CMSC 430 Project 3

Author: Tyler D Clark
Date: 30 November 2021

Description The third project involves modifying the attached interpreter so that it interprets programs for the complete language.

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 then transferred the lexer and the generator code from the previous project to this project. This was so that I could properly view errors, the lexer could identify the correct tokens and the parser could parse the given language. The first changes I made was to fully list all the tokens to be used in
the language in values.h. I then made sure the proper token was being returned in scanner.l and in the case of operators, yylval.oper was being set. Next, I added the code to process each different type of statement and lastly added them to the correct productions in the bison file.

Test cases

Test case 1

The first test case was to test real values, multiple variables, arithmetic operators, and case statements. The following code was used to test the interpreter:

```
-- Tests real evaluation, arithmetic operators, case
expression evaluation, multiple variables.

function test1 a: integer returns real;
    b: real is 12.3 + 4.5;
    c: real is 6.78 - 9.10;
    d: real is 12.3 / 4.5;
begin
    case a is
        when 1 => b * 6.7;
        when 2 => c * (8.9 rem 1.0);
        others => d ** 2;
    endcase;
end;
```

the following output was produced:

```
⌥⌘2 tylerclark@MacBook-Pro:~/Repos/CMSC430/project3
tylerclark ~/ … /CMSC430/project3
λ ./compile < tests/test1.txt 1                                    (main x!?)

   1  -- Tests real evaluation, arithmetic operators, case expression evaluation, multiple variables.
   2
   3  function test1 a: integer returns real;
   4    b: real is 12.3 + 4.5;
   5    c: real is 6.78 - 9.10;
   6    d: real is 12.3 / 4.5;
   7  begin
   8    case a is
   9            when 1 => b * 6.7;
  10            when 2 => c * (8.9 rem 1.0);
  11            others => d ** 2;
  12    endcase;
      end;
Compiled Successfully
Result = 112.56
tylerclark ~/ … /CMSC430/project3
λ ./compile < tests/test1.txt 2                                    (main x!?)

   1  -- Tests real evaluation, arithmetic operators, case expression evaluation, multiple variables.
   2
   3  function test1 a: integer returns real;
   4    b: real is 12.3 + 4.5;
   5    c: real is 6.78 - 9.10;
   6    d: real is 12.3 / 4.5;
   7  begin
   8    case a is
   9            when 1 => b * 6.7;
  10            when 2 => c * (8.9 rem 1.0);
  11            others => d ** 2;
  12    endcase;
      end;
Compiled Successfully
Result = -2.088
tylerclark ~/ … /CMSC430/project3
λ ./compile < tests/test1.txt 3                                    (main x!?)

   1  -- Tests real evaluation, arithmetic operators, case expression evaluation, multiple variables.
   2
   3  function test1 a: integer returns real;
   4    b: real is 12.3 + 4.5;
   5    c: real is 6.78 - 9.10;
   6    d: real is 12.3 / 4.5;
   7  begin
   8    case a is
   9            when 1 => b * 6.7;
  10            when 2 => c * (8.9 rem 1.0);
  11            others => d ** 2;
  12    endcase;
      end;
Compiled Successfully
Result = 7.47111
tylerclark ~/ … /CMSC430/project3
λ                                                                  (main x!?)
```

Test case 2

For the next test case, I tested boolean literals, the not operator and the if expression. I used the following code:

```
-- Tests boolean literal evaluation, not operator, if
expression.

function test2 a: real returns boolean;
```

```
    b: boolean is true;
begin
    if a > 0 then
        b;
    else
        not b;
    endif;
end;
```

The following output was produced:

```
⌥⌘2          tylerclark@MacBook-Pro:~/Repos/CMSC430/project3

tylerclark ~/ … /CMSC430/project3
λ ./compile < tests/test2.txt 25.5                                  (main x!?)

   1  -- Tests boolean literal evaluation, not operator, if expression.
   2
   3  function test2 a: real returns boolean;
   4    b: boolean is true;
   5  begin
   6    if a > 0 then
   7            b;
   8    else
   9            not b;
  10    endif;
      end;
Compiled Successfully
Result = 1
tylerclark ~/ … /CMSC430/project3
λ ./compile < tests/test2.txt -25.5                                 (main x!?)

   1  -- Tests boolean literal evaluation, not operator, if expression.
   2
   3  function test2 a: real returns boolean;
   4    b: boolean is true;
   5  begin
   6    if a > 0 then
   7            b;
   8    else
   9            not b;
  10    endif;
      end;
Compiled Successfully
Result = 0
tylerclark ~/ … /CMSC430/project3
λ                                                                   (main x!?)
```

