

SCS 2312 - Computational Models and Programming Language Concepts

Take Home Assignment

R.Jananganan

23000686

1) Context-Free Grammar (CFG)

$\langle \text{program} \rangle$	\rightarrow	$\langle \text{statement_list} \rangle$
$\langle \text{statement_list} \rangle$	\rightarrow	$\langle \text{statement} \rangle \langle \text{statement_list} \rangle$ ϵ
$\langle \text{statement} \rangle$	\rightarrow	$\langle \text{declaration} \rangle$ $\langle \text{print_statement} \rangle$
$\langle \text{declaration} \rangle$	\rightarrow	$\text{int } \langle \text{identifier} \rangle = \langle \text{expression} \rangle ;$
$\langle \text{print_statement} \rangle$	\rightarrow	$\text{print} (\langle \text{identifier} \rangle);$
$\langle \text{expression} \rangle$	\rightarrow	$\langle \text{term} \rangle \langle \text{expression}' \rangle$
$\langle \text{expression}' \rangle$	\rightarrow	$+ \langle \text{term} \rangle \langle \text{expression}' \rangle$ $- \langle \text{term} \rangle \langle \text{expression}' \rangle$ ϵ
$\langle \text{term} \rangle$	\rightarrow	$\langle \text{factor} \rangle \langle \text{term}' \rangle$
$\langle \text{term}' \rangle$	\rightarrow	$* \langle \text{factor} \rangle \langle \text{term}' \rangle$ $/ \langle \text{factor} \rangle \langle \text{term}' \rangle$ ϵ
$\langle \text{factor} \rangle$	\rightarrow	$\langle \text{identifier} \rangle$ $\langle \text{number} \rangle$
$\langle \text{identifier} \rangle$	\rightarrow	$\text{letter} (\text{letter} \mid \text{digit})^*$
$\langle \text{number} \rangle$	\rightarrow	$\text{digit} +$

This grammar explains about the Context-Free Grammar of the built parser.c file. This c programming file supports variable declarations, arithmetic operations with proper operator precedence and print statements also.

2) Tokenizer Implementation

Here the function fgetc() is used in the code for the tokenizer to read the input file character by character, so that it classifies input into tokens such as keywords, identifiers, numbers, operators and symbols. Blank whitespaces are ignored by using the isspace() function. isalpha() and isalnum() functions are used to identify the keywords and identifiers and then isdigit() function is used to detect numbers. And finally the function ungetc() is used to push back the characters that doesn't belong to the current token into the code again so that it can be detected and read in the next step.

3) Parser

The parser is implemented in such a way that each non-terminal elements in the CFG corresponds to a function such as parseExpression(), parseTerm() and parseFactor(). The advance() function is used to retrieve the next token and expect() function is used to ensure that the correct token appears according to the grammar. If an unexpected token is met then a syntax error will be generated accordingly.

4) Semantic Analysis

A symbol tyable is implemented using arrays to store variable names and their corresponding values. The function getVar() ensures variables are declared before use. Semantic checks include detection of undeclared variables, devision by zero and optional redeclaration errors.

5) Execution & Evaluation

Its done during parsing. Operator precedence is maintained by separating expression parsing into three levels such as:

- a) expression(+ , -)
- b) term(*, /)
- c) factor(identifier or number)

This ensures correct mathematical evaluation

6) Testing

i) For correct input

C parser.c X

```
C parser.c > main(int, char * [])
322     void pars main(int, char * []) (function)
332         else{
335     }
336
337     void parseProgram(){
338         while(currentToken.type != TOKEN_EOF){
339             parseStatement();
340         }
341     }
342
343
344     int main( int argc, char *argv[]){
345
346         if(argc<2){
347             printf("Usage: %s <inputfile>\n", argv[0]);
348             return 1;
349         }
350
351         FILE *file =fopen(argv[1], "r");
352
353         if(file == NULL){
354             exit(1);
355         }
356     }
357 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- jananganan@MacBookAir CMPL_Assignment % gcc parser.c -o parser
- jananganan@MacBookAir CMPL_Assignment % ./parser input.txt
25
- jananganan@MacBookAir CMPL_Assignment % █

ii) For Error examples

a) Int x = y + 2;

The terminal window title is "CMPL_Assignment — vim test2.txt". The code in the buffer is:

