EBNF Grammar for the programming language

lar9482

February 2024

⟨*program*⟩ ::= ⟨*definitions*⟩

⟨*definitions*⟩ ::= ⟨*function declaration*⟩ ⟨*definitions*⟩
| ⟨*global declaration*⟩ ⟨*definitions*⟩
| EPSILON

⟨*global declaration*⟩ ::= '`:global`' ⟨*identifierDecl*⟩ ⟨*declaration*⟩ '`;`'

⟨*declaration*⟩ ::= ⟨*varDecl*⟩
| ⟨*multiVarDecl*⟩
| ⟨*arrayDeclaration*⟩

⟨*varDecl*⟩ ::= ⟨*optional_single_value*⟩

⟨*multiVarDecl*⟩ ::= '`,`' ⟨*identifierDeclList*⟩ ⟨*optional_multiple_value*⟩

⟨*optional_single_value*⟩ ::= '`=`' ⟨*Expr*⟩
| EPSILON

⟨*optional_multiple_value*⟩ ::= '`=`' ⟨*Expr*⟩ '`,`' ⟨*exprList*⟩
| EPSILON

⟨*arrayDeclaration*⟩ ::= '`[`' '`]`' ⟨*singleOrMulti_Array_Expr*⟩
| '`[`' ⟨*number*⟩ '`]`' ⟨*singleOrMulti_Array_Static*⟩

⟨*singleOrMulti_Array_Expr*⟩ ::= '`=`' ⟨*arrayInitial*⟩
| '`=`' ⟨*ProcedureCall*⟩
| '`[`' '`]`' '`=`' '`{`' ⟨*multiDimArrayInitial*⟩ '`}`'
| '`[`' '`]`' '`=`' ⟨*ProcedureCall*⟩

⟨*singleOrMulti_Array_Static*⟩ ::= '`[`' ⟨*number*⟩ '`]`'
| EPSILON

⟨*identifierDeclList*⟩ ::= ⟨*identifierDecl*⟩ '`,`' ⟨*identifierDeclList*⟩
| ⟨*identifierDecl*⟩

⟨*identifierDecl*⟩ ::= ⟨*identifier*⟩ ‘`:`’ ⟨*PrimitiveType*⟩

⟨*exprList*⟩ ::= ⟨*Expr*⟩ ‘`,`’ ⟨*exprList*⟩
| ⟨*Expr*⟩

⟨*arrayInitial*⟩ ::= ‘`{`’ ⟨*multipleExprs*⟩ ‘`}`’

⟨*multiDimArrayInitial*⟩ ::= ⟨*arrayInitial*⟩ ‘`,`’ ⟨*multiDimArrayInitial*⟩
| ⟨*arrayInitial*⟩

⟨*identifierDecl*⟩ ::= ⟨*identifier*⟩ ‘`:`’ ⟨*primitive_type*⟩

⟨*primitive_type*⟩ ::= ‘`int`’
| ‘`bool`’

⟨*functionDeclaration*⟩ ::= ⟨*identifier*⟩ ‘`(`’ ⟨*paramsOptional*⟩ ‘`)`’ ‘`:`’ ⟨*returnTypesOptional*⟩
⟨*Block*⟩

⟨*paramsOptional*⟩ ::= ⟨*identifier*⟩ ⟨*Type*⟩ ⟨*paramsList*⟩
| EPSILON

⟨*paramsList*⟩ ::= ‘`,`’ ⟨*identifier*⟩ ⟨*Type*⟩ ⟨*paramsList*⟩
| EPSILON

⟨*Type*⟩ :: ⟨*primitiveType*⟩ ⟨*typeArray*⟩

⟨*typeArray*⟩ ::= ‘`[`’ ‘`]`’ ⟨*typeMultiDimArray*⟩
| ‘`[`’ ‘`]`’
| EPSILON

⟨*typeMultiDimArray*⟩ ::= ‘`[`’ ‘`]`’
| EPSILON

⟨*returnTypesOptional*⟩ ::= ⟨*Type*⟩ ⟨*returnTypeList*⟩
| EPSILON

⟨*returnTypeList*⟩ ::= ‘`,`’ ⟨*Type*⟩ ⟨*returnTypeList*⟩
| EPSILON

⟨*block*⟩ ::= ‘`{`’ ⟨*statements*⟩ ‘`}`’

⟨*statements*⟩ ::= ⟨*statement*⟩ ⟨*statements*⟩
| EPSILON

⟨*statement*⟩ ::= ⟨*identifierStatement_All*⟩ ‘`;`’
| ⟨*Conditional*⟩
| ⟨*whileLoop*⟩
| ⟨*forLoop*⟩
| ⟨*return*⟩ ‘`;`’

