Versión de JAVA

Las funciones sucede y precede

Ahora ya sabemos como mandarle un número a un procedimiento, y probablemente ya habrás intentado poner operaciones como suma, resta o multiplicación, sin embargo lamanto decirte que ninguna de estas operaciones están soportadas en Karel.

Por otro lado, existen dos funciones que nos permiten sumarle 1 a un número y restarle 1.

Pero... ¿qué es una función? Una función es una instrucción que devuelve un valor, es decir, reciben un parámetro (o más) que luego procesa, para al final regresar un valor; por ejemplo, la función booleana junto-a-zumbador devuelve verdadero si Karel está parado junto a un zumbador y falso si no lo está. En Karel no se pueden declarar funciones nuevas, pero se pueden usar las que ya existen.

Las funciones sucede y precede son dos instrucciones que reciben un parámetro, posteriormente, devuelven un número más y un número menos (respectivamente) que el que le enviamos.

La función sucede se escribe así:

...

sucede(xxx);

...

donde xxx es un número o un parámetro, y la función precede se escribe así:

...

precede(xxx);

...

donde xxx es un número o un parámetro.

Debido a que devuelven un número, solo nos pueden servir poniendolas en alguna instrucción o sentencia que reciba un número, como repetir/veces, otro sucede o precede o una instrucción personal que reciba un parámetro.

Por ejemplo, el siguiente trozo de código pone n + 1 zumbadores en donde Karel se encuentra:

...

repetir sucede(n) veces inicio

  deja-zumbador;

fin;

...

nota que n se "incrementa" (se le suma uno). Si en vez de sucede, pusieramos precede, Karel dejaría n - 1 zumbadores, porque la n se "decrementa" (se le quita uno) cuando se pone dentro de una función precede.

Ejercicio 11: Escribe una nueva instrucción que reciba un número n y mueva a Karel n + 2 veces (validando el choque contra paredes), y posteriormente, coloque n - 2 zumbadores en la posición en donde está. PISTA: Usa un repetir/veces para mover a Karel y otro para colocar los zumbadores. NO se vale colocar instrucciones fuera de los ciclos repetir/veces.

Subir

La función si-es-cero

La última función en Karel, es la función si-es-cero, que nos ayuda a saber si un número es cero. Devuelve verdadero si el número es cero y falso si no lo es.

Es evidente de que si ponemos "si-es-cero(0)" no es muy útil, ya que sabemos perfectamente que cero es

cero (:S). Sin embargo es muy útil cuando se está manejando parámetros. Por ejemplo, queremos hacer una instrucción que avance "n" lugares, pero si el parámetro es cero, gire a la izquierda.

Por razones didácticas, en esta ocasión te daremos la solución:

```
...

define-nueva-instruccion avanza-si-no-es-cero (n) como inicio

  si si-es-cero (n) entonces inicio

    gira-izquierda;

  fin

  sino inicio

    repetir n veces inicio

      avanza;

    fin;

  fin;

fin;

...
```

Ahora si, te toca a ti:

Ejercicio 12: Define una nueva instrucción que haga que Karel ponga "n" zumbadores en donde se encuentra, pero si "n" es cero, recoja 1 zumbador.

Subir

Recursividad con parámetros

¿Recuerdas la recursión (recursividad)? ¿Que pasaría si a una instrucción con parámtro la llamaramos usando precede o sucede?

Todo esto, en Karel es posible, definamos una instrucción con parámetros y volvámosla recursiva usando sucede:

```
...
define-nueva-instruccion recursiva2 (n) como inicio
  si frente-libre entonces inicio
    avanza;
    recursiva2 ( sucede (n) );
  fin
  sino inicio
    repite n veces inicio
      deja-zumbador;
    fin;
  fin;
fin;
...
```

Este código hace exactamente lo mismo que el de la sección anterior, con la diferencia de que la complejidad que existía en la pila de llamadas es eliminada, y colocada como una instrucción base. En cada paso de la recursión, se aumenta uno, y cuando llegas a la base, tiene un número que tal vez pudas usar para tu beneficio. No te olvides que también puedes usar la función precede y si-es-cero y que en la llamada recursiva puedes llamarla con el parámetro sin modficar.

Como punto importante, hay que destacar que en la pila de llamdas, además de la instrucción que sigue, también se guarda el valor actual del parámetro.

Ejercicio 13: Realiza el Ejercicio 10 pero con Recursividad con parámetros.

