A Toy Language Project

fun & learning

https://github.com/arkenidar/structure-js

The challenge that started it all

I asked myself whether I could build somehow a custom programming language. The only constraint posed was “make it solve the fizz-buzz game”.

I made it in Lua, while learning Lua. The project was dubbed Pang as a codename for a rewrite (previous attempts were not satisfying). So “LuaPang” reached several goals:

LuaPang solved Fizz-Buzz game test

This means in this code there is a way to have:

1. variables
2. loops
3. multiple if
4. formulas
5. etc

Other results

I managed to make it support function definition with function parameters, local variables, a call stack, a return statement, support for recursion in recursive functions.

And another result was to make it use associative arrays and strings, basically for a “count the vowels” game.

To keep it simple I added all the necessary for the goals and nothing more, but the features are still generic for what the can do, they can do it for other purposes. Anyway source code availability make it possible to add the missing parts.

Script and REPL

The LuaPang project then was chosen to embody the script+REPL tradition so files containing programs are executed within or without a ReadEvaluatePrintLoop.

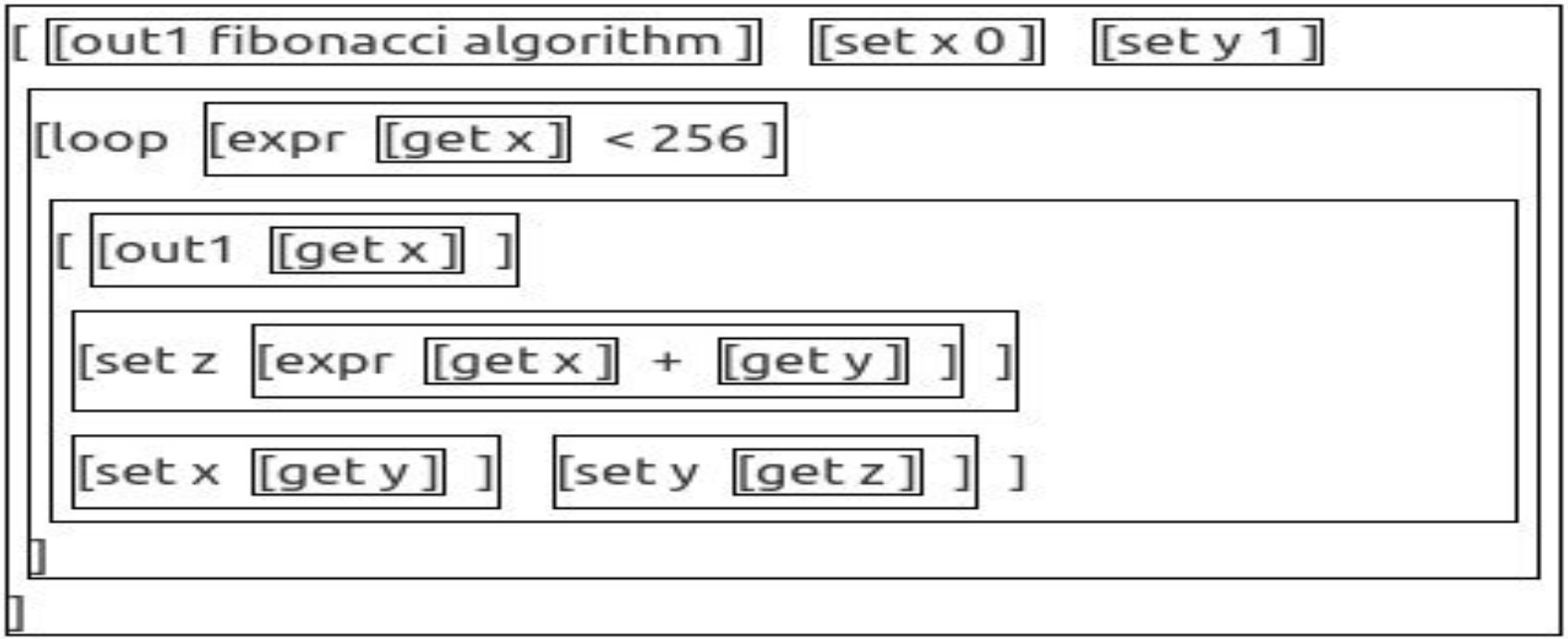
I used also a way to superficially translate Pang words in Italian so I made some scripts containing only Italian language. This could be relatively readable but I used Polish-Notation from the beginning so this choice counts. No parenthesis used in expressions and other peculiar details attracted interest as a relative novelty

How it may look (textually)

```
out1|fibonacci algorithm|
    set|x|0|set|y|1|
    loop|expr|get|x|<|256|
    begin|
        out1|get|x|
        set|z|expr|get|x|+|get|y|
        set|x|get|y|
        set|y|get|z|
        end|
```

https://arkenidar.github.io/tree-structure/public/

How it may look (diagram)



Graphical Interface

A graphical interface was planned in DHTML so I translated LuaPang in TypescriptPang, then I rewrote TypescriptPang in TypescriptStructure, a project that is aimed to allow the list-of-lists structure as a way to store intermediate programs, used during program execution and derived from polish-notation based textual source code. The amout of novelty required for the graphical interface goals seems too hard to be fun and for lack of a simpler path the “program-tree graphical interface” still awaits to be explored.

This was part of an effort to have a really graphical program editor, but source code still plays a big big role. I envisioned a two-way transformation from one to the other but this too has unsolved design challenges. I am the only author and this is meant to be a Toy Language, possibly useful as excercise and learning.

Structure of a js-structure program (in this case “fibonacci test”)

https://arkenidar.com/code/pang/fibonacci.structure.txt

how it may look (textually)

```
def_func|factorial|1|recursive-definition|

  out1|factorial of 4:|out1|factorial|4|

  def_func|"factorial"|1|
      if3|expr|arg|0|==|0|1|
      expr|arg|0|*|factorial|expr|arg|0|-|1|
```

… and more!

See the demos/tests in the source code, see my github. But some of them are obsoleted as I grew the language, but they still have historical interest.
https://github.com/arkenidar/structure-js