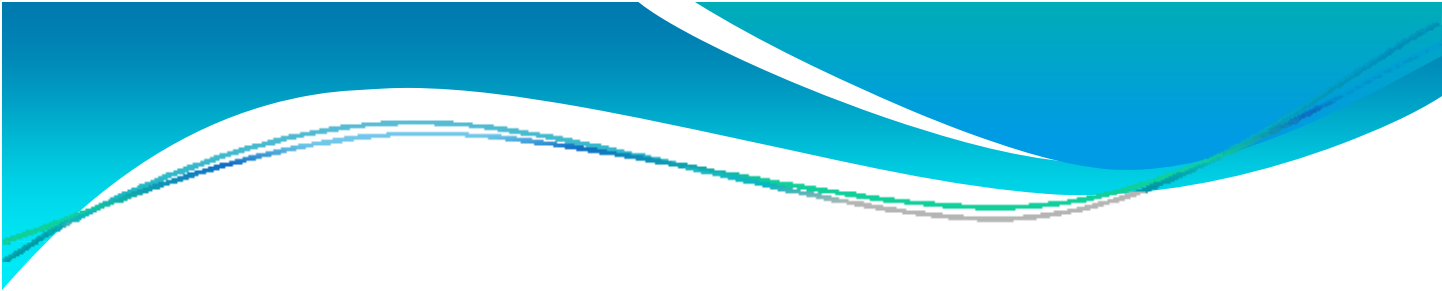




# Gira-Izquierda

Karel para todos!





¿Estás interesado en aprender a programar en lenguaje de Karel? ¿Te interesa la Olimpiada Mexicana de Informática? Esta es tu oportunidad de unirte al equipo OMI y ser parte de la mejor competencia nacional de programación.

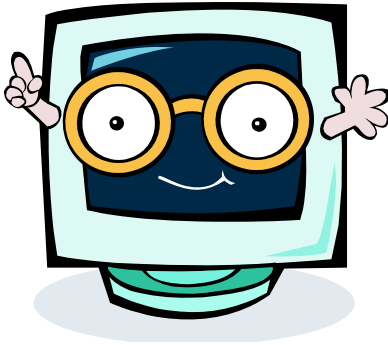
Este pequeño libro te enseña paso por paso como usar el programa Karel. El libro contiene todo lo que necesitas saber para lograr problemas de dificultad básica y media. Practicando en línea en la página de OMIJAL, [www.omijal.org.mx](http://www.omijal.org.mx), podrás convertirte en un experto en Karel. Recuerda, la práctica hace al maestro.

**Agradecimiento:**

A Javier Lomelín Urrea por su dedicación y esmero en este proyecto.

**Manuel Fernando Guzmán Muñoz**  
**Presidente OMIJal**





# Gira-Izquierda



Olimpiada de Informática del Estado de Jalisco  
OMIJal  
Derechos Reservados 2008-2010

# Índice

## Capítulo 1: Introducción a Karel 4

- 1.1: El Programa de Karel 5
- 1.2: El Mundo de Karel 8
- 1.3: Empezando a Programar 10
- 1.4: Instrucciones Básicas 11

## Capítulo 2: Toma de Decisiones 12

- 2.1: Decisiones Simples 13
- 2.2: Decisiones con otro Caso 15
- 2.3: Decisiones Anidadas 16

## Capítulo 3: Repeticiones 17

- 3.1: Repeticiones Fijas 18
- 3.2: Repeticiones con Condición 19

## Capítulo 4: Instrucciones Nuevas 21

# I. Introducción a Karel

Karel es un lenguaje sencillo y divertido que ayuda al aprendizaje de algoritmos y al desarrollo de la lógica. Karel es un robot virtual que aparece como una pequeña flecha azul que viaja a través de su mundo. El mundo de Karel es un cuadrado lleno de calles y avenidas que él puede recorrer, a menos que ésta esté bloqueada.

