	Falso
rightIsBlocked()	Derecha bloqueado	Verdadero

En la siguiente imagen o situación, **Karel se encuentra orientado hacia el Oeste, a la derecha y a la izquierda de él, hay paredes bloqueando el camino.**



Sensores para las paredes dentro del mundo de Karel		
Condición	Traducción	Respuesta
frontIsClear()	Frente despejado	Verdadero
frontIsBlocked()	Frente bloqueado	Falso
leftIsClear()	Izquierda despejado	Falso
leftIsBlocked()	Izquierda bloqueado	Verdadero
rightIsClear()	Derecha despejado	Falso
rightIsBlocked()	Derecha bloqueado	Verdadero

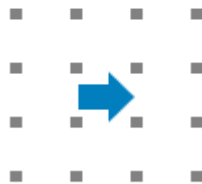
- **Sensores para la orientación o dirección de Karel.**

Estas 8 condiciones le **ayudan a Karel a detectar hacia donde esta viendo**, con respecto a su dirección u orientación, esto lo hace **en base a su brújula virtual**.

Condición	Traducción
facingNorth()	Estoy orientado o viendo hacia el norte
facingSouth()	Estoy orientado o viendo hacia el sur
facingEast()	Estoy orientado o viendo hacia el este
facingWest()	Estoy orientado o viendo hacia el oeste
notFacingNoth()	No estoy orientado o viendo hacia el norte
notFacingSouth()	No estoy orientado o viendo hacia el sur
notFacingEast()	No estoy orientado o viendo hacia el este
notFacingWest()	No estoy orientado o viendo hacia el oeste

Una vez más, para que quede más claro, te mostraré algunas situaciones dentro del mundo de Karel y las respuestas a las condiciones de los sensores con respecto a la orientación o dirección de Karel.

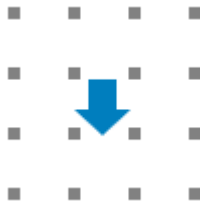
En la siguiente imagen, **Karel está orientado hacia el este**.



Sensores para la orientación de Karel		
Condición	Traducción	Respuesta
facingNorth()	Estoy orientado o viendo hacia el norte	Falso
facingSouth()	Estoy orientado o viendo hacia el sur	Falso
facingEast()	Estoy orientado o viendo hacia el este	Verdadero
facingWest()	Estoy orientado o viendo hacia el oeste	Falso
notFacingNorth()	No estoy orientado o viendo hacia el norte	Verdadero
notFacingSouth()	No estoy orientado o viendo hacia el sur	Verdadero
notFacingEast()	No estoy orientado o viendo hacia el este	Falso
notFacingWest()	No estoy orientado o viendo hacia el oeste	Verdadero

MANUAL DEL USUARIO PLATAFORMA.KARELOGIC.NET

En la siguiente imagen o situación, **Karel** **esta orientado hacia el sur**.



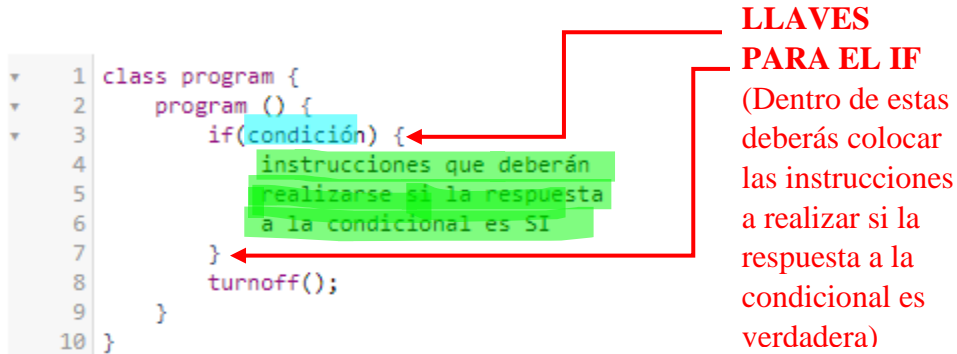
Sensores para la orientación de Karel		
Condición	Traducción	Respuesta
facingNorth()	Estoy orientado o viendo hacia el norte	Falso
facingSouth()	Estoy orientado o viendo hacia el sur	Verdadero
facingEast()	Estoy orientado o viendo hacia el este	Falso
facingWest()	Estoy orientado o viendo hacia el oeste	Falso
notFacingNorth()	No estoy orientado o viendo hacia el norte	Verdadero
notFacingSouth()	No estoy orientado o viendo hacia el sur	Falso
notFacingEast()	No estoy orientado o viendo hacia el este	Verdadero
notFacingWest()	No estoy orientado o viendo hacia el oeste	Verdadero