```
int x=y+2;
print(x);
~  
~  
~  
~  
~n", argv[0]);  
~
```

The code contains a syntax error: "int x=y+2;" instead of "int x = y + 2;".

The interface shows the "TERMINAL" tab selected. The terminal output is:

- jananganan@MacBookAir CMPL_Assignment % gcc parser.c -o parser
- jananganan@MacBookAir CMPL_Assignment % ./parser input.txt
25
- ✖ jananganan@MacBookAir CMPL_Assignment % ./parser test2.txt
Semantic Error : Variable 'y' used before declaration
- jananganan@MacBookAir CMPL_Assignment % █

b) int x=5

 print(x);

The terminal window title is "CMPL_Assignment — vim test1.txt". The code in the buffer is:

```
int x=5
print(x);
~  
~  
~  
~  
~n", argv[0]);
~  
~  
~
```

The code contains a syntax error: "int x=5" instead of "int x = 5;".

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

- jananganan@MacBookAir CMPL_Assignment % gcc parser.c -o parser
- jananganan@MacBookAir CMPL_Assignment % ./parser input.txt
25
- ⊗ jananganan@MacBookAir CMPL_Assignment % ./parser test2.txt
Semantic Error : Variable 'y' used before declaration
- ⊗ jananganan@MacBookAir CMPL_Assignment % ./parser test1.txt
Syntax Error : Expected ; but found TOKEN_PRINT
- jananganan@MacBookAir CMPL_Assignment % █

c) int = 5;

CMPL_Assignment — vim test3.c

```
int =5;

~
~
~
~
~n", argv[0]);
~
~
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

- jananganan@MacBookAir CMPL_Assignment % gcc parser.c -o parser
- jananganan@MacBookAir CMPL_Assignment % ./parser input.txt
25
- ⊗ jananganan@MacBookAir CMPL_Assignment % ./parser test2.txt
Semantic Error : Variable 'y' used before declaration
- ⊗ jananganan@MacBookAir CMPL_Assignment % ./parser test1.txt
Syntax Error : Expected ; but found TOKEN_PRINT
- ⊗ jananganan@MacBookAir CMPL_Assignment % ./parser test3.txt
Syntax Error: Expected identifier after int
- jananganan@MacBookAir CMPL_Assignment % █

d) Division by zero

A screenshot of a terminal window titled "CMPL_Assignment — vim test4.txt —". The code in the terminal is:

```
int x=10;
int y=0;
int z=x/y;
print(z);
~  
~, argv[0]);
~  
~
```

The terminal shows several cursor symbols (~) indicating where the code was last edited.

A screenshot of the VS Code interface showing the "PROBLEMS" tab selected. The list of errors is as follows:

- jananganan@MacBookAir CMPL_Assignment % gcc parser.c -o parser
- jananganan@MacBookAir CMPL_Assignment % ./parser input.txt
25
- ✖ jananganan@MacBookAir CMPL_Assignment % ./parser test2.txt
Semantic Error : Variable 'y' used before declaration
- ✖ jananganan@MacBookAir CMPL_Assignment % Open file in editor (cmd + click) Syntax Error : Expected ; but found TOKE
- ✖ jananganan@MacBookAir CMPL_Assignment % ./parser test3.txt
Syntax Error: Expected identifier after int
- ✖ jananganan@MacBookAir CMPL_Assignment % ./parser test4.txt
Semantic Error: Division by zero
- jananganan@MacBookAir CMPL_Assignment %