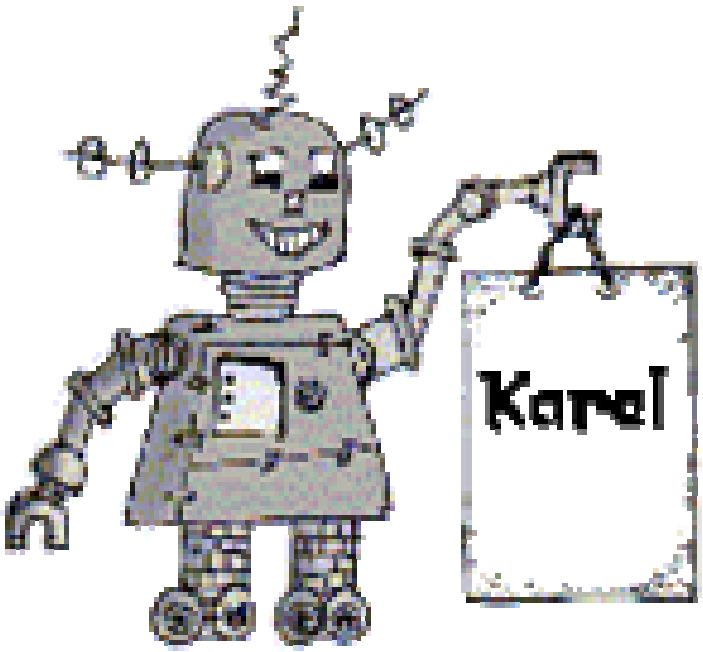
Manuel Antonio Ortiz Montalvo

MANUAL

KAREL EL ROBOT



H. Veracruz, Ver. a Julio del 2006

INDICE

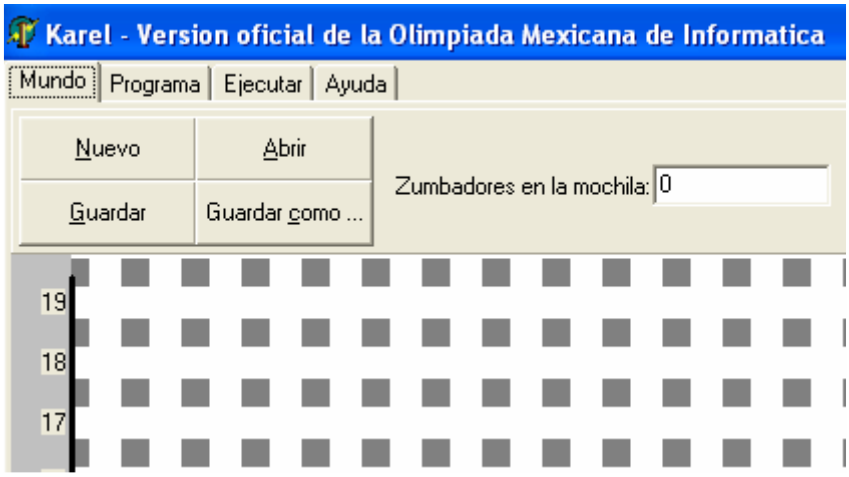
I. INTRODUCCIÓN A KAREL

Karel es un lenguaje orientado al aprendizaje de algoritmos. Karel simula un robot (aunque realmente es una flechita azul), que viaja en torno a su mundo, como si fuera una ciudad, donde hay calles y avenidas, pero estas pueden estar cerradas (o sea, que puede no haber paso).

Karel lleva consigo una mochila, donde guarda unos zumbadores, (mas entendible, unos trompos), y con ellos puede hacer muchas cosas, entre ellas, contar. Puede dejar los zumbadores en una esquina, (formada por el cruce de una calle y una avenida), o puede recogerlos y guardarlos en su mochila.

1.1 Ambiente de Karel

El Ambiente está conformado por 4 secciones:



a) Mundo, que es donde se editará el lugar donde Karel actuará de acuerdo a las condiciones del programa codificado.

b) Programa, sección donde se codifica el programa, y se puede elegir entre 2 lenguajes diferentes: Pascal y java.

Cuenta con 5 botones:

- Nuevo. Crea un nuevo archivo de texto para guardar el código fuente de un programa.

- Abrir. Abre un archivo de texto con un programa previamente codificado y guardado.

- Guardar. Guarda el programa que esté actualmente abierto.

- Guardar como. Guarda el programa que está actualmente abierto, y se puede elegir la ubicación donde se guardará y el nombre del archivo.

- Compilar. Compila el archivo abierto, para poder ejecutarse.

c) Ejecutar, en ésta sección se corre el programa (lo empieza a ejecutar), pero tiene que estar previamente codificado y compilado. Aquí es donde vemos gráficamente el resultado del programa que se elaboró.

Cuenta con 4 botones y dos cuadros de texto:

- Adelante. Ejecuta la siguiente línea de código.

- Correr. Empieza a ejecutar el programa desde la última línea en la que se quedó.

- Detener. Pausa la ejecución del programa

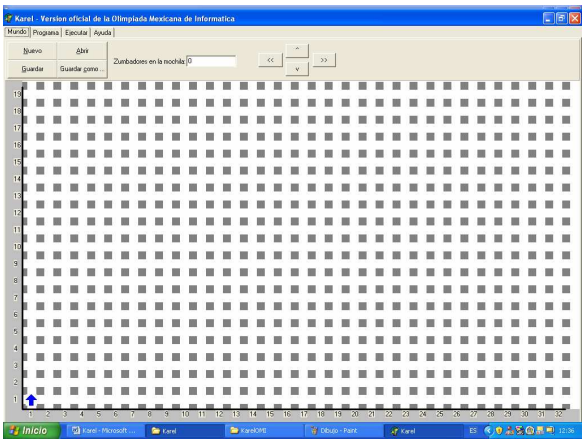
- Inicializar. Regresa a Karel a su posición y orientación inicial, y sitúa al comienzo del programa, el renglón de ejecución.

Zumbadores en la mochila, es la cantidad que inicialmente Karel carga en la mochila.

Retardo ejecución. Tiempo en milisegundos que tardará el programa, en ejecutar una línea de código, desde la ejecución de la última línea.

d) Ayuda, contiene un sencillo tutor acerca del uso de Karel. También viene la sintaxis de las instrucciones.

1.2 El mundo de Karel



El mundo está formado por 100 calles y 100 avenidas, y puede haber paredes, que obstruyan el paso; las calles son horizontales, y las avenidas verticales.

El puede girar en su mundo solamente hacia la izquierda, y siempre a 90º, entonces solo puede estar orientado hacia 4 puntos, que son Norte, Sur, Este y Oeste.

Para poner una pared que impida el paso de Karel por una calle, solamente se tiene que dar un click entre 2 manzanas (cuadritos grises), las cuales deben estar contiguas (juntas) verticalmente u horizontalmente. Y para quitarla, simplemente se la da click a la pared que se desee eliminar. Antes de la calle 1 y de la avenida 1, se considera pared, al igual que después de la calle 102 y de la avenida 101.

En el mundo también se puede definir el número inicial de zumbadores que Karel traerá en su mochila. Para poner que no se le agoten los zumbadores se usa la palabra INFINITO.

Se puede establecer la orientación de Karel antes de que el programa empiece a ejecutarse, de la siguiente manera: se da click derecho sobre cualquier esquina, y se abrirá un menú emergente, elegiremos “Situar a Karel”, y posteriormente la orientación que se desea.

