Lab 3 Semantic Analysis

Sorin Macaluso

April 7, 2024

1 Test Cases

These are a few repeat test cases from the other examples as well as some of the ones from the lab itself. This will only show semantic analysis output since the lexer and the parser have already been established. All test cases where ran individually since there would be to much output to grab just the semantic stuff. Test number 9 is proof that I can work all together.

1.1 Test Case 1

```
/*1 valid*/
/*Long Test Case - Everything Except Boolean Declaration */
{
  /* Int Declaration */
  int a
  int b
  a = 0
  b=0
  /* While Loop */
  while (a != 3) {
    print(a)
    while (b != 3) {
      print(b)
      b = 1 + b
      if (b == 2) {
        /* Print Statement */
        print("there is no spoon" /* This will do nothing */ )
      }
    }
    b = 0
    a = 1+a
  }
}
$

Semantic Analysis starting :)
AST for Program 1 Done
<block>

-<var_decl>

--[int]

```

```
--[a]

-<var_decl>

--[int]

--[b]

-<assignment_statment>

--[a]

--[0]

-<assignment_statment>

--[b]

--[0]

-<while_statment>

--[while]

--[(]

--[a]

--[!=]

--[3]

--[)]

--<block>

---<print_statment>

----[print]

----[a]

---<while_statment>

----[while]

----[(]

----[b]

----[!=]

----[3]

----[)]

----<block>

-----<print_statment>

------[print]

------[b]

-----<assignment_statment>

------[b]

------<+>

-------[1]

-------[b]

------<if_statment>

-------[if]

-------[(]

-------[b]

-------[==]

-------[2]

-------[)]

-------<block>

--------<print_statment>

---------[print]

---------["there is no spoon"]

-<assignment_statment>

--[b]

--[0]

-<assignment_statment>

--[a]

--<+>

```

```
---[1]

---[a]

-[$]

Scope 0:
Variable Name: a, Type: int, IsUsed: true, IsInitialized: true
Variable Name: b, Type: int, IsUsed: true, IsInitialized: true
End of program 1
```

1.2 Test Case 2

```
/*2 valid*/
{
    int a
    boolean b
    {
        string c
        a = 5
        b = true /* no comment */
        c = "inta"
        print(c)
    }
    print(b)
    print(a)
}$

Semantic Analysis starting :)
AST for Program 1 Done
<block>

-<var_decl>

--[int]

--[a]

-<var_decl>

--[boolean]

--[b]

-<block>

--<var_decl>

---[string]

---[c]

--<assignment_statment>
```

```

---[a]

---[5]

--<assignment_statment>

---[b]

---[true]

--<assignment_statment>

---[c]

---["inta"]

--<print_statment>

---[print]

---[c]

-<print_statment>

--[print]

--[b]

-<print_statment>

--[print]

--[a]

-[$]

Scope 0:
Variable Name: a, Type: int, IsUsed: true, IsInitialized: true
Variable Name: b, Type: boolean, IsUsed: true, IsInitialized: true
Scope 1:
Variable Name: c, Type: string, IsUsed: true, IsInitialized: true
End of program 1
```

1.3 Test Case 3

```
/*3 invalid (b not in scope)*/
{
    int a
    {
        boolean b
        a = 1
    }
    print(b)
}$

Semantic Analysis starting :)
AST for Program 1 Done
<block>

-<var_decl>

--[int]

--[a]

-<block>

--<var_decl>

---[boolean]

---[b]

--<assignment_statment>

---[a]

---[1]

-<print_statment>

--[print]

--[b]

-[$]

Variable b line num 8 used in print statement not found.
Scope 0:
Variable Name: a, Type: int, IsUsed: true, IsInitialized: true
End of program 1
```

1.4 Test Case 4

```
/*4 valid proof of scope manuvering*/
{
int a
{
    boolean b
    {
        string c
        {
            a = 5
            b = false
            c = "inta"
        }
        print(c)
        }
    print(b)
    }
print(a)
}$

Semantic Analysis starting :)
AST for Program 1 Done
<block>

-<var_decl>

--[int]

--[a]

-<block>

--<var_decl>

---[boolean]

---[b]

--<block>

---<var_decl>

----[string]

----[c]

---<block>

----<assignment_statment>

-----[a]

-----[5]

----<assignment_statment>

-----[b]

-----[false]

----<assignment_statment>

-----[c]

-----["inta"]

```

```
-<print_statment>

--[print]

--[c]

-<print_statment>

--[print]

--[b]

-<print_statment>

--[print]

--[a]

-[$]

Scope 0:
Variable Name: a, Type: int, IsUsed: true, IsInitialized: true
Scope 1:
Variable Name: b, Type: boolean, IsUsed: true, IsInitialized: true
Scope 2:
Variable Name: c, Type: string, IsUsed: true, IsInitialized: true
End of program 1
```

1.5 Test Case 5

```
/*5 valid*/
{
    boolean a
    a = (1 == (2 == 3))
}$

Semantic Analysis starting :)
AST for Program 1 Done
<block>

-<var_decl>

--[boolean]

--[a]

-<assignment_statment>

--[a]

--[(]

--[1]
```

```

--[==]

--[(]

--[2]

--[==]

--[3]

--[)]

--[)]

-[$]

Scope 0:
Variable Name: a, Type: boolean, IsUsed: true, IsInitialized: true
End of program 1
```

1.6 Test Case 6

```
/*6 invalid type missmatch error*/
{
    string a
    a = 2
}$

Semantic Analysis starting :)
AST for Program 1 Done
<block>

-<var_decl>

--[string]

--[a]

-<assignment_statment>

--[a]

--[2]

-[$]

Type mis-match error at 4 at 9 declared string but comparing int.
Scope 0:
Variable Name: a, Type: string, IsUsed: true, IsInitialized: false
End of program 1
```

1.7 Test Case 7

```
/*7 valid (checking for boolop stuff inside while)*/
{
    while((1 == (2 == 3)) != 1){
        print("hello")
    }
}$

Semantic Analysis starting :)
AST for Program 1 Done
<block>

-<while_statment>

--[while]

--[(]

--[(]

--[1]

--[==]

--[(]

--[2]

--[==]

--[3]

--[)]

--[)]

--[!=]

--[1]

--[)]

--<block>

---<print_statment>

----[print]

----["hello"]

-[$]

End of program 1
```

```

/*no print out since there are no varabils in this program*/
```

1.8 Test Case 8

```
/*8 valid (checking for boolop stuff inside if)*/
{
    if((1 == (2 == 3)) != 1){
        print("hello")
    }
}$

Semantic Analysis starting :)
AST for Program 1 Done
<block>

-<if_statment>

--[if]

--[(]

--[(]

--[1]

--[==]

--[(]

--[2]

--[==]

--[3]

--[)]

--[)]

--[!=]

--[1]

--[)]

--<block>

---<print_statment>

----[print]

----["hello"]
```

```

-[$]

End of program 1

/*no print out since there are no varabils in this program*/
```

1.9 Test Case 9

```
{
    int a
    a = 3
}$

{
    int a
    a = 4
}$

{
    int a
    a = 5
}$

Semantic Analysis starting :)
AST for Program 1 Done
<block>

-<var_decl>

--[int]

--[a]

-<assignment_statment>

--[a]

--[3]

-[$]

Scope 0:
Variable Name: a, Type: int, IsUsed: true, IsInitialized: true
End of program 1

Semantic Analysis starting :)
AST for Program 2 Done
<block>

-<var_decl>

--[int]
```

```

--[a]

-<assignment_statment>

--[a]

--[4]

-[$]

Scope 0:
Variable Name: a, Type: int, IsUsed: true, IsInitialized: true
End of program 2

Semantic Analysis starting :)
AST for Program 3 Done
<block>

-<var_decl>

--[int]

--[a]

-<assignment_statment>

--[a]

--[5]

-[$]

Scope 0:
Variable Name: a, Type: int, IsUsed: true, IsInitialized: true
End of program 3
```

2 References

Java Regular Expression
Enum
Switch cases (Was more of an idea)
Double Quote For Regular Expressions
Groups in Regular Expressions
More Group stuff
Index for Regular Expressions
Check for end of line
Full Syntax Java Regx
Quote In Regx

Links that where already used/shown in previous lab reports. For semantic analysis I was either using the stuff that was mentioned in the slides or having to build my own thing completely. Especially the semantic analysis, traversing through that is something that I completely made on my own.