Lua Programming Language

...

Adam Shkolnik, Liam Murphy, Karam Khan, and Liam Rich

Background/Rationale

- Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes created Lua in 1993
- Built to be a **scripting language** , not just a dynamic language
- Lua is a "**Freestanding Application** "
- Has a **garbage collector** to reduce storage waste
- **Tables** inspired by VDM and AWK's data structures
- **First-class functions** necessary due to compilation method
- **Closures** implemented to reduce restrictions on first-class functions
- Concurrency **not included** due to race condition issue
- **Asymmetrical Coroutines** used for multitasking instead

Code Sample

C++

```cpp
lua_State *lua = luaL_newstate(); // create lua VM
luaL_openlibs(lua); // allow use of lua std lib

int result = luaL_dofile(lua, "src/main.lua"); // load script

if(result != LUA_OK) // check if failed to load script
{
    return; // exit program
}

lua_getglobal(lua, "add_two"); // get function
if(lua_isfunction(lua, -1)) // check if the value is a func.
{
    int a = 3; int b = 5; int sum;

    lua_pushinteger(lua, a); // place param on stack
    lua_pushinteger(lua, b); // place param on stack
    lua_pcall(lua, 2, 1, 0); // call lua function

    sum = lua_tointeger(lua, -1); // get return value from stack
    lua_pop(lua, 1); // remove return value from stack
    fmt::println("sum: {}\n", sum); // print value
}
```

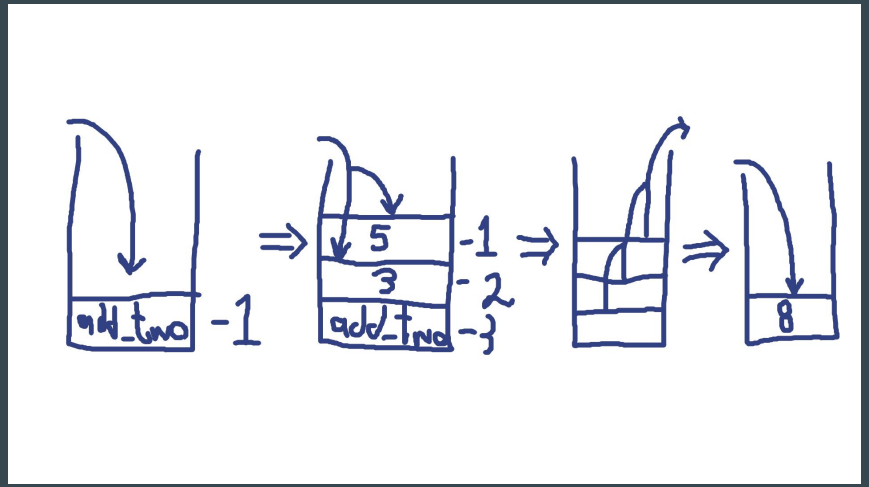
Lua

```lua
function add_two(a,b)
    return a + b;
end
```

Output

```
sum: 8
```

Stack Diagram



Classification & Taxonomy

- Multiparadigm Nature: Lua is a multiparadigm language, primarily supporting procedural, functional, imperative, and object-oriented programming.
- Data Abstraction: Lua supports data abstraction using tables and metatables, allowing developers to implement both object-oriented and functional programming.
- Simplicity and Flexibility: Lua's simple syntax and dynamic typing make it accessible and flexible..
- Concurrency: Lua provides coroutines for cooperative multitasking, enabling programmers to write concurrent programs easily. Although it doesn't offer true parallelism out of the box, coroutines enable concurrency without other complexities

Data Types:

Nil
Boolean
Number
String
- Immutable

Table
Function
- Functions are first-class values in Lua, meaning they can be stored in variables, passed as arguments, and returned from other functions.

Userdata
- A type that allows arbitrary C data to be stored in Lua variables, primarily used to interface with C libraries.

Evaluation

Readability

- Lua's syntax is minimal, due to its similarity with other high-level languages like Python.
- The consistent use of data types and control structures provides a straightforward learning curve.
- Its concise syntax ensures readable and easily maintainable code.

Writability/Productivity

- Lua's lightweight design makes it simple for devs to write code, which makes a big difference especially when embedding in larger applications.
- It supports first-class functions, closures, and metatables, adding flexibility without complexity.
- The versatile table structure simplifies data handling and allows for expressive code.

Reliability

- Lua's dynamic typing can lead to runtime errors, requiring careful handling.
- Basic error handling via pcall and xpcall covers exceptions effectively.
- The flexibility of tables can lead to aliasing issues if not managed properly.

Evaluation - Cont.

Cost

- Lua is much easier compared to most languages when it comes to learning which **reduces training overhead.**
- Its lightweight nature allows for much more efficient software development and **lower maintenance costs.**
- With **fast execution and low resource usage,** Lua is a major cost-effective choice for performance-critical tasks.

Others

- Lua is highly **portable** , running on a variety of platforms, from PCs to embedded systems.
- While it is known for **general-purpose use** , Lua really excels in **game scripting** and **embedded systems** .
- It's **well-documented** , with a clear manual, making it easy to implement across platforms.
- Lua's **community** is strong in gaming, with projects like **Roblox** and **World of Warcraft** , though its ecosystem is smaller than other languages.
- Used in major projects like **Angry Birds** and **World of Warcraft** , Lua shows its power in game development and beyond.
- Lua's **lightweight and fast language architecture** makes it and ideal choice for customizable, performance-driven scripting.

Conclusion and Demo

- Useful for extending the functionality of applications
- Good for defining data for application
- Let C/C++ engine code do as much of the heavy lifting to allow for a simple and easy Lua interface

Questions?

Resources

- Lua Videos by Javidx9
  - #1 Embedding Lua in C++
  - #2 Automating Sequences via Lua Coroutines in C++
  - #3 Meh... Just use Sol...
- Programming in Lua Book
- Demo Code
- Lua Template