Karel tiene una mochila donde guarda zumbadores. Los zumbadores son unos objetos que pueden ser simulados como canicas. Éstos le ayudan a Karel a realizar sus tareas ya que pueden ser utilizados para muchas cosas como contar, marcar lugares especiales o caminos importantes, etc.

Para que Karel pueda hacer su trabajo, tú como programador tienes que escribir un código con órdenes o comandos que Karel obedece. El código tiene que estar correcto, ya que, lamentablemente, Karel no es un robot listo y necesita toda tu ayuda para poder funcionar bien.

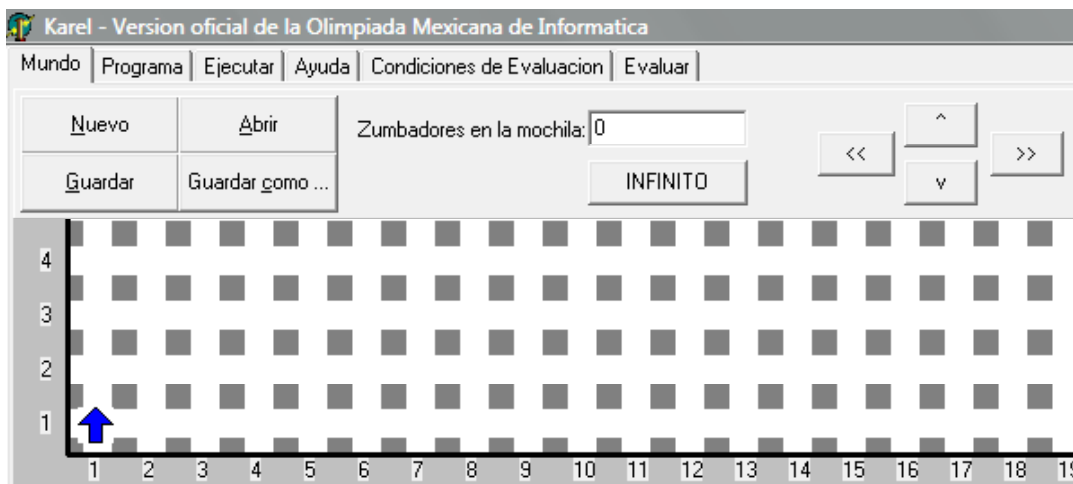


# 1.1: El Programa de Karel

El programa de Karel es una aplicación muy sencilla que sólo tiene cuatro secciones o ventanillas.

## Mundo

Es el lugar en el cual Karel realiza sus tareas y puede ser diseñado como tú quieras mientras la tarea de Karel pueda ser realizada ahí.



## Ayuda

Este botón contiene un pequeño tutorial acerca del uso de Karel y viene la sintaxis correcta de las instrucciones.

Las dos ventanillas restantes son parte de un evaluador que viene con Karel pero no son necesarias para el uso del programa.

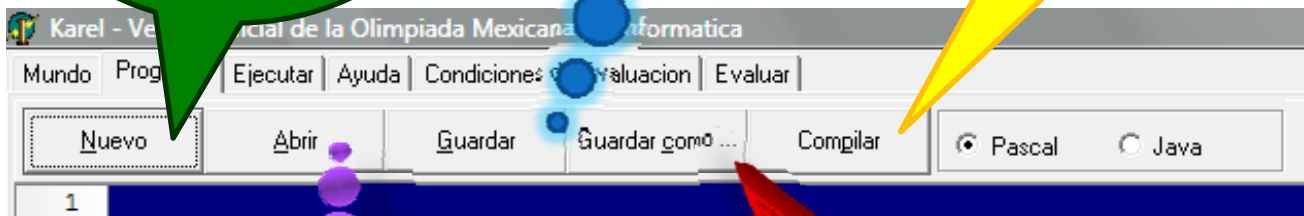
## Programa

Aquí es donde tú escribes tu código de instrucciones que Karel va a realizar. Es la parte más importante del programa porque es lo que hace que Karel funcione. Se pueden elegir dos tipos de código: Pascal y Java. Nosotros sólo vamos a usar Pascal.

