Op language (unfinished yet)

What is Op ?

Op is a language based on operator.
I call it operator but in fact you can see them just as function with infix notation.
It's a project used to learn more about lexing, parsing, interpreting, compiling and isn't focused on being practical.
For now, Op is a interpreted language in Ocaml.
I'm open to all kind of thought you can have on this project, you can contact me on :
email : akskwyx@gmail.com
discord : .skwyx

Objectives (features) :

- Make a usable language
- Being able to set precendence of operators
- Being able to create operators with infix notation and use regex to declare operator (see : examples)
- (A feature to add and remove parameters and so change arity of an operator is an idea but isn't a priority and at state of pure thinking for now)

Examples

read notice for detail (not done yet)

Fibonnaci function

```
fibo n -> n <= 1 ? 1 : fibo (n-1) + fibo (n-2)
```

Iter function

```
iter <- i <- i+1
k <- 0
iter >> k ** k <= 5
```

// Declare iter function
// Initialize variable k to 0
// While k is lower or equal to 5, iter k

Thought example

```
{ (\[a\]+*\[b\])+ } <- ((
    + !< 2
    \* !< 3
```

```
    a.i + b.i * a.(i+1) + b.(i+1) ** i < (a.len - 1)
))

var <- 1 +* 2 +* 3 +* 4
$var
```

// Declare a regex operator with the pattern (value +* value)+
// Reassign '+' precedence to 2 (3 by default)
// Reassign '' *precedence to 3 (2 by default)*
// So now + has a higher priority than
// Sum each couple of a and b and multiply all of them
// Call it and put the result in var
// Print the content of var : 1 + 2 * 3 + 4 = 3 * 7 = 21 (don't forget assignment of precedence)

TODO :

- Project :
  - Add documentation
- Parsing :
  - Change list type to enable adding at end in O(1)
  - Add panic mode to parser to handle multiple error