Reggie Fisher

Programming Languages 403

**CLOX INTERPRETER DOCUMENTATION**

**INTRODUCTION**

Below is a testing and assessment of my lox interpreter that I have built using c++. The functionality and operation of the interpreter is similar to the walkthrough provided in the book “Crafting Interpreters”. To understand how to build and run this program, check out the README file that was provided within the same directory as this file.

**TESTING OF THE INTERPRETER**

Below are test cases that I have used to ensure that my interpreter works as expected. The code will be given in blue and the expected results are given in red. I have also split the tests cases into categories for better understanding.

**TESTS**

1. **Basic Arithmetic and Logical Operations**

* Addition: print 5 + 3;
  + Expected Output: ‘8’
* Addition: print 20 + 7 + 1;
  + Expected Output: ‘28’
* Subtraction: print 11 – 15;
  + Expected Output: ‘-4’
* Subtraction: print 15 - 8 - 5;
  + Expected Output: ‘2’
* Multiplication: print 4 \* 2;
  + Expected Output: ‘8’
* Multiplication: print 11 \* 4 \* 2;
  + Expected Output: ‘88’
* Division: print 8 / 2;
  + Expected Output: ‘4’
* Equality: print 12 == 12;
  + Expected Output: ‘true’
* Mixed Expression: print 5 \* 10 - 2;
  + Expected Output: ‘48’
* Mixed Expression: print 3 + 6 \* 11 - 1;
  + Expected Output: ‘68’
* Logical Operations: print true and false;
  + Expected Output: ‘false’
* Logical Operations: print true or false;
  + Expected Output: ‘true’

1. **Variables and Scoping**

* Variable Declaration:

var a = 5;

print a;

* + Expected Output: ‘5’
* Variable Re-Assignment:

var b = 5;

b = b + 2;

print b;

* + Expected Output: ‘7’
* Scope Test:

{

var c = "first";

print c; // expect: first

}

{

var c = "second";

print c;

}

* + Expected Output: first

Second

* Scope Test:

{

var a = "outer";

{

print a; //expect: outer

var a = "inner";

print a; //expect: inner

}

}

* + Expected Output: outer

inner

1. **Control Structures**

* If – else:

var condition = false;

if (condition) {

print "It's true!";

} else {

print "It's false!"; //Expect: It's false!

}

* + Expected Output: ‘It’s false’
* if – else:

var score = 85;

var grade = "";

if (score >= 90) {

grade = "A";

} else if (score >= 80) {

grade = "B";

} else if (score >= 70) {

grade = "C";

} else {

grade = "D";

}

print "Your grade is " + grade;

* + Expected Output: ‘Your grade is B’
* While Loop:

var a = 0;

while (a < 3) {

print a;

a = a + 1;

}

* + Expected Output: ‘0 1 2’
* For Loop:

for (var i = 0; i < 5; i = i + 1) {

print i;

}

* + Expected Output: ‘0 1 2 3 4’

1. **Functions**

* Function Definition and Call:

fun add(a, b) {

return a + b;

}

print add(3, 4);

* + Expected Output: ‘7’
* Recursive Function:

fun fibonacci(n) {

if (n <= 0) return 0;

if (n == 1) return 1;

return fibonacci(n - 1) + fibonacci(n - 2);

}

// Calculate and print the 10th Fibonacci number

var result = fibonacci(10);

print result;

* + Expected Output: ‘55’

1. **Classes and Inheritance**

* Class Definition and Object Creation:

class Box {

init(value) {

this.value = value;

}

}

var b = Box(10);

print b.value;

* + Expected Output: ‘10’
* Inheritance:

class Foo {

methodOnFoo() { print "foo"; }

override() { print "foo"; }

}

class Bar < Foo {

methodOnBar() { print "bar"; }

override() { print "bar"; }

}

var bar = Bar();

bar.methodOnFoo(); // expect: foo

bar.methodOnBar(); // expect: bar

bar.override(); // expect: bar

* + Expected Output:

‘foo’

‘bar’

‘bar’

1. **Error Handling**

* Division by Zero: print 5 / 0;
  + Expected Output: ‘inf’
* Undefined Variable: print x;
  + Expected Output:

Undefined variable ‘x’.

[line 1]

* Missing semicolon: print 5 + 10
  + Expected Output:

[line 1] Error at end: Expect ‘;’ after value.

* Missing if condition:

if {

print "Cookies";

}

* + Expected Output:

[line 1] Error at ‘{‘: Expect ‘(‘ after ‘if’.

[line 3] Error at ‘}’: Expect expression.

* Missing for loop condition: for (var a = 1; {}; a = a + 1) {}
  + Expected Output:

[line 1] Error at ‘{‘: Expect expression.

[line 1] Error at ‘)’: Expect ‘;’ after expression.

* Undefined function call: add(5, 2);
  + Expected Output:

Undefined variable ‘add’.

[line 1]

* Class inherits itself: class Foo < Foo {}
  + Expected Output:

[line 1] Error at ‘Foo’: A class can’t inherit from itself.

* Stack overflow:

fun foo() {

var a1;

var a2;

var a3;

var a4;

var a5;

var a6;

var a7;

var a8;

var a9;

var a10;

var a11;

var a12;

var a13;

var a14;

var a15;

var a16;

foo(); // expect runtime error: Stack overflow.

}

foo();

* + Expected Output:

Segmentation fault (core dumped)

1. **Edge Cases**

* Empty Block: { }
  + Expected Output: No output
* String Concatenation: print “Hello, “ + “world!”;
  + Expected Output: Hello, world!

**Weaknesses**

After fulling testing my interpreter, I was unable to find any weaknesses within my implementation. All functions and test cases work as expected.