Compiler Design

February 22, 2018

Team Members

Madhukar Jaiswal (201551066)
Prashant Chaurasiya (201551088)
Kamlesh Kumar (201551057)
Vikesh Meena (201551026)

1 Grammar Tokens:

Token Matching Table		
Pattern	Token	Purpose
(	LP	To specify opening bracket
)	RP	To specify closing bracket
OR	OR	To denote logical OR Operator
+	PLUS	To denote plus operator
-	MINUS	To denote minus operator
=	ASSIGN	To denote assignment operator
:	COLON	To denote start of a function body or start of statements following a condition or loop
;	SEMICOLON	To denote end of a statement
char	CHAR	To denote character data type
int	INT	To denote integer data type
float	FLOAT	To denote float data type
[a-z]+[A-Z]+[0-9]*	ID	to denote variable and function names
if	IF	To denote if conditional statement
else	ELSE	To denote else conditional statement
ef	EF	To denote else if conditional statement
endi	ENDIF	To denote end of if conditional statement
[	LSB	To denote left square brackets
]	RSB	To denote right square brackets
”	DQ	To denote double quotations

Token Matching Table		
Pattern	Token	Purpose
bool	BOOL	To denote boolean data type
*	MUL	To denote multiplication operator
/	DIV	To denote division operator
AND	AND	To denote logical AND operator
NOT	NOT	To denote logical NOT operator
<	LT	To denote less than operator
>	GT	To denote greater than operator
<=	LE	To denote less than or equal to operator
>=	GE	To denote greater than or equal to operator
==	EQ	To denote equals operator
!=	NEQ	To denote notequals operator
looptill	LOOP	To specify beginning of iteration
endl	ENDLOOP	To specify the end of loop
break	BREAK	To specify the stopping of iteration
continue	CONTINUE	To specify the continuation of iteration
obj	OBJ	To specify object type to store return value of a function
.	DOT	To access various elements of the return list from a function
return	RETURN	To specify the return of a function
endf	ENDF	To denote end of function

2 Grammar Productions:

Now we are going to display the production rules of our grammar. All the possible strings that are found to be generated by our grammar rules are valid for this language. In case a string is found not to be derived by these set of production rules, then it leads to an error.

3 Start Of Program

START: <functions> <main>

<functions> :<function><functions> | EPSILON

<main> : MAIN LP RP RETURN LP RP COLON <stmts>ENDFUNCTION

4 Function Declaration and Assignments

1. <function>:FUNC ID LP <param_list >RP RETURN LP <param_list>RP COLON <stmts>ENDFUNCTION

2. <param_list>: <type ><var> COMMA<param_list>| EPSILON

3. <return_value>: ID | CONSTANT | EPSILON

4. <type> : CHAR | INT | FLOAT | BOOL

5. <stmts> : <stmt> <stmts> | EPSILON

6. <stmt> : <assign_stmts> | <conditional_stmts> | < declaration_stmts> | <io_stmts> | <loop_stmt> | proc_callstmts> | BREAK |CONTINUE

7. <declaration_stmts> : <type> <var_list>

8. <var_list> : ID | ID COMMA <var_list> | IDCOMMA <assign_list>

9. <assign_stmts> : <type> <assign_list> | <assign_stmt>

10. <assign_list> : <assign_stmt>|<assign_stmt> COMMA <assign_list>|<var_list>

11. <assign_stmt> : ID ASSIGN <value>

12. <value> : ID | CONSTANT | <expr>

13. <expr> : <expr> PLUS <term> | <expr> MINUS <term> | <term>

14. <term> : <term> MUL <factor> | <term> DIV <factor> | <factor>

15. <factor> : ID | CONSTANT | LP <expr> RP

5 Conditional Statements:

16. <conditional_stmts >: IF LP<conditional_expr>RP COLON <stmts ><else_if ><condition >

17. <else_if >: ELSEIF LP conditional_expr >RP <stmts><else_if> | EPSILON

18. <condition >: ELSE COLON <stmts>ENDIF | ENDIF

6 Condition

19. <conditional_expr >: LP <conditional_expr> RP <logical_op> LP < conditional_expr > RP

20. <conditional_expr >: <elem><rel_op><elem>

21. <conditional_expr >: NOT LP <conditional_expr>RP | BOOL_LIT

22. <logical_op> : AND | OR

23. < rel_op >: LT | GT | LE | GE | EQ | NEQ

24. <elem> : <var> | <expr> | INT_LIT | REAL_LIT

7 Looping

25. <loop_stmt>: LOOP LP <conditional_expr> RP COLON <stmts> ENDL

8 IO Statements

26. <io_stmts> : INPUT LP <var> RP SEMICOLON

27. <io_stmts> : OUTPUT LP <out_stmt> RP SEMICOLON

28. <out_stmt> : < var> <out> |DQUOTE TEXT DQUOTE <out> | DQUOTE NL DQUOTE <out>

29. <out> : + <out_stmt> | + DQUOTE TAB DQUOTE | <out_stmt> | EPSILON

30. <var> : ID <_var>

31. <_var> : LSB <dims> RSB | EPSILON

9 proc_call stmts:

32. <proc_call stmts> : FUNC_NAME LP < param_list > RP RETURN LP <param_list>RP SEMICOLON