

CMSC 430 Project 4

Author: Tyler D Clark

Date: 12 December 2021

Description The fourth project involves modifying the semantic analyzer for the attached compiler by adding checks for semantic errors.

Approach

Like the previous projects, I started by reading up on the reading materials for the assignment. Again, I watched all the videos as well to get a good understanding before continuing. I also read up on the assignment and the project requirements. I began by transferring over code from the previous project and comparing it against the skeleton code provided for the project. I then started to modify the code to fit the project requirements and lastly wrote tests to ensure that the code was correct.

Test cases

The test case were designed based on the assignment rubric

Test case 1

```
-- Tests for remainder with non-integer operands.  
  
function test1 a: integer returns boolean;  
begin  
    a rem 2.0;  
end;
```

Output:

```
tylerclark@MacBook-Pro:~/Repos/CMSC430/Project4
tylerclark ~/.../CMSC430/Project4
λ ./compile < tests/test1.txt (main ?)

1  -- Tests for remainder with non-integer operands.
2
3  function test1 a: integer returns boolean;
4  begin
5    a rem 2.0;
Semantic Error, Integer Type Required
    end;
Lexical Errors: 0
Syntax Errors: 0
Semantic Errors: 1
tylerclark ~/.../CMSC430/Project4
λ (main ?)
```

Test case 2

```
-- Generates semantic error when if and then types don't
match
-- Generates semantic error for return type mismatch

function test2 a: boolean returns boolean;
  b: integer is 1;
  c: real is 1.0;

begin
  if a then
    b;
  else
    c;
  endif;
end;
```

Output:

```

tylerclark@MacBook-Pro:~/Repos/CMSC430/Project4
tylerclark ~/.../CMSC430/Project4
λ ./compile < tests/test2.txt (main ?)

1 -- Generates semantic error when if and then types don't match
2 -- Generates semantic error for return type mismatch
3
4 function test2 a: boolean returns boolean;
5     b: integer is 1;
6     c: real is 1.0;
7
8 begin
9     if a then
10         b;
Semantic Error, Type Mismatch on Function Return
11     else
12         c;
Semantic Error, Type Mismatch on Function Return
13     endif;
Semantic Error, Type Mismatch on Then and Else Statements
    end;
Lexical Errors: 0
Syntax Errors: 0
Semantic Errors: 3
tylerclark ~/.../CMSC430/Project4
λ (main ?)

```

Test Case 3

```

-- Generates semantic error when case types don't match
-- Case Expression Not Integer

function test3 a: real returns boolean;
begin
    case a is
        when 1 => true;
        when 2 => 1;
        others => 1.2;
    endcase;
end;

```

Output:

```

tylerclark@MacBook-Pro:~/Repos/CMSC430/Project4
tylerclark ~/.../CMSC430/Project4
λ ./compile < tests/test3.txt (main ?)

1  -- Generates semantic error when case types don't match
2  -- Case Expression Not Integer
3
4  function test3 a: real returns boolean;
5  begin
6    case a is
Semantic Error, Case Expression must be Integer Type
7      when 1 ⇒ true;
8      when 2 ⇒ 1;
Semantic Error, Type Mismatch on Function Return
Semantic Error, Type Mismatch on When Statement
9      others ⇒ 1.2;
Semantic Error, Type Mismatch on Function Return
10     endcase;
Semantic Error, Type Mismatch on Others Statement
    end;
Lexical Errors: 0
Syntax Errors: 0
Semantic Errors: 5
tylerclark ~/.../CMSC430/Project4
λ (main ?)

