**Lua** (*лу́а*, с [порт.](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%82%D1%83%D0%B3%D0%B0%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) — «луна») — [скриптовый](https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), разработанный в подразделении Tecgraf (*Computer Graphics Technology Group*) [Католического университета Рио-де-Жанейро](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%B0%D1%82%D0%BE%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82_%D0%A0%D0%B8%D0%BE-%D0%B4%D0%B5-%D0%96%D0%B0%D0%BD%D0%B5%D0%B9%D1%80%D0%BE&action=edit&redlink=1) ([Бразилия](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D0%B7%D0%B8%D0%BB%D0%B8%D1%8F)). Интерпретатор языка является [свободно распространяемым](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), с открытыми исходными текстами на [языке Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)).

Lua разработана подразделением Tecgraf (группа технологий компьютерной графики) Католического университета Рио-де-Жанейро в Бразилии, история языка ведёт отсчёт с 1993 года. Авторы языка — [Роберто Иерусалимский](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D1%83%D1%81%D0%B0%D0%BB%D0%B8%D0%BC%D1%81%D0%BA%D0%B8%D0%B9,_%D0%A0%D0%BE%D0%B1%D0%B5%D1%80%D1%82%D0%BE), Луиш Энрике ди Фигейреду и Валдемар Селиш. Lua распространяется свободно, с открытыми исходными текстами на [языке Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)).

Историческими родителями Lua были языки конфигурирования и описания данных *SOL* (Simple Object Language) и *DEL* (Data-Entry Language). Они были независимо разработаны в Tecgraf в 1992—1993 годах для добавления некоторой гибкости в два отдельных проекта (оба были интерактивными графическими приложениями для конструкторских нужд в компании [Petrobras](https://ru.wikipedia.org/wiki/Petrobras)). В SOL и DEL отсутствовали какие-либо управляющие конструкции, и Petrobras чувствовал растущую необходимость в добавлении к ним полноценного программирования.

Как пишет автор языка в *The Evolution of Lua*:

“В атмосфере «сделай сам», которая тогда царила в Tecgraf, было вполне естественно, что мы решили разработать наш собственный скриптовый язык. Из-за того, что большинство пользователей не были профессиональными программистами, языку следовало избегать замысловатого синтаксиса и семантики. Реализация нового языка должна быть легко [портируема](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F" \o "Портирование программного обеспечения), так как клиенты Tecgraf имели очень разнообразные платформы. Наконец, поскольку мы ожидали, что другим продуктам Tecgraf также понадобится встроенный скриптовый язык, новый язык должен предоставляться в виде библиотеки с [API](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) на [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)).”

Управляющие конструкции Lua в основном заимствованы из [Modula](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B0-2) (if, while, repeat/until), хотя на них также повлияли [CLU](https://ru.wikipedia.org/wiki/CLU) ([параллельное присваивание](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%B8%D1%81%D0%B2%D0%B0%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), множественное [возвращаемое значение функции](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B2%D0%BE%D0%B7%D0%B2%D1%80%D0%B0%D1%89%D0%B0%D0%B5%D0%BC%D0%BE%D0%B3%D0%BE_%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) как более простая альтернатива вместо [передачи параметров по ссылке](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B0_%D0%BF%D0%BE_%D1%81%D1%81%D1%8B%D0%BB%D0%BA%D0%B5) или явных [указателей](https://ru.wikipedia.org/wiki/%D0%A3%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85))), [C++](https://ru.wikipedia.org/wiki/C%2B%2B) («отличная идея объявлять [локальные переменные](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F) лишь тогда, когда они нужны»), [SNOBOL](https://ru.wikipedia.org/wiki/%D0%A1%D0%BD%D0%BE%D0%B1%D0%BE%D0%BB) и [AWK](https://ru.wikipedia.org/wiki/AWK) ([ассоциативные массивы](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2)). Создатели Lua также признают, что единый вездесущий механизм структурирования данных в LISP и Scheme ([связный список](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA)) оказал большое влияние на их решение о выборе таблиц в качестве основной структуры данных для Lua.

