

EBNF Grammar for the programming language

lar9482

February 2024

$\langle program \rangle ::= \langle definitions \rangle$

$\langle definitions \rangle ::= \langle function\ declaration \rangle \langle definitions \rangle$
| $\langle global\ declaration \rangle \langle definitions \rangle$
| EPSILON

$\langle global\ declaration \rangle ::= \text{'global'} \langle identifierDecl \rangle \langle declaration \rangle \text{';'}$

$\langle declaration \rangle ::= \langle varDecl \rangle$
| $\langle multiVarDecl \rangle$
| $\langle arrayDeclaration \rangle$

$\langle varDecl \rangle ::= \langle optional_single_value \rangle$

$\langle multiVarDecl \rangle ::= \text{' ,' } \langle identifierDeclList \rangle \langle optional_multiple_value \rangle$

$\langle optional_single_value \rangle ::= \text{'=' } \langle Expr \rangle$
| EPSILON

$\langle optional_multiple_value \rangle ::= \text{'=' } \langle Expr \rangle \text{' ,' } \langle exprList \rangle$
| EPSILON

$\langle arrayDeclaration \rangle ::= \text{'[' ']' } \langle singleOrMulti_Array_Expr \rangle$
| $\text{'[' } \langle number \rangle \text{']' } \langle singleOrMulti_Array_Static \rangle$

$\langle singleOrMulti_Array_Expr \rangle ::= \text{'=' } \langle arrayInitial \rangle$
| $\text{'=' } \langle ProcedureCall \rangle$
| $\text{'[' ']' '=' '{' } \langle multiDimArrayInitial \rangle \text{'}'}$
| $\text{'[' ']' '=' } \langle ProcedureCall \rangle$

$\langle singleOrMulti_Array_Static \rangle ::= \text{'[' } \langle number \rangle \text{']'}$
| EPSILON

$\langle identifierDeclList \rangle ::= \langle identifierDecl \rangle \text{' ,' } \langle identifierDeclList \rangle$
| $\langle identifierDecl \rangle$

$\langle identifierDecl \rangle ::= \langle identifier \rangle \text{'.'} \langle PrimitiveType \rangle$
 $\langle exprList \rangle ::= \langle Expr \rangle \text{' ,' } \langle exprList \rangle$
 $\quad | \quad \langle Expr \rangle$
 $\langle arrayInitial \rangle ::= \text{'{' } \langle multipleExprs \rangle \text{'}'}$
 $\langle multiDimArrayInitial \rangle ::= \langle arrayInitial \rangle \text{' ,' } \langle multiDimArrayInitial \rangle$
 $\quad | \quad \langle arrayInitial \rangle$
 $\langle identifierDecl \rangle ::= \langle identifier \rangle \text{'.'} \langle primitive_type \rangle$
 $\langle primitive_type \rangle ::= \text{'int'}$
 $\quad | \quad \text{'bool'}$
 $\langle functionDeclaration \rangle ::= \langle identifier \rangle \text{'(' } \langle paramsOptional \rangle \text{')' ' : ' } \langle returnTypesOptional \rangle$
 $\quad \langle Block \rangle$
 $\langle paramsOptional \rangle ::= \langle identifier \rangle \langle Type \rangle \langle paramsList \rangle$
 $\quad | \quad \text{EPSILON}$
 $\langle paramsList \rangle ::= \text{' ,' } \langle identifier \rangle \langle Type \rangle \langle paramsList \rangle$
 $\quad | \quad \text{EPSILON}$
 $\langle Type \rangle :: \langle primitiveType \rangle \langle typeArray \rangle$
 $\langle typeArray \rangle ::= \text{'[' ']' } \langle typeMultiDimArray \rangle$
 $\quad | \quad \text{'[' ']'}$
 $\quad | \quad \text{EPSILON}$
 $\langle typeMultiDimArray \rangle ::= \text{'[' ']'}$
 $\quad | \quad \text{EPSILON}$
 $\langle returnTypesOptional \rangle ::= \langle Type \rangle \langle returnTypeList \rangle$
 $\quad | \quad \text{EPSILON}$
 $\langle returnTypeList \rangle ::= \text{' ,' } \langle Type \rangle \langle returnTypeList \rangle$
 $\quad | \quad \text{EPSILON}$