Estatuto de control condicional simple y compuesto (“if” y “if-else”)

Un **estatuto de control condicional** o **instrucción condicional** es una **instrucción** que permite **cuestionar la información para tomar una decisión**, normalmente consiste en **decidir si un grupo de instrucciones se hace o no**, como su nombre lo dice, la decisión se toma conforme a la respuesta a la condición: **SI** o **NO**. **Se tiene que cumplir la condición para continuar y llevar a cabo ciertas instrucciones.**

➤ **Estatuto de control condicional simple “if”:**

Los sensores necesitan ser utilizados de manera correcta y con los comandos adecuados, uno de ellos es la **instrucción “if”**. Esta instrucción **permite condicionar la ejecución de un grupo de instrucciones**, pues **las instrucciones dentro de esta se realizarán si la respuesta a su condición es igual a SI**, por otro lado, **si la respuesta a su condición es NO**, las **instrucciones de su bloque no se realizarán**. En seguida te muestro su escritura básica:



```

1 class program {
2     program () {
3         if(condición) {
4             instrucciones que deberán
5             realizarse si la respuesta
6             a la condicional es SI
7         }
8         turnoff();
9     }
10 }
    
```

LLAVES PARA EL IF
(Dentro de estas deberás colocar las instrucciones a realizar si la respuesta a la condicional es verdadera)

Ahora que conoces su escritura básica, te mostraré un ejemplo en la que se utilizará la instrucción **“if”**:

En este ejemplo le indicaremos a Karel avance sobre una calle y si, mientras va avanzando, se encuentra un trompo, lo levante y siga su camino.

Instrucción “if”				
	Observaciones	Codigo	Mundo inicial	Mundo final
Condicional = SI	En el codigo le pedimos a Karel que avance 3 veces y que SI en algún momento se encuentra un trompo debajo de él, lo tome.	<pre> 1 class program { 2 program () { 3 iterate(3) { 4 move(); 5 if(nextToABeeper()) { 6 pickbeeper(); 7 } 8 } 9 turnoff(); 10 } 11 }</pre>		
	Aquí se pide exactamente lo mismo, pero en este caso NO hay trompos en el camino de Karel, por lo tanto, Karel solo avanza las tres cuadras y se apaga.			

➤ Estatuto de control condicional compuesto “if-else”:

La instrucción **if** tiene una extensión que nos permite saber cuando la condición es falsa y con esa información tomar una segunda decisión en caso de que termine siendo así.

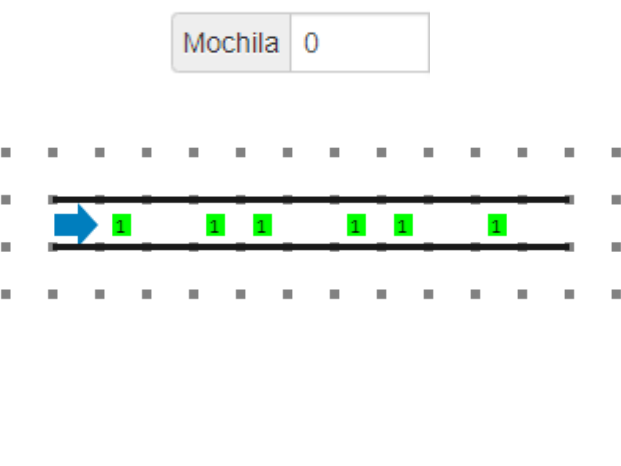

El comando **else**, le permite a la instrucción **if** ejecutar un grupo de instrucciones si la respuesta a la condición es NO o la condición es falsa. Este comando (**else**) solo se puede utilizar para complementar la instrucción if, **no puede utilizarse de manera independiente**.

