

MiniC Language Manual

Name: Swetanjali Dutta

Roll Number: 20171077

31/08/2020

1 Macro Syntax Specification using Context Free Grammars

1.1 Meta Notation:

- $\langle \text{foo} \rangle$ means foo is a non terminal.
- **foo**(in bold font) means foo is a terminal i.e a token.
- $[x]$ means zero or one occurrence of x i.e x is optional. Note that *brackets in quotes* i.e '[' and ']' are terminals.
- x^* means zero or more occurrences of x .
- x^+ means one or more occurrences of x .
- $x^+,$ means a comma separated list of one or more x s. Comma is a terminal.
- $\{\}$ i.e large braces are used for grouping. Note that *braces in quotes* i.e '{' and '}' are terminals.
- $|$ separates alternatives.
- Punctuation like round brackets, braces, semicolons and commas are terminals. Please note that they have not been written in bold in the CFG.

1.2 Production Rules:

1. $\langle \text{program} \rangle \rightarrow \langle \text{decl} \rangle^+$
2. $\langle \text{decl} \rangle \rightarrow \langle \text{var_decl} \rangle \mid \langle \text{method_decl} \rangle$
3. $\langle \text{var_decl} \rangle \rightarrow \langle \text{type} \rangle \langle \text{identifier} \rangle^+, ;$
4. $\langle \text{method_decl} \rangle \rightarrow \{ \langle \text{type} \rangle \mid \mathbf{VOID} \} \mathbf{ID} ([\{ \langle \text{type} \rangle \langle \text{identifier} \rangle^+,]) \langle \text{block} \rangle$
5. $\langle \text{block} \rangle \rightarrow \{ ' \langle \text{var_decl} \rangle^* \langle \text{statement} \rangle^* \}$
6. $\langle \text{type} \rangle \rightarrow \mathbf{INT} \mid \mathbf{UINT} \mid \mathbf{BOOL} \mid \mathbf{CHAR}$
7. $\langle \text{statement} \rangle \rightarrow \langle \text{assignment} \rangle^+, ;$
| $\langle \text{method_call} \rangle;$
| $\mathbf{IF} (\langle \text{expr} \rangle) \langle \text{block} \rangle [\mathbf{ELSE} \langle \text{block} \rangle]$
| $\mathbf{FOR} ([\langle \text{assignment} \rangle^+,]; [\langle \text{expr} \rangle^+,]; [\langle \text{assignment} \rangle^+,]) \langle \text{block} \rangle$
| $\mathbf{WHILE} (\langle \text{expr} \rangle) \langle \text{block} \rangle$
| $\mathbf{BREAK};$
| $\mathbf{CONTINUE};$
| $\langle \text{block} \rangle$
| $\mathbf{RETURN} [\langle \text{expr} \rangle];$
| $\mathbf{PRINT} (\langle \text{expr} \rangle);$

8. $\langle \text{assignment} \rangle \rightarrow \langle \text{identifier} \rangle \text{ ASSIGN } \langle \text{expr} \rangle$
9. $\langle \text{method_call} \rangle \rightarrow \text{ID} ([\langle \text{expr} \rangle^+,])$
10. $\langle \text{expr} \rangle \rightarrow \langle \text{identifier} \rangle$
 $\quad | \langle \text{expr} \rangle \langle \text{arithmetic_op} \rangle \langle \text{expr} \rangle$
 $\quad | \langle \text{expr} \rangle \langle \text{relational_op} \rangle \langle \text{expr} \rangle$
 $\quad | \langle \text{expr} \rangle \langle \text{conditional_op} \rangle \langle \text{expr} \rangle$
 $\quad | \langle \text{expr} \rangle \langle \text{equality_op} \rangle \langle \text{expr} \rangle$
 $\quad | \langle \text{expr} \rangle \text{ THEN } \langle \text{expr} \rangle \text{ OTHERWISE } \langle \text{expr} \rangle$
 $\quad | \langle \text{literal} \rangle$
 $\quad | \langle \text{method_call} \rangle$
 $\quad | \text{NOT } \langle \text{expr} \rangle$
 $\quad | \text{NEGATE } \langle \text{expr} \rangle$
 $\quad | (\langle \text{expr} \rangle)$
 $\quad | \text{READ_INT}()$
 $\quad | \text{READ_CHAR}()$
 $\quad | \text{READ_BOOL}()$
11. $\langle \text{identifier} \rangle \rightarrow \text{ID} | \text{ID} \{ '[' \langle \text{expr} \rangle ']' \}^*$
12. $\langle \text{literal} \rangle \rightarrow \text{INT_LIT} | \text{FLOAT_LIT} | \text{CHAR_LIT} | \text{STRING_LIT} | \langle \text{bool_lit} \rangle$
13. $\langle \text{bool_lit} \rangle \rightarrow \text{TRUE} | \text{FALSE}$
14. $\langle \text{arithmetic_op} \rangle \rightarrow \text{ADD} | \text{SUB} | \text{MUL} | \text{DIV} | \text{MOD}$
15. $\langle \text{relational_op} \rangle \rightarrow \text{LT} | \text{GT} | \text{LE} | \text{GE}$
16. $\langle \text{conditional_op} \rangle \rightarrow \text{AND} | \text{OR}$
17. $\langle \text{equality_op} \rangle \rightarrow \text{EQ} | \text{NE}$

1.3 Start Symbol:

- program

2 Micro Syntax Specification using Regular Expressions

2.1 Meta Notation:

- Token Type \rightarrow Lexeme

2.2 Rules:

1. FALSE \rightarrow false
2. TRUE \rightarrow true
3. NOT \rightarrow !
4. NEGATE \rightarrow ~
5. VOID \rightarrow void
6. INT \rightarrow int
7. UNINT \rightarrow uint
8. CHAR \rightarrow char
9. BOOL \rightarrow bool
10. THEN \rightarrow ?

11. OTHERWISE \rightarrow :
12. FOR \rightarrow for
13. WHILE \rightarrow while
14. IF \rightarrow if
15. ELSE \rightarrow else
16. BREAK \rightarrow break
17. CONTINUE \rightarrow continue
18. RETURN \rightarrow return
19. ADD \rightarrow +
20. SUB \rightarrow -
21. MUL \rightarrow *
22. DIV \rightarrow /
23. MOD \rightarrow %
24. LT \rightarrow <
25. GT \rightarrow >
26. LE \rightarrow <=
27. GE \rightarrow >=
28. AND \rightarrow &&
29. OR \rightarrow ||
30. EQ \rightarrow ==
31. NE \rightarrow !=
32. ASSIGN \rightarrow =
33. PRINT \rightarrow print
34. READ_INT \rightarrow read_int
35. READ_CHAR \rightarrow read_char
36. READ_BOOL \rightarrow read_bool
37. , \rightarrow ,
38. ; \rightarrow ;
39. (\rightarrow (
40.) \rightarrow)
41. { \rightarrow {
42. } \rightarrow }
43. INT_LIT \rightarrow [0-9][0-9]*
44. FLOAT_LIT \rightarrow [0-9][0-9]*([0-9][0-9]*)?
45. CHAR_LIT \rightarrow '[a-zA-Z0-9 _.,:]'
46. ID \rightarrow [a-zA-Z][a-zA-Z0-9_]*
47. STRING_LIT \rightarrow "[a-zA-Z0-9 _.,:]*"

3 Lexical Considerations

4 Semantic Checks