```

Test Case 4

```

-- Generates semantic error when if condition is not Boolean

function test4 a: integer returns boolean;

begin
  if a then
    true;
  else
    false;
  endif;
end;

```

Output:

```
tylerclark@MacBook-Pro:~/Repos/CMSC430/Project4
tylerclark ~/ ... /CMSC430/Project4
λ ./compile < tests/test4.txt (main ?)

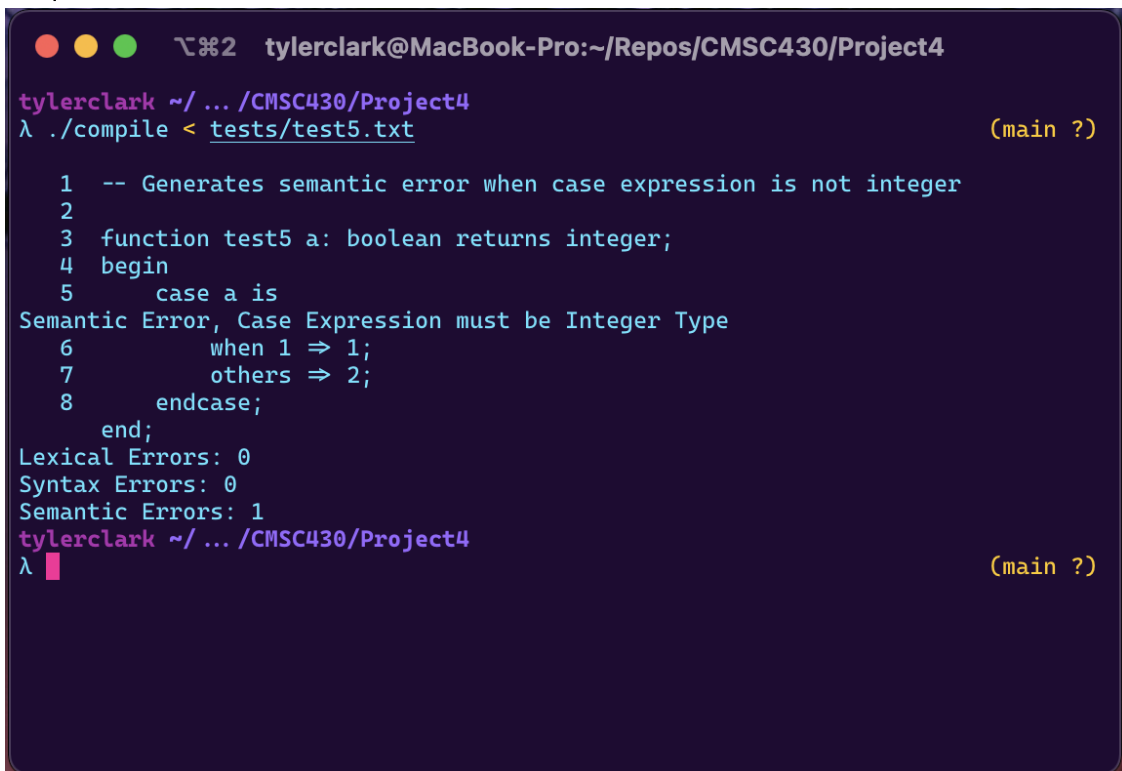
1  -- Generates semantic error when if condition is not Boolean
2
3  function test4 a: integer returns boolean;
4
5  begin
6    if a then
Semantic Error, If Condition must be Boolean Type
7      true;
8    else
9      false;
10   endif;
    end;
Lexical Errors: 0
Syntax Errors: 0
Semantic Errors: 1
tylerclark ~/ ... /CMSC430/Project4
λ (main ?)
```

Test Case 5

```
-- Generates semantic error when case expression is not
integer

function test5 a: boolean returns integer;
begin
  case a is
    when 1 => 1;
    others => 2;
  endcase;
end;
```

Output:



```
tylerclark@MacBook-Pro:~/Repos/CMSC430/Project4
tylerclark ~/.../CMSC430/Project4
λ ./compile < tests/test5.txt (main ?)

1  -- Generates semantic error when case expression is not integer
2
3  function test5 a: boolean returns integer;
4  begin
5      case a is
Semantic Error, Case Expression must be Integer Type
6          when 1 ⇒ 1;
7          others ⇒ 2;
8      endcase;
    end;
Lexical Errors: 0
Syntax Errors: 0
Semantic Errors: 1
tylerclark ~/.../CMSC430/Project4
λ (main ?)
```

Test Case 6

```
-- Generates semantic error on narrowing initialization

function test6 a: real returns integer;
    b: integer is a;
begin
    b;
end;
```

