# Recursividad en lenguajes

## Recursivity in languages

Autor : Angie Valery Jaramillo Guerrero
*Facultad de ingenierías, ingeniería de sistemas, Universidad tecnológica de Pereira, Pereira, Colombia*
Correo-e: angievalery.jaramillo@utp.edu.co

***Resumen*— El objetivo principal de este trabajo es presentar el concepto de recursión, su historia y evolución dentro de la Teoría de la Computabilidad y la Lógica Matemática.**

***Palabras clave*— recursividad, función.**

***Abstract*— the main goal of this paper is to introduce the concept of recursion, its history, and its evolution within Computability Theory and Mathematical Logic.**

***Key Word* —computabilidad, recursividad, Sistema, retroalimentacion,**

## I. INTRODUCCIÓN

- En matemáticas se da el nombre de *recursión* a la técnica consistente en definir una función en términos de sí misma. Puesto que en C una función puede llamar a otras funciones, se permite que una función también pueda llamarse a sí misma.

- Toda función definida recursivamente debe contener al menos una definición explícita para alguno de sus argumentos. De no ser así la función puede caer en un bucle infinito.

## II. CONTENIDO

La recursividad hace referencia sobre aquellos elementos de un sistema, que puedan estar en constante intercambio de información, y que para poder llegar a cumplir objetivos existe la retroalimentación dentro de la organización de los elementos de los sistemas, dentro de un departamento puede haber una serie de vínculos para que dicha información fluya de un lugar a otro generando recursividad entre los elementos de forma conjunta.

La recursividad es una técnica muy utilizada en programación informática. Se suele utilizar para resolver problemas cuya solución se puede hallar resolviendo el mismo problema, pero para un caso de tamaño menor.

Cuando en informática se escribe un programa con un algoritmo recursivo, en las propias sentencias del algoritmo hay una llamada a sí mismo, es decir una de las sentencias llama al algoritmo recursivo en el que está insertada, aunque para solucionar un caso más sencillo.

Los métodos recursivos se pueden usar en cualquier situación en la que la solución pueda ser expresada como una sucesión de pasos o transformaciones gobernadas por un conjunto de reglas claramente definidas.

### A. historia.

No hay ninguna duda de que el concepto de recursión surgió en el seno de la Teoría de la Computabilidad (una rama especializada de la Lógica Matemática). Su origen se remonta al siglo XIX y fue ya utilizado por autores como Dedekind o Peano (Soare 1996, 1999, 2009).

Sin embargo, parece que no siempre ha sido empleado de la misma manera. Así, dentro de tales disciplinas significó tanto *definición por recursión* -o definición recursiva- como *computabilidad* (y lo mismo vale para 'recursivo' y 'computable'; véase, por ejemplo, Soare 2009). Tal como nos indica Kleene (1952), una definición recursiva es un método que sirve para definir una función (o predicado).[1] Así, una función es recursiva, esto es, está definida, en un sentido técnico, por recursión cuando para definir un argumento *y* hacemos uso de sus propios valores previamente computados para argumentos menores que y; pudiendo emplearse también funciones previamente definidas (Soare 1996, 1999, 2009; véase también Gödel 1931; Kleene 1952, 2002; Cutland 1980). Éste es el sentido que constituye su significado original.

Fecha de Recepción: (Letra Times New Roman de 8 puntos)
Fecha de Aceptación: Dejar en blanco

Epstein y Carnielli (1989) indican que, en su forma más simple, la definición recursiva de una función *f* sería como sigue (siendo *g* una función previamente definida): $f(0) = m$; $f(n+1) = gf(n))$. Este sistema de ecuaciones recursivas puede instanciarse fácilmente y de una manera perspicua en la función suma (cf. Boolos y Jeffrey 1974).

$\alpha+0 = \alpha$ /a+1 = a' [caso base],
$\alpha+(b+1) = (\alpha+b)+1$ [paso recursivo]

B. definición

Se llama recursividad a un proceso mediante el que una función se llama a sí misma de forma repetida, hasta que se satisface alguna determinada condición. El proceso se utiliza para computaciones repetidas en las que cada acción se determina mediante un resultado anterior. Se pueden escribir de esta forma muchos problemas iterativos.

C. Condiciones para la recursividad.

Se deben satisfacer dos condiciones para que se pueda resolver un problema recursivamente:

Primera: El problema se debe escribir en forma recursiva.