Para poner una cantidad de zumbadores iniciales en cualquier esquina, se hace abriendo el menú emergente en cualquier esquina, (con click derecho) y seleccionando la cantidad deseada, que puede ser de 0 a 9, una cantidad definida o Infinito zumbadores.

Los puntos cardinales se indican como muestra la figura siguiente:

EJERCICIO 1.

Objetivo. Hacer un mundo propio para practicar el uso de esta sección.

1) Dar un click en la avenida 1, entre las calles 5 y 6, como se muestra en la figura 1, y deberá aparecer una pared como se ve en la figura 2.

2) Hacer lo mismo pero ahora en la avenida 2

3) Seguir sucesivamente hasta llegar a la avenida numero 5. El mundo deberá quedar como se ve en la figura 3

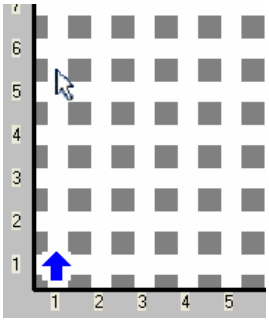


Figura 1

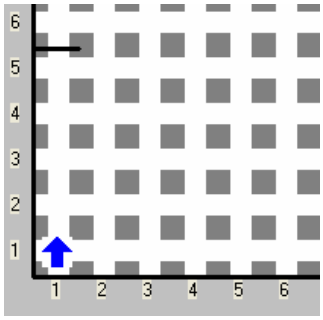


Figura 2

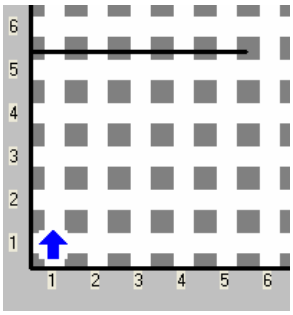


Figura 3

4) Dar click en la calle 5, entre las avenidas 5 y 6 (como se muestra en la figura 4), y deberá aparecer una pared verticalmente, (como en la figura 5).

5) Seguir poniendo paredes verticales hasta llegar a la calle 1, para que queden rodeadas las 5 primeras calles con las 5 primeras avenidas. En la figura 6 se muestra como debe quedar el mundo con las paredes.

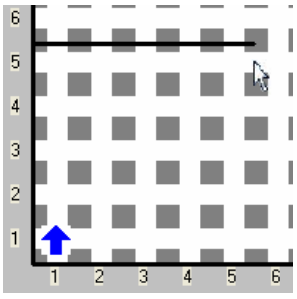


Figura 4

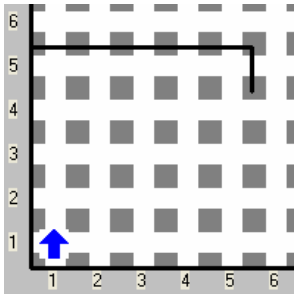


Figura 5

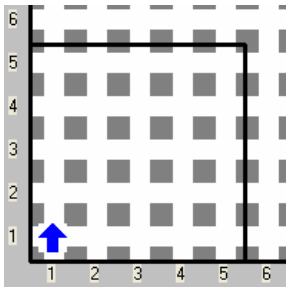


Figura 6

Ya que hemos puesto las paredes seguiremos con poner zumbadores en el mundo.

6) Click derecho sobre la esquina (1,5) Avenida 1, calle 5. y elegir la opción 8 zumbadores como se ve en la figura 7.

7) Hacer lo mismo en las esquinas (5,5) y la esquina (5,1), el resultado se puede observar en la figura 8.

8) Para orientar a Karel, debemos dar click derecho en la esquina donde se cruza la calle 3 con la avenida 3 y aparecerá un menú emergente, (como se muestra en la figura 9).

9) Seleccionar la opción “Situar a Karel”, y después elegir “Orientado al Este”, ver figura 10).

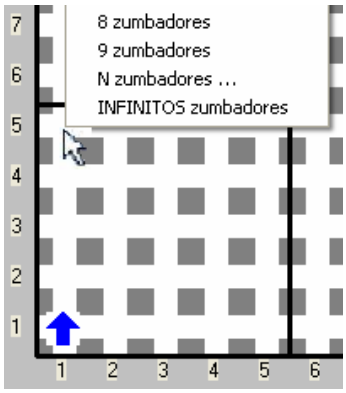


Figura 7

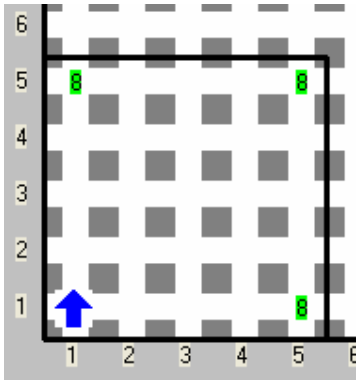


Figura 8

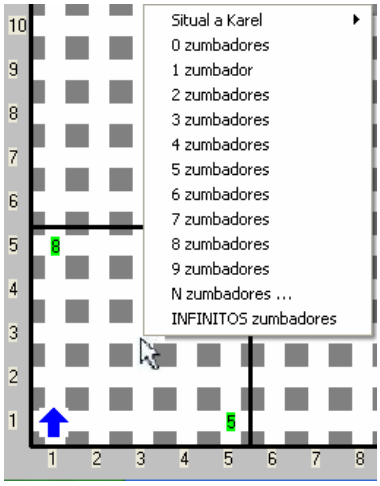


Figura 9

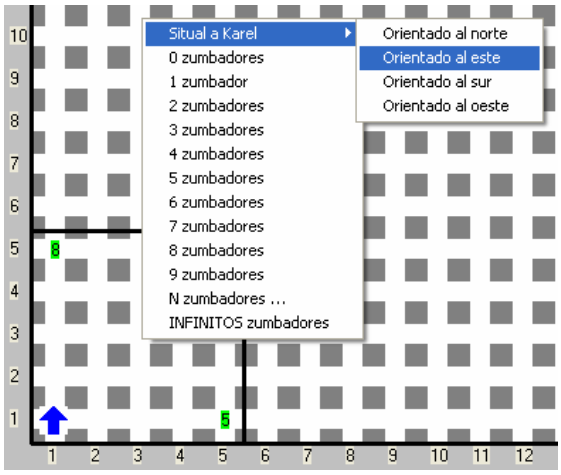
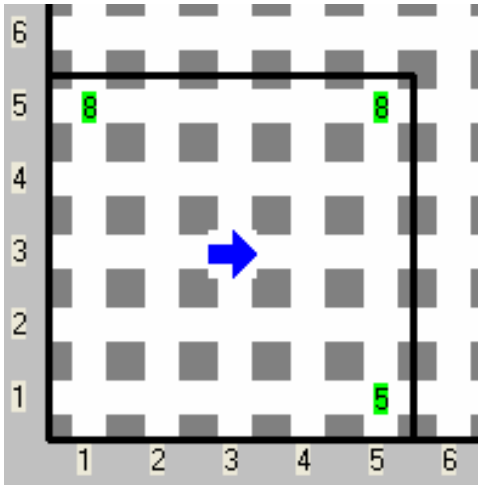


Figura 10

¡¡¡FELICIDADES!!! Has creado tu primer mundo de Karel. Ahora pasaremos a la sección de programar a Karel, para que haga sus tareas en los mundos que diseñes. El mundo que hiciste debió quedar como el de la figura de la derecha. Ahora guarda el mundo con el nombre mundo1, (recuerda usar el botón Guardar como)



1.2 Como hacer un programa básico

La estructura de un programa en Karel cuenta con 2 simples secciones:

1) iniciar-programa … finalizar-programa: Indica donde irá todo el código fuente del programa; donde inicia y donde finaliza.