**Nuevo** crea un archivo de texto donde puedes empezar a escribir un programa desde el inicio.

**Guardar** te permite guardar el texto que está actualmente abierto.

**Compilar** revisa tu código para que no haya errores de ortografía y Karel lo pueda utilizar.



**Abrir** es un botón que te permite abrir un texto que ya habías escrito y guardado antes.

**Guardar Como** guarda el programa que está abierto pero tú le puedes poner la ubicación donde éste quedará guardado.



## Ejecutar

En esta sección puedes ver a Karel realizar las instrucciones que habías escrito. Ojo, tu código tiene que estar compilado para que funcione. Esta sección hace que veas gráficamente el resultado de tu código.

The screenshot shows the Karel IDE interface with several callouts:

- Adelante** ejecuta la siguiente línea que hay en tu código. (Callout pointing to the 'Adelante' button)
- Detener** pausa el programa. (Callout pointing to the 'Detener' button)
- Correr** empieza a ejecutar el programa desde la línea en la que se encuentra hasta que termina el programa o lo detienes. (Callout pointing to the 'Correr' button)
- Inicializar** regresa a Karel a su posición inicial y sitúa al inicio del programa el renglón de ejecución. (Callout pointing to the 'Inicializar' button)

The interface includes a menu bar (Mundo, Programa, Ejecutar, Ayuda, Condiciones de E), a toolbar with buttons for 'Adelante', 'Detener', 'Correr', and 'Inicializar', and input fields for 'Zumbadores en la mochila' (set to 1) and 'Retardo de ejecución (ms)' (set to 500). A grid on the right shows Karel's position at (1,1).

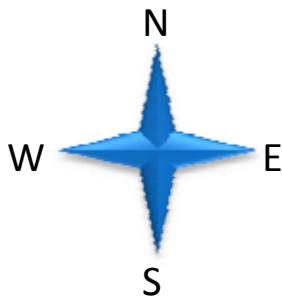
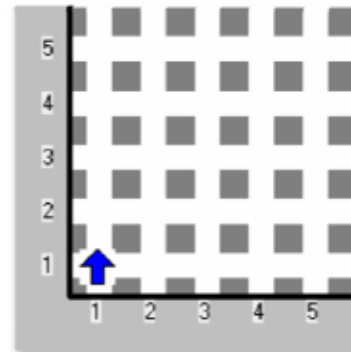
*Zumbadores en la mochila* indica cuantos zumbadores está cargando Karel en ese momento.

*Retardo de ejecución* que indica que tan rápido avanza Karel. Entre menor sea el número de retardo, es mayor la velocidad de Karel.



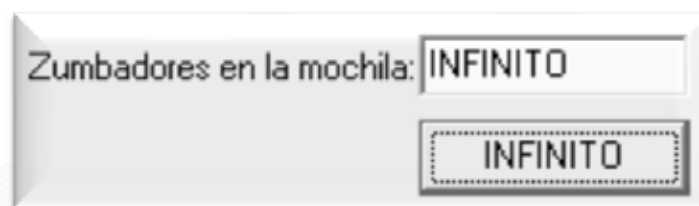
## 1.2: El Mundo de Karel

El mundo de Karel está formado por cien avenidas y cien calles. Las avenidas son verticales y las calles son horizontales. Se pueden agregar y quitar paredes en cualquier parte del mundo, así como zumbadores, excepto en los límites de éste.

