Lenguaje de Programación: Usos comunes de la Recursión

José Luis Alonzo Velázquez

Universidad de Guanajuato

Agosto 2010

Propiedades de las funciones recursivas

El proceso para escribir una función recursiva es muy parecido al que usamos para escribir un ciclo:

1. Hay que encontrar la condición de paro o caso base. Para esto hay que intentar responder la pregunta ¿ Cuál es el caso más simple que puede resolverse?
2. ¿Qué es lo que Karel tiene que hacer en el caso base?
3. Encontrar una manera de revolver un pedazo pequeño del problema cuando no estemos en el caso base. Esto es llamado "*reducir el problema al caso general*" .
4. Asegurarse que la reducción nos lleve al caso base.

NOTA

La diferencia más importante entre las soluciones recursivas y las iterativas es que un ciclo iterativo tiene que terminar cada iteración antes de poder empezar la siguiente. Mientras que una instrucción recursiva generalmente inicia una nueva copia antes de haber completado la actual. En esos casos, la copia actual es suspendida temporalmente esperando a que termine la nueva copia y cuando eso suceda la copia actual podrá continuar la ejecución de sus instrucciones. Lo que generalmente sucede es que la nueva copia no se completa sin antes generar otra nueva copia. Cada copia debe ser finalizada en orden, siendo la primera que se creó la última en terminar.

Contar pasos

Pasos

Para motivar el ingenio de Karel, el Dr. Isaac le prometió una docena de beepers si lograba realizar el siguiente reto: "debe caminar hasta topar con el primer beeper y luego caminar, al norte, tantas veces como caminaste desde tu punto inicial hasta el beeper".

Problema

Karel se encuentra a n pasos de una esquina X marcada con un beeper. Karel debe colocarse en la esquina que esta justamente a n pasos al norte de la esquina X.

Consideraciones

- No importa en que posición y orientación tenga que finalizar Karel al finalizar.
- Desconoces la distancia entre Karel y el beeper.
- Desconoces la orientación de Karel. Solo sabes que esta mirando al beeper.
- No importa la orientación final de Karel, solo su posición inicial.

Veamos los 4 pasos

1. ¿Cuál es el caso base? Karel está sobre el beeper.
2. ¿Qué es lo que tiene que hacer Karel en el caso base? Voltear al norte.
3. ¿Cuál es el caso general y que hay que hacer en el? Karel no está sobre el beeper. Lo que hay que hacer es dar un paso al frente y luego llamar a la función recursiva, para después dar otro paso al frente. Este segundo paso será dado en todas las copias excepto en la que sea el caso base, ocasionando con esto que Karel de tantos pasos como los que tuvo que dar para llegar al caso base pero después de haber encontrado este, lo cual significa que Karel ya está viendo al norte.
4. ¿La reducción nos lleva al caso base? Si, si hay un beeper enfrente de Karel.

Moviendo beeper

Ahora imaginemos que en lugar de ir al norte del beeper, quiere llevárselo al doble de la distancia entre el inicio y el beeper. ¿Cuál es el caso base? Por supuesto es el mismo, es cuando Karel está sobre el beeper.

1. ¿Qué hay que hacer en el caso base? Recoger el beeper.
2. ¿Qué hay que hacer en el caso general? Dar un paso, buscar beeper y luego caminar dos pasos más.
3. ¿La reducción nos lleva al caso base? Si, si hay un beeper enfrente de Karel.

La función quedaria así:

```
define doble_distancia_beeper()
        {
        if(nextToABeeper)
                pickbeeper();
        else
                {
                move();
                doble_distancia_beeper();
                move();
                move();
                }
        }
```

Contar beepers

Clonando

El Dr. Karelov es muy estudioso pero siempre requiere hacer copias de los montones de beepers que investiga. Hacer copias significa colocar la misma cantidad de beepers, que hay en una esquina, en otra esquina.

Problema

Karel como buen robot que es, debe hacer una copia del montón de beepers sobre el que esta. La copia la debe dejar en la esquina que tiene enfrente.

Consideraciones

- Dos y dos son cuatro, cuatro y dos son seis.
- Karel lleva los suficientes beepers para hacer la clonación.
- No importa la posición final ni orientación de Karel.
- El montón clonado debe ser la esquina que tiene enfrente Karel cuando inicia.

Esta es una de las aplicaciones más usadas en Karel para recursión

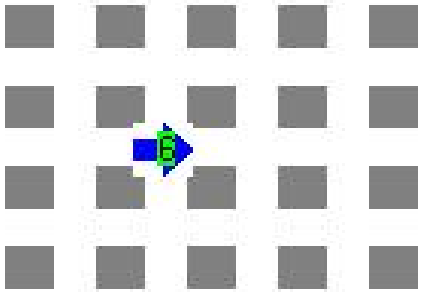

Figura: Esquina con 6 beepers

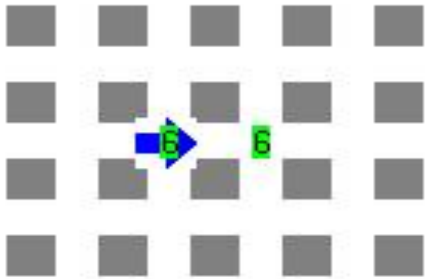

Figura: Estado final

1. ¿Cuál es el caso base? No hay ningún beeper.
2. ¿Qué hay que hacer en el caso base? Moverse a la siguiente esquina.
3. ¿Cuál es el caso general? Hay beepers en la esquina.
4. ¿Qué hay que hacer en el caso base? Para reducir, hay que recoger un beeper. Luego llamar a la recursión, y después poner un beeper en cada una de las esquinas.

Código

```
define pon_beeper_y_regresa()
        {
        putbeeper();
        halfturn();
        move();
        }
define copia_cantidad()
        {
        if(nextToABeeper)
                {
                pickbeeper();
                copia_cantidad();
                pon_beeper_y_regresa();
                pon_beeper_y_regresa();
                }
        else
                move();
        }
```


Edgar Alfredo Duéñez Guzmán & Edgar Said Hernández Sánchez & Marte Alejandro Ramírez Ortegón *Los Dilemas de Karel*. CIMAT, Mayo 2006.