Tablas

Lua implementa un tipo de base con el que construir diferentes tipos de estructuras de datos. Se llaman tablas y ofrecen las características necesarias para la creación de estructuras de datos complejas que pueden ser manejadas de una manera eficiente.

En este capítulo vamos a ver qué es una tabla en Lua y diferentes tipos de estructuras de datos que pueden ser implementadas usando las tablas, para disponer de estructuras específicas para casos particulares, como arrays o diccionarios.

Tablas

Las tablas, *tables* por su denominación en inglés, en Lua son colecciones de pares clave/valor, cuyos valores pueden ser de cualquier tipo excepto `nil`. Constituyen la única estructura de datos disponible en Lua y sirve de base para la creación de otras estructuras de datos. Las tablas se indexan por la clave, así, se puede recuperar cualquier valor de la tabla si conocemos la clave. Dado que las claves y valores de las tablas en Lua pueden ser de cualquier tipo, una clave o un valor pueden ser a su vez de tipo tabla. Las tablas en Lua son dinámicas: no tienen un tamaño definido y pueden crecer según las necesidades.

A partir de una tabla podemos construir cualquier tipo de estructura de datos más compleja o específica, como por ejemplo arrays, arrays multidimensionales, matrices, listas, colas, etc.

Para facilitar la manipulación de las tablas, Lua dispone de una librería estándar que define una serie de funciones para operar con las tablas.

Sintaxis de las tablas

Las tablas se definen por medio de los constructores de tablas, que se representan por los caracteres de llaves `{` y `}`. Una tabla vacía se define como a continuación.

```
> tabla = {}
> print( tabla)
table: 00000000007d4230
```

En el ejemplo anterior el valor retornado por la función print es el tipo y el identificador del objeto. Para poder recuperar un valor contenido en la tabla debemos hacer referencia a un elemento por su clave.

Uso de las tablas

La tabla es un tipo complejo y flexible que permite la gestión de estructuras complejas de datos. A continuación tratamos las diferentes posibilidades de uso.

Definición y asignación

Una vez definida una tabla, podremos acceder a cada uno de sus elementos a través de la clave. Para insertar valores en una tabla, damos valor a la clave y le asignamos un valor.

```
> tabla = {}
> tabla["clave"] = "Lua"      -- Insertamos con la clave 'clave' el valor 'Lua', ambos cadenas
> tabla[1] = 1000             -- Insertamos con la clave '1' el valor '1000', ambos enteros
> = tabla["clave"]            -- Recuperamos el valor de la clave 'clave'
Lua
> = tabla[1]                  -- Recuperamos el valor de la clave '1'
1000
```

Como ves en el ejemplo anterior, las claves pueden ser de cualquier tipo, pudiéndose incluso mezclar. Si bien, por lo general, en muchos tipos de estructuras de datos las claves suelen ser del mismo tipo. Del mismo modo, los tipos de la clave y el valor pueden ser diferentes también. En todos los casos la clave nunca puede ser `nil` ni `NaN` (not a number).

```
> tabla = {}
> clave = {}
> funcion = function () end
> tabla[clave] = 250            -- Asignamos 250 a la clave 'clave' que es una tabla
> tabla[funcion] = 500          -- Asignamos 500 a la clave 'funcion' que es una funcion
> = tabla[clave]                -- Recuperamos el valor de la clave 'clave'
250
> = tabla[funcion]              -- Recuperamos el valor de la clave 'funcion'
500
> tabla[nil] = 125              -- Asignar un valor a una clave 'nil' produce un error
stdin:1: table index is nil
stack traceback:
        stdin:1: in main chunk
        [C]: in ?
> tabla[0/0] = 450              -- Asignar un valor a una clave 'NaN' también produce error
stdin:1: table index is NaN
stack traceback:
        stdin:1: in main chunk
        [C]: in ?
```

Cuando se dice que una clave puede ser de cualquier tipo, significa también que una clave puede ser el resultado de una expresión válida cualquiera.

```
> tabla = {}
> tabla[1/0] = 750                                    -- Clave basada en una operación aritmética
> tabla[0/1] = 1000                                   -- Otro ejemplo de operación aritmética
> = tabla[1/0]
750
> = tabla[0/1]
1000
> tabla[10^3] = 1250                                  -- Operación aritmética: exponenciación
> = tabla[10^3]
1250
> tabla[string.gsub("aa", "a", "b", 1)] = 1500     -- Clave basada en el retorno de la llamada
a una función
> = tabla[string.gsub("aa", "a", "b", 1)]
1500
```

Acceso a claves no definidas

En el caso de que recuperemos el valor de una clave que no existe, no se produce ningún error y devuelve `nil`.

```
> tabla = {}
> = tabla[100]
nil
```

Supresión de pares

Cualquier clave que hayamos declarado puede ser eliminada asignándole el valor `nil`.

```
> tabla = {}
> tabla["clave"] = 10
> = tabla["clave"]
10
> tabla["clave"] = nil
> = tabla["clave"]
nil
```

Acceso simplificado a claves tipo string

Las tablas también ofrecen un modo de acceso sencillo para las claves de tipo `string`, reduciendo el código necesario para acceder a una clave. Los únicos requisitos para este tipo de claves es respetar que sean de tipo `string` y que tengan la siguiente sintaxis:

> *Una cadena que contenga letras, números y guión bajo (_ underscore). No puede comenzar por un número.*

```
> tabla = {}
> tabla.clave = 1000
> = tabla.clave
1000
> tabla.1 = 500
stdin:1: syntax error near '.1'
> tabla._1 = 500
> = tabla._1
500
```

Definición y asignación avanzada

Hasta ahora hemos visto una forma para la definición de una tabla: primero declaramos la tabla y luego le asignamos los valores. Sin embargo Lua ofrece la posibilidad de declarar e inicializar una tabla en una sola expresión. Veamos un ejemplo de definición y asignación en una sola sentencia.

```
tabla = {["clave"] = "valor", [123] = 1000}
= tabla.clave
valor
= tabla["clave"]
valor
= tabla[123]
1000
```

En el ejemplo anterior, puedes observar que en la primera línea, a la variable `tabla`, se le asigna una tabla en la cual los pares clave valor le son definidos en la misma línea.

Revisión #2
Creado 31 julio 2023 11:58:25 por Guillermo
Actualizado 4 octubre 2023 05:58:39 por Guillermo