Общая характеристика

Lua обычно называют [мультипарадигменным языком](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BF%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Он обеспечивает небольшой набор базовых механизмов, которые могут быть расширены для решения различных задач, а не набор сложных жёстких спецификаций, обеспечивающих программирование в единой [парадигме](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F).

Lua — это[динамически типизированный](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) язык, предназначенный для использования в качестве расширения или в качестве скриптового языка, и при этом достаточно компактный, чтобы поместиться на различных [исполняющих платформах](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0).

Синтаксис

Синтаксис Lua в основном построен на основе поздних паскалеподобных языков, таких как [Модула-2](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B0-2) или [Оберон](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D1%80%D0%BE%D0%BD_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)). Формат записи текста - свободный, команды в тексте программы разделяются любыми пробельными символами. Допускается, но не является обязательным применение точки с запятой для разделения операций.

**Лексика**

Основной алфавит языка — английский, в строковых литералах допускается использование символов других языков. Идентификаторы могут состоять из букв, цифр и знака подчёркивания, но не могут начинаться с цифры или совпадать с одним из ключевых слов. Руководство по языку не рекомендует использовать идентификаторы, начинающиеся с подчёркивания, так как такие идентификаторы используются для системных целей.

Язык регистро-зависимый, все ключевые слова пишутся в нижнем регистре, идентификаторы, отличающиеся только регистром букв, считаются различными.

**Комментарии**

Для комментариев используется следующий синтаксис, близкий к языкам [Ada](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [SQL](https://ru.wikipedia.org/wiki/SQL) и [VHDL](https://ru.wikipedia.org/wiki/VHDL)

*-- Простой однострочный комментарий в Lua начинаются с двойного дефиса и продолжается до конца строки.*

*-- Строчный комментарий не обязан начинаться с начала строки,*

*--[[Многострочный комментарий начинается с идущих подряд за двумя минусами двух открывающихся квадратных скобок и продолжается до двух подряд закрывающихся квадратных скобок. Как здесь: ]]*

**Типы данных**

Lua представляет собой язык с неявным [динамическим](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) определением [типов данных](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). [Переменная](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) языка может содержать значения любого типа. Все значения в Lua могут храниться в переменных, использоваться в качестве [аргументов](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%B3%D1%83%D0%BC%D0%B5%D0%BD%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) при вызове [функций](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и возвращаться в виде результата их выполнения.

В Lua восемь основных типов:

* **nil** (неопределенный)
* **boolean** (логический)
* **number** (числовой)
* **string** (строковый)
* **function** (функция)
* **userdata** (пользовательские данные)
* **thread** (поток)
* **table** (таблица)

К типу *number* относятся обычно вещественные числа (double).

Тип *string* jбозначает массивы символов. Строки Lua могут содержать любые 8-битные символы, включая ноль. Многострочные литералы ограничиваются двумя подряд открывающимися и двумя подряд закрывающимися квадратными скобками.

Функции в Lua являются полноправными объектами, допускающими присваивание, передачу в функцию в параметре и возврат из функции как одного из значений. Тип **thread** имеют сопрограммы, тип **userdata** предназначен для представления внешних данных, полученных или предоставляемых из/в код на другом языке (главным образом, на C/C++).

**Таблицы**

Таблица в Lua — это[динамический](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2) [гетерогенный](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D1%82%D0%B5%D1%80%D0%BE%D0%B3%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [ассоциативный массив](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2), то есть множество пар «ключ-значение». Ключами могут быть значения любых типов Lua, кроме nil. Ключи также могут быть идентификаторами Lua. Запись nil в элемент таблицы равносильна удалению данного элемента.