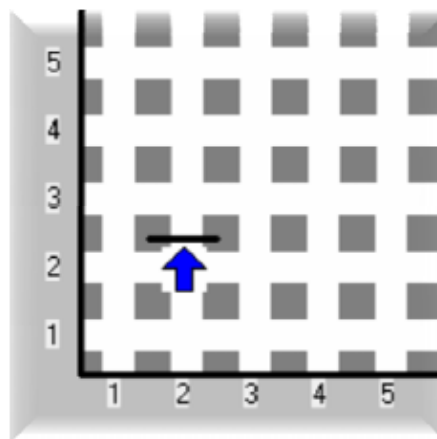


Karel solamente tiene la habilidad de girar hacia su izquierda, y siempre a 90°. Por lo tanto, Karel siempre va a estar orientado hacia alguno de los puntos cardinales: norte, sur, este y oeste.

En el mundo también se puede definir el número de zumbadores que carga Karel en su mochila. Existe la opción de que Karel tenga zumbadores ilimitados y nunca se agoten. Esta opción se activa oprimiendo el botón INFINITO en el cuadro de texto del mundo.



Las paredes son obstáculos que se ponen entre las calles que Karel no puede saltar, bloqueando su paso por completo. Las paredes se ponen simplemente haciendo un clic con el ratón entre dos calles o dos avenidas. No existen paredes que vallan de una calle a una avenida, es decir, no hay paredes en diagonal. Para quitarla simplemente se hace un clic en la pared. Los límites del mundo son considerados como pared.



Se puede establecer con que ubicación inicia Karel el programa y hacia donde está orientado. Así como agregar y quitar zumbadores en cualquier parte del mundo.



## 1.3: Empezando a Programar

Los programas de Karel cuentan con dos simples secciones:

**iniciar-programa... finalizar-programa:** Indica donde irá todo el código fuente del programa; donde inicia y donde termina.

**inicia-ejecucion... finaliza-ejecucion:** Esto indica cual es el área del programa que se va a ejecutar.

Además de estas dos secciones, el código debe contar con la línea *apagate* para finalizar la ejecución por completo.

```
1  iniciar-programa
2      inicia-ejecucion
3          apagate ;
4      termina-ejecucion
5  finalizar-programa
```

El programa anterior sólo tiene una instrucción para Karel: *apagate*. Es importante saber que después de cada instrucción se debe poner un punto y coma (;). Las líneas de *iniciar-programa*, *inicia-ejecucion*, *termina-ejecucion* y *finalizar-programa*, no llevan punto y coma, ya que no son instrucciones. Estas no le ordenan a Karel a hacer algo, simplemente marcan el inicio y el fin de una sección.

## 1.4: Instrucciones Básicas

Karel cuenta con cinco instrucciones básicas para hacer todas sus tareas. Las instrucciones son las siguientes:

**avanza:** Karel avanza una cuadra hacia donde esté orientado. Si hay una pared enfrente, el programa marcará un error y dejará de ejecutarse.

**gira-izquierda:** Karel gira hacia la izquierda 90°, cambiando su orientación.

**coge-zumbador:** Karel recoge un zumbador en el lugar donde está parado. Si no hay zumbador en ese lugar, el programa marcará error y termina la ejecución.

**deja-zumbador:** Karel deja un zumbador en el lugar que está parado. Si Karel no tiene zumbadores en la mochila, entonces no podrá dejar el zumbador y el programa marcará error.

**apagate:** Finaliza la ejecución del programa. Karel ya no podrá hacer más cosas porque ya está apagado.



## II. Toma de Decisiones

Muchas veces, Karel necesita tomar una decisión cuando está en una situación apretada como cuando está enfrente de una pared y necesita saber si avanza o si no. Las condiciones que Karel puede detectar para poder hacer su decisión se listan a continuación:

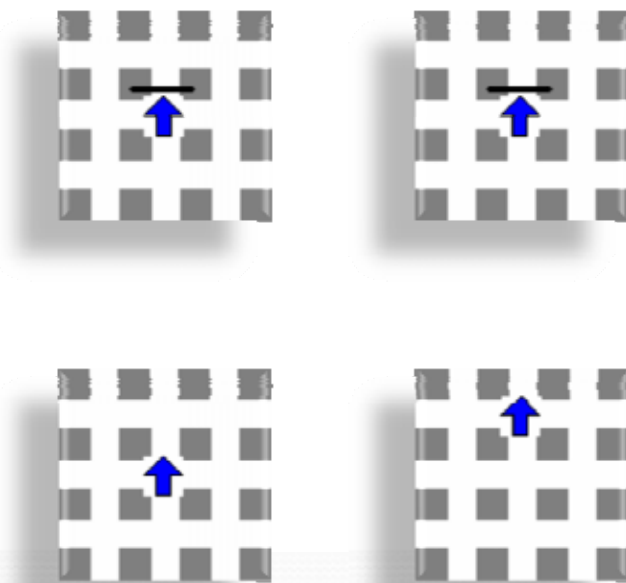
<i>frente-libre</i>	<i>junto-a-zumbador</i>	<i>orientado-al-este</i>
<i>frente-bloqueado</i>	<i>no-junto-a-zumbador</i>	<i>orientado-al-oeste</i>
<i>izquierda-libre</i>	<i>algun-zumbador-en-la-mochila</i>	<i>no-orientado-al-norte</i>
<i>izquierda-bloqueada</i>	<i>ningun-zumbador-en-la-mochila</i>	<i>no-orientado-al-sur</i>
<i>derecha-libre</i>	<i>orientado-al-norte</i>	<i>no-orientado-al-este</i>
<i>derecha-bloqueada</i>	<i>orientado-al-sur</i>	<i>no-orientado-al-oeste</i>

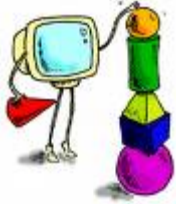
## 2.1: Decisiones Simples

Hay veces que Karel puede hacer una serie de instrucciones si se cumple una condición previamente establecida. La condición puede ser cualquiera de las mencionadas anteriormente. Por ejemplo:

```
si frente-libre entonces inicio  
    avanza;  
fin;
```

En las líneas anteriores se muestra una pequeña decisión que va a hacer Karel. Si en frente de él no hay pared, entonces avanza; si hay pared entonces no hace nada.

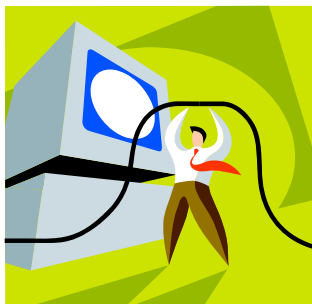




Es importante saber cómo escribir el código de las condiciones. Lo que está escrito en negritas anteriormente es la condición que tú le das a Karel y puede ser cualquiera de las de la lista. Entre *inicio* y *fin* pueden existir las líneas de código que sean necesarias.

También pueden existir múltiples condiciones en la misma línea de código como: *si **frente-libre y orientado-al-norte** entonces inicio...* Si cualquiera de las dos condiciones no se cumple, Karel se salta esa instrucción.

La opción 'o' está también disponible: *si **frente-libre o junto-a-zumbador** entonces inicio...* Aquí Karel tomaría una decisión si se cumple cualquiera de las dos condiciones.



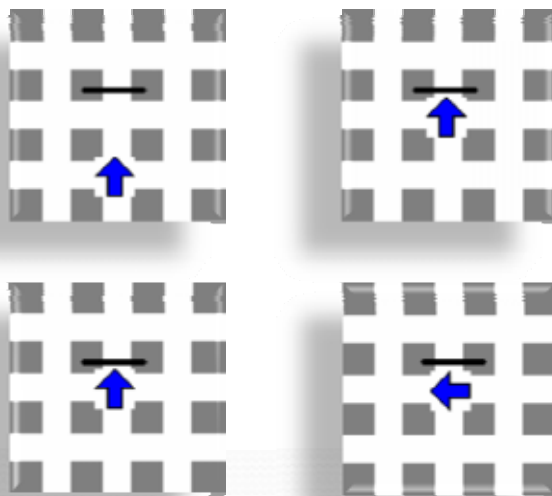


## 2.2: Decisiones con otro Caso

En este tipo de decisiones, Karel puede hacer una serie de instrucciones si se cumple la condición y si no se cumple Karel puede hacer otra serie de instrucciones. Ejemplo:

```
si frente-libre entonces inicio
    avanza;
fin
sino inicio
    gira-izquierda;
fin;
```