2) inicia-ejecucion … finaliza-ejecucion: Indica cual será la primera línea en ejecutarse al correr el programa y cual la última. La primera será la que se encuentre inmediatamente después del "inicia-ejecucion" y la última será la que se encuentre inmediatamente antes del "finaliza-ejecucion"

Y además para terminar la ejecución del programa, debe estar la instrucción apágate. Esta indica el final del programa.

Ejemplo:

```
1 iniciar-programa
2     inicia-ejecucion
3         apagate;
4     termina-ejecucion
5 finalizar-programa
6
```

Como se darán cuenta, el programa anterior sólo tiene una instrucción de lo que Karel tiene que hacer. Después de escribir una instrucción, ésta debe terminar con punto y coma (;). El iniciar-programa, inicia-ejecucion, termina-ejecucion y finalizar-programa, no llevan punto y coma, ya que no son instrucciones, o sea, no le ordenan a Karel hacer algo, simplemente marca el inicio y fin de una sección.

El ejemplo anterior, lo que hace, es sólo ordenarle a Karel que se apague. Entonces Si se ejecuta el programa, no se verá que se haga algo en sí. Sólo finalizará inmediatamente.

1.4 Instrucciones Básicas

Karel cuenta con 5 instrucciones básicas, con las cuales le diremos que es lo que tiene que hacer:

avanza	Avanza una cuadra hacia enfrente (depende de la orientación en la que se encuentre)
gira-izquierda	Gira hacia la izquierda a 90 grados
coge-zumbador	Recoge un zumbador de la esquina en la que se encuentra (claro está, si existe un zumbador en la esquina)
deja-zumbador	Deja un zumbador en la esquina en la que se encuentre, si es que tiene algún zumbador en la mochila
apagate	Finaliza la ejecución del programa, ya que no puede hacer mas cosas, porque estará apagado.

Los comandos se escriben tal y como se detallaron, en minúsculas y sin acentos.

```
1  iniciar-programa
2      inicia-ejecucion
3          avanza;
4          gira-izquierda;
5          coge-zumbador;
6          deja-zumbador;
7          apagate;
8      termina-ejecucion
9  finalizar-programa
10
```

El programa abortará anormalmente (dejará de ejecutarse el programa con error) si se dan los casos siguientes:

1) Si se trata de dejar un zumbador en una esquina, y no se tienen zumbadores en la mochila.

2) Si se trata de recoger un zumbador, en una esquina donde no hayan zumbadores.

3) Si se trata de avanzar y enfrente hay una pared (obvio, como va a pasar encima de la pared, Karel no salta).

II. TOMA DE DECISIONES

Karel muchas veces necesita tomar decisiones, de acuerdo a las situaciones en las que se encuentre. Las condiciones que Karel puede detectar (al cabo es un robot) se listan a continuación:

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

2.1 Decisiones Simples

Karel, puede hacer una serie de instrucciones, si es que se cumple con la condición establecida, que puede ser cualquiera de las que se mencionaron anteriormente. Por ejemplo, si tengo hambre voy a comer, y sino, pues no pasa nada.

```
si condicion entonces
inicio
        instruccion1;
        instruccion2;
            .
            .
        instruccionN;
fin;
```

Si se cumple la condición establecida (cualquiera de la lista previamente mostrada), llevará acabo todas las instrucciones que se encuentren dentro del inicio y fin. Si la condición no se cumple, simplemente no llevará a cabo ninguna instrucción.

2.2 Decisiones con otro caso

En este tipo de decisión, Karel, puede hacer una serie de instrucciones si se cumple la condición, y si no se cumple, puede llevar a cabo otra serie de instrucciones diferentes. Ejemplo: si tengo hambre voy a comer y sino, voy a jugar fútbol.

```
si condicion entonces
inicio
        instruccionSi1;
            .
            .
        instruccionSiN;
fin
sino inicio
        instruccionSino1;
            .
            .
        instruccionSinoN;
fin;
```

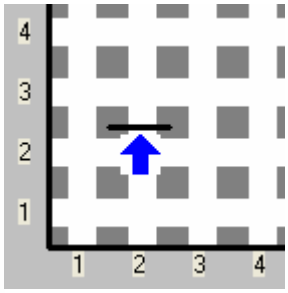
Si la condición establecida se cumple, realizará el bloque de instrucciones que se encuentran entre el “inicio” y el “fin”, y si no se cumple, entonces se ejecutan las instrucciones que se encuentran entre el “sino inicio” y el “fin;”.

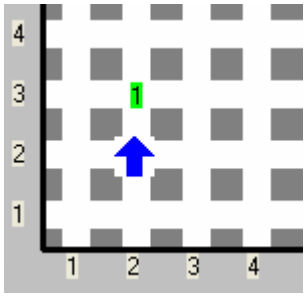
EJERCICIO 2.

Objetivo. Practicar el uso de decisiones simples y a la vez con las instrucciones básicas.

Codifica el siguiente programa (escríbelo en la computadora), en la sección de programa.

```
1  iniciar-programa
2      inicia-ejecucion
3          si frente-libre entonces inicio
4              avanza;
5          fin;
6          gira-izquierda;
7          si junto-a-zumbador entonces inicio
8              coge-zumbador;
9          fin;
10         si algun-zumbador-en-la-mochila entonces inicio
11             gira-izquierda;
12             gira-izquierda;
13             si frente-libre entonces inicio
14                 avanza;
15             fin;
16             deja-zumbador;
17         fin;
18         apagate;
19     termina-ejecucion
20 finalizar-programa
```





Ejecútalo con los 2 mundos mostrados arriba (en la sección ejecutar). Obviamente deberás crear los mundos antes de ejecutar el programa. Karel no debe tener zumbadores en la mochila.

En las líneas de la 3 a la 5 vemos, que Karel, solamente avanzará si no tiene paredes enfrente de él. Después girará a la izquierda, sin depender de ninguna condición, en la línea 6.

Posteriormente, Karel recogerá un solo zumbador, si se encuentra junto a él (líneas 7 a 9). Después, Karel hará otras acciones, sólo si recogió el zumbador, en las líneas de la 10 a la 17.

Si Karel recogió un zumbador, quiere decir que trae un zumbador en la mochila, y si es así, girará 2 veces a la izquierda, para quedar de espaldas a como estaba inicialmente. Karel avanzará sólo si tiene el frente libre, y posteriormente, dejará el zumbador que recogió, en la esquina.

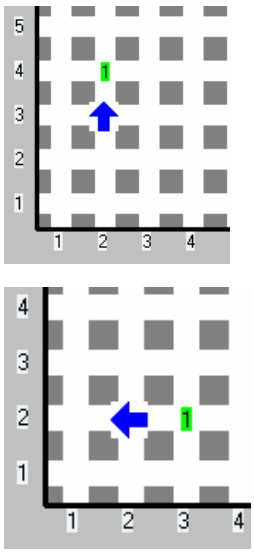
Después de hacer todo esto, Karel se apagará, finalizando el programa.

EJERCICIO 3.

Objetivo. Ver como se usan las decisiones que tienen otro caso.