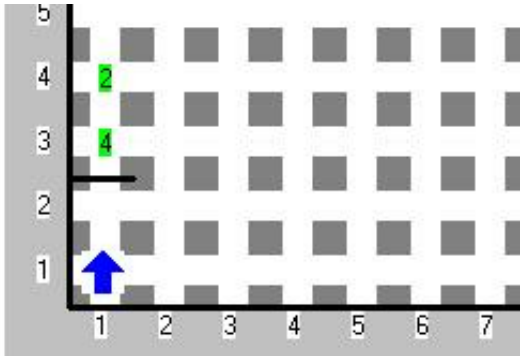
Ejercicio 14: Realiza una instrucción que avance a Karel tantas veces como zumbadores tenga en su mochila utilizando Recursividad con parámetros, NO se vale usar un ciclo mientras.

Subir

Recursividad mixta

Si ya dominas los dos tipos de recursividad anteriores, esto será muy fácil para ti, ya que la recursividad mixta no es más que usar los dos tipos de recursividad al mismo tiempo.

Veamos el siguiente problema:



Teniendo un mundo como el de arriba y con zumbadores infinitos en la mochila, coloca en 1,1 el resultado de la multiplicación.

Este es uno de los problemas más clásicos de recursividad mixta, sin ella la complejidad del problema sería muy elevada.

Para este problema usaremos dos instrucciones, uno con recursividad simple y el otro con parámetro.

La mecánica de solución es la siguiente: Se van a tomar todos los zumbadores de una posición con recursividad con parámetros, para que al llegar a la base, el parámetro tenga el número de zumbadores que hay en dicha posición, posteriormente, ese número se pasa como parámetro a la instrucción recursiva simple (si, aunque la recursividad es simple, también tiene parámetro), y en cada paso, se le va a llamar recursivamente con el mismo parámetro, de esa forma, en la pila de llamadas, el parámetro va a estar tantas veces como zumbadores haya en la segunda posición, y vuala. Tal vez quede más claro si ves las instrucciones y las sigues paso a paso en Karel.

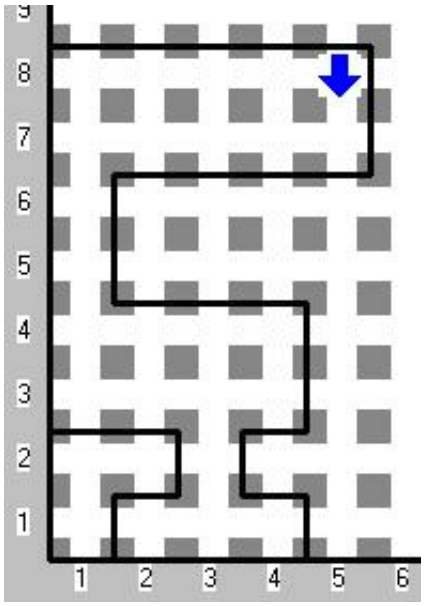
...

```
define-nueva-instruccion multiplica (n) como inicio

  si junto-a-zumbador entonces inicio

    coge-zumbador;

    multiplica(n);
```

```
    repetir n veces inicio
      deja-zumbador;
    fin;
  fin
  sino inicio
    veAlInicio;
  fin;
fin;
define-nueva-instruccion cuenta (n) como inicio
  si junto-a-zumbador entonces inicio
    coge-zumbador;
    cuenta( sucede (n) );
  fin
  sino inicio
    avanza;
    multiplica (n);
  fin;
fin;
...
```

Esta casi completa, pero le falta implementar la instrucción veAlInicio (que debe ser de lo más trivial) y el posicionamiento inicial, tal vez en un principio no entiendas bien, revisa el código paso a paso hasta que lo comprendas, ¡Suerte!

Ejercicio 15: Teniendo un mundo como el de abajo:



en donde puede haber cualquier cantidad de calles y cualquier ancho en las avenidas (siempre cerrado el circuito y sin "islas" o bifurcaciones),deje en la esquina izquierda más inferior (2,1 en el ejemplo), tantos zumbadores como número de calles de ancho 1 que hay, Karel lleva infinitos zumbadores en su mochila. En resultado del ejemplo es 3.

En este problema se usan casi todos los temas vistos en el curso, emplealos y resuélvelo.

Subir

Ahora ya debes de estar listo para resolver los problemas de la sección Recursividad con parámetros

Origen del curso: Curso de Karel OMI

Creador del módulo: Félix Rafael Horta Cuadrilla