Project 2

Terrence Jackson

CMSC 430: Compiler Theory and Design

Professor McDonald

September 10, 2024

Approach

I began the project by compiling the provided skeleton code and running the initial test files to ensure everything worked as expected. My first major task was to replace the existing lexical analyzer code with my code from the previous project, which successfully passed initial tests.

Then, I followed the suggested approach and started with modifying the type and primary productions. Although the primary production was not explicitly provided in the grammar, I deduced that adding REAL_LITERAL was necessary, and this allowed me to pass test5.

Next, I focused on the if statement. A key challenge here was realizing that I had implemented “elseif” instead of “elsif”, which caused the token to be incorrectly identified as an IDENTIFIER. After debugging, I corrected this error and successfully handled semicolon usage, passing the relevant tests.

For multiple variables in functions, I introduced recursion in the grammar to handle the possibility of multiple variable declarations. I modified the optional_variable production to allow 0 or more variables. I renamed it optional_variables for clarity, and test8 compiled successfully.

The next modification was handling parameter declarations in the function header. This involved replacing the EBNF brace metasymbol in the parameters production. I applied recursive logic similar to the one used for the cases/case and optional_variable productions, successfully passing test9 and test10.

Adding exponentiation, modulus, and unary negation required careful implementation of the operator precedence and associativity rules. I modified the grammar to include these operators, ensuring the correct order of operations through recursion. Although the parser doesn’t explicitly

verify associativity, I implemented these based on textbook knowledge and successfully passed test11.

I grouped logical operators together in terms of precedence, followed by relational operators, as indicated by the requirements. This step was slightly unclear, but I believe it aligns with the intended behavior that the ! (NOT) operator should have the highest precedence among logical operators.

Finally, I incorporated error productions in key areas, such as the function header, variable declarations, and the WHEN clause. The provided function body already had an error production, and I focused on case handling based on the syntax4.txt test file. Throughout, I ensured that all changes were incremental and tested after each modification to avoid introducing shift/reduce or reduce/reduce conflicts.

Test Plan

Before executing the test plan, I compiled the program using the provided makefile by navigating to the project directory in the terminal and running the command make. Once the compilation was successful, I tested the program using the provided test files. Running the program with each test file as input allowed me to verify the correctness of the changes and ensure that the program behaved as expected across all test cases.

Test Case	Input	Expected Output	Pass?

syntax1	// Missing Operator in Expression
	
	function main returns integer;
	begin
	8 + 2 9 * 3;
	end;
		1 syntax error in function body	Y

syntax2	// Error in Function Header, Missing Colon
	
	function main a integer returns integer;
	b: integer is 3 * 2;
	begin
	if a <= 0 then
	b + 3;
	else
	b * 4;
	endif;
	end;
		1 syntax error in function header	Y

syntax3	// Error in Variable Declaration
	
	function main a: integer returns integer;
	b integer is
	if a > 5 then
	a * 3;
	else
	2 + a;
	endif;
	c: real is 3.5;
	begin
	if a <= 0 then
	b + 3;
	else
	b * 4;
	endif;
	end;
		1 syntax error in variable declaration	Y

syntax4	// Multiple Errors, Error in Case Clause and Missing Others Clause
	
	function main a: integer returns integer;
	begin
	switch a is
	case 1 => a 2;
	case 2 => 5;
	endswitch;
	end;
		2 syntax errors, one in case 1 and one at endswitch	Y

syntax5	-- Multiple Errors
	
	function main a integer returns real;
	b: integer is * 2;
	c: real is 6.0;
	begin
	if a > c then
	b + / .4;
	else
	switch b is
	case => 2;
	case 2 => c;
	endswitch;
	endif;
	end;
		5 syntax errors
		1 in function header
		1 in variable declaration
		1 in if statement
		1 in else statement
		1 at endswitch
			Y

test1	// Function with Arithmetic Expression
	
	function main returns integer;
	begin
	7 + 2 * (5 + 4);
	end;
		Compiled Successfully.	Y