Karel deberá encontrar un zumbador, que solamente se puede encontrar enfrente de él o detrás de él, y deberá recogerlo. A continuación se muestra el código y ejemplos de cómo pueden ser los mundos:

```
 1 iniciar-programa
 2     inicia-ejecucion
 3         avanza;
 4         si junto-a-zumbador entonces inicio
 5             coge-zumbador;
 6         fin
 7         sino inicio
 8             gira-izquierda;
 9             gira-izquierda;
10             avanza;
11             avanza;
12             coge-zumbador;
13         fin;
14         apagate;
15     termina-ejecucion
16 finalizar-programa
```



En las líneas (3 a 6), busco si enfrente de Karel, existe un zumbador. En la línea 7, sólo si Karel no se encuentra junto a un zumbador, llevará a cabo otra serie de instrucciones, (De la línea 8 a 14). Lo que hará es darse la vuelta y buscar en la parte de atrás de dónde se encontraba inicialmente.

En la línea 12 simplemente recogerá el zumbador que se encuentre allí. Si recuerdan que las instrucciones dicen que forzosamente debe haber un zumbador ya sea enfrente, o detrás de Karel. Al no encontrar Karel un zumbador enfrente, obviamente debe estar detrás. Entonces, al legar a la línea 12, no encontró zumbador enfrente y ahora simplemente recojo el zumbador que se debe encontrar detrás de Karel.

2.3 Decisiones con múltiples condiciones

Las estructuras que se mencionaron anteriormente, sólo toman en cuenta una sola condición, pero también se pueden tomar en cuenta más de una condición en la misma sentencia.

Para unir mas de una condición debemos usar los operadores "*y*" y "*o*", siempre en medio de dos condiciones. O sea que si son 3 condiciones, debemos usar 2 operadores (iguales o diferentes dependiendo del problema).

```
si condicion1 y condicion2 o condicion3 entonces
inicio
        instruccion1;
                .
                .
        instruccionN;
fin;
```

Con el primer operador "y", la condición total se cumplirá sólo si las 2 condiciones que está uniendo se cumplen. "si condicion1 y condicion2 entonces"

Con el operador "o", la condición total será verdadera, si cualquiera de las 2 condiciones es verdadera. "si condicion1 o condicion2 entonces"

Ejemplos de esto pueden ser**:**

si tengo-hambre y es-hora-de-comer entonces inicio voy-a-comer; fin sino inicio voy-a-jugar; fin;	Fíjense, que sólo comerá si tiene hambre y además es la hora de la comida. El irá a jugar, si no tiene hambre o si no es la hora de la comida.
si tengo-hambre o es-hora-de-comer entonces inicio voy-a-comer; fin sino inicio voy-a-jugar; fin;	El irá a comer, si se cumple cualquiera de las condiciones, y él irá a jugar, sino se cumple ninguna condición, o sea, sino tiene hambre, sino es hora de la comida y además, si su mamá no quiere que coma.

Nota, los ejemplos anteriores, son sólo ejemplos ilustrativos, no funcionan en Karel, sólo es para entenderlo mejor en el lenguaje natural.

Para preguntar lo contrario se ocupa la palabra no, separada de la condición.

Ej. "si no derecha-libre entonces", estoy usando otra sentencia en lugar de preguntar por "si derecha-bloqueada entonces", que al final actúa de la misma manera, pero escrito diferente.

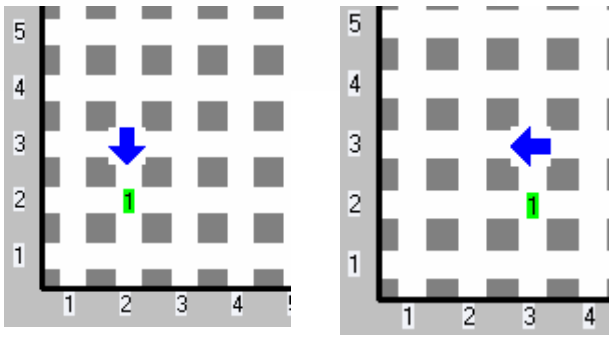
EJERCICIO 4.

Objetivo. Que se entienda con claridad el uso de las condiciones en todos sus aspectos.

Karel deberá recoger un zumbador que puede haber o no, en la parte de debajo de él (una esquina hacia el sur). Si hacia el sur no hay pared, quiere decir que si hay zumbador, pero si hay pared, Karel no recogerá nada. Para que Karel pueda trabajar deberá empezar orientado al sur, sino está orientado al sur, no hará nada.

A continuación se muestran mundos de ejemplo y el código fuente.

```
1 iniciar-programa
2     inicia-ejecucion
3         si frente-libre y orientado-al-sur entonces inicio
4             avanza;
5             coge-zumbador;
6         fin;
7         apagate;
8     termina-ejecucion
9 finalizar-programa
```



Karel simplemente avanzará y recogerá el zumbador, si se encuentra orientado al sur y además tiene el frente libre. Ya que puede estar orientado al sur, pero si frente a él exista una pared, para él sería imposible avanzar.

2.4 Decisiones Anidadas

Después de haber tomado una decisión, dentro de la misma se puede tomar otra u otras, a esto se le llama decisiones anidadas.

Las decisiones anidadas tienen las mismas características que las anteriores, y pueden tener la estructura siguiente:

```
si condicion1 entonces
inicio
        instruccion1;
        instruccion2;
            .
        si condicion2 entonces
        inicio
                instruccionSi1;
                    .
                    .
                instruccionSiN;
        fin
        sino inicio
                instruccionSino1;
                    .
                    .
                instruccionSinoN
        fin;
            .
        instruccionN;
fin;
```

Entonces, dentro de un si, se puede hacer otra instrucción si (o varios), y también dentro de un sino, se puede meter un si (o varios).

En la estructura de ejemplo, la condicion2, solo se va a tomar en cuenta, cuando la condicion1 sea verdadera, o sea, cuando se cumpla.

EJERCICIO 5.

Objetivo. Que se entienda con claridad el uso de las condiciones en todos sus aspectos.

Karel se encontrará en alguna esquina del mundo y orientado hacia cualquier punto. Deberás hacer que Karel se oriente hacia el sur y posteriormente se apague.

```
 1 iniciar-programa
 2     inicia-ejecucion
 3         si no-orientado-al-sur entonces inicio
 4             si orientado-al-norte entonces inicio
 5                 gira-izquierda;
 6                 gira-izquierda;
 7             fin
 8             sino inicio
 9                 si orientado-al-este entonces inicio
10                     gira-izquierda;
11                     gira-izquierda;
12                     gira-izquierda;
13                 fin
14                 sino inicio
15                     gira-izquierda;
16                 fin;
17             fin;
18         fin;
19         apagate;
20     termina-ejecucion
21 finalizar-programa
```

FIGURA 5.1

En la línea 3 preguntamos si Karel no se encuentra orientado al sur, ya que si se encuentra orientado al sur, pues ya no haríamos nada. Si Karel no se encuentra orientado al sur, preguntamos si se encuentra orientado al norte, si es así, pues simplemente giramos 2 veces para orientarnos al sur, líneas 4 a 7. (Ver Figura 5.1).

Si no estoy orientado al sur, ni al norte, ahora pregunto si estoy orientado al este. Si estoy orientado al este, entonces tengo que girar 3 veces para llegar al sur, como lo vemos en las líneas (9 a 13). Pero si tampoco estoy orientado al este, quiere decir que estoy orientado al oeste (ya que anteriormente había preguntado por el sur y por el norte), y sólo me queda girar una vez.

Listo, entonces Karel siempre quedará orientado al sur, sin importar hacia donde esté.

III. REPETICIONES

Muchas veces, Karel, necesita repetir una serie de instrucciones varias veces. A una repetición se le puede llamar "ciclo". Un ejemplo del uso de ciclos: Karel debe avanzar 50 veces hacia delante, lo que tendríamos que hacer es escribir 50 veces la instrucción avanza; como podrán ver es un trabajo pesado. Para disminuir el trabajo, se tienen sentencias de repeticiones.

