Recursividad

Recursion.

Evelyn Tabares Valencia
Ingeniería de sistemas y computación, Universidad Tecnológica de Pereira, Pereira, Colombia
evelyn.tabares@utp.edu.co

***Resumen*— La recursividad es una técnica de programación que se utiliza para realizar una llamada a una función desde ella misma, de allí su nombre. El ejemplo más utilizado por su fácil comprensión es el cálculo de números factoriales. Pero increíblemente no solo está presente en la programación, personalidades como: Kurt Gödel, el más grande lógico del siglo XX, Maurits Cornelis Escher, un artista gráfico holandés que era capaz de plasmar en una hoja de papel enigmáticas paradojas y Johann Sebastián Bach el gran compositor barroco que, entre muchas otras obras, creó las embriagantes fugas para órgano, aplicaron esta llamada recursividad a disciplinas tan diferentes entre sí como lo son las matemáticas, el dibujo y la música.**

***Palabras clave*— *Recursividad*, programación, Kurt Gödel, Maurits Cornelis Escher, Johann Sebastián Bach.**

***Abstract*— Recursion is a programming technique used to make a call to a function from itself, hence its name. The most used example for its easy comprehension is the calculation of factorial numbers. But incredibly not only is present in the programming, personalities such as: Kurt Gödel, the greatest logician of the twentieth century, Maurits Cornelis Escher, a Dutch graphic artist who was able to capture on a sheet of paper enigmatic paradoxes and Johann Sebastian Bach the great Baroque composer who, among many other works, created intoxicating organ leaks, applied this so-called recursion to disciplines as different as mathematics, drawing and music.**

***Key Word* —: Recursion, programming, Kurt Gödel, Maurits Cornelis Escher, Johann Sebastián Bach.**

I. INTRODUCCIÓN

Un concepto que siempre les cuesta bastante a los programadores que están empezando es el de recursión o recursividad (se puede decir de las dos maneras). [1]

Aunque es un concepto que puede llegar a ser muy complejo, en esencia es muy sencillo: [1]

La recursividad consiste en funciones que se llaman a sí mismas, evitando el uso de bucles y otros iteradores. [1]

Un ejemplo fácil de ver y que se usa a menudo es el cálculo de la factorial de un número entero. El factorial de un número se define como ese número multiplicado por el anterior, éste por el anterior, y así sucesivamente hasta llegar a 1. Así, por ejemplo, el factorial del número 5 sería: 5x4x3x2x1 = 120. [1]

Tomando el factorial como base para un ejemplo, ¿cómo podemos crear una función que calcule el factorial de un número? Bueno, existen multitud de formas. La más obvia quizá sería simplemente usar un bucle determinado para hacerlo, algo así en JavaScript: [1]

```
function factorial(n){
  var res = 1;
  for(var i=n; i>=1; i--){
    res = res * i;
    }
  return res;
 }
```

[1]

Sin embargo, hay otra forma de hacerlo sin necesidad de usar ninguna estructura de bucle que es mediante recursividad. Esta versión de la función hace exactamente lo mismo, pero es más corta, más simple y más elegante: [1]

```
function factorial(n) {
 if (n<=1) return 1;
 return n* factorial(n-1);
}
```

[1]

Aquí lo que se hace es que la función se llama a sí misma (eso es recursividad), y deja de llamarse cuando se cumple la condición de parada: en este caso que el argumento sea menor o igual que 1 (que es lo que hay en el condicional). [1]

Es decir, cuando llamamos a la primera función, ésta se llama a sí misma, pero pasándole un número menos y así sucesivamente hasta llegar a la última (la que recibe un 1 y por lo tanto deja de hacer más llamadas). En el momento en el que alguna de ellas empieza a devolver valores "hacia atrás", regresa la llamada a cada una de ellas, los valores devueltos se van multiplicando por el parámetro original en cada una de ellas, hasta llegar arriba del todo en el que la primera llamada devuelve el valor buscado. [1]

Fecha de Recepción:
Fecha de Aceptación: Dejar en blanco

El paradigma de la publicidad bien podría ser una imagen que se publicita a sí misma. Eso mismo debieron de pensar los directores publicitarios de la marca de productos alimenticios Droste en los años 60. [2]

