Правила написания программ на языке Луа

13 февраля 2024 г.

1. Введение

Настоящий документ содержит требования, предъявляемые к программам на языке Луа, сдаваемым студентами II курса кафедры ИУ-1 МВТУ в рамках курса «Информатика» и «Практикум на ЭВМ». Предполагается, что читатель уже знаком с аналогичным документом для языка Си.

Требования предъявляются к трём аспектам программ: переносимости, структуре и исходному тексту. С точки зрения сдачи преподавателям (и, следовательно, получения оценки за курс) программы, не удовлетворяющие настоящему документу, приравниваются к ненаписанным.

В рамках этого документа принимаются следующие обозначения.

1. Некоторые требования содержат фразу «только при наличии веского обоснования». Это означает, что использование того или иной приёма считается допустимым только если при сдаче программы будет представлено удовлетворительное объяснение (устное), почему без этого приёма в данном конкретном случае обойтись нельзя.
2. В примерах фрагменты кода, не противоречащие требованиям, называются *допустимыми*.
3. В примерах кода допустимый код помещен в чёрную рамку, недопустимый — в красную.
4. В примерах выражение `--[[ ... ]]` означает пропуск нерелевантной части кода.

В качестве дальнейшего чтения по вопросам правил и подходов, помогающих в написании кода, можно порекомендовать работы [1], [2, главы 12–18], [3, главы 7,10–13,17,19,31,32].

Наконец, напомним важное правило:

Исходный текст программ предназначен
в первую очередь для понимания человеком.

2. Правила написания программ

2.1. Требования переносимости

1. Программа должна удовлетворять спецификация языка Луа версии 5.4 [4].

2.2. Требования к тексту

2.2.1. Символы

1. Файл исходного кода должен быть в кодировке UTF8.
2. Символы, отличные то базового набора ASCII (коды 0–127), допускаются только в комментариях.
3. Ширина строки кода не должна превышать 80 символов.

2.2.2. Выражения и строки

1. На каждой строке кода может располагаться не более одного оператора. (Это требование можно понимать так: не более одного действия на строку.)
2. Допускается инициализация переменной значением-функцией в одну строку, если функция содержит не более одного оператора.
   Пример. `local f = function(x) return cos(2*x) end`

2.2.3. Отступы и пробелы

1. Допускается использование в роли отступов двух пробелов, четырёх пробелов или символа табуляции.
2. В программе допускается использование только одного вида отступов.
3. Строка с открывающей операторной скобкой должны быть на том же уровне вложенности, что и конструкция, порождающая блок (объявление функции, цикла или типа).
4. Строка с закрывающей операторной скобкой должна быть на том же уровне вложенности, что и строка с открывающей.
5. Программа должна использовать только один способ расположения открывающих скобок: либо на той же строке, что и порождающая блок конструкция, либо на следующей.
6. Открывающая и закрывающая скобки не могут находиться на одной строке.
7. Все строчки между двумя парными операторными скобками должны быть на один уровень вложенности выше, чем эти скобки.
8. Бинарные операторы (+, *, ...) должны быть отделены пробелами от аргументов.
9. Унарные операторы пробелами не отделяются.
10. После запятой всегда ставится пробел.
11. В случае, когда последовательность вложенных операторов выбора является взаимоисключающей, должен использоваться оператор `elseif` без увеличения вложенности на подветках.

2.2.4. Разбиение на несколько строк

1. При инициализация полей таблиц, не помещающееся в одну строчку, каждое поле должно инициализироваться на отдельной строчке.
   Пример. Фрагмент слева является допустимым, справа — **недопустимым**.

```
d = {
  file = "input.txt",
  width = 100,
  height = 100
}
```

```
d={file="input.txt",width=100,height=100}
```

2. При необходимости разбиения вызова функции или её объявления на несколько строк, каждый параметр должен располагаться на отдельной строке; строки, начиная со второй, должны иметь отступ на единицу выше строки вызова (объявления).
   Пример. Следующие фрагменты кода являются допустимыми.

```
cfg = readCfgParams(file,
  section,
  errorHandler);
```

```
cfg = readCfgParams(
  file,
  section,
  errorHandler
);
```

3. При разбиении выражения на несколько строк каждая строка, начиная со второй, должна иметь глубину отступа на единицу выше первой.

