Bucles

En ocasiones necesitamos hacer algunas tareas repetitivas y necesitamos que el programa ejecute una serie de instrucciones repetidamente. Para este tipo de necesidad existen los bucles, que ejecutan una serie de instrucciones de manera repetida hasta que se detengan en base a una condición. Esta condición debe cumplirse para que empiecen a ejecutarse y seguirán haciéndolo mientras esta condición sea válida. En el momento en que la condición ya no sea válida el bucle se detiene.

Los ciclos de funcionamiento de un bucle se llaman iteraciones. Una iteración empieza con la verificación de la condición y si esta es true, se ejecuta el bloque de código. Las iteraciones se repiten hasta que la condición sea false. Sin embargo, existe una excepción que veremos a continuación.

Lua ofrece los siguientes tipos de bucles:

Tipo de bucle	Descripción
while	El bucle *while* define una condición y un bloque de código
for	El bucle *for* define una secuencia y un bloque de código
repeat	El bucle *repeat* es similar al bucle *while* pero invertido

Todos los bucles en Lua se pueden anidar, esto es, que pueden ponerse bucles dentro de otros bucles, para construir así operaciones más complejas.

Bucle while

El bucle *while* es el más polivalente de todos los bucles. Acepta una condición de cualquier tipo permitiendo mucha flexibilidad a la hora de controlarlo. Este tipo de bucle se puede ejecutar cero o más veces.

La sintaxis de un bucle *while* es:

```
while(condition)
do
    -- Instrucciones
end
```

A continuación se muestra un ejemplo de bucle *while*:

```
i = 1

while(i <= 10)
do
    print(i)
    i = i + 1
end
```

Que produce la siguiente salida:

```
1
2
3
4
5
6
7
8
9
10
```

Bucle for

El bucle *for* está especializado en la iteración de tablas y secuencias. Permite principalmente definir el número de veces que el bucle debe ejecutarse. El bucle *for* puede ejecutarse cero o más veces.

La sintaxis del bucle *for* es:

```
for inicializacion, objetivo, incremento
do
    -- Instrucciones
end
```

Veamos en más detalle los tres componentes que debemos definir en el bucle *for*:

- **inicializacion**: se define una variable de control a la cual se le dará un valor numérico inicial. Esta inicialización se realiza una sola vez. Esta variable será utilizada para llevar el

control del número de iteraciones.
- **objetivo**: es el valor máximo o mínimo que la variable de control puede alcanzar. El bucle se ejecutará hasta que la variable de control alcance el máximo o el mínimo definido.
- **incremento**: indica en cuantas unidades se debe incrementar la variable de control. Este incremento ocurre después de cada iteración.

En el bucle *for*, cuando se ejecuta la primera iteración, se realiza la inicialización de la variable de control. Luego se verifica que la variable de control no es igual al valor máximo/mínimo y si es diferente se ejecutan las instrucciones. Al finalizar el bloque de instrucciones, se incrementa la variable de control con el valor definido en `incremento`. En las siguientes iteraciones, se vuelve a comprobar la variable de control con el objetivo, si aun no ha sido alcanzado, se vuelven a ejecutar las instrucciones y se vuelve a incrementar la variable de control. Estas iteraciones se repetirán hasta que la variable de control tenga el mismo valor que el objetivo definido. Todas estas operaciones son realizadas por el intérprete automáticamente.

En el siguiente ejemplo se imprime una lista de números del 1 al 10:

```
i = 1
for i = 1, 10, 1
do
    print(i)
end
```

El resultado del ejemplo anterior es:

```
1
2
3
4
5
6
7
8
9
10
```

Bucle repeat

A diferencia de los bucles **while** o **for**, el bucle **repeat...until** ejecuta el bloque de instrucciones al menos una vez, siempre antes de verificar la condición. Una vez ejecutado el bloque de instrucciones se verifica la condición, si esta es *false*, se vuelve a ejecutar el bloque de instrucciones, si es *true*, la ejecución finaliza.

El bucle *repeat...until* es similar al bucle *while*, la única diferencia es que el bucle *repeat...until* verifica la condición después de ejecutar el bloque de código.

La sintaxis del bucle **repeat...until** es:

```
repeat
    -- Instrucciones
until(condiciones)
```

Como puedes observar en el ejemplo anterior, la expresión condicional se encuentra después del bloque de instrucciones, de este modo el bloque de instrucciones siempre se ejecuta al menos una vez antes de que se verifique la condición.

