Brian Cullinan

Project 5: Advanced Language Design

10/08/08

My approach remains the same, to create a small syntax language. All the keywords are 1 or 2 characters long. The code will eventually compile into PHP code which can then be interpreted automatically. My new language spec includes lots of new features. The features as listed on the site, as well as arrays. I believe arrays will be easy to implement because of the way PHP handles arrays.

Spec

// language spec

// The objective of my language is to be as shorthanded as possible.

Program ::= Command-List

Command-List ::= Command EOL Command-List

| Command EOL

Command ::= Assignment

| Declaration

| For-Loop

| While-Loop

| If-Statement

| Function-Def

| Function-Call

| Return-Call

| p Expression // internal print functionality

| Identifier ( Expression )

Assignment ::= Identifier = Expression

Expression ::= Literal

| V-Name

| Operator Expression

| ( Expression )

| Logical Expression

Function-Def ::= f F-Name(Param-Def)[Command-List]

Function-Call ::= F-Name Param-Def

Return-Call ::= r Expression // returns to line after call just like in assembly

F-Name ::= Identifier

Param-Def ::= Identifier

| Identifier, Param-Def

For-Loop ::= fo(Assignment,Expression,Assignment)[Command-List]

While-Loop ::= w(Expression)[Command-List]

If-Statement ::= if(Expression)[]

| if(Expression)[]el[]

| if(Expression)[]el If-Statement

V-Name ::= Identifier

::= Identifier[Integer] // list reference

Declaration ::= v Assignment // global var

| c Assignment // constant

| l Assignment // local var

Type ::= Identifier

Operator ::= + | - | \* | / | ^

Logical ::= < | > | == | <= | >= | || | && | !

Identifier ::= Letter | Identifier Letter | Identifier Digit

Literal ::= i Integer-Literal

| r Real-Literal

| h Hex-Literal

| b Binary-Literal

Integer-Literal ::= Digit | Integer-Literal Digit

Real-Literal ::= Integer-Literal . Integer-Literal

Hex-Literal ::= [A-Z0-9]

Binary-Literal ::= 1

| 0

| 1 Binary-Literal

| 0 Binary-Literal

Comment ::= # Letter EOL

Code Sample (prints out Fibonacci numbers)

v fib1 = 0

v fib2 = 1

fo(i = 0, i < 100, i = i + 1)

[

print fib1

print fib2

fib1 = fib1 + fib2

l newfib = fib2 + fib1

fib2 = newfib

]

if(fib1 > 5)[

print fib1

]el[

p fib2

]

f print(pvar)[

p pvar #prints out 5.08

]