3.1 Repeticiones fijas

Tú le puedes decir a Karel exactamente cuantas veces va a repetir una misma instrucción o serie de instrucciones. Para ello usamos la instrucción repetir N veces, la cual tiene la siguiente estructura:

```
repetir N veces
inicio
        instruccion1;
        instruccion2;
                .
                .
        instruccionN;
fin;
```

donde N es un número que indicará la cantidad de veces que se repetirán las instrucciones que se encuentren entre el inicio y fin (dentro de la instrucción repetir).

Para hacer que Karel avance 20 veces hacia delante, usaremos la instrucción repetir de la siguiente manera:

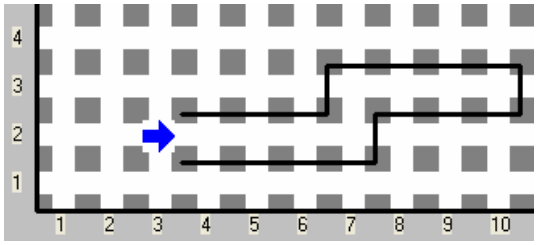
```
1  iniciar-programa
2      inicia-ejecucion
3          repetir 20 veces
4          inicio
5              avanza;
6          fin;
7          apagate;
8      termina-ejecucion
9  finalizar-programa
10
```

EJERCICIO 6.

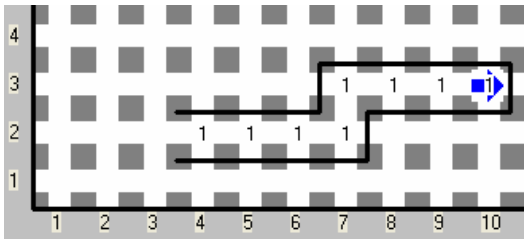
Objetivo. Practicar el uso y entendimiento de la funcionalidad del ciclo repetir.

Se pide que Karel, deje 1 zumbador en cada esquina que se encuentre dentro de las paredes. Al iniciar la ejecución Karel deberá tener en su mochila 10 zumbadores.

A continuación se muestra el mundo inicial y el mundo como deberá quedar después de la ejecución.



Mundo Inicial



Mundo Final

```
 1 iniciar-programa
 2     inicia-ejecucion
 3         repetir 4 veces inicio
 4             avanza;
 5             deja-zumbador;
 6         fin;
 7         gira-izquierda;
 8         avanza;
 9         repetir 3 veces inicio
10             gira-izquierda;
11         fin;
12         repetir 4 veces inicio
13             deja-zumbador;
14             si frente-libre entonces
15             inicio
16                 avanza;
17             fin;
18         fin;
19         apagate;
20     termina-ejecucion
21 finalizar-programa
```

Karel Avanza cuatro y veces, y va dejando un zumbador en cada paso (líneas 3-6). Al llegar a la esquina, gira hacia la izquierda (Al norte). Y avanza. Para llegar a la calle 3.

Después hace 3 giros a la izquierda (recuerden que no puede girar hacia la derecha), para poder orientarse a la derecha.

Por último va a repetir cuatro veces, dejar un zumbador y avanzar, solo si el frente está libre (ver líneas 12-18). Al finalizar esta repetición, Karel estará hasta el fondo y ya habrá terminado correctamente su trabajo.

Se debe notar que en la ultima repetición se pregunta si está el frente libre, porque aquí primero se deja un zumbador, y posteriormente se avanza. Al llegar al fondo, si no se preguntara, Karel dejaría un zumbador e intentaría pasar por encima de la pared, entonces nuestro programa estaría mal.

En la primera repetición no se preguntaba por el frente libre, ya que Karel primero avanzaba y posteriormente dejaba el zumbador. Entonces Karel al llegar al fondo, simplemente deja el zumbador, y ahí finaliza el ciclo.

3.2 Repeticiones condicionales

Si necesitas que Karel repita una serie de instrucciones mientras se cumpla una condición, se usa la sentencia mientras…hacer. Esto se puede ejemplificar de la siguiente manera: Andas en una bicicleta y vas a pedalear mientras tengas energía.

Lo que vas a repetir es pedalear, y lo vas a repetir mientras tengas energía, pero no sabes cuantas veces vas a pedalear antes de que se te acabe la energía.

Entonces, la sentencia mientras, la usas cuando no sabes exactamente cuantas veces vas a repetir las instrucciones, y la repetición depende sólo de una condición (también puede ser un conjunto de condiciones unidas como se vio en el tema 2.3).

La estructura de la sentencia es la siguiente:

```
mientras condicion hacer
inicio
        instruccion1;
        instruccion2;
                .
                .
        instruccionN;
fin;
```

Donde la condición, es cualquiera que pueda detectar Karel (las que se vieron en el capitulo 2), y dentro del inicio y fin, van la serie de instrucciones (aunque puede ser solo una) que se van a repetir dependiendo de la condición.

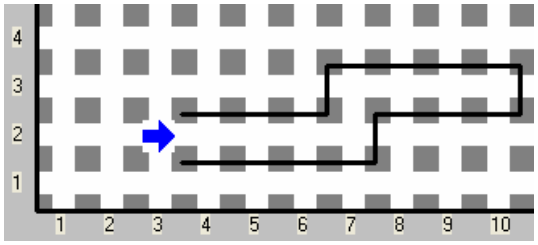
Un ejemplo, es que Karel avance hasta topar con una pared. En este punto cabe recordar que si intenta avanzar y hay una pared enfrente, el programa abortará, o sea, terminará, pero anormalmente, ya que no pasó por la instrucción apagate.

```
1  iniciar-programa
2      inicia-ejecucion
3          mientras frente-libre hacer
4          inicio
5              avanza;
6          fin;
7          apagate;
8      termina-ejecucion
9  finalizar-programa
10
```

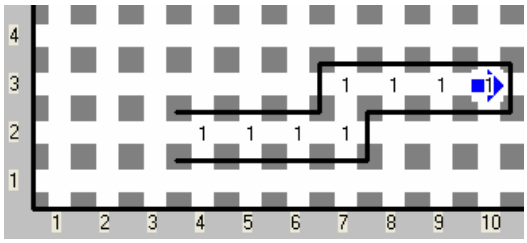
EJERCICIO 7.

Objetivo. Determinar las diferencias y similitudes entre los diferentes tipos de ciclos.

Karel resolverá el mismo problema anterior, pero ahora sin usar la sentencia repetir, sino ahora con el mientras. Recuerden los mundos:



Mundo Inicial



Mundo Final

```
1  iniciar-programa
2      inicia-ejecucion
3          mientras frente-libre hacer inicio
4              avanza;
5              deja-zumbador;
6          fin;
7          gira-izquierda;
8          avanza;
9          mientras no-orientado-al-este hacer
10         inicio
11             gira-izquierda;
12         fin;
13         deja-zumbador;
14         mientras frente-libre hacer inicio
15             avanza;
16             deja-zumbador;
17         fin;
18         apagate;
19     termina-ejecucion
20 finalizar-programa
```

Aquí Karel, avanzará mientras tenga el frente libre, o sea, dejará de avanzar cuando se encuentre con una pared. Y mientras va avanzando, va dejando un zumbador. (Líneas 3-6).

Al topar con pared, Karel girará a la izquierda, y avanzará, para situarse en la calle 3 (Ver líneas 7-8). Posteriormente se orientará al Este, para continuar con su trabajo. Para ello, girará hacia la izquierda, hasta que se encuentre orientado al Este, como se ve en las líneas (9-12).

