Closures

Un *closure* es una función que ha sido definida dentro de otra función. Así un closure es toda función anidada, ya sea anónima o no.

Los *closures* se caracterizan por tener acceso a las variables locales que han sido definidas dentro de la función que las contiene.

Veamos un ejemplo de *closure*.

```
function externa(a)
    local b = 10
    local c = 5
    return function()
        return a + b + c
    end
end

local f = externa(1)
print(f())
```

Hemos definido una función `externa()` que acepta un argumento. Dentro de ella se definen dos variables `b` y `c`. Esta función devuelve una función anónima que realiza la suma de las variables `a`, `b`, `c`.

Al ejecutar el código del ejemplo se obtiene el siguiente resultado.

```
16
```

El concepto más importante a retener dentro de las *closures* es aquel del acceso a las variables locales de la función envolvente, a que ahí reside su utilidad. Veamos otro ejemplo para entender mejor el funcionamiento de los *closures*.

```
function contador()
    local i = 0
    return function()
        i = i + 1
        return i
    end
```

```
end

local contador1 = contador()
local contador2 = contador()

contador1()
contador2()
contador2()

print (contador1(), contador2())
```

En este ejemplo hemos definido de nuevo una función anónima dentro de otra, pero en esta ocasión no acepta ningún argumento. Dentro de la función hemos declarado una variable `i` que será usada para mantener el estado de la función. La función anónima devuelta incrementa `i` y devuelve su valor.

Para probar la función hemos declarado dos variables que son inicializadas con sendas referencias a la función `contador()`. Esto creará dos instancias de la función `contador()` cada una de ellas con su contador interno.

La primera llamada a `contador1()` incrementa `i` en una unidad. Del mismo modo las llamadas a `contador2()` incrementan el contador en una unidad cada una. Finalmente, durante la llamada a `print()`, volvemos a llamar tanto a `contador1()` como a `contador2()` lo cual incrementa de nuevo los contadores y devuelve su valor final. Así, el resultado que obtenemos es:

```
2       3
```

Revisión #1
Creado 8 abril 2024 13:00:05 por Guillermo
Actualizado 8 abril 2024 13:00:39 por Guillermo