Un ejemplo de uso del bucle *repeat...until* es el siguiente:

```
mensajes = {}

repeat
  if (#(mensajes) == 0) then
      print("No hay mensajes")
  else
      print(table.remove(mensajes, 1))
  end
until(#(mensajes) == 0)
```

El resultado del ejemplo anterior es:

```
No hay mensajes
```

Como ves, el bloque de código se ha ejecutado una sola vez, ya que la tabla `mensajes` está vacía. Probemos ahora con algunos mensajes:

```
mensajes = {
  "Mensaje 1",
  "Mensaje 2",
  "Mensaje 3"
}

repeat
  if (#(mensajes) == 0) then
    print("No hay mensajes")
  else
```

```
    print(table.remove(mensajes, 1))
  end
until(#(mensajes) == 0)
```

Cuyo resultado es:

```
Mensaje 1
Mensaje 2
Mensaje 3
```

En este caso ya no aparece *No hay mensajes*. Al ejecutarse el bloque de instrucciones `mensajes` tiene una longitud de 3, con lo que se extrae el primer elemento de la lista y se imprime (*Mensaje 1*). Durante la verificación de la condición, esta no se cumple porque `mensajes` tiene una longitud de 2. El bloque se ejecuta una vez más, extrayendo el primer elemento de la lista. Al comprobar la condición, `mensajes` tiene una longitud de 1, con lo que no se cumple y la ejecución continua. El bloque se ejecuta de nuevo, extrae el primer elemento de la lista y lo imprime. Durante la verificación de la condición, esta vez `mensajes` tiene una longitud de 0, por lo que el bucle finaliza.

Bucles anidados

El anidamiento de bucles permite la realización de tareas complejas, como por ejemplo, el recorrido de arrays multidimensionales. Cuando anidamos bucles, lo podemos hacer con cualquier tipo de bucle, pudiendo mezclarlos si es preciso.

A continuación se muestran algunos ejemplos:

```
-- Bucle while anidado en un bucle for

for indice, valor, incremento
do
  while(condicion)
  do
    -- instrucciones
  end

  -- instrucciones
end
```

```
-- Bucle while anidado en bucle while

while(condicion)
```

```
do
  while(condicion)
  do
    -- instrucciones
  end

  -- instrucciones
end
```

```
-- Bucle for anidado en otro bucle for

for indice, valor, incremento
do
  for indice, valor, incremento
  do
    -- instrucciones
  end
  -- instrucciones
end
```

Instrucción break

La instrucción **break** permite controlar la ejecución del bucle. Con ella podemos finalizar el bucle en todo momento, sin que sea necesario que la condición se cumpla.

Cuando el intérprete encuentra una instrucción *break* dentro de un bucle, sale del bucle y continua ejecutando las instrucciones siguientes. Si el la instrucción *break* se encuentra dentro de un bucle anidado, la ejecución pasará al bucle padre.

El siguiente ejemplo muestra el funcionamiento de la instrucción *break*:

```
print("Antes del bucle")

for i = 1, 5, 1
do
  if (i > 3) then
    break
  end

  print("-> " .. i)
```

```
  for j = 1, 5, 1
  do
    if (j > 3) then
      break
    end

    print("   |- " .. j)
  end
end

print("Despues del bucle")
```

Al ejecutar el código anterior obtenemos el siguiente resultado:

```
Antes del bucle
-->
|- 1
|- 2
|- 3
--> 2
|- 1
|- 2
|- 3
--> 3
|- 1
|- 2
|- 3
Despues del bucle
```

Observa como el uso de la instrucción *break* nos ha permitido modificar el funcionamiento del bucle. En lugar de imprimir hasta el número 5, que es el máximo en ambos bucles, hemos puesto una condición que sale del bucle cuando llega a 3. Cuando esto sucede en el bucle anidado, la ejecución continua en el bucle padre. Cuando el bucle padre llega a 3, finaliza y continua con la última instrucción del ejemplo.

Bucle infinito

Un bucle es infinito cuando la condición es siempre válida. Sirve generalmente para casos en los que se necesita que el bucle se ejecute continuamente como base del funcionamiento del programa. Para ello se usa generalmente el bucle **while** al que le ponemos como condición la

palabra clave **true**, de este modo forzamos que la condición sea siempre válida. Un ejemplo de bucle infinito es:

```
while( true)
do
 -- instrucciones
end
```

Siempre puedes finalizar el bucle en todo momento con la instrucción *break*.

Revisión #2
Creado 23 mayo 2023 08:33:24 por Guillermo
Actualizado 31 julio 2023 12:02:47 por Guillermo