FUNCIONES DE ORDEN SUPERIOR

```
fun calculadora(n1:Int, n2:Int, fn: (Int, Int) -> Int):Int{
    return fn(n1, n2)
}

fun suma(x: Int, y: Int): Int {return  x + y}
fun resta(x: Int, y: Int): Int {return  x - y}

println("La suma de 40 y 30 es " + calculadora( n1: 40,  n2: 30, ::suma))
println("La resta de 65 y 32 es " + calculadora( n1: 65,  n2: 32, ::resta))
```

LAMBDAS

```
var funcion = {x:Int, y:Int -> x + y}

println("La suma de 40 y 30 es " + calculadora( n1: 40,  n2: 30, funcion))

funcion = {x:Int, y:Int -> x - y}

println("La resta de 40 y 30 es " + calculadora( n1: 40,  n2: 30, funcion))
```

Utilizo la misma estructura pero la modifico entre una llamada y otra. Las lambdas son estas funciones pero anónimas, por ejemplo:

```
println("La suma de 40 y 30 es " + calculadora( n1: 40,  n2: 30, {x:Int, y:Int -> x + y}))
println("La suma de 40 y 30 es " + calculadora( n1: 40,  n2: 30) { x: Int, y: Int -> x - y })
```

Parámetros, flecha y cuerpo.

Ejemplo para una lamba con “más cuerpo”:

```
println("La potencia de 2 elevado a 6 es " + calculadora( n1: 2,  n2: 6) { x, y ->
    var valor = 1
    for (i in 1 ≤ .. ≤ y) {
        valor *= x
    }
    valor ^calculadora // Sin el return

})
```

```
var arrayó = IntArray( size: 10){i -> 5*i}
println("Arrayó:")
for (i in arrayó.indices) print(arrayó[i])
println()
```

En resumen una lambda es una función anónima (no está definida en ningún otro lugar) que se pasa como parámetro en una función de orden superior.