Posteriormente conocido como efecto Droste es uno de los ejemplos más sencillos de recursividad aplicado a una imagen. Digo más sencillos porque las imágenes recursivas ofrecen un gran número de posibilidades creativas, tan variadas como originales. Aunque no se trataba de una idea original pues las imágenes recursivas ya se conocían seiscientos años antes lo que si es cierto es que la publicidad en aquel siglo no tenía el mismo impacto en la población que la del siglo XX. [2]

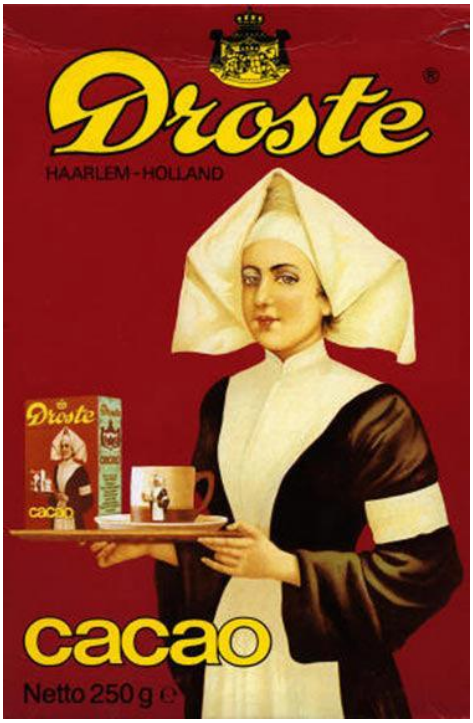
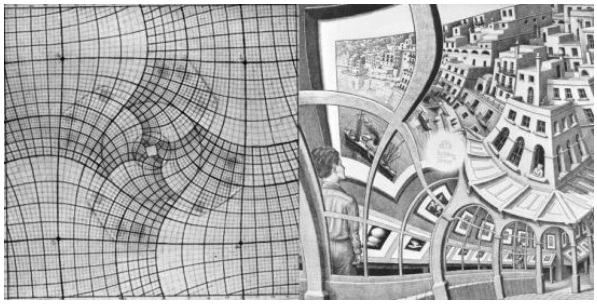


Imagen publicitaria de la marca de productos alimenticios Droste en los años 60.

Imagen tomada de:
https://mimoriarty.wordpress.com/2011/04/11/imagenes-recursivas-efectos-droste-y-escher/

¿Qué es la recursividad? Según la definición de Wikipedia recursividad es la forma en la cual se especifica un proceso basado en su propia definición. Esto es un poco difícil de adaptar al mundo de la imagen o el diseño gráfico, por ese motivo es mejor recurrir a un ejemplo más visual. Imaginad una cuadrícula bidimensional. Una cuadrícula es un ejemplo perfecto de imagen recursiva sin tener que recurrir a las matemáticas. Cada cuadrado está formado por cuadrados de menor tamaño que a su vez están formados por otros cuadrados más pequeños y así una y otra vez. No importa realmente que sean cuadrados, triángulos equiláteros, rectángulos áureos, curvas de Koch, … No importa tanto la forma como que el contenido de la imagen se utiliza a sí misma hasta el infinito. [2]

Uno de los tipos de imágenes recursivas que en ocasiones se confunden con el efecto Droste son las originadas a partir de la técnica efecto Escher en honor al pintor holandés Maurits Cornellis Escher (1898-1972) cuyas litografías exploraron diferentes técnicas especialmente enfocas a jugar con el espacio. De esta forma en sus obras se destacan las teselas, retículas, imágenes recursivas, estudios de las superficies y la partición regular del plano. Es posible obsevar como trabajaba M. C. Escher en la imagen inferior. A la izquierda la cuadrícula base inicial de la que parte la obra, PrintGallery, a la derecha. [2]



Obras de M. C. Escher.

