¿Qué es?

Técnica utilizada en programación que nos permite ejecutar una serie de instrucciones cierto número definido de veces determinado por nosotros.

Facilitará algunas acciones siempre que sepamos en qué momento debemos utilizarla. teniendo en cuenta que no todas las funciones pueden ser o hacerse recursivas debido a eficiencia o a tratamiento de memoria del sistema.

¿Cuándo es?

En programación estructurada, se entiende por recursividad o función recursiva, toda aquella función que durante su ejecución se invoca directa o indirectamente a sí misma.

Para el uso de la piscina, solamente utilizamos la manera directa, aunque a veces tambien se podrá utilizar la manera indirecta.

¿Qué reglas debe cumplir?

1.- Tendrá al menos una condición de corte con una solución controlada a la misma. En caso de no contemplarlo, podemos caer en ejecuciones infinitas o desbordamientos de pila.

2.- Al menos tenemos que tener una llamada recursiva o entrada en el proceso de recursión.

3.- En cada llamada recursiva, habremos realizado las operaciones necesarias para acercarnos a la condición de finalización o salida.

4.- Al llegar a la condición de corte, debemos obtener la solución total al problema.

Normalmente se utiliza para solucionar problemas matemáticos que por otros medios quedan más complicados de entender. P. Ej: el factorial de un número.

1 int rs;

2 if (x == 0)

3 rs = 1;

4 else

5 rs = x \* factorial (x-1);

6 return (rs);

En el ejercicio, la condición de corte o solución viene dada por la orden x == 0 que hará que el código devuelva un 1 y por tanto no vuelva a entrar en recursión.

Si llamamos a la función con el valor del factorial de 3 el resultado será el siguiente, el programa primeramente solucionará el factorial de 3, pero para ello, necesita solucionar el factorial de dos, como todavía no llegamos a la solución trivial, deberemos solucionar a su vez el factorial de 1 y por supuesto, para resolver este necesitamos la solución trivial de nuestro factorial de 0.

Al solucionar y llegar a esa primera solución, recorreremos en orden ascendente nuevamente los resultados, sustituyendo el valor correspondiente hasta obtener el resultado final.

Recursividad indirecta

Una función llamará a otra y la segunda función llamará a la primera nuevamente, necesitamos una condición de salida fuerte y bien estudiada para evitar que se creen procesos infinitos.

Cosas a tener en cuenta

1.- En las funciones recursivas no podrán existir estructuras iterativas, o es iterativo o es repetitivo, en ningún momento puedo colocar un bucle en el interior de algo recursivo

2.- la iteración utiliza estructuras de repetición (for, while, etc). La recursión utilizará en su interior estructuras de selección.

3.- La iteración consigue invocar a la repetición de forma explícita con estructuras repetitivas, la recursión debe conseguir lo mismo mediante llamadas sucesivas.a la función.

4.- el bucle o iteración termina cuando falla la condición de corte, la estructura recursiva, acaba cuando se cumple la condición de salida.

5.- se vuelve infinita cuando no damos correctamente con la condición de salida.

6.- al invocarse continuamente a si misma, puede sobrecargar el tiempo de proceso o la pila de memoria del sistema.

¿Cuándo utilizarla?

En una solucion recursiva, el equipo debe controlar y resolver las invocaciónes y en ese camino mantiene informacion acerca de los resultados de las mismas. puede ser costoso en tiempo de proceso y espacio de memoria.

Por lo tanto, se recomienda el uso de la recursión solo en el caso de que:

la solucion no pueda ser facilmente expresable por otros métodos.

la eficiéncia de la solucion recursiva sea tan buena como la solucion iteractiva.

Lo mande la molinillo