Recursividad

Introducción a la Recursividad

Valentina Bonilla Bedoya
Risaralda, Universidad Tecnológica de Pereira
valentina.bonilla@utp.edu.co

***Resumen*— la recursividad significa el hecho de que un sistema, este compuesto a su vez de objetos que también son sistemas. En general que un sistema sea subsistema de otro más grande. Representa la jerarquización de todos los sistemas existentes es el concepto unificador de la realidad y de los objetos. El concepto de recursividad se aplica a sistemas dentro de sistemas mayores. La recursividad es una técnica de programación muy potente que puede ser utilizada en lugar de la interacción. Permite diseñar algoritmos recursivos que dan soluciones elegantes y simples, y generalmente bien estructuradas y modulares, a problemas de gran complejidad.**

Palabras claves: Recursividad, Programación, Sistemas.

***Abstract*— Recursion means the fact that a system, this in turn composed of objects that are also systems. In general, one system is subsystem of another larger one. Represents the hierarchy of all existing systems is the unifying concept of reality and objects. The concept of recursion applies to systems within larger systems. Recursion is a very powerful programming technique that can be used instead of interaction. It allows to design recursive algorithms that give elegant and simple solutions, and generally well-structured and modular, to very complex problems.**

Key words: Recursion, Programming, Systems.

I. INTRODUCCIÓN

Es un concepto muy abstracto y complejo que tiene que ver tanto con la lógica como también con la matemática y otras ciencias. Podemos definir a la recursividad como un método de definir un proceso a través del uso de premisas que no dan más información que el método en sí mismo o que utilizan los mismos términos que ya aparecen en su nombre, por ejemplo cuando se dice que la definición de algo es ese algo mismo.

Ventajas y desventajas de la Recursividad:

Ventajas:

- No es necesario definir la secuencia de pasos exacta para resolver el problema.
- Soluciones simples, claras.
- Soluciones elegantes.
- Soluciones a problemas complejos.

Desventajas:

- Podría ser menos eficiente.
- Sobrecarga asociada con las llamadas a sub-algoritmos
- Una simple llamada puede generar un gran número de llamadas Recursivas. (Fact(n) genera n llamadas recursivas)
- El valor de la recursividad reside en el hecho de que se puede usar para resolver problemas sin fácil solución iterativa.
- La ineficiencia inherente de algunos algoritmos recursivos.

La recursividad tiene como característica principal la sensación de infinito, de algo que es continuo y que por tanto no puede ser delimitado en el espacio o el tiempo porque se sigue replicando y multiplicando de manera lógica y matemática. Así, es común encontrar casos de recursividad por ejemplo en imágenes de espejos que hacen que la imagen sea replicada al infinito, una dentro de otra hasta que deja de verse pero no por eso deja de existir.
Producto de dos números: Un ejemplo de solución a un problema de forma recursiva sería calcular el producto de dos números de forma recurrente sin emplear el operador de multiplicación. Tomaremos como ejemplo la multiplicación de 3 y 4. Como caso base se conoce que el resultado de multiplicar cualquier número por 1 será el mismo número.

3 * 4 se puede descomponer en 3 + 3 * 3 Del mismo modo 3 * 3 se puede descomponer en 3 + 3 * 2 Y así sucesivamente La figura muestra la secuencia de pasos completa que da solución al problema.

Fecha de Recepción: 04/06/2019
Fecha de Aceptación:

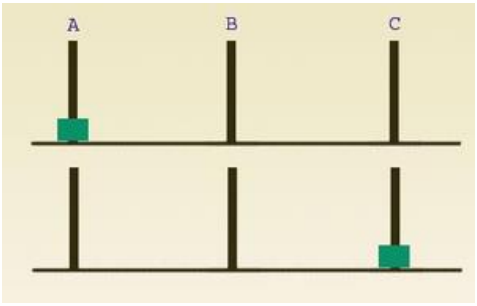
```
3 * 4 = 3 + 3 * 3                3 * 3 = 3 + 3 * 2
      = 3 + 3 + 3 * 2            3 * 2 = 3 + 3 * 1
      = 3 + 3 + 3 + 3 * 1        3 * 1 = 3
      = 3 + 3 + 3 + 3
      = 12
```

Torres de Hanoi

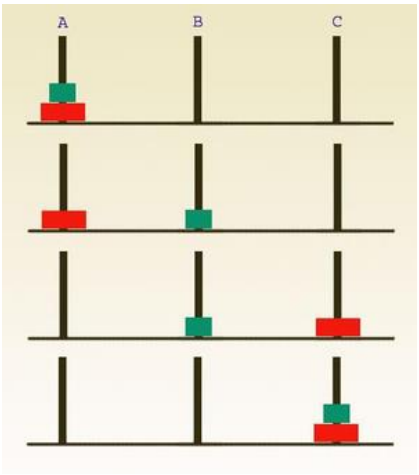
Otro ejemplo es el problema de las Torres de Hanoi. El problema es el siguiente: Se tiene tres postes: A, B y C. En el poste A se coloca una cantidad determinada de discos de diámetro diferente, de modo que un disco de diámetro menor siempre queda sobre uno de diámetro mayor. El objetivo es mover los discos al poste C usando el poste B como auxiliar. Sólo puede moverse el disco superior de cualquier poste a otro poste, y un disco mayor no puede estar en ningún momento sobre uno menor.

Como se puede observar este problema no está planteado en términos recursivos, sin embargo se puede buscar una forma recursiva de dar solución al mismo.

La solución al problema de mover los n discos puede existir a partir de encontrar la solución al movimiento de n – 1 discos, pues al restar sucesivamente n – 1 llegaremos al caso base, mover un disco, cuya solución sería realizar el movimiento del disco de la torre A a la C, como se observa en la figura.



Si se deseara mover dos discos desde la posición inicial a la final los movimientos serían los que se muestran en la imagen.



Conociendo la solución para el movimiento de dos discos, es posible buscar la solución para tres. De la misma forma en que se realizó el movimiento de dos discos desde A hasta C utilizando B como auxiliar, podemos desplazar estos dos discos de C hacia B empleando como auxiliar a A.

En este punto tendríamos los dos discos de menor tamaño en B y el mayor en A, el siguiente paso sería mover un disco de A a C como se hizo en el caso base y a continuación mover los discos restantes de B a C usando la torre A como auxiliar.

Siguiendo estas reglas, se puede hallar la solución al problema de n discos, conociendo la solución para n – 1.

En términos generales, la solución sería la siguiente:

- Si n es 1 mover el disco de A a C y terminar.

- Si n es mayor que 1 entonces:

- Mover el disco superior desde A hasta B n – 1 veces utilizando C como auxiliar.

- Mover el disco restante desde A hasta C.

- Mover los n – 1 discos desde B hasta C empleando A como auxiliar.

Como se ha podido comprobar, la recursividad es una técnica que puede ayudar a encontrar soluciones a problemas cuya solución de manera iterativa resulta bastante engorrosa. El estudio de la misma resulta de gran ayuda para los programadores a la hora de dar respuesta a diversos problemas. [1].

CONCLUSIONES

La recursividad, también llamada recursión o recurrencia, es la forma en la cual se especifica un proceso basado en su propia definición. O sea, si se tiene un problema de tamaño N, este puede ser dividido en instancias más pequeñas que N del mismo problema y conociendo la solución de las instancias más simples, se puede aplicar inducción a partir de estas asumiendo que quedan resueltas.

La recursividad consiste en funciones que se llaman a sí mismas, evitando el uso de bucles y otros iteradores.

No todas las funciones pueden llamarse a sí mismas, sino que deben estar diseñadas especialmente para que sean recursivas, de otro modo podrían conducir a bucles infinitos, o a que el programa termine inadecuadamente. Tampoco todos los lenguajes de programación permiten usar recursividad. C++ permite la recursividad.

Cada vez que se llama a una función, se crea un juego de variables locales, de este modo, si la función hace una

llamada a sí misma, se guardan sus variables y parámetros, usando la pila, y la nueva instancia de la función trabajará con su propia copia de las variables locales. Cuando esta segunda instancia de la función retorna, recupera las variables y los parámetros de la pila y continúa la ejecución en el punto en que había sido llamada.

REFERENCIAS

[1]. C. Agüero, " EcuRed, Ingreniero en Ciencias Informáticas, *Recursividad*.

[2]. E. J. Ramirez.

http://hector11300819.mex.tl/1921868_RECURSIVIDAD.html