Imagen tomada de:
https://mimoriarty.wordpress.com/2011/04/11/imagenes-recursivas-efectos-droste-y-escher/

II. CONTENIDO

Programar implica abrir tu mente a una nueva forma de pensar. Esto incluye asimilar conceptos que, en otro contexto, pueden parecer abstractos. Uno de los conceptos que más dolores de cabeza suele dar a los estudiantes de programación es el de la Recursividad o Recursión. [3]

A pesar de lo anterior, el uso de recursión no deja de ser una técnica más al momento de escribir código, y puede volverse tremendamente útil para mejorar tu forma de escribir código. A continuación, te muestro que es la recursión y como utilizarla en tus programas sin perderte en el camino. [3]

En programación, la recursión, también conocida como recursividad o recurrencia, es un concepto abstracto que hace referencia a que una función o método se invoque a sí mismo dentro del bloque de código que lo define. Para que quede más claro, puedes ver el siguiente fragmento de código que hace uso de recursión: [3]

```
def contador(n):
    if n>0:
        print (n)
        return contador(n-1)
```

En el código anterior, creamos una función que imprimirá una cantidad n de números de forma regresiva, sin necesidad de utilizar ciclos while o for. [3]

Lo que ocurre al invocar la función con un argumento de, por ejemplo, 3, es que se comprobará que 3 es mayor a 0, imprimirá el número y después volverá a iniciar la función con el argumento 3-1, como 2 es mayor a 0 volverá a imprimir el número e invocará la función con 2-1 como argumento, se repite el procedimiento. El caso base, o último llamado de la función recursiva será 1-1, como 0 no es mayor a 0, la función terminará. [3]

Lo anterior queda explicado de forma visual en el siguiente diagrama:

Diagrama de recursión

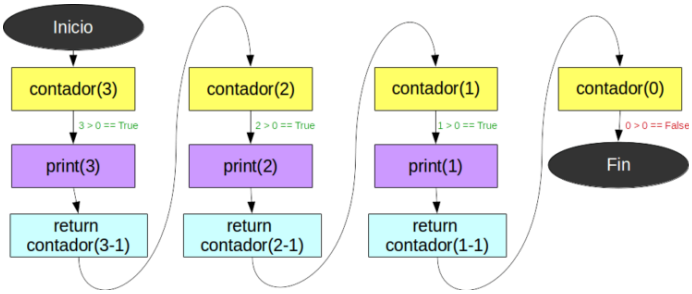


Imagen tomada de:
https://www.technodyan.com/recursividad-aplicar-recursion-programas/

Un punto clave del uso de la recursión es el de contar con un caso base que marque el final de las llamadas a la función, de otra forma, terminarás con un ciclo infinito de llamadas recursivas que forzarán el cierre del programa. En la función anterior, se llega al final de la recursividad cuando n es igual o menor a 0. [3]

Ahora que sabemos cómo avanza la recursión, hay que mencionar otra propiedad: El «retroceso» de los valores una vez se llega al caso base, o al último llamado de recursión. Por ejemplo, consideremos el siguiente programa para obtener el factorial de un número: [3]

```
def factorial(n):
        if n>1:
                return n*factorial(n-1)
        return 1
```

En el código anterior definimos una función factorial que recibe un número n. Si n es mayor a 0, se regresa n multiplicado por la misma función factorial con n-1 como argumento. Si n no es mayor a 0, la función regresa el entero 1. Este código se aprovecha de los valores que retorna cada llamado a la recursión para obtener el resultado deseado, como se ve a continuación. [3]

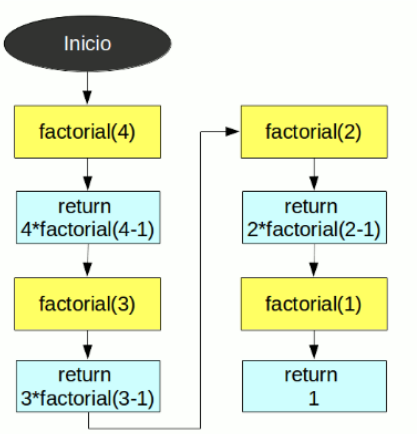


Imagen tomada de:
https://www.technodyan.com/recursividad-aplicar-recursion-programas/