⟨*identifierStatement_All*⟩ ::= ⟨*identifier*⟩ ‘:’ ⟨*primitiveType*⟩ ⟨*declaration*⟩
| ⟨*identifier*⟩ ⟨*assignOrMutate*⟩
| ⟨*identifier*⟩ ⟨*procedureCall*⟩
| ⟨*identifier*⟩ ⟨*multiAssign_Or_MultiCallAssign*⟩

⟨*identifierStmt_DeclAssignMutate*⟩ ::= ⟨*identifier*⟩ ‘:’ ⟨*primitiveType*⟩ ⟨*declaration*⟩
| ⟨*identifier*⟩ ⟨*assignOrMutate*⟩

⟨*assignOrMutate*⟩ ::= ⟨*assign*⟩
| ⟨*mutate*⟩
| ‘`[`’ ⟨*Expr*⟩ ‘`]`’ ⟨*arrayAssignOrMutate*⟩

⟨*assign*⟩ ::= ‘`=`’ ⟨*Expr*⟩

⟨*mutate*⟩ ::= + +
| - -

⟨*arrayAssignOrMutate*⟩ ::= ‘`[`’ ⟨*Expr*⟩ ‘`]`’ ⟨*multiDimArrayAssignOrMutate*⟩
| assign
| mutate

⟨*multiDimArrayAssignOrMutate*⟩ ::= assign
| mutate

⟨*multiAssign_Or_MultiCallAssign*⟩ ::= ‘,’ ⟨*identifierList*⟩ ‘=’ ⟨*Expr*⟩ ‘,’ ⟨*ExprList*⟩
| ‘,’ ⟨*identifierList*⟩ ‘=’ ⟨*ProcedureCall*⟩

⟨*identifierList*⟩ ::= ‘`,`’ ⟨*identifier*⟩ ⟨*identifierList*⟩
| ⟨*identifier*⟩

⟨*ExprList*⟩ ::= ⟨*Expr*⟩ ‘`,`’ ⟨*ExprList*⟩
| ⟨*Expr*⟩

⟨*Conditional*⟩ ::= ‘`if`’ ‘`(`’ ⟨*Expr*⟩ ‘`)`’ ⟨*block*⟩ ⟨*elseIfConditional*⟩ ⟨*elseConditional*⟩

⟨*elseIfConditional*⟩ ::= '`else`' '`if`' '`(`' ⟨*Expr*⟩ '`)`' ⟨*block*⟩ ⟨*elseIfConditional*⟩
| EPSILON

⟨*elseConditional*⟩ ::= '`else`' ⟨*block*⟩
| EPSILON

⟨*whileLoop*⟩ ::= '`while`' '`(`' ⟨*Expr*⟩ '`)`' ⟨*block*⟩

⟨*forLoop*⟩ ::= '`for`' '`(`' ⟨*identifierStmt_DeclAssignMutate*⟩ '`;`' ⟨*Expr*⟩ '`;`' ⟨*identifier*⟩
⟨*assignOrMutate*⟩ '`)`' ⟨*block*⟩

⟨*return*⟩ ::= '`return`' ⟨*ExprList*⟩?

⟨*Expr*⟩ ::= ⟨*Expr*⟩ BINOP ⟨*Expr*⟩
| UNOP ⟨*BaseExpr*⟩
| BaseExpr

⟨*ExprLL*⟩ ::= UNOP ⟨*ExprLL*⟩ ⟨*ExprPrime*⟩
| ⟨*BaseExpr*⟩ ⟨*ExprPrime*⟩

⟨*ExprPrime*⟩ ::= BINOP ⟨*ExprLL*⟩ ⟨*ExprPrime*⟩
| EPSILON

⟨*BaseExpr*⟩ ::= '(' ⟨*ExprLL*⟩ ')'
| ⟨*Loc*⟩
| ⟨*ProcedureCall*⟩
| ⟨*Lit*⟩

⟨*Loc*⟩ ::= ⟨*identifier*⟩ ⟨*LocArrayAccess*⟩

⟨*LocArrayAccess*⟩ ::= '`[`' ⟨*ExprLL*⟩ '`]`'
| EPSILON

⟨*ProcedureCall*⟩ ::= ⟨*identifier*⟩ '`(`' ⟨*ArgsOptional*⟩ '`)`'

⟨*ArgsOptional*⟩ ::= ⟨*ExprLL*⟩ ⟨*ArgsList*⟩
| EPSILON

⟨*ArgsList*⟩ ::= '`,`' ⟨*ExprLL*⟩ ⟨*ArgsList*⟩
| EPSILON

⟨*Lit*⟩ ::= ⟨*numberLiteral*⟩
| ⟨*boolLiteral*⟩
| ⟨*charLiteral*⟩
| ⟨*stringLiteral*⟩