Recursividad, breve introducción al concepto.

Recursion.

Autor: Nicolas Vega Jaramillo
Ingeniería de Sistemas y Computación., Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: nicolas.vega@utp.edu.co

***Resumen*— Se define como el hecho de que una función se llame a si misma, esto para facilitar las cosas en cuanto a código, una manera fácil de construir una lógica para formar cosas.**

***Palabras clave*— Recursividad, Programa, Función, Lógica.**

***Abstract*— It´s defined as the fact that a function calls itself, this to make the code easier, an easy way to build a logic to form things.**

***Key Word* —Recursion, Program, Function, Logic.**

I. INTRODUCCIÓN

Una función que se llame a su misma, en esas pocas palabras es lo que se define la recursividad, pero estas funciones tienen unas claras características, y es importante cumplirlas, ya que con estas se logra un código organizado.

La recursividad facilita mucho las cosas, a tal nivel que permite ahorrar muchísimas líneas de código si se sabe emplear de una buena manera, también permite entender todo mejor, y esto a la hora de trabajar con cualquier tipo de programa es algo que se agradece.

Al principio puede ser un poco raro, pero todo es cuestión de práctica, con esto se logra formar una buena lógica que permite a los programadores tener una vida mas fácil.

II. CONTENIDO

Una función recursiva debe cumplir ciertas características, las cuales permiten que, a la hora de hacerlas, todo sea mas fácil. Aquí el siguiente ejemplo: (DrRacket, lenguaje R5RS)

```
; Programa que calcula el factorial de un número.

(define (facto n)     ; Definición de la Función
     (if (= n 0)            ; Condición de parada
         1                  ; Condición verdadera
       (* n (recursiva (- n 1))  ; Condición falsa
       )                         ;Fin de la condicional
)                                ;Fin de la Función
```

El primer paso es definir la función, luego poner una condición, la condicional no siempre es la misma, es depende de lo que se quiera lograr, en este caso, como se quiere calcular el factorial de un valor n, lo que se hace es poner la condicional como 0, ya que se le ira restando a n 1 cada vez que pasa por el bucle, ya que cuando n no es 0 se multiplica n por n-1, lo que indica que se esta restando una unidad y a su vez multiplicando por n, después si n sigue siendo diferente de 0 se repite el proceso, formando un bucle hasta que n sea igual a 0.

La recesividad tiene tres niveles, los cuales se definirán a continuación. Cada nivel con un ejemplo,

1) Recesividad nivel I. El nivel más sencillo de la recesividad, ya que solo es cuando una función se llama a sí misma, sin nada más.

Aquí otro ejemplo.

```
; Programa que suma un numero n, x veces.

(define (Suma n x)
  (if (= x 0)
     0
     (+ n (Suma3 n (- x 1)))
      )
  )
```

La anterior función Suma recibe a n como un numero y x como las veces que se desea sumar el número. La instrucción recursiva se basa en sumar n hasta que x sea 0, ahí el bucle se detiene.

2) Recursividad II. Este tipo de recursividad se basa en una función recursiva y en una función normal que se apoya a la recursiva.

```
; Función que recibe un numero entero n y suma sus dígitos pares.
```

Fecha de Recepción: 30 mayo 2019
Fecha de Aceptación:

```
(define (par n)
  (if (= (remainder n 2) 0)
     n
     0
     )
  )

(define (solución n)
  (if (= n 0)
    0
    (+ (par (remainder n 10))
      (solución (quotient n 10)))
    ))
```

Lo que hace la función de apoyo es definir cuando un dato es par, la instrucción “remainder” da el residuo de un numero entro otro, en este caso muestra el residuo de un numero entre 2 si el residuo es 0, devuelve el dato, ya que esto señala si el numero es par, si no lo es devuelve 0, puesto que se trata de sumar números pares. La función recursiva evalúa el ultimo digito con la función par, y lo suma a los dígitos sobrantes de n en la función solución.

3) Recursividad III. Este nivel se define como una solución recursiva que requiere de 2 o mas funciones recursivas y una de apoyo.

; Función que calcula el factorial de la cantidad de primos que tiene un numero entero.

; Primos y contador

```
(define (primos n)
  (if (or (= n 1) (= n 2)(= n 3)
        (= n 5) (= n 7))
     1
     0
     )
  )

(define (ContI n)
  (if (= n 0)
     0
     (+ (primos (remainder n 10))
       (ContI (quotient n 10)))
     )
  )

;Factorial

(define (facto n)
  (if (= n 0)
     1
     (* n (facto (- n 1)))
     )
  )

;Funcion Sol
(define (FactDigPrim n)
  (if (= n 0)
     0
     (facto (ContI n))
     )
  )
```

La función primos define cuando un digito es primo, si lo es devuelve 1, si no lo es devuelve 0, ya que se trata de contar los dígitos primos. La función ContI es un contador el cual recibe un entero y cuenta la cantidad de primos que hay en este, evaluando cada digito, como en el ejemplo de **Recesividad II.** La función final lo que hace es calcular el factorial con la función antes vista, pero esta vez evaluando a n en el ContI.

III. CONCLUSIONES

La recursividad es una de las herramientas más importantes a la hora de programar, ya que con esta se pueden hacer muchísimas aplicaciones, tenerla en cuenta es vital.

RECOMENDACIONES

Agradecimientos a mi profesor Omar Iván Trejos Buriticá, por haberme enseñado este concepto.

.