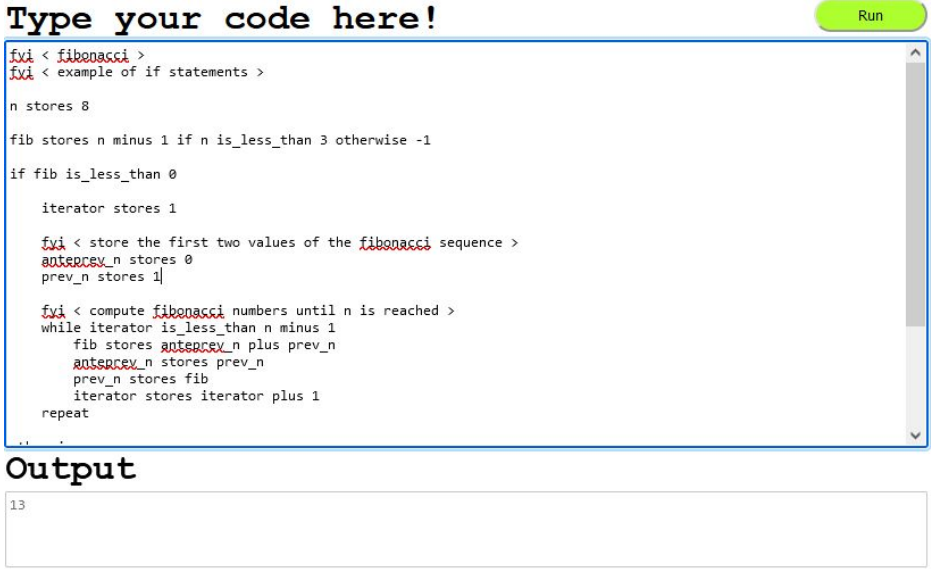
Codeable

A sleek language written and directed by Joseph Hale, Jacob Janes, Rithvik Arun, Sai Nishanth Vaka, and Jacob Hreshchyshyn

Copyright © 2022 Arizona Board of Regents

Language

- Codeable utilizes a non-technical language that can be understood and written by people with minimal coding knowledge.
- For loops are called by “For i from 0 to 5”
- Comments are tagged with “FYI”
- Variables can be identified with the keyword “stores”
- Simplicity is embraced, yet the language is still turing complete



Copyright © 2022 Arizona Board of Regents

Grammar

The grammar we built is easy to understand and simple to read.

Our grammar includes:

Grammar	**Examples**
command	Commands contain comments, assignment, loop, show, selection
program	Programs contain commands
boolean	r is_less_than n minus 1, r is greater_than n minus 1
number	1,2,3,4,5,6,7,8,9
expression	a stores x plus y, a stores x minus y, a stores x times y, a stores x divided_by y

Copyright © 2022 Arizona Board of Regents

Grammar Cont.

string	<Codeable>
assignment	n stores 8, a stores <Codeable>
ternary	if n is_less_than 3 otherwise -1
if_statement	if n is_less_than 3 otherwise
loop_for	for i from 0 to a by 1 repeat
loop_while	while iterator is_less_than n minus 1 repeat
print	show <Codeable>
comment	fyi < uses i as an iterator for a >

Copyright © 2022 Arizona Board of Regents

Compiler

- When we are trying to compile Codeable, we use Node JS to tokenize our language, which is fed into Tau Prolog.

- Tau is where our tokenized list is fed into our Prolog interpreter, and we get our results fed back into JS.

```
program(P, [fyi,<,fibonacci,>,fyi,
<,example,of,if,statements,>,n,stores,8,fib,stores,n,minus,1,if,n,is_less_than,3,otherwise,-
1,if,fib,is_less_than,0,iterator,stores,1,fyi,
<,store,the,first,two,values,of,the,fibonacci,sequence,>,anteprev_n,stores,0,prev_n,stores,1,fyi,
<,compute,fibonacci,numbers,until,n,is,reached,>,while,iterator,is_less_than,n,minus,1,fib,stores,anteprev_n,plus,p
rev_n,anteprev_n,stores,prev_n,prev_n,stores,fib,iterator,stores,iterator,plus,1,repeat,otherwise,message,stores,
<,try,a,higher,value,of,n,for,more,fun,>,show,message,move_on,show,fib], []), write(P), eval(P, [], EnvOut,
ValueOut).
```

This is a figure of our TAU compiler with a tokenized list provided by our JS front end.

Copyright © 2022 Arizona Board of Regents

Intermediate Form

- Both the parser, and the evaluator are completely in prolog.
- Due to the parser outputting a parse tree from a list of tokens, our intermediate form is actually our output from the parser.

The following is a parse tree provided by our compiler, and it is then fed into our prolog evaluator to feed a result back to JS.

```
prog(cmd(fyi(fibonacci),cmd(fyi(example of if
statements),cmd(assign(n,expr_term(term_factor(factor_numeric(8)))),cmd(assign(fib,ternary(is_less_than(expr_term(t
erm_factor(factor_identifier(n))),expr_term(term_factor(factor_numeric(3)))),expr_minus(term_factor(factor_identifi
er(n)),expr_term(term_factor(factor_numeric(1)))),expr_term(term_factor(factor_numeric(-1))))),cmd(if(is_less_than(
expr_term(term_factor(factor_identifier(fib))),expr_term(term_factor(factor_numeric(0)))),cmd(assign(iterator,expr_
term(term_factor(factor_numeric(1)))),cmd(fyi(store the first two values of the fibonacci
sequence),cmd(assign(anteprev_n,expr_term(term_factor(factor_numeric(0)))),cmd(assign(prev_n,expr_term(term_factor(
factor_numeric(1)))),cmd(fyi(compute fibonacci numbers until n is
reached),while(is_less_than(expr_term(term_factor(factor_identifier(iterator))),expr_minus(term_factor(factor_ident
ifier(n)),expr_term(term_factor(factor_numeric(1))))),cmd(assign(fib,expr_plus(term_factor(factor_identifier(antepr
ev_n)),expr_term(term_factor(factor_identifier(prev_n))))),cmd(assign(anteprev_n,expr_term(term_factor(factor_ident
ifier(prev_n)))),cmd(assign(prev_n,expr_term(term_factor(factor_identifier(fib)))),assign(iterator,expr_plus(term_f
actor(factor_identifier(iterator)),expr_term(term_factor(factor_numeric(1))))))))))),cmd(assign(message,try a
higher value of n for more fun),show_identifier(message))),show_identifier(fib)))))))
```

Copyright © 2022 Arizona Board of Regents

Runtime

- Our runtime environment is produced by JS interfacing with Tau Prolog

- When we provide a CLI command, it reads our file, produces a parse tree based on the provided Codable code, and outputs to the terminal.

```
PS C:\...\Documents\...\GitHub\SER502-Spring2022-Team15\src\runtime> node .\codeable_cli.js "../../data/fibonacci.code"
[INFO] Loaded interpreter source code
Successfully parsed query!
[output,try a higher value of n for more fun][output,1]
PS C:\...\Documents\...\GitHub\SER502-Spring2022-Team15\src\runtime> node .\codeable_cli.js "../../data/factorial.code"
[INFO] Loaded interpreter source code
Successfully parsed query!
[output,24]
```

Copyright © 2022 Arizona Board of Regents