En este caso, si el frente de Karel está libre entonces avanza. Si está bloqueado entonces gira hacia la izquierda. Ojo, únicamente el *fin* de la última línea lleva punto y coma.

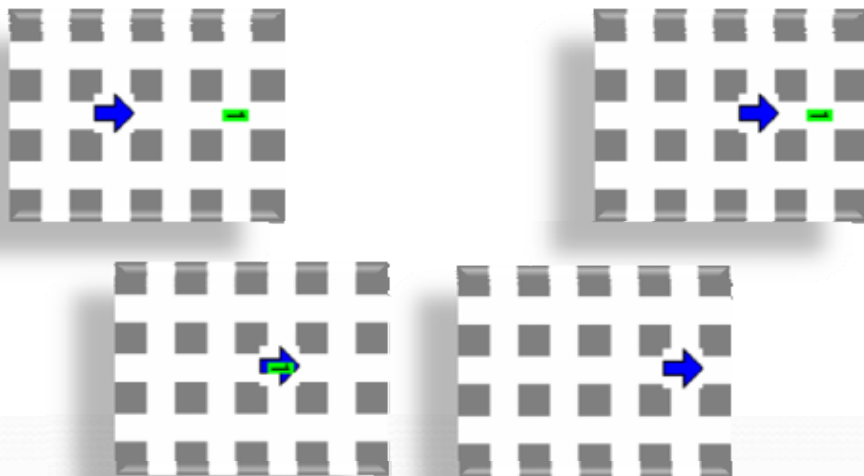


## 2.3: Decisiones Anidadas

Después de haber tomado ya una decisión, dentro de la misma, Karel puede tomar múltiples decisiones. A esto se le conoce como decisiones anidadas. Las decisiones anidadas tienen las mismas características que las decisiones anteriores y tienen estructuras similares a la siguiente:

```
si frente-libre entonces inicio  
    avanza;  
    si junto-a-zumbador entonces inicio  
        coge-zumbador;  
    fin;  
fin;
```

Dentro de cada decisión pueden existir las instrucciones que sean necesarias, así como las decisiones con otro caso. Si es necesario, muchas decisiones pueden existir dentro de otras.



## III. Repeticiones

Hay veces que Karel necesita repetir una serie de instrucciones y decisiones muchas veces. Digamos que Karel tiene que avanzar cincuenta veces para llegar a un lugar importante. Lo que tendríamos que hacer sería escribir la palabra *avanza* cincuenta veces, y es un trabajo pesado. Para ahorrar trabajo, espacio y tiempo, podemos hacer repeticiones, o “ciclos”, como se les llama comúnmente.

Las repeticiones que puede hacer Karel son las repeticiones fijas y las repeticiones con condición. En este capítulo aprenderás a hacer los dos tipos de repeticiones y así verás como todo se hace más fácil.

