Specification document

Problem to be solved

A reasonably performing programming language interpreter, for a nice functional language.

Targets

- Functional
- Light syntax
- Mostly immutable
- Macro system

Example

The language **might** look something like this:

```
stuff = [42 36 72]
add = (arg1, arg2) ->
    arg1 + arg2
main = (args) ->
    print(reduce(add, stuff)) # 150
    for range(100), (i) ->
        print(i)              # 0..99
    length(args) == 0 ?
        print("No args")
    !?
        printf("Args: {}", args)
```

Algorithms and data structures

Data structure	For	Time Complexity	Space Complexity
Hash Table	Scoping, use in programming	mostly `O(1)`	`O(n)`
Dynamic Array	Use in programming	mostly `O(n)`	`O(n)`
Stack	Programs run on a stack	`O(1)`	`O(n)`
Some tree	Representation of source, programming	`O(log(n))`	`O(n)`
Heap	Use in programming	`O(log(n))`	`O(n)`

Algorithm	For	Time Complexity	Space Complexity
Pratt parser	Creates an AST tree from the tokens	LL(1) `O(n)`	`O(n)`
Some tokenizer	Creates tokens from the source code	`O(n)`	`O(n)`
Some GC	For collecting garbage	Not sure	Not sure
Some sort	For programming	`O(n log(n))`	Not sure