Test case 3

For the last test case, I tested relation operators, the logical operators and finally multiple parameters. I used the following code:

```
-- This tests relation operators, logical operators, and
multiple parameters

function test3 a: integer, b: integer returns boolean;

begin
    case a is
        when 1 => (a < b);
```

```
            when 2 => (a > b);
            when 3 => (a >= b);
            when 4 => (a <= b);
            when 5 => (a /= b) and (a = b);
            others => (a /= b) or (a = b);
        endcase;
    end;
```

the following output was produced:

```
⌥⌘2          tylerclark@MacBook-Pro:~/Repos/CMSC430/project3

tylerclark ~/ … /CMSC430/project3
λ ./compile < tests/test3.txt 1 2                                    (main x!?)

   1  -- This tests relation operators, logical operators, and multiple parameters
   2
   3  function test3 a: integer, b: integer returns boolean;
   4
   5  begin
   6      case a is
   7          when 1 ⇒ (a < b);
   8          when 2 ⇒ (a > b);
   9          when 3 ⇒ (a ≥ b);
  10          when 4 ⇒ (a ≤ b);
  11          when 5 ⇒ (a ≠ b) and (a = b);
  12          others ⇒ (a ≠ b) or (a = b);
  13      endcase;
      end;
Compiled Successfully
Result = 1
tylerclark ~/ … /CMSC430/project3
λ ./compile < tests/test3.txt 2 2                                    (main x!?)

   1  -- This tests relation operators, logical operators, and multiple parameters
   2
   3  function test3 a: integer, b: integer returns boolean;
   4
   5  begin
   6      case a is
   7          when 1 ⇒ (a < b);
   8          when 2 ⇒ (a > b);
   9          when 3 ⇒ (a ≥ b);
  10          when 4 ⇒ (a ≤ b);
  11          when 5 ⇒ (a ≠ b) and (a = b);
  12          others ⇒ (a ≠ b) or (a = b);
  13      endcase;
      end;
Compiled Successfully
Result = 0
tylerclark ~/ … /CMSC430/project3
λ ./compile < tests/test3.txt 3 3                                    (main x!?)

   1  -- This tests relation operators, logical operators, and multiple parameters
   2
   3  function test3 a: integer, b: integer returns boolean;
   4
   5  begin
   6      case a is
   7          when 1 ⇒ (a < b);
   8          when 2 ⇒ (a > b);
   9          when 3 ⇒ (a ≥ b);
  10          when 4 ⇒ (a ≤ b);
  11          when 5 ⇒ (a ≠ b) and (a = b);
  12          others ⇒ (a ≠ b) or (a = b);
  13      endcase;
      end;
Compiled Successfully
Result = 1
tylerclark ~/ … /CMSC430/project3
λ                                                                    (main x!?)
```

```
tylerclark@MacBook-Pro:~/Repos/CMSC430/project3

tylerclark ~/ … /CMSC430/project3
λ ./compile < tests/test3.txt 3 3                                    (main x!?)

   1  -- This tests relation operators, logical operators, and multiple parameters
   2
   3  function test3 a: integer, b: integer returns boolean;
   4
   5  begin
   6      case a is
   7          when 1 ⇒ (a < b);
   8          when 2 ⇒ (a > b);
   9          when 3 ⇒ (a ≥ b);
  10          when 4 ⇒ (a ≤ b);
  11          when 5 ⇒ (a /= b) and (a = b);
  12          others ⇒ (a /= b) or (a = b);
  13      endcase;
      end;
Compiled Successfully
Result = 1
tylerclark ~/ … /CMSC430/project3
λ ./compile < tests/test3.txt 4 5                                    (main x!?)

   1  -- This tests relation operators, logical operators, and multiple parameters
   2
   3  function test3 a: integer, b: integer returns boolean;
   4
   5  begin
   6      case a is
   7          when 1 ⇒ (a < b);
   8          when 2 ⇒ (a > b);
   9          when 3 ⇒ (a ≥ b);
  10          when 4 ⇒ (a ≤ b);
  11          when 5 ⇒ (a /= b) and (a = b);
  12          others ⇒ (a /= b) or (a = b);
  13      endcase;
      end;
Compiled Successfully
Result = 1
tylerclark ~/ … /CMSC430/project3
λ ./compile < tests/test3.txt 5 5                                    (main x!?)

   1  -- This tests relation operators, logical operators, and multiple parameters
   2
   3  function test3 a: integer, b: integer returns boolean;
   4
   5  begin
   6      case a is
   7          when 1 ⇒ (a < b);
   8          when 2 ⇒ (a > b);
   9          when 3 ⇒ (a ≥ b);
  10          when 4 ⇒ (a ≤ b);
  11          when 5 ⇒ (a /= b) and (a = b);
  12          others ⇒ (a /= b) or (a = b);
  13      endcase;
      end;
Compiled Successfully
Result = 0
tylerclark ~/ … /CMSC430/project3
λ                                                                    (main x!?)
```

```
⌥⌘2  tylerclark@MacBook-Pro:~/Repos/CMSC430/project3
        (main x!?)

  1  -- This tests relation operators, logical operators, and multiple parameters
  2
  3  function test3 a: integer, b: integer returns boolean;
  4
  5  begin
  6      case a is
  7          when 1 ⇒ (a < b);
  8          when 2 ⇒ (a > b);
  9          when 3 ⇒ (a ≥ b);
 10          when 4 ⇒ (a ≤ b);
 11          when 5 ⇒ (a /= b) and (a = b);
 12          others ⇒ (a /= b) or (a = b);
 13      endcase;
     end;
Compiled Successfully
Result = 1
tylerclark ~/ ... /CMSC430/project3
λ                                                          (main x!?)
```

Lessons Learned

For this project, I learned a lot about syntax directed translation. I gained valuable information on bison and flex files and took this opportunity to brush up on my C++ skills. The videos for the week allowed me to get a good understanding on how these different types of files worked together. The real challenge was understanding how and when to interpret values for different statements. I ran into major hiccups trying to figure out how to use actions within the bison code. Overall, it was a huge learning experience.