Después Karel, dejará un zumbador y volverá a hacer el primer ciclo, que era avanzar y dejar un zumbador, mientras no haya pared. (Líneas 13-17).

Cuando Karel encuentre la pared, habrá finalizado la ejecución.

3.3 Ciclos anidados

Al igual que en la sentencia si, en un ciclo (sea repetir o mientras) se pueden meter otros ciclos, uno solo, o varios, y no importan si son del mismo tipo o de diferente. También dentro de un ciclo, se puede meter la sentencia si.

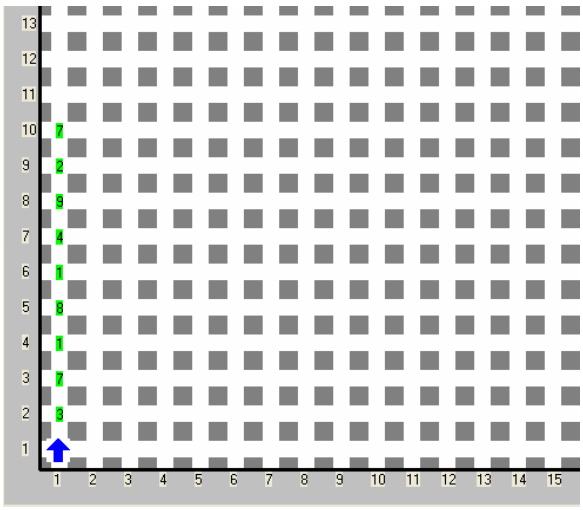
La estructura podría ser como las siguientes:

```
repetir N veces
inicio
        sentencia1;
        mientras condicion hacer
        inicio
                sentenciaMientras1;
                sentenciaMientras2;
                        .
                        .
                sentenciaMientrasN;
        fin;
        sentencia2;
                .
                .
        sentenciaN;
fin;
```

```
mientras condicion hacer
inicio
        repetir N veces
        inicio
                sentencia1;
                sentencia2;
                        .
                        .
                sentenciaN;
        fin;
        sentenciaMientras1;
        sentenciaMientras2;
                .
                .
        sentenciaMientrasN;
fin;
```

Un ejemplo de esto: Karel, caminará de frente, por las 10 primeras calles, en las cuales habrán zumbadores tirados; él tendrá que recoger todos los zumbadores que encuentre en las primeras 10 calles hacia su frente.

```
1  iniciar-programa
2      inicia-ejecucion
3          repetir 10 veces
4          inicio
5              mientras junto-a-zumbador hacer
6              inicio
7                  coge-zumbador;
8              fin;
9              avanza;
10         fin;
11         apagate;
12     termina-ejecucion
13 finalizar-programa
14
```

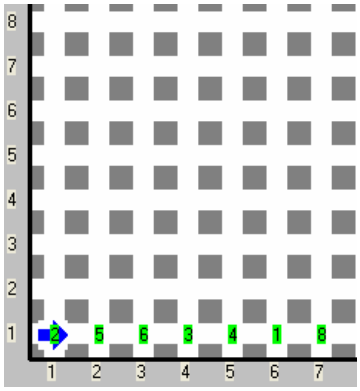


EJERCICIO 8.

Objetivo. Entender 100% el uso y aplicación de los ciclos.

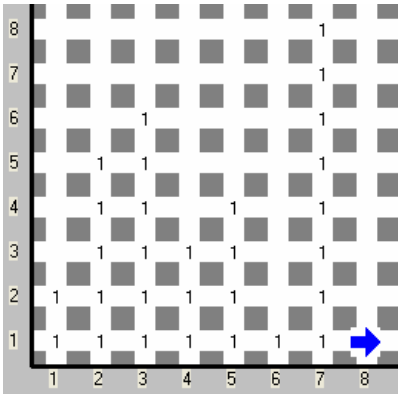
Karel deberá hacer una grafica de barras con los zumbadores que se encuentren en las esquinas frente de él. Todos los montones están juntos sobre la calle 1, por lo que el último montón, es el que está antes de una esquina vacía.

Karel deberá hacer una fila de zumbadores en cada avenida, con cada montón que encuentre en la primera calle. Se puede observar un mundo inicial de ejemplo y el mundo como deberá quedar, al finalizar la ejecución. Karel al iniciar no tendrá zumbadores en su mochila.



<- Mundo inicial

Mundo Final ->



Mientras Karel se encuentre junto a un montón, girará a la izquierda para orientarse al Norte, posteriormente, recogerá todos los zumbadores (líneas 3-7), e irá dejando uno por uno en cada esquina hacia el norte (líneas 8-12).

Después regresará a la calle 1, dando vuelta y avanzando hasta la pared, (ver líneas 13-17).

```
1  iniciar-programa
2      inicia-ejecucion
3          mientras junto-a-zumbador hacer inicio
4              gira-izquierda;
5              mientras junto-a-zumbador hacer inicio
6                  coge-zumbador;
7              fin;
8              mientras algun-zumbador-en-la-mochila hacer
9              inicio
10                 deja-zumbador;
11                 avanza;
12             fin;
13             gira-izquierda;
14             gira-izquierda;
15             mientras frente-libre hacer inicio
16                 avanza;
17             fin;
18             gira-izquierda;
19             avanza;
20         fin;
21         apagate;
22     termina-ejecucion
23 finalizar-programa
```

Y ahora girará a la izquierda y avanzará, para posicionarse sobre otro montón sobre la calle 1, si es que existe, para repetir nuevamente lo explicado antes. Karel finalizará cuando al final ya no se encuentre junto a un montón de zumbadores (o a un solo zumbador).

IV. NUEVAS INSTRUCCIONES

Con el programa, nosotros podemos crear nuestras instrucciones, apoyándonos con otras ya creadas previamente por nosotros, o con las que Karel ya trae.

Como se dijo anteriormente, Karel sólo gira a la izquierda. Pero nosotros, podemos crear, por ejemplo, nuestra instrucción gira-derecha. Y nos apoyaríamos de la instrucción gira-izquierda y la usaríamos 3 veces, ya que como Karel gira a 90º, si gira 3 veces a la izquierda, quedará exactamente a la derecha de donde empezó.

La estructura para definir una nueva instrucción es:

```
define-nueva-instruccion nombre como
inicio
        instruccion1;
                .
                .
        instruccionN;
fin;
```

donde nombre, es como se llamará nuestra nueva instrucción, y dentro del inicio y fin, irán todas las instrucciones que llevará a cabo nuestra instrucción.

Como ejemplo, para crear la instrucción gira-derecha quedaría:

```
1  iniciar-programa
2      define-nueva-instruccion gira-derecha como
3      inicio
4          gira-izquierda;
5          gira-izquierda;
6          gira-izquierda;
7      fin;
8      inicia-ejecucion
9          gira-derecha;
10         apagate;
11     termina-ejecucion
12 finalizar-programa
13
```

Como podrán observar, la nueva instrucción se escribe en la sección que se encuentra entre el “iniciar-programa” y el “inicia-ejecucion”, o sea, antes que Karel empiece a funcionar.

Una vez que Karel esté haciendo su labor, ya puedo darle las nuevas órdenes que hemos creado. Pueden observar que el programa llama a gira-derecha (previamente creado) y posteriormente se apaga.

Una instrucción también puede tener un parámetro, que es un valor entero (0,1,2…) que se guarda en una variable, y después se puede hacer operaciones con él. Se define de la siguiente manera:

Al crear una nueva instrucción, después de poner el nombre de la misma, se le ponen paréntesis “( ) “ y dentro de ellos el nombre de la variable que guardará el valor entero.