test2	// Function with When Statement
	
	function main returns integer;
	begin
	when 3 < 2 & 6 < 7, 7 * 2 : 7 + 2;
	end;
		Compiled Successfully.	Y

test3	// Function with a Switch Statement
	
	function main returns character;
	a: integer is 2 * 2 + 1;
	begin
	switch a is
	case 1 => 'a';
	case 2 => 'b';
	others => 'c';
	endswitch;
	end;
		Compiled Successfully.	Y

test4	// Function with a List Variable
	
	function main returns integer;
	primes: list of integer is (2, 3, 5, 7);
	begin
	primes(1) + primes(2);
	end;
		Compiled Successfully.	Y

test5	// Function with a Real and Character Variables and Literals
	
	function main returns character;
	a: real is 7.8e-1;
	begin
	when a < .45, '\n' : 'A';
	end;
		Compiled Successfully.	Y

test6	// Function with an If Statemnt
	
	function main a: integer returns integer;
	begin
	if a < 10 then
	1;
	elsif a < 20 then
	2;
	elsif a < 30 then
	3;
	else
	4;
	endif;
	end;
		Compiled Successfully.	Y

test7	// Left and Right Fold Statement
	
	function main returns integer;
	a: list of integer is (1, 2, 3);
	begin
	switch a(1) is
	case 1 =>
	fold right - (2,3, 4)
	endfold;
	case 2 =>
	fold left + a endfold;
	others =>
	0;
	endswitch;
	end;
		Compiled Successfully.	Y

test8	// Multiple Integer Variable Initializations
	
	function main returns integer;
	b: integer is 5 + 1 - 4;
	c: integer is 2 + 3;
	begin
	b + 1 - c;
	end;
		Compiled Successfully.	Y

test9	// Single Parameter Declaration
	
	function main a: integer returns integer;
	begin
	a + 1;
	end;
		Compiled Successfully.	Y

test10	// Two Parameter Declarations
	
	function main a: integer, b: real returns real;
	c: real is .7;
	begin
	a + b * c;
	end;
		Compiled Successfully.	Y

test11	// Arithmetic Operators
	
	function main returns integer;
	begin
	9 + ~2 - (5 - 1) / 2 % 3 * 3 ^ 1 ^ 2;
	end;
		Compiled Successfully.	Y

test12	// Relational and Logical Operators
	
	function main returns integer;
	begin
	when 5 > 8 & 3 = 3 | 9 < 1 & !(3 <> 7) | 6 <= 7 & 3 >= 9, 1 : 0;
	end;
		Compiled Successfully.	Y

test13	// Comprehensive Test with Nested If
	
	function main a: integer, b: character, c: real returns real;
	d: integer is #8e;
	e: real is 3.75;
	f: character is 'A';
	g: list of integer is (1, 3, 5);
	begin
	if ~a > 5 & a < 1 & !(c = 5.8 | c <> .7E4) then
	if c >= 7.5E-2 & c <= 5.2 then
	when a >= d, a + 2 - 7.9E+2 / 9 * 4 : 8.9;
	elsif g(1) = a ^ 2 % 3 then
	a % 2 - 5 / c;
	else
	fold left + (1, 2, 3) endfold;
	endif;
	else
	fold right - g endfold;
	endif;
	end;
		Compiled Successfully.	Y

test14	// Comprehensive Test with Nested Switch
	
	function main a: integer, b: real returns real;
	c: integer is 8;
	d: real is .7E2;
	begin
	switch a is
	case 1 => a * 2 / d ^ 2;
	case 2 => a + 5.3E+2 - b;
	case 3 =>
	switch d is
	case 1 => a % 2;
	others => 9.1E-1;
	endswitch;
	case 4 => a / 2 - c;
	others => a + 4.7 * b;
	endswitch;
	end;
		Compiled Successfully.	Y

Syntax Test Case 1:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skelet

   1  // Missing Operator in Expression
   2
   3  function main returns integer;
   4  begin
   5      8 + 2 9 * 3;
syntax error, unexpected INT_LITERAL, expecting ';'
   6  end;
   7