A esta extensión del **if**, le llamaremos **if-else**. Te mostraré la manera correcta de



Ahora que hemos visto la manera correcta de escribir el if-else, te mostraré un ejemplo utilizando el **if-else**:

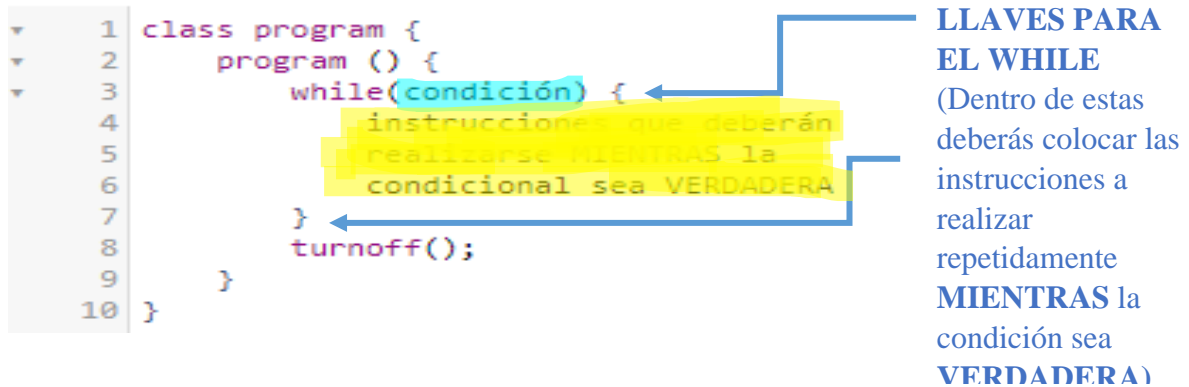
En este ejemplo Karel el cartero y deberá recoger las cartas en los buzones de una calle de 10 casas, en caso de que sus buzones esten vacíos deberá continuar caminando

Instrucción “if-else”		
	Observaciones	Codigo
Condional = SI	En el codigo le pedimos a Karel que avance 10 veces, SI se encuentra con un trompo (carta) deberá recogerlo y guardarlo en su mochila	<pre> 1 class program { 2 program () { 3 iterate(10) { 4 if(nextToABeeper()) { 5 pickbeeper(); 6 move(); 7 } 8 else { 9 move(); 10 } 11 } 12 turnoff(); 13 } 14 }</pre>
		<div>Mundo inicial</div>  <div>Mundo final</div> 
Condional = NO	Aquí se pide exactamente lo mismo, pero en caso de que NO haya trompos, solamente deberá avanzar a la siguiente casa.	

Como puedes observar se llevaron acabo tanto el grupo de instrucciones de if y el grupo de instrucciones de else, eso fue porque en el camino Karel si se encontró con algunos trompos (y siguió las instrucciones dentro de las llaves de if, ya que la condicional era VERDADERA), pero también había celdas sin trompos (y siguió las instrucciones dentro de las llaves de else, ya que la condicional era FALSA y no estaba sobre ningún trompo).

Estatuto de control de repetición condicional “while”

Este comando funciona de manera similar al estatuto de control de repetición “iterate(n), pero el “while” hace que un grupo de instrucciones se ejecute una y otra vez **MIENTRAS** la condición sea **VERDADERA**. La manera de escribir esta instrucción es la siguiente:



Ahora que conocemos su forma básica de escritura, te explicaré como funciona a más detalle con ayuda de un ejemplo:

Le pediremos a Karel que camine dejando un camino de trompos hasta que encuentre una pared frente a él.

Estatuto de control de repetición condicional “while”	
Código	<pre> 1 class program { 2 program () { 3 while(frontIsClear()) { 4 move(); 5 putbeeper(); 6 } 7 turnoff(); 8 } 9 }</pre>
Mundo inicial	
Mundo final	

MANUAL DEL USUARIO PLATAFORMA.KARELOGIC.NET

Como podrás observar la mochila de Karel tiene trompos infinitos para que pueda seguir colocándolos por su camino hasta que choque con alguna pared, cuando chocó con la pared inmediatamente se apagó porque no le dimos más instrucciones para realizar después de haber llegado a la pared.

Enlace del video tutorial de como navegar en plataforma.karelogic.net

https://www.youtube.com/watch?v=HC6PviE17KE&ab_channel=GilbertoReyes

Curso básico de Lenguaje C/C++; Prof. Gilberto Reyes B.

Para más información o dudas, comunicarse por whatsapp

con el Prof. Gilberto Reyes Barrera al

teléfono 8126799244 mayo 2023