Ejemplo: define-nueva-instrucción contar(num) como ….

En el ejemplo anterior, se definió una instrucción que se llama contar y además recibe un valor entero y lo guarda en una variable que se llama num.

Cuando se hace la llamada a ésta instrucción, se le debe mandar el valor que guardará num.

Ej. contar(5);

El 5 se guardará en la variable num.

Los comentarios, se refieren a textos que podemos escribir en el programa, que no son instrucciones y que sólo los usamos como notas. Pero el programa no los tomará en cuenta para su funcionamiento. Los comentarios son útiles, cuando se desea tener un programa claro, y además para recordar que es lo que hicimos, si es que lo revisamos en un buen lapso después de haberlo hecho.

Para poner un texto como un comentario, debe ser de la siguiente forma: antes del texto que será un comentario se pone un paréntesis que abre, seguido de un asterisco (* y al final del texto se pone un asterisco y un paréntesis que cierra *).

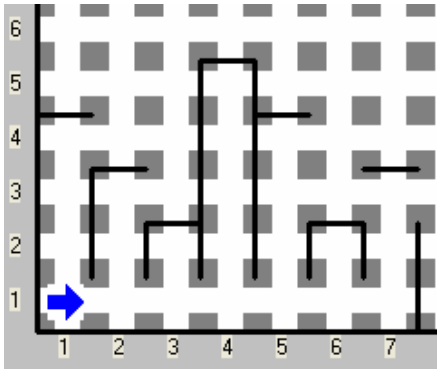
```
1  iniciar-programa
2      (* MI INSTRUCCION PARA AVANZAR HASTA
3          QUE SE ENCUENTRE CON UNA PARED *)
4      define-nueva-instruccion avanza-hasta-pared como
5      inicio
6          (* Avanzar mientras no hayan paredes*)
7          mientras frente-libre hacer inicio
8              avanza;
9          fin;
10     fin;
11     inicia-ejecucion
12         avanza-hasta-pared; (*Llama a la instruccion*)
13         apagate;
14     termina-ejecucion
15 finalizar-programa
16
```

EJERCICIO 9.

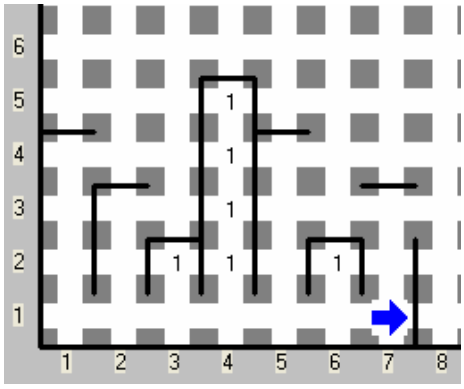
Objetivo. Practicar el uso de crear nuevas instrucciones, para familiarizarse con esto que es muy usual.

Karel deberá rellenar cada cuarto con zumbadores, pero solo los cuartos que no tengan puerta al final, o sea, tengan pared al norte, pared al este y pared al oeste.

Karel finalizará su trabajo cuando se tope con una pared en la calle 1. Karel tendrá INFINITO zumbadores en su mochila.



Mundo Inicial



Mundo Final

Karel, va ir avanzando sobre la calle 1 hasta que se encuentre junto a un zumbador (cuando tope con pared, dejará un zumbador para marcar el fin de la calle), y en cada esquina que avance, va a girar hacia la izquierda, para quedar orientado hacia el norte, y va a avanzar hasta topar con pared, o sea, hasta el fondo del cuarto (ver líneas 15-18). Estando al fondo del cuarto, Karel, verificará que las paredes de su izquierda y derecha se encuentren libres (línea 19).

En éste punto Karel, tomará la decisión de dejar un zumbador en cada esquina hasta llegar a la calle 1 (líneas 20-25), o simplemente regresar a la calle 1 sin dejar nada (líneas 26-28).

Después de tomar su decisión, y realizar lo decidido, girará a la izquierda, para estar orientado al este y avanzará a la siguiente avenida si no existe pared (como se muestra en las líneas 29-31).

Y se repite la misma historia, Karel vuelve a realizar lo del principio, hasta que en este punto se encuentre junto a un zumbador, que marca el final de la calle. Para finalizar, simplemente recogerá el zumbador que le sirvió como marca, para sabes cual era el final de la calle.

En la página siguiente, se muestra el código fuente para resolver éste problema.

```
1  iniciar-programa
2      define-nueva-instruccion avanza-hasta-pared como inicio
3          mientras frente-libre hacer
4              avanza;
5      fin;
6      define-nueva-instruccion vuelta como inicio
7          gira-izquierda;
8          gira-izquierda;
9      fin;
10     define-nueva-instruccion gira-derecha como inicio
11         vuelta;
12         gira-izquierda;
13     fin;
14     inicia-ejecucion
15         mientras no-junto-a-zumbador hacer inicio
16             si frente-bloqueado entonces deja-zumbador;
17             gira-izquierda;
18             avanza-hasta-pared;
19             si izquierda-bloqueada y derecha-bloqueada
20             entonces inicio
21                 vuelta;
22                 mientras frente-libre hacer inicio
23                     deja-zumbador;
24                     avanza;
25                 fin;
26             fin sino inicio
27                 vuelta;
28                 avanza-hasta-pared;
29             fin;
30             gira-izquierda;
31             si frente-libre entonces avanza;
32         fin;
33         coge-zumbador;
34         apagate;
35     termina-ejecucion
36 finalizar-programa
```

V. RECURSIVIDAD

La recursividad es algo que se define a si mismo; y en Karel se refiere a cuando una instrucción se llama a sí misma.

Cuando una nueva instrucción hecha por nosotros, en su bloque de instrucciones, tiene una llamada a la misma instrucción que hicimos, se dice que es una instrucción recursiva. Ejemplo:

```
1  iniciar-programa
2      define-nueva-instruccion avanza-hasta-pared como
3      inicio
4          si frente-libre entonces
5          inicio
6              avanza;
7              avanza-hasta-pared;
8          fin;
9      fin;
10     inicia-ejecucion
11         avanza-hasta-pared;
12         apagate;
13     termina-ejecucion
14 finalizar-programa
15
```

Dentro de la instrucción avanza-hasta-pared se hace una llamada así misma (como se observa en la línea 7), entonces es recursiva.

Una instrucción recursiva, siempre debe contener una condición de paro, que es la que indicará si la recursividad terminará o si se vuelve a llamar a la instrucción. En el código de ejemplo pueden observar la condición en la línea 4; si la condición se cumple, entonces vuelve a llamar a la instrucción, y sino, ya no la llama. O sea, que si la condición no se cumple, la recursividad finaliza.

Si la condición no existiese y hubiera una llamada recursiva, la instrucción sería infinita (se dice que el programa se ciclaría), ya que siempre se estaría llamando a la instrucción, y la instrucción se volvería a llamar, y así sucesiva e infinita mente.

Existe instrucción que nos dice si una constante o variable, es un cero.

Sintaxis: si-es-cero(N)

donde: N puede ser una constante (1,25,6…) o una variable (a, num, n…)

Ejemplo:

```
si si-es-cero(5) entonces
inicio
        avanza;
fin
sino inicio
        gira-izquierda
fin;
```

Obviamente siempre girará a la izquierda, ya que el 5 nunca podrá ser un 0. O sea, el 5 es una constante, porque siempre tiene el mismo valor.

```
si si-es-cero(num) entonces inicio
        avanza;
fin
sino inicio
        gira-izquierda
fin;
```

Aquí puede avanzar o girar a la izquierda, pero todo depende del valor que tenga guardado la variable num.

num es una variable, ya que su valor puede variar.