Lexical Errors 0
Syntax Errors 1
Semantic Errors 0
```

Syntax Test Case 2:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Error in Function Header, Missing Colon
    2
    3  function main a integer returns integer;
syntax error, unexpected INTEGER, expecting ':'
    4      b: integer is 3 * 2;
    5  begin
    6      if a <= 0 then
    7          b + 3;
    8      else
    9          b * 4;
   10      endif;
   11  end;
   12

Lexical Errors 0
Syntax Errors 1
Semantic Errors 0
```

Syntax Test Case 3:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Error in Variable Declaration
    2
    3  function main a: integer returns integer;
    4      b integer is
syntax error, unexpected INTEGER, expecting ':'
    5          if a > 5 then
    6              a * 3;
    7          else
    8              2 + a;
    9          endif;
   10      c: real is 3.5;
   11  begin
   12      if a <= 0 then
   13          b + 3;
   14      else
   15          b * 4;
   16      endif;
   17  end;
   18

Lexical Errors 0
Syntax Errors 1
Semantic Errors 0
```

Syntax Test Case 4:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$ ./co

    1  // Multiple Errors, Error in Case Clause and Missing Others Clause
    2
    3  function main a: integer returns integer;
    4  begin
    5      switch a is
    6          case 1 => a 2;
  syntax error, unexpected INT_LITERAL, expecting ';'
    7          case 2 => 5;
    8      endswitch;
  syntax error, unexpected ENDSWITCH, expecting CASE or OTHERS
    9  end;
   10

  Lexical Errors 0
  Syntax Errors 2
  Semantic Errors 0
```

Syntax Test Case 5:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

   1  -- Multiple Errors
   2
   3  function main a integer returns real;
syntax error, unexpected INTEGER, expecting ':'
   4      b: integer is * 2;
syntax error, unexpected MULOP
   5      c: real is 6.0;
   6  begin
   7      if a > c then
   8          b + / .4;
syntax error, unexpected MULOP
   9      else
  10          switch b is
  11              case => 2;
syntax error, unexpected ARROW, expecting INT_LITERAL
  12              case 2 => c;
  13          endswitch;
syntax error, unexpected ENDSWITCH, expecting CASE or OTHERS
  14      endif;
  15  end;
  16

Lexical Errors 0
Syntax Errors 5
Semantic Errors 0
```

Test Case 1:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

   1  // Function with Arithmetic Expression
   2
   3  function main returns integer;
   4  begin
   5      7 + 2 * (5  + 4);
   6  end;
Compiled Successfully.
```

Test Case 2:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Function with When Statement
    2
    3  function main returns integer;
    4  begin
    5      when 3 < 2 & 6 < 7, 7 * 2 : 7 + 2;
    6  end;
 Compiled Successfully.
```

Test Case 3:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Function with a Switch Statement
    2
    3  function main returns character;
    4      a: integer is 2 * 2 + 1;
    5  begin
    6      switch a is
    7          case 1 => 'a';
    8          case 2 => 'b';
    9          others => 'c';
   10      endswitch;
   11  end;
 Compiled Successfully.
```

Test Case 4:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Function with a List Variable
    2
    3  function main returns integer;
    4      primes: list of integer is (2, 3, 5, 7);
    5  begin
    6      primes(1) + primes(2);
    7  end;
 Compiled Successfully.
```

Test Case 5:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Function with a Real and Character Variables and Literals
    2
    3  function main returns character;
    4      a: real is 7.8e-1;
    5  begin
    6      when a < .45, '\n' : 'A';
    7  end;
  Compiled Successfully.
```

