

## FINAL CONTEXT FREE GRAMMAR B++ Programming Language

The grammar  $G = (V, T, S, P)$  of our interpreter is defined by:

$V = \{ \langle \text{source} \rangle, \langle \text{block} \rangle, \langle \text{statement} \rangle, \langle \text{functionDecl} \rangle, \langle \text{expression} \rangle, \langle \text{assignment} \rangle, \langle \text{ifStatement} \rangle, \langle \text{forStatement} \rangle, \langle \text{whileStatement} \rangle, \langle \text{doWhileStatement} \rangle, \langle \text{missingSemiColonAss} \rangle, \langle \text{ifStat} \rangle \langle \text{elseIfStat} \rangle, \langle \text{elseStat} \rangle, \langle \text{missingLBraceForLoop} \rangle \langle \text{missingRBraceForLoop} \rangle, \langle \text{missingLBraceWhileLoop} \rangle, \langle \text{missingRBraceWhileLoop} \rangle, \langle \text{missingLBraceDoWhileLoop} \rangle, \langle \text{missingRBraceDoWhileLoop} \rangle, \langle \text{missingSemiColonLoop} \rangle, \langle \text{functionCall} \rangle, \langle \text{idList} \rangle, \langle \text{exprList} \rangle, \langle \text{expression} \rangle, \langle \text{indexes} \rangle, \langle \text{list} \rangle, \langle \text{PRINTLN} \rangle, \langle \text{INPUT} \rangle, \langle \text{IF} \rangle, \langle \text{ELSE} \rangle, \langle \text{FOR} \rangle, \langle \text{WHILE} \rangle, \langle \text{TO} \rangle, \langle \text{DO} \rangle, \langle \text{NULL} \rangle, \langle \text{IDENTIFIER} \rangle, \langle \text{NUMBER} \rangle, \langle \text{STRING} \rangle, \langle \text{letters+} \rangle, \langle \text{digits+} \rangle} \}$

$T = \{a, b, \dots, z, A, B, \dots, Z, 0, 1, \dots, 9, ;, :, \{, \}, [, ], \_ , ', ", +, -, *, /, \%, \&, |, !\} \cup \{\text{other special characters}\}$

$S = \langle \text{source} \rangle$

$P$  has the following productions:

### Production for the program

$\langle \text{source} \rangle \rightarrow \langle \text{block} \rangle \text{ EOF}$

$\langle \text{block} \rangle \rightarrow (\langle \text{statement} \rangle \mid \langle \text{functionDecl} \rangle)^* (\text{RETURN} \langle \text{expression} \rangle \text{ SCOLON})?$

### Productions for variable declaration

$\langle \text{assignment} \rangle \rightarrow \text{VAR IDENTIFIER} \langle \text{indexes} \rangle? '=' \langle \text{expression} \rangle$   
 $\rightarrow \text{IDENTIFIER} \langle \text{indexes} \rangle? '=' \langle \text{expression} \rangle$

### Productions for statements

$\langle \text{statement} \rangle \rightarrow \langle \text{assignment} \rangle \text{ SCOLON}$   
 $\rightarrow \langle \text{functionCall} \rangle \text{ SCOLON}$   
 $\rightarrow \langle \text{ifStatement} \rangle$   
 $\rightarrow \langle \text{forStatement} \rangle$   
 $\rightarrow \langle \text{whileStatement} \rangle$   
 $\rightarrow \langle \text{doWhileStatement} \rangle$   
 $\rightarrow \langle \text{missingSemiColonAss} \rangle$

### Productions for expressions

$\langle \text{expression} \rangle \rightarrow '-' \langle \text{expression} \rangle$   
 $\rightarrow '!' \langle \text{expression} \rangle$   
 $\rightarrow \langle \text{assoc=right} \rangle \langle \text{expression} \rangle '^' \langle \text{expression} \rangle$   
 $\rightarrow \langle \text{expression} \rangle \langle \text{op} \rangle = ( '*' \mid '/' \mid \% ) \langle \text{expression} \rangle$   
 $\rightarrow \langle \text{expression} \rangle \langle \text{op} \rangle = ( '+' \mid '-' ) \langle \text{expression} \rangle$   
 $\rightarrow \langle \text{expression} \rangle ( '++' \mid '***' \mid '// ' \mid '\%' \mid '--' ) \langle \text{expression} \rangle$

→ <expression> <op>=( '>=' | '<=' | '>' | '<' )  
     <expression>  
 → <expression> <op>=( '==' | '!=' ) <expression>  
 → <expression> '&&' <expression>  
 → <expression> '||' <expression>  
 → NUMBER  
 → NULL  
 → BOOL  
 → STRING <indexes>?  
 → IDENTIFIER <indexes>?  
 → '(' <expression> ')' <indexes>?  
 → <expression> ')'  
 → '(' <expression>  
 → INPUT '(' STRING? ')'

### Other Productions

<ifStatement>	→ <ifStat> <elseIfStat>* <elseStat>?
<ifStat>	→ IF <expression> LBRACE <block> RBRACE
<elseIfStat>	→ ELSE IF <expression> LBRACE <block> RBRACE
<elseStat>	→ ELSE LBRACE <block> RBRACE
<forStatement>	→ FOR IDENTIFIER '=' <expression> TO expression LBRACE block RBRACE
	→ <missingLBraceForLoop>
	→ <missingRBraceForLoop>
<missingLBraceForLoop>	→ FOR IDENTIFIER '=' <expression> TO expression block RBRACE
<missingRBraceForLoop>	→ FOR IDENTIFIER '=' <expression> TO expression LBRACE block
<WhileStatement>	→ WHILE <expression> LBRACE block RBRACE
	→ <missingLBraceWhileLoop>
	→ <missingRBraceWhileLoop>
<missingLBraceWhileLoop>	→ WHILE <expression> block RBRACE
<missingRBraceWhileLoop>	→ WHILE <expression> LBRACE block
<missingSemiColonAss>	→ <assignment>
	→ <functionCall>
<DoWhileStatement>	→ DO LBRACE <block> RBRACE WHILE <expression> SCOLON
	→ <missingLBraceDoWhileLoop>
	→ <missingRBraceDoWhileLoop>
	→ <missingSemiColonLoop>
<missingLBraceDoWhileLoop>	→ DO <block> RBRACE WHILE <expression> SCOLON
<missingRBraceDoWhileLoop>	→ DO LBRACE <block> WHILE <expression> SCOLON
<missingSemiColonLoop>	→ DO LBRACE <block> RBRACE WHILE <expression>
<functionCall>	→ IDENTIFIER '(' exprList? ')' → PRINTLN '(' expression? ')' → PRINT '(' expression? ')' → PRINTLN expression? ')' → PRINTLN '(' expression?
<exprList>	→ <expression> ( ',' expression )*
<indexes>	→ ( '[' <expression> ']' )+

## Keywords

<PRINTLN>	→ 'brint'
<INPUT>	→ 'bnput'
<IF>	→ 'bf'
<ELSE>	→ 'blse'
<FOR>	→ 'bor'
<RETURN>	→ 'beturn'
<WHILE>	→ 'bhile'
<TO>	→ 'bto'
<DO>	→ 'bdo'
<NULL>	→ 'bull'
<BOOL>	→ 'brue'
	→ 'balse'
<NUMBER>	→ INT ( '.' Digit *)?
<IDENTIFIER>	→ [a-zA-Z_] [a-zA-Z_0-9]*