BÁCH KHOA E-LEARNING

Câu hỏi 1

Không hoàn thành

Chấm điểm của 1,00

In CSEL, a program consists of many declarations: variable declation (vardecl), constant declaration (constdecl), function declaration (funcdecl). Given the grammar of CSEL as follows:

```
program: decl+ EOF;

cseltype: INT | FLOAT | BOOLEAN;

decl: vardecl decltail | constdecl decltail | funcdecl decltail;

decltail: vardecl decltail | constdecl decltail | funcdecl decltail | ;

vardecl: LET single_vardecls SEMI;

single_vardecls: single_vardecl single_vardecltail;

single_vardecl: ID COLON cseltype;

single_vardecltail: COMMA single_vardecl single_vardecltail | ;

constdecl: CONST single_constdecl SEMI;

single_constdecl: ID COLON cseltype EQ expr;

expr: INTLIT | FLOATLIT | BOOLEANLIT;

funcdecl: FUNCTION ID LR paramlist RR SEMI;

paramlist: single_vardecls | ;

LET: 'Let';

CONST: 'Constant';

FUNCTION: 'Function';

SEMI: ';';

COLON: ':';

COMMA: ',';

LR: '(';

RR: ')';

EQ: '=';

INT: 'Int';

FLOAT: 'Float';

BOOLEAN: 'Boolean';

INTLIT: [0-9]+;

FLOATLIT: [0-9]+ '.' [0-9]+;

BOOLEANLIT: 'True' | 'False';

ID: [a-zA-Z]+;

WS: [ \t\r\n\f]+ -> skip;
```

and AST classes as follows:

class Program(ABC): # decl: List[Decl]

class Type(ABC): pass

class IntType(Type)

class FloatType(Type)

class BooleanType(Type)

class LHS(ABC): pass

class Id(LHS): # name: str