Test Case 6:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Function with an If Statemnt
    2
    3  function main a: integer returns integer;
    4  begin
    5      if a < 10 then
    6          1;
    7      elsif a < 20 then
    8          2;
    9      elsif a < 30 then
   10          3;
   11      else
   12          4;
   13      endif;
   14  end;
  Compiled Successfully.
```

Test Case 7:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Left and Right Fold Statement
    2
    3
    4  function main returns integer;
    5      a: list of integer is (1, 2, 3);
    6  begin
    7      switch a(1) is
    8          case 1 =>
    9              fold right - (2,3, 4) endfold;
   10          case 2 =>
   11              fold left + a endfold;
   12          others =>
   13              0;
   14      endswitch;
   15  end;
 Compiled Successfully.
```

Test Case 8:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Multiple Integer Variable Initializations
    2
    3  function main returns integer;
    4      b: integer is 5 + 1 - 4;
    5      c: integer is 2 + 3;
    6  begin
    7      b + 1 - c;
    8  end;
 Compiled Successfully.
```

Test Case 9:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Single Parameter Declaration
    2
    3  function main a: integer returns integer;
    4  begin
    5      a + 1;
    6  end;
 Compiled Successfully.
```

Test Case 10:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Two Parameter Declarations
    2
    3  function main a: integer, b: real returns real;
    4      c: real is .7;
    5  begin
    6      a + b * c;
    7  end;
Compiled Successfully.
```

Test Case 11:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Arithmetic Operators
    2
    3  function main returns integer;
    4  begin
    5      9 + ~2 - (5 - 1) / 2 % 3 * 3 ^ 1 ^ 2;
    6  end;
Compiled Successfully.
```

Test Case 12:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$ ./compile

    1  // Relational and Logical Operators
    2
    3  function main returns integer;
    4  begin
    5      when 5 > 8 & 3 = 3 | 9 < 1 & !(3 <> 7) | 6 <= 7 & 3 >= 9, 1 : 0;
    6  end;
Compiled Successfully.
```

Test Case 13:

```
● tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$

    1  // Comprehensive Test with Nested If
    2
    3  function main a: integer, b: character, c: real returns real;
    4      d: integer is #8e;
    5      e: real is 3.75;
    6      f: character is 'A';
    7      g: list of integer is (1, 3, 5);
    8  begin
    9      if ~a > 5 & a < 1 & !(c = 5.8 | c <> .7E4) then
   10          if c >= 7.5E-2 & c <= 5.2 then
   11              when a >= d, a + 2 - 7.9E+2 / 9 * 4 : 8.9;
   12          elsif g(1) = a ^ 2 % 3 then
   13              a % 2 - 5 / c;
   14          else
   15              fold left + (1, 2, 3) endfold;
   16          endif;
   17       else
   18           fold right - g endfold;
   19       endif;
   20  end;
   21
  Compiled Successfully.
```

Test Case 14:

```
tjackson@UMGC:~/Documents/CMSC-430/Project_2/Project 2 Skeleton Code$
    1  // Comprehensive Test with Nested Switch
    2
    3  function main a: integer, b: real returns real;
    4      c: integer is 8;
    5      d: real is .7E2;
    6  begin
    7      switch a is
    8          case 1 => a * 2 / d ^ 2;
    9          case 2 => a + 5.3E+2 - b;
   10          case 3 =>
   11              switch d is
   12                  case 1 => a % 2;
   13                  others => 9.1E-1;
   14              endswitch;
   15          case 4 => a / 2 - c;
   16          others => a + 4.7 * b;
   17      endswitch;
   18  end;
Compiled Successfully.
```

Lessons Learned

This project deepened my understanding of operator precedence and associativity, particularly with recursive grammar. Implementing these rules required careful thought, especially for ensuring correct recursion for left- and right-associative operators. Also, debugging tokenization issues, like my confusion between elseif and elsif, highlighted the importance of small details in grammar and how incorrect token definitions can propagate unexpected behavior. By making incremental changes and testing thoroughly after each step, I was able to isolate and resolve issues as they arose. Looking back, I believe there may be room for improving some of the recursion logic to be more concise, especially around handling lists of parameters and variables. However, the overall approach worked well in ensuring the stability of the syntactic analyzer, and I was able to successfully meet the project requirements without major conflicts.