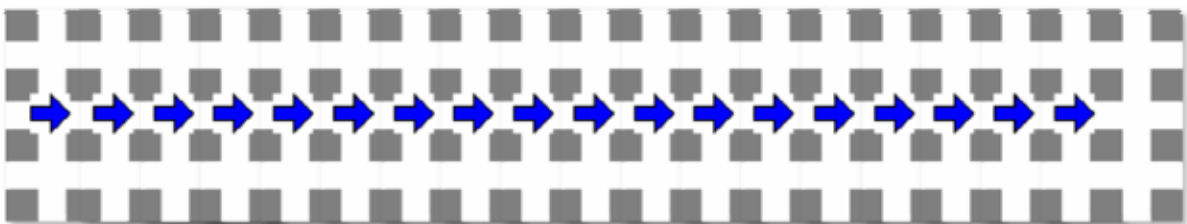


## 3.1: Repeticiones Fijas

En el programa de Karel, tú le puedes dar un número exacto de veces que quieres repetir una instrucción o serie de instrucciones. Esto se usa con la instrucción de *repetir*. Ejemplo:

```
repetir 50 veces inicio  
  
    avanza;  
  
fin;
```

Así te ahorras muchísimo trabajo y esfuerzo. Tú le puedes poner el número de veces que quieres repetir la instrucción. El número de veces que Karel va a hacer la serie de instrucciones que tú le das tiene que ser un número entero, no puede haber fracciones ni decimales.



## 3.2: Repeticiones con Condición

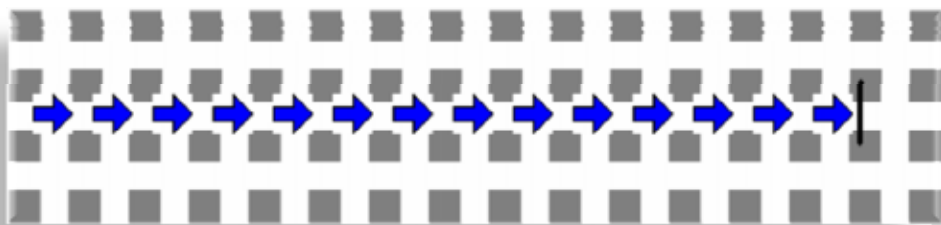
En ciertas ocasiones, Karel tiene que hacer una instrucción pero tú no sabes cuándo hacer que se pare. Por ejemplo: Karel tiene que avanzar hasta que encuentra una pared. Tú sabes que Karel tiene que avanzar, pero no sabes que tan lejos. Esto se logra de la siguiente manera.

*mientras frente-libre hacer inicio*

*avanza;*

*fin;*

En este caso, mientras Karel no tenga una pared en frente, va a avanzar. Cuando encuentre la pared, dejará de hacer la repetición. Las condiciones que llevan las repeticiones son las mismas que en el capítulo anterior.



También existen las repeticiones anidadas como en el capítulo anterior, y se pueden combinar las decisiones con las repeticiones. Ejemplo:

```
si junto-a-zumbador entonces inicio  
    mientras junto-a-zumbador hacer inicio  
        coge-zumbador;  
    fin;  
fin;
```

Aquí, Karel toma una decisión y luego hace una repetición. Este código hace que Karel encuentre un montón de zumbadores y los recoja todos. Si no está junto a un zumbador, Karel no recoge nada.



## IV. Instrucciones Nuevas

Para hacer las cosas más fáciles y ahorrar trabajo, nosotros podemos crear nuestras propias instrucciones usando instrucciones existentes, u otras que ya habíamos creado.

Como ya lo saben, Karel solamente puede girar hacia la izquierda. Y hay veces que es muy tardado escribir tres veces *gira-izquierda* para que Karel gire hacia la derecha. Entonces, nosotros podemos crear una instrucción que se llame *gira-derecha* de la siguiente manera:

```
define-nueva-instrucción gira-derecha como inicio  
gira-izquierda;  
gira-izquierda;  
gira-izquierda;  
fin;
```

Esto tiene que ser escrito entre las líneas *iniciar-programa* e *inicia-ejecucion al inicio de tu código*. Entonces dentro del código tú escribes la instrucción *gira-derecha* y Karel gira tres veces hacia la izquierda automáticamente. Las instrucciones nuevas pueden llamarse como tú quieras.







Si te interesa practicar para convertirte en un experto, ingresa a [www.omijal.org.mx](http://www.omijal.org.mx) y se parte de la leyenda.