Variables

Las variables son referencias a espacios de memoria donde se almacenan los datos de la aplicación. Las variables pueden ser de cualquier tipo soportado por Lua. En todo caso, antes de poder usar una variable esta debe ser declarada. Si intentamos acceder a una variable que no ha sido declarada se devuelve `nil`.

Existen tres tipos de variables: locales, globales y campos de tabla. Las variables globales son accesibles accesibles por todo el script. Las variables locales son accesibles solamente en el bloque donde fueron declaradas. Para declarar una variable como local debemos usar la palabra clave `local` delante del identificador de la variable:

```
local a = 10    --> Variable local
```

Las variables globales se declaran directamente sin usar ninguna palabra clave:

```
a = 10    --> Variable global
```

En el siguiente ejemplo vamos a definir una variable global y una local, para luego intentar acceder a éstas desde los dos ámbitos:

```
a = 10

function Local()
    local b = 5

    print("Local: a -> " .. a .. ", b -> " .. b)
end

Local()
print("Global: a -> " .. a .. ", b -> " .. b)
```

Cuyo resultado es:

```
$ lua TestAmbitosVariables.lua
Local: a -> 10, b -> 5
lua: TestAmbitosVariable.lua:10: attempt to concatenate global 'b' (a nil value)
stack traceback:
TestAmbitosVariable.lua:10: in main chunk
```

```
[C]: in ?
```

Como puedes observar, la llamada a la función `Local()` se ejecuta correctamente, la variable global `a` y la variable local `b` se pueden acceder desde la función sin problemas. Sin embargo, la última llamada a `print()` produce un error: b es `nil`. Dado que `b` fue definida como local dentro de la función, por lo que, esta está disponible solamente dentro de dicha función, fuera de ella, simplemente no existe.

Declaración y asignación

Toda variable en Lua debe ser declarada e inicializada antes de ser utilizada. A diferencia de lenguajes de tipos estáticos como C, C++, C#, Java, etc., Lua no permite la declaración de una variable sin inicializarla. Solo existe una excepción a esta regla y es en el caso de los parámetros de funciones u otras estructuras del lenguaje, como el bucle `for`. En estos casos sí se declara una variable sin inicializar, posteriormente, en tiempo de ejecución, el intérprete le asignará los valores en función del flujo del programa. El siguiente código es correcto en C#:

```
string miCadena;
bool condicion = true;

if (condicion)
    miCadena = "Verdadero";
else
    miCadena = "Falso";

Console.WriteLine(miCadena);
```

Cuyo resultado es:

```
Verdadero
```

Sin embargo, el código anterior no funcionaría en Lua. En la primera línea se declara una variable de tipo `string` a la cual no se le asigna ningún valor. Esta variable tendrá entonces un valor `null`. Más tarde en el bloque condicional se le asigna un valor a esa variable. En Lua no podemos declarar una variable sin asignarle un valor y esto es debido a los tipos dinámicos. Si en Lua intentamos declarar una variable sin asignarle un valor se produce un error.

Lua maneja dinámicamente el tipo de las variables. Como no podemos definir el tipo del mismo modo que se hace en C#, el único modo que tiene el intérprete de conocer el tipo es por la expresión que se le asigne. Esto se llama inferencia de tipos.

En el siguiente código de ejemplo, declaramos una función y una tabla donde asignamos varios elementos de diferentes tipos:

```
function Funcion(valor)
    print(type(valor))
end

a = {
    "Texto",
    15.9,
    Funcion,
    false,
    {}

}

for k, v in pairs(a) do
    Funcion(v)
end
```

Cuando ejecutamos este código obtenemos la siguiente salida:

```
$ lua TestTiposAsignados.lua
string
number
function
boolean
table
```

Como puedes observar en el ejemplo anterior, Lua ha reconocido los tipos de las variables a partir de las expresiones asignadas. En el apartado Tipos de datos tienes más información acerca de los diferentes tipos disponibles en Lua y sus expresiones.

Asignación de múltiples variables

Lua permite la asignación de varias variables al mismo tiempo:

```
a, b = "texto", 5
```

Los valores son asignados a las variables en el mismo orden, por lo que el ejemplo anterior equivale a:

```
a = "texto"
b = 5
```

En el caso de la asignación de variables cruzadas, Lua primero, evaluará las variables a la derecha para luego hacer la asignación. De este modo podemos invertir dos variables:

```
x, y = y, x                 --> Inversión del valor de las variables x e y
a[i], a[j] = a[j], a[i]     --> Inversión del valor de los elementos i y j
```

Si el número de elementos de la izquierda no es igual al de elementos en la derecha se siguen las siguientes reglas:

- **Menos elementos por la derecha**: se asignan valores a las variables de la izquierda. Cuando no queden valores para asignar por la derecha, se asignará `nil` a las variables restantes.
- **Menos elementos por la izquierda**: se asignan valores a las variables de la izquierda. Los valores que no pueden ser asignados a ninguna variable se descartan.
-

```
> x, y, z = 1, 2
> print(x, y, z)
1 2 nil
> x, y = x+1, y+2, z+3
> print(x, y)
2, 4
> x, y, z = 0
> print(x, y, z)
0 nil nil
```

También se pueden asignar directamente a varias variables el retorno de una función que devuelva varios valores:

```
x, y = obtenerCoordenadas()
```

En este ejemplo, se asume que la función `obtenerCoordenadas()` devuelve dos valores que serán asignados a las variables `x` e `y`.

Es posible el uso de variables locales en las asignaciones múltiples.

Índice

Los índices de las tables se pueden especificar de dos modos:

- Usando corchetes []. Este método puede ser usado con todos los elemento de tipo lista y clave/valor.
- Usando un punto '.'. Este método sólo es válido para los elementos de tipo clave/valor.

```
table[i]
table.i
```

```
> table = { 10, 20, i3 = 30 }
> print(table[1])
10
> print(table.2)
stdin:1: ')' expected near '.2'  -- Produce un error porque es un elemento de tipo lista
> print(table[2])
20
> printe(table.i3)      -- Es un elemento de tipo clave/valor
30
```

Revisión #4
Creado 23 mayo 2023 07:42:40 por Guillermo
Actualizado 31 julio 2023 12:02:47 por Guillermo