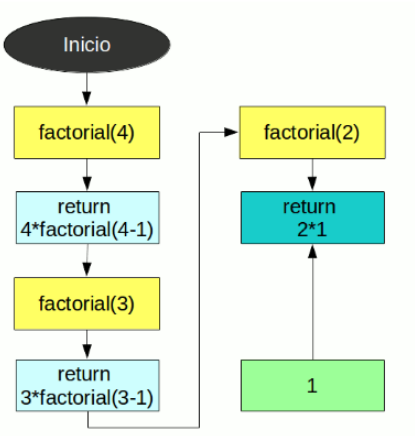


Imagen tomada de:
https://www.technodyan.com/recursividad-aplicar-recursion-programas/

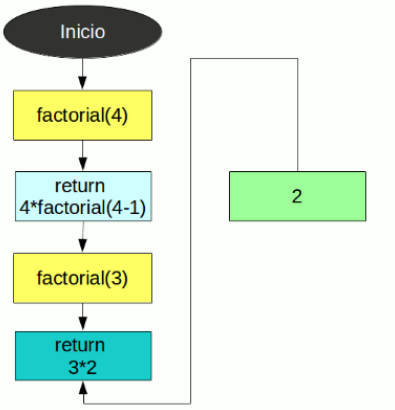


Imagen tomada de:
https://www.technodyan.com/recursividad-aplicar-recursion-programas/

III. ¿Cuándo usar recursión en nuestros programas?

A pesar de que puede llegar a ser una técnica muy elegante al momento de programar, la recursión no es la mejor opción en términos de rendimiento, pues implica ejecutar la misma función por cada iteración del problema, teniéndose que evaluar de nuevo todos los pasos de la función. [3]

Al momento de ejecutar la función contador que ya mencionamos, se puede representar su complejidad así:

La función contador hace un sólo llamado a la recursividad a la vez, por lo tanto, es una función de complejidad lineal.

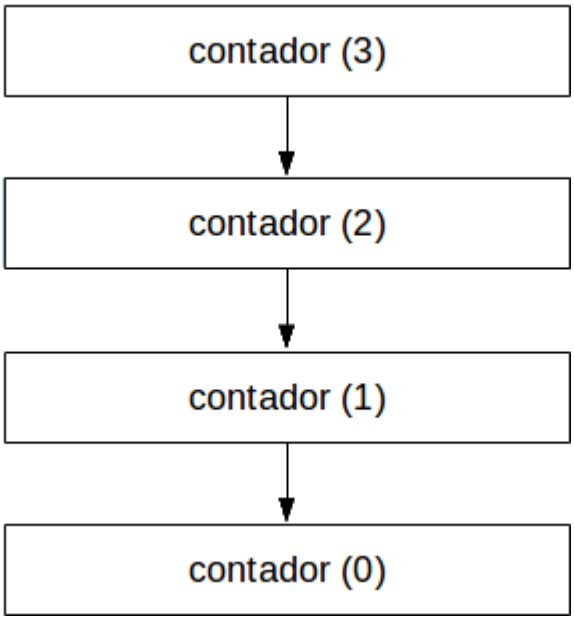


Imagen tomada de:
https://www.technodyan.com/recursividad-aplicar-recursion-programas/

Cuando comprendemos como utilizar la recursión en nuestros programas, es común que se compare con el uso de bucles; a fin de cuentas, la utilidad de ambos es resolver problemas repetitivos con variaciones de datos. Sin embargo, no son exclusivos entre sí y cada uno tiene su momento para ser implementado. [3]

Como ya mencioné anteriormente, la recursión gana mucho en legibilidad, pero pierde igualmente en rendimiento, por lo que sólo es recomendable para resolver problemas que no requieran de muchos cálculos por parte del procesador, mientras que los bucles o ciclos administran de forma mucho más eficiente los recursos de la computadora. [3]

Otra cosa importante a tener en cuenta es que, cada vez que se hace una llamada a una función desde otra función (aunque sea a sí misma), se crea una nueva entrada en la pila de llamadas del intérprete. Ésta tiene un espacio limitado por lo que puede llegar un punto en el que si se hacen demasiadas se sature y se produzca un error. A este error se le denomina "Desbordamiento de pila" o "Stack Overflow". Ahora ya sabes de donde viene el nombre del famoso sitio para dudas de programadores sin el que la programación moderna no sería posible ;-) [1]