Таблицы являются единственным в Lua составным типом данных. Они являются фундаментом для всех пользовательских типов данных, таких как [структуры](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [массивы](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [множества](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)) и другие:

**Операции**

**Присваивание**

Оператором присваивания служит символ **=**. Lua опускает [параллельное присваивание](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%B8%D1%81%D0%B2%D0%B0%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). В простейшем случае это позволяет писать выражения вида:

x, y = y, x

В отличие, например, от языка [Go](https://ru.wikipedia.org/wiki/Go" \o "Go), параллельное присваивание в Lua очень либерально, оно не требует точного соответствия числа переменных числу значений. Если значений больше, чем переменных, то последние значения отбрасываются, если значений меньше — последние переменные в списке получают значение nil.

**Сравнения и логические операторы**

Операторы сравнения: **<, >, <=, >=, ==, ~=** (последние два означают, соответственно, сравнение на равенство и на неравенство). Сравнения на равенство и неравенство применимы к любым типам данных, прочие — только к строкам и числам.

Логические операторы: **and, or, not** — логические «и», «или» и «не», соответственно, могут применяться к значениям любых типов и возвращают true либо false. В параметрах этих операция значения nil, которая возвращает постоту, и false считаются ложными, любые другие значения — истинными. В отличие от C и ряда других языков, даже числовой нуль (0) или пустая строка ("") с точки зрения логических операторов представляют значение «истина». Вычисление бинарных логических операторов традиционно выполняется до тех пор, пока результат не будет определён, причём в качестве результата возвращается последнее вычисленное подвыражение.

**Арифметические операторы**

**+** — сложение;

**-** — вычитание;

**\*** — умножение;

**/** — деление;

**-** — унарный минус;

**%** — остаток от деления;

**^** — возведение в степень.

**Замыкания**

Lua поддерживает концепцию [замыканий](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)).

Каждый раз, когда вызывается makeaddfunc, создаётся новое замыкание для переменной x, так что каждая возвращаемая анонимная функция будет ссылаться на свой параметр x. Как и у любого другого объекта Lua, временем жизни замыкания управляет сборщик мусора.

**Достоинства и недостатки**

Во-первых, как уже было отмечено, своей компактностью, а вкупе с тем, что исходники написаны на С, вы получаете полное взаимодействие с одним из популярнейших языков на планете и широкий спектр доступных платформ.

Во-вторых, он быстрый. Если[взглянуть](http://benchmarksgame.alioth.debian.org/) на сравнительную характеристику с другими языками, то можно заметить, что хоть Lua и не укладывает на лопатки C или Python, но в некоторых тестах показывает отличные результаты.

В-третьих, он очень удобен для изучения даже не самыми опытными программистами. Наверняка даже те, кто просто знает английский язык, поняли минимум 80% описанного выше кода и без проблем смогут воспроизвести.

Четвертый аргумент - у вас не возникнет никаких проблем с поиском информации. У Lua есть много полезной информации непосредственно [на официальном сайте](https://www.lua.org/docs.html). Кроме того, очень недурно развито [сообщество](http://www.lua.org/community.html) на [StackOverFlow](http://stackoverflow.com/questions/tagged/lua) и [IRC-чате](http://lua-users.org/wiki/IrcChannel), а в США разработчики и создатели так и вовсе встречаются каждый год.

Любителей современных тенденций привлечет тот факт, что на Lua можно писать функциональный код. Да что там, если вы хотя бы пару недель уделите программированию на Lua, то наверняка найдёте ещё не один десяток плюсов.

В Lua недостатки носят локальный характер. Так, например, язык по умолчанию не поддерживает Unicode, но это исправляется с использованием специальной библиотеки. Или ограниченные [возможности обработки ошибок и исключений](http://www.lua.org/pil/8.4.html), хотя многие сочтут это за благо. Или необходимость ставить оператор return исключительно последним в блоке, но опять-таки для многих это естественное правило хорошего кода.

**Использование**

Язык широко используется для создания тиражируемого программного обеспечения (например, на нём написан графический интерфейс пакета Adobe Lightroom). Также получил известность как язык программирования уровней и расширений во многих играх (в том числе World of Warcraft) из-за удобства встраивания, скорости исполнения кода и лёгкости обучения