

# Recursividad en programación

---

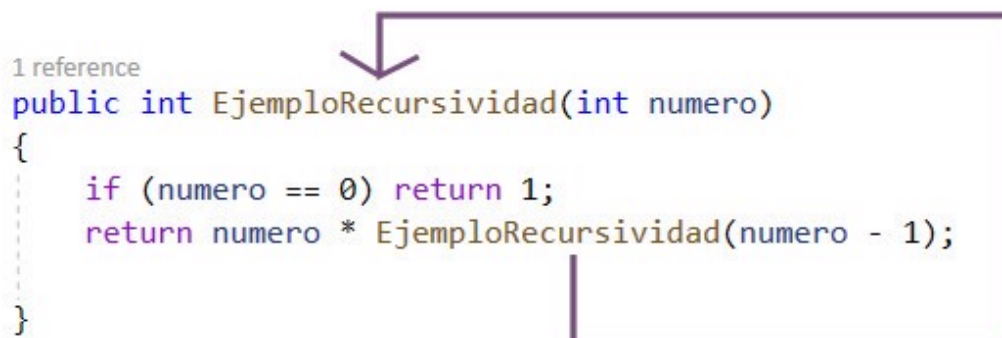
## Índice

- 1 - Qué es la recursividad en programación?
- 2 - Cuando utilizar recursividad?
- 3 - La pila de recursión

En el post de hoy trataremos un tema muy importante dentro de la programación, como es, la recursividad, es cierto que es un termino que cuando lo estudias por primera vez cuesta, pero es muy sencillito. Hay que decir que no está destinado únicamente al entorno .Net, sino que sirve para todos los lenguajes de programación.

### 1 - Qué es la recursividad en programación?

La recursividad es un concepto que se indica cuando un método se llama a si mismo. Cuando creamos un método recursivo debemos tener en cuenta que este tiene que terminar por lo que dentro del método debemos asegurarnos de que no se está llamando a si mismo todo el rato, Lo que quiere decir que el ciclo es finito.



```
1 reference
public int EjemploRecursividad(int numero)
{
    if (numero == 0) return 1;
    return numero * EjemploRecursividad(numero - 1);
}
```

Debemos tener mucho cuidado cuando realizamos llamadas recursivas ya que si la utilizamos sin control podríamos desbordar la memoria del ordenador, causando que el programa se rompa.

### 2 - Cuando utilizar recursividad?

Como se puede apreciar de la descripción anterior. Podemos utilizar recursividad para reemplazar cualquier tipo de bucle. A pesar de ello en el mundo laboral no se utiliza demasiado, debido a que un error puede ser trágico en la memoria, así como tener una lista con millones de datos, puede hacer que utiliza mucha memoria. Aun así, la gran mayoría de las veces, utilizamos recursividad para algoritmos de búsqueda u ordenación.

Se denomina caso base a la condición de terminación de la recursividad.

## Ejemplo 1

**Ejemplo es calcular el factorial de un numero** \*multiplicar todos los números entre dos enteros. En matemáticas se expresa con n!, donde n es el último número a comprobar. Debemos empezar desde el 1.

### Ejemplo utilizando un método iterativo

```
public static int FactorialIterativo(int numero)
{
    int i, resultado = 1;
    for(i=1; i <= numero; i++)
    {
        resultado = resultado * i;
    }
    return resultado;
}
```

### El ejemplo utilizando método recursivo

```
public static int FactorialRecursivo(int numero)
{
    if (numero == 0) return 1;
    return numero * FactorialRecursivo(numero - 1);
}
```

## 3 - La pila de recursión

La memoria de un ordenador se divide en 4 segmentos

- Segmento de código: almacena las instrucciones del programa en código máquina.
- Segmento de datos: almacena las variables estáticas o constantes.
- Montículo: almacena las variables dinámicas
- Pila del programa: Parte destinada a las variables locales y parámetros de la función que se está ejecutando.

Cuando llamamos a una función (o método) el proceso es el siguiente

Se reserva espacio en la pila para los parámetros de la función y sus variables locales.

Se guarda en la pila la dirección de la línea del código desde donde se ha llamado al método.

Se almacenan los parámetros de la función y sus valores en la pila.

Finalmente se libera la memoria asignada en la pila cuando la función termina y se vuelve a la llamada de código original.

En cambio, cuando la función es recursiva:

- Cada llamada genera una nueva llamada a una función con los correspondientes objetos locales.
- Volviéndose a ejecutar completamente, hasta la llamada a si misma. Donde vuelve a crear en la pila los nuevos parámetros y variables locales. Tantos como llamadas recursivas generemos.
- Al terminar, se van liberando la memoria en la pila, empezando desde la última función creada hasta la primera, la cual será la última en liberarse.