Además, hay que tener mucho cuidado con la condición de parada. Ésta se refiere a la condición que se debe comprobar para determinar que ya no se harán más llamadas a la función. Es en ese momento en el que empiezan a devolverse los valores hacia "arriba", retornando a la llamada original. [1]

Si no tienes la condición de parada controlada pueden pasar varias cosas (todas malas), como, por ejemplo:

Que se sature la pila y se produzca un desbordamiento.
Que ocupes cada vez más memoria.
Que se produzcan desbordamientos de variables al ir acumulando resultados.
En JavaScript, que el navegador -al cabo de unos segundos- avise al usuario de que hay un script descontrolado y que lo detenga. [1]

IV.Gödel, Escher, Bach: Un Eterno y Grácil Bucle

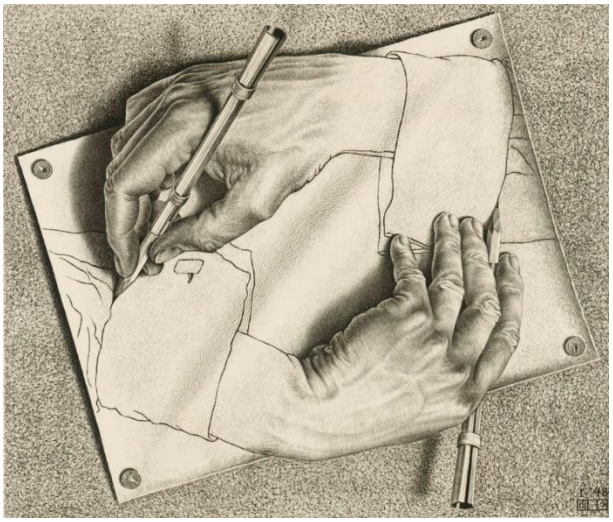
¿Qué cree usted que tienen en común personalidades como Kurt Gödel, Maurits Cornelis Escher y Johann Sebastián Bach?.

A primera vista es difícil imaginar algún ente integrador. Se trata sólo de tres destacados personajes, que brillaron en disciplinas tan disímiles como las matemáticas, el dibujo y la música. [4]

Sin embargo, sí existe un elemento común y ese no es otro que la auto referencia (o recursividad), aquel extraño y perturbador fenómeno que permite que un evento se contenga a sí mismo, hasta el infinito. Como dos espejos, puestos uno frente al otro, cuyas imágenes se enfrentan y proyectan hasta la eternidad. Aprovechando ese puente de comunión, Douglas R. Hofstadter, un destacado académico y escritor, nos presenta el libro “Gödel, Escher, Bach; Un eterno y grácil bucle” (Tusquets editores S.A., serie Metatemas) [4]

La obra gira entorno a kurt Gödel, quien en el siglo pasado remeció al mundo intelectual, al enunciar y demostrar un inquietante teorema conocido como de la incompletitud, el cual tuvo un profundo impacto en las matemáticas y la filosofía, al poner en evidencia la imposibilidad de acceder a una verdad numérica absoluta y completa. Gödel utilizó el razonamiento matemático para inspeccionar el propio razonamiento matemático (a este auto referencia el autor la denomina bucle), y a través de ese camino encontró sentido a situaciones paradójicas que por milenios habían puesto en aprietos al razonamiento humano. [4]

Junto a Gödel, aparece la figura de Maurits Cornelis Escher, aquel genial artista que llevó al papel situaciones imposibles, utilizando para ello ciclos autocontenidos e infinitos. Entre sus muchas obras podemos recordar aquella que muestra dos manos entrelazadas, en donde cada una dibuja a la otra; o ese grabado que presenta una cascada que se alimenta eternamente de la misma agua que precipita. [4]



El tercer protagonista es Johann Sebastián Bach, el gigante de la música, que en muchas de sus composiciones utilizó estructuras recursivas, las que permitieron dar vida a maravillosas piezas musicales, como sus famosas fugas y cánones. [4]