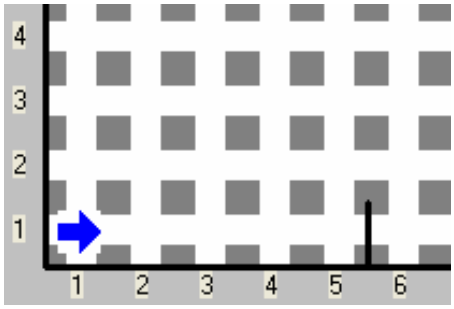
EJERCICIO 10.

Objetivo. Razonar y entender con más profundidad el uso de la recursividad.

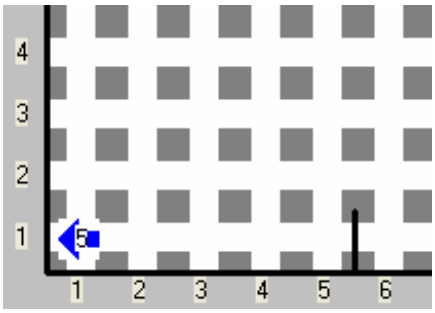
Karel deberá contar cuantas esquinas hay desde la esquina (1,1) hasta topar con una pared, que deberá haber en la calle 1. Karel deberá poner la cantidad de zumbadores igual a la cantidad de esquinas de distancia, los deberá poner en la esquina (1,1). Como se muestra en las figuras siguientes:

```
1  iniciar-programa
2     define-nueva-instruccion cuenta como inicio
3          si frente-libre entonces
4          inicio
5              avanza;
6              cuenta;
7              deja-zumbador;
8          fin
9          sino inicio
10             gira-izquierda;
11             gira-izquierda;
12             mientras frente-libre hacer
13                 avanza;
14         fin;
15     fin;
16     inicia-ejecucion
17         cuenta;
18         deja-zumbador;
19         apagate;
20     termina-ejecucion
21 finalizar-programa
```

Mundo Inicial



Mundo Final



Primeramente se llama a la función cuenta, que será la función recursiva. La función recursiva se estará llamando mientras el frente esté libre, como se ve en las líneas 2 y 3. Lo primero que se condiciona es que el frente se encuentre libre, si esto es así entonces avanzará una esquina, y volverá a llamar a la función cuenta, que volverá a hacer lo mismo (líneas 3-6). Si el frente no está libre, se dará la vuelta y regresará a la esquina 1,1 como se muestra en las líneas (10-13). En ese momento la recursión termina, porque ya no se vuelve a llamar a la función cuenta, ni una vez más.

Entonces en cada llamada a la función cuenta que se hizo, pasará a la siguiente línea en que se quedó (pasa a la línea 7), o sea, estará dejando un zumbador por cada llamada que se hizo. El número de llamadas que se hicieron es igual a la cantidad de esquinas menos 1, ya que para avanzar a la primera esquina, no se llamó a la función, porque Karel ya se encontraba en la primera esquina.

Al finalizar completamente la recursión, simplemente dejará otro zumbador para completar la cantidad de esquinas que existen (línea 18).

VI. ARITMETICA CON KAREL

Karel cuenta con otras 2 funciones:

1) sucede : lo que hace esta función, es incrementar en uno un valor constante, o un valor que esté guardado en una variable.

Con lo antes mencionado, si la aplico de la siguiente manera: sucede(5), lo que me va a regresar sería un 6.

Sintaxis:

sucede(N); donde N es un valor entero, o puede ser una variable que esté dentro de una función.

2) precede : decrementa en uno una constante o una variable. Si hacemos un precede(n) y n vale 8, la función me regresará un 7.

Sintaxis:

precede(N); donde N es un valor entero, o puede ser una variable que tenga una función.

Un ejemplo de su uso sería dejar 10 zumbadores a lo largo de la calle (obviamente debe tenerlos en su mochila):

```
1  iniciar-programa
2      define-nueva-instruccion dejalos(num) como
3      inicio
4          si no si-es-cero(num) entonces
5          inicio
6              deja-zumbador;
7              avanza;
8              dejalos(precede(num));
9          fin;
10     fin;
11     inicia-ejecucion
12         dejalos(10);
13         apagate;
14     termina-ejecucion
15 finalizar-programa
16
```

Empieza a contar desde 10 y va dejando un zumbador y después avanza, posteriormente disminuye su cuenta en 1, en el primer caso a 9 y vuelve a repetir el proceso, hasta que llegue a 0. Cuando llega a 0 ya no hace nada.

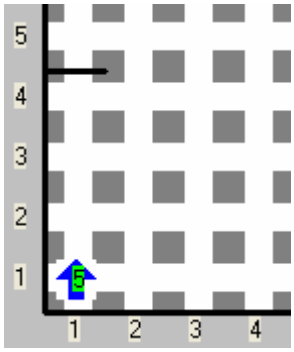
EJERCICIO 11.

Objetivo. Practicar el uso de estas 2 nuevas instrucciones, y ver su funcionabilidad en conjunto con la recursividad y la función si-es-cero().

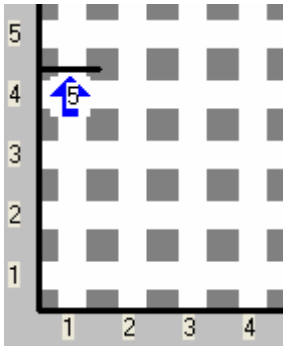
Karel deberá recoger todos los zumbadores que se encuentren junto de él, y llevarlos hasta que se encuentre con una pared frente de él. Tendrá INFINITO zumbadores en la mochila al iniciar el programa.

```
 1 iniciar-programa
 2     define-nueva-instruccion deja(m) como inicio
 3         si no si-es-cero(m) entonces
 4         inicio
 5             deja-zumbador;
 6             deja(precede(m));
 7         fin;
 8     fin;
 9     define-nueva-instruccion cuenta(n) como inicio
10         si junto-a-zumbador entonces inicio
11             coge-zumbador;
12             cuenta(sucede(n));
13         fin
14         sino inicio
15             mientras frente-libre hacer
16                 avanza;
17             deja(n);
18         fin;
19     fin;
20     inicia-ejecucion
21         cuenta(0);
22         apagate;
23     termina-ejecucion
24 finalizar-programa
```

Mundo Inicial



Mundo Final



En la línea 21, se llama a la función que será la encargada de realizar el programa, se llama con un valor de 0 que se le asignará a la variable n.

Si Karel se encuentra junto a un zumbador, lo recogerá y volverá a llamar a la función pero incrementando el valor de la variable n en uno (va contando uno a uno la cantidad de zumbadores, ver líneas 10-12). Y si ya no hay zumbadores junto de él, avanzará hasta topar con pared, y llamará a la función deja, que será la encargada de contar cuantos zumbadores se deben dejar, y se le asigna el valor de n a la variable m (guardarán el número de zumbadores que recogió, ver líneas 14-18).

La función deja, pregunto si la variable n vale 0, si no vale cero, entonces deja un zumbador, y vuelve a llamar a la función deja, pero ahora decrementando el valor de m, (para marcar que se tiene un zumbador menos en la mochila, como se ve en las líneas 2-8).

Cuando la variable m, valga 0, entonces se habrán dejado todos los zumbadores que se recogieron, y ya no se llamará nuevamente a ninguna función, por ende el programa habrá finalizado.