2.3. Требования к структуре программы

2.3.1. Состав программы

1. Программа должна представлять собой набор взаимосвязанных функций.
2. Ответ на вопрос, что делает та или иная функция, должен состоять из одной простой фразы, причём из неё должен быть понятен смысл аргументов и возвращаемого значения функции. Если эта фраза не очевидна из сигнатуры функции, то рекомендуется поместить её в комментарий перед функцией.
3. Код каждой функции должен быть понятен без знания кода, эту функцию вызывающего.
4. Код, вызывающий функцию, должен быть понятен без знания кода вызываемой функции.
5. Не рекомендуется давать функциям более трёх параметров.
6. Использование у функции шести и более параметров допустимо только при наличии веского обоснования.
7. Не рекомендуется делать функции длиннее 30–50 строк.
8. Использование функций длинной 100 строк и более возможно только при наличии веского обоснования.

2.3.2. Выбор имён

1. Имена (идентификаторы) должны выбираться в соответствии с тем, для чего они используются.
2. Имена переменных (в том числе функций) не должны совпадать с именами, предоставляемых стандартной библиотекой Луа.
3. Имя переменной не должно содержать указания её типа. **Примечание.** Указание единицы измерения является допустимым, например, `distance_km`, `angle_deg`.
4. Имена переменных и функций должны начинаться с строчной буквы.
5. Имена переменных и функций могут быть либо в `camelCase`, либо в `snake_case`, при этом во всей программе для этого должна использоваться только одна схема именования.
6. При использовании слова естественного языка как части имени допустимо использование только слов английского языка и сокращений от них.
   Пример. Следующие имена являются допустимыми: `my_number`, `num_elements`, `value`, `is_off`.
   Следующие имена являются **недопустимыми**: `mnu_chiselko`, `imya_faila`, `kol_elementov`, `znach`, `is_vykl`.

2.3.3. Языковые конструкции

1. Использование глобальных переменных возможно только при наличии веского обоснования.
2. Использование в выражении трёх и более побочных эффектов возможно только при наличии веского обоснования.

3. Запрещено использование более одного побочного эффекта в логических выражениях.
4. Использование вложенности блоков уровня четыре и выше возможно только при наличии веского обоснования.
5. Переменные должны объявляться (и, если возможно, инициализироваться) как можно ближе к месту своего первого использования.
6. Внутри тела цикла `for` запрещено менять значение счётчика цикла или итератора.
7. Использование оператора безусловного перехода (`goto`) допустимо только в следующих случаях.
   (a) Для досрочного выхода из двойных, тройных и более вложенных циклов.
   (b) Для организации освобождения выделенных в начале блока ресурсов (памяти, файлов, ...).
8. Не допускается использование числовых констант, отличных от 0 и 1.
9. Обращение к полю таблицы, чьё имя известно до выполнения, должно быть через оператор выбора поля (точку). Оператор индексирования (квадратные скобки) должен использоваться лишь для ключей, которые не могут являться идентификаторами, либо для ключей, чьё значение известно только на этапе выполнения.
10. Обход элементов последовательности (таблицы с последовательными натуральными ключами, начиная от 1) должен осуществляться через числовой вариант оператора `for` (т.е. не через генератор `ipairs`).
11. Код программы не должен дублировать задачи, решаемые стандартной библиотекой Луа.

2.3.4. Комментарии

1. Комментарии должны использоваться для описания вещей, не очевидных непосредственно из кода.
   Пример. При объявлении функции комментарии могут описывать следующие аспекты её аргументов:
   - Единица измерения, если она есть (пиксель, байт, килобайт,...).
   - Свойства, которые обязаны выполняться для корректной работы функции. (Например, ограничения на диапазон принимаемых значений для скалярных типов.)
2. Комментарии не должны дублировать код.

Список литературы

[1] Столяров А.В. Оформление программного кода. Макс-Пресс, Москва, 2019.

[2] Ousterhaut J. A philosophy of software design. Yaknyam Press, Palo Alto, 2018.

[3] McConnel S. Code complete. Microsoft Press, 2004. Перевод: Макконел С. Совершенный код. Русская редакция, Москва, 2010.

[4] Lua 5.4 Reference Manual. https://lua.org/manual/5.4/