El alemán compuso sus piezas musicales pensando en Fuga y Canon, que son elementos musicales que usan la repetición de melodías en diferente escala o velocidad con o sin contrapunto, pero que no precisamente son la misma melodía. El uso del canon y la fuga es similar a la naturaleza fractal pero este término no se conocía en 1685 sino hasta 1975 cuando fue acuñado por el matemático Benoît Mandelbrot. [5]

En piezas autosemejantes de Bach los mismos motivos son repetidos una y otra vez con distintas variaciones dentro de una región mayor de la pieza. Así, por ejemplo, distintas voces se repiten al doble de velocidad la melodía de la voz principal. [5]

A partir de la música de Bach se descubrió que repetir un proceso con el objetivo de alcanzar una meta deseada, en conjunto con las matemáticas, resultan elementos que se aplican como una extensión de la composición musical. Así, los fractales proveen inesperadas conexiones entre las artes musicales y muchos procesos naturales, ya que mezclan cualidades determinantes y probables para producir naturalmente un agradable balance entre lo que se espera y la novedad, pues al escuchar la repetición de cada una de las notas por separado, la pieza resultaría un caos, al igual que una imagen que no entra en armonía con todos los elementos. [5]

Bach fue uno los compositores más representativos de la música barroca, creó La tocata y fuga en re menor, obra que se caracteriza por desprenderse de una armadura de clave, es decir: un conjunto de alteraciones en las notas sensibles, propias de la época barroca. Esta pieza musical de órgano se ha adaptado para ser tocada en piano y violín, además de ser utilizada en diversos filmes, música rock, obras teatrales, videojuegos, anime, etc. y que cuenta con una composición de Fuga y Canon. [5]

Las piezas musicales de Sebastian Bach resultan un juego mental en el que cada una de las notas integran el resultado final de la obra. [5]

V. CONCLUSIONES

La recursividad es un concepto muy abstracto y voluble, tanto que como hemos podido evidenciar en el presente documento es aplicable a disciplinas tan diferentes de la programación como lo son las matemáticas, el arte y la música, en las cuales la recursividad toma forma de manera visual, auditiva e incluso matemática, demostrando su gran alcance y aplicabilidad.

La recursividad no es más que una función que se llama a si misma hasta que la condición de esta se vea negada para darse por terminada, si esto no pasara el proceso se repetiría de manera infinita o hasta que se dé un “desbordamiento de pila” por eso es importante definir de manera adecuada la condición, para que este desbordamiento de pila o algún error no aparezca. Si bien la recursividad como su nombre lo indica hace que un proceso que realizaríamos a través de otros métodos sea más corto, sin embargo, crea la desventaja de que ocupa más memoria por un proceso legible y lógico.

Depende de cada persona elegir si usar o no este recurso, sin embargo, es sin duda practico, por su corta solución a problemas que con métodos como bucles sería demasiado largo y tedioso realizar.

WEBGRAFIA

[1] J. M. Alarcón, «campusmvp,» 27 enero 2016. [En línea]. Available: https://www.campusmvp.es/recursos/post/Que-es-la-recursividad-o-recursion-Un-ejemplo-con-JavaScript.aspx. [Último acceso: 2 junio 2019].

[2] mimoriarty, «mimoriarty.wordpress.com,» [En línea]. Available: https://mimoriarty.wordpress.com/2011/04/11/imagenes-recursivas-efectos-droste-y-escher/. [Último acceso: junio 2 2019].

[3] N. R. Guerra, «technodyan.com,» 15 junio 2017. [En línea]. Available: https://www.technodyan.com/recursividad-aplicar-recursion-programas/. [Último acceso: 2 junio 2019].

[4] H. J. Goldenberg, «Tauzero,» 3 marzo 2009. [En línea]. Available: http://tauzero.org/2009/03/godel-escher-bach-un-eterno-y-gracil-bucle/. [Último acceso: 2 junio 2019].

[5] «culturacolectiva,» 27 agosto 2013. [En línea]. Available: https://culturacolectiva.com/musica/los-fractales-en-la-musica-de-sebastian-bach. [Último acceso: 2 junio 2019].