Output:

```
tylerclark@MacBook-Pro:~/Repos/CMSC430/Project4
tylerclark ~/.../CMSC430/Project4
λ ./compile < tests/test6.txt (main ?)

1 -- Generates semantic error on narrowing initialization
2
3 function test6 a: real returns integer;
4   b: integer is a;
Semantic Error, Narrowing Variable Initialization is Illegal
5 begin
6   b;
   end;
Lexical Errors: 0
Syntax Errors: 0
Semantic Errors: 1
tylerclark ~/.../CMSC430/Project4
λ (main ?)
```

Test Case 7

```
-- Generates semantic error for duplicate variables

function test7 a: integer returns integer;
  a: integer is 1;
begin
  a;
end;
```

Output:

```
tylerclark@MacBook-Pro:~/Repos/CMSC430/Project4
tylerclark ~/ ... /CMSC430/Project4
λ ./compile < tests/test7.txt (main ?)

1 -- Generates semantic error for duplicate variables
2
3 function test7 a: integer returns integer;
4     a: integer is 1;
Semantic Error, Duplicate Identifier: a
5 begin
6     a;
7 end;

Lexical Errors: 0
Syntax Errors: 0
Semantic Errors: 1
tylerclark ~/ ... /CMSC430/Project4
λ (main ?)
```

Test Case 8

```
-- Tests for undeclared identifiers

function test8 a: integer returns integer;
b: integer is a + c;
begin
    b;
end;
```


Output:

```

tylerclark@MacBook-Pro:~/Repos/CMSC430/Project4
tylerclark ~/.../CMSC430/Project4
λ ./compile < tests/test8.txt (main ?)

1 -- Tests for undeclared identifiers
2
3 function test8 a: integer returns integer;
4 b: integer is a + c;
Semantic Error, Undeclared c
5 begin
6     b;
    end;
Lexical Errors: 0
Syntax Errors: 0
Semantic Errors: 1
tylerclark ~/.../CMSC430/Project4
λ (main ?)

```

Testing Multiple errors

```

-- Test of Multiple Semantic Errors

function test a: integer returns integer;
    b: integer is
        if a + 5 then
            2;
        else
            5;
        endif;
    c: real is 9.8 - 2 + 8;
    d: boolean is 7 = f;
begin
    case b is
        when 1 => 4.5 + c;
        when 2 => b;
        others => c;
    endcase;
end;

```

Output:

```
tylerclark@MacBook-Pro:~/Repos/CMSC430/Project4
λ ./compile < tests/test_multiple.txt (main ?)

1  -- Test of Multiple Semantic Errors
2
3  function test a: integer returns integer;
4      b: integer is
5          if a + 5 then
Semantic Error, If Condition must be Boolean Type
6              2;
7          else
8              5;
9          endif;
10         c: real is 9.8 - 2 + 8;
11         d: boolean is 7 = f;
Semantic Error, Undeclared f
12     begin
13         case b is
14             when 1 ⇒ 4.5 + c;
15             when 2 ⇒ b;
16             others ⇒ c;
Semantic Error, Narrowing Function Return is Illegal
17         endcase;
Semantic Error, Type Mismatch on Others Statement
18     end;
Lexical Errors: 0
Syntax Errors: 0
Semantic Errors: 4
tylerclark ~/.../CMSC430/Project4
λ (main ?)
```

Lessons Learned

For this project, I had to learn to about semantic error checking and how to implement this is bison and C++ code. I did not experience any big issues with this project, but I did get good experience with C++ as well as Flex and Bison. Much of the required code was modelled off of the already provided skeleton code. For instance, `checkIf()` and `checkRemainder()` were modelled after `checkAssignment()` and so on. I had one small error that I stumped me for a while, but it was due to one of my design decisions early on. In `listing.cc`, I was tallying the semantic, syntax and lexical errors based on the `ErrorCategory` enums and I had not properly planned for all cases. I eventually figured out my mistake and moved on. Overall, I worked through all hardships in the project and learned a lot in the process.