$\langle block \rangle ::= \{ \langle statements \rangle \}$
 $\langle statements \rangle ::= \langle statement \rangle \langle statements \rangle$
 $\quad \mid \text{ EPSILON}$
 $\langle statement \rangle ::= \langle identifierStatement_All \rangle \text{ ';'}$
 $\quad \mid \langle Conditional \rangle$
 $\quad \mid \langle whileLoop \rangle$
 $\quad \mid \langle forLoop \rangle$
 $\quad \mid \langle return \rangle \text{ ';'}$
 $\langle identifierStatement_All \rangle ::= \langle identifier \rangle \text{ ':' } \langle primitiveType \rangle \langle declaration \rangle$
 $\quad \mid \langle identifier \rangle \langle assignOrMutate \rangle$
 $\quad \mid \langle identifier \rangle \langle procedureCall \rangle$
 $\quad \mid \langle identifier \rangle \langle multiAssign_Or_MultiCallAssign \rangle$
 $\langle identifierStmt_DeclAssignMutate \rangle ::= \langle identifier \rangle \text{ ':' } \langle primitiveType \rangle \langle declaration \rangle$
 $\quad \mid \langle identifier \rangle \langle assignOrMutate \rangle$
 $\langle assignOrMutate \rangle ::= \langle assign \rangle$
 $\quad \mid \langle mutate \rangle$
 $\quad \mid \text{ '[' } \langle Expr \rangle \text{ ']' } \langle arrayAssignOrMutate \rangle$
 $\langle assign \rangle ::= \text{ '=' } \langle Expr \rangle$
 $\langle mutate \rangle ::= \text{ ++ }$
 $\quad \mid \text{ -- }$
 $\langle arrayAssignOrMutate \rangle ::= \text{ '[' } \langle Expr \rangle \text{ ']' } \langle multiDimArrayAssignOrMutate \rangle$
 $\quad \mid \text{ assign}$
 $\quad \mid \text{ mutate}$
 $\langle multiDimArrayAssignOrMutate \rangle ::= \text{ assign}$
 $\quad \mid \text{ mutate}$
 $\langle multiAssign_Or_MultiCallAssign \rangle ::= \text{ ';' } \langle identifierList \rangle \text{ '=' } \langle Expr \rangle \text{ ',' } \langle ExprList \rangle$
 $\quad \mid \text{ ';' } \langle identifierList \rangle \text{ '=' } \langle ProcedureCall \rangle$
 $\langle identifierList \rangle ::= \text{ ',' } \langle identifier \rangle \langle identifierList \rangle$
 $\quad \mid \langle identifier \rangle$
 $\langle ExprList \rangle ::= \langle Expr \rangle \text{ ',' } \langle ExprList \rangle$
 $\quad \mid \langle Expr \rangle$
 $\langle Conditional \rangle ::= \text{ 'if' '(' } \langle Expr \rangle \text{ ')' } \langle block \rangle \langle elseIfConditional \rangle \langle elseConditional \rangle$

$\langle \text{elseIfConditional} \rangle ::= \text{'else' 'if' '(' } \langle \text{Expr} \rangle \text{' ')} \langle \text{block} \rangle \langle \text{elseIfConditional} \rangle$
| EPSILON

$\langle \text{elseConditional} \rangle ::= \text{'else' } \langle \text{block} \rangle$
| EPSILON

$\langle \text{whileLoop} \rangle ::= \text{'while' '(' } \langle \text{Expr} \rangle \text{' ')} \langle \text{block} \rangle$

$\langle \text{forLoop} \rangle ::= \text{'for' '(' } \langle \text{identifierStmt_DeclAssignMutate} \rangle \text{' ;' } \langle \text{Expr} \rangle \text{' ;' } \langle \text{identifier} \rangle$
 $\langle \text{assignOrMutate} \rangle \text{' ')} \langle \text{block} \rangle$

$\langle \text{return} \rangle ::= \text{'return' } \langle \text{ExprList} \rangle ?$

$\langle \text{Expr} \rangle ::= \text{UNOP } \langle \text{Expr} \rangle \langle \text{ExprPrime} \rangle$
| $\langle \text{BaseExpr} \rangle \langle \text{ExprPrime} \rangle$

$\langle \text{ExprPrime} \rangle ::= \text{BINOP } \langle \text{Expr} \rangle \langle \text{ExprPrime} \rangle$
| EPSILON

$\langle \text{BaseExpr} \rangle ::= \text{'(' } \langle \text{Expr} \rangle \text{' ')} \langle \text{Loc} \rangle$
| $\langle \text{ProcedureCall} \rangle$
| $\langle \text{Lit} \rangle$

$\langle \text{Loc} \rangle ::= \langle \text{identifier} \rangle \langle \text{LocArrayAccess} \rangle$

$\langle \text{LocArrayAccess} \rangle ::= \text{'[' } \langle \text{ExprLL} \rangle \text{' ']} \langle \text{Lit} \rangle$
| EPSILON

$\langle \text{ProcedureCall} \rangle ::= \langle \text{identifier} \rangle \text{'(' } \langle \text{ArgsOptional} \rangle \text{' ')} \langle \text{Lit} \rangle$

$\langle \text{ArgsOptional} \rangle ::= \langle \text{Expr} \rangle \langle \text{ArgsList} \rangle$
| EPSILON

$\langle \text{ArgsList} \rangle ::= \text{' ,' } \langle \text{Expr} \rangle \langle \text{ArgsList} \rangle$
| EPSILON

$\langle \text{Lit} \rangle ::= \langle \text{numberLiteral} \rangle$
| $\langle \text{boolLiteral} \rangle$
| $\langle \text{charLiteral} \rangle$
| $\langle \text{stringLiteral} \rangle$