Segunda: La sentencia del problema debe incluir una condición de fin.

.

**Se dará prioridad a los artículos tipo a, b, c ya que son los de mayor impacto en el tema de recursividad.**

Primero debemos decir que la recursividad no es una estructura de datos, sino que es una técnica de programación que nos permite que un bloque de instrucciones se ejecute n veces. Remplaza en ocasiones a estructuras repetitivas como se muestra en la fig 1.

La recursividad es un concepto difícil de entender en principio, pero luego de analizar diferentes problemas aparecen puntos comunes.
En Java como en otros lenguajes como c, los métodos pueden llamarse a sí mismos. Si dentro de un método existe la llamada a sí mismo decimos que el método es recursivo.
Cuando un método se llama a sí mismo, se asigna espacio en la pila para las nuevas variables locales y parámetros.

Al volver de una llamada recursiva, se recuperan de la pila las variables locales y los parámetros antiguos y la ejecución se reanuda en el punto de la llamada al método.

FIGURA 1
EJEMPLO DE RECURSIVIDAD EN EL LENGUAJE C

**Ejemplo 1: Posteriormente se muestra nuevamente el problema de la resolución del factorial, pero pidiendo el valor que deseamos hallar por teclado.**

```
#include <stdio.h>

main()
{
int n;
long int factorial (int n);

printf("Introducir la cantidad entera a la que le queremos hallar el factorial: ");
scanf("%d", &n);

printf("%d! = %d\n", n, factorial(n));
}

long int factorial (int n)  /* Calcular el factorial */
{
      if (n <= 1)
            return(1);
      else
            return(n * factorial (n-1));
}
```

Tabla 1. Ejemplo de tabla en artículo.

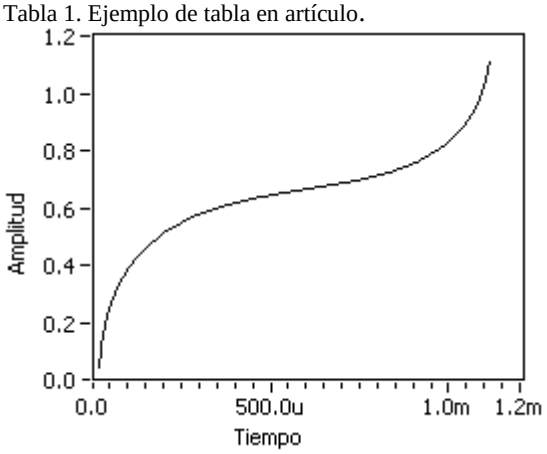


Figura 1. Ejemplo de figura en un artículo.

Si en el artículo se utilizan ecuaciones, estas deberán tener numeración consecutiva, así no las cite o use en el texto. Se debe definir su procedencia.

## III. CONCLUSIONES

Se puede decir que la recursividad es una técnica de programación bastante útil y muy interesante de estudiar. A través de los ejemplos que el individuo pueda revisar, aprenderá con más rapidez y sencillez lo que es programar recursivamente e incluir esta técnica cuando se le presente un problema como los que fueron mencionados anteriormente. La asignación de memoria, sea estática o dinámica, en realidad se tendrá que aplicar en cualquier programa al momento de su codificación; tomando en cuenta que cada programador tiene su estilo de programar.

RECOMENDACIONES

Utilizar recursividad cuando:
Clarifique el algoritmo y el programa que soluciona u problema.
No halla fuertes restricciones de memoria o tiempo de ejecución
Como consecuencia de estos dos puntos, y dado que la trasformación de un algoritmo recursivo a su versión iterativa es un proceso totalmente mecánico

## REFERENCIAS

[1]. https://books.google.com.co/books?id=25jqDQAAQBAJ&pg=PA154&lpg=PA154&dq=recomendaciones+en+la+recursividad&source=bl&ots=f1udwemdzg&sig=ACfU3U2s9psvs4LJ5STLmkijEL9PXWyw6A&hl=es&sa=X&ved=2ahUKEwi0vIuMhN3iAhXhwVkKHWddAEcQ6AEwDXoECAkQAQ#v=onepage&q=recomendaciones%20en%20la%20recursividad&f=false
[2]. http://decsai.ugr.es/~jfv/ed1/c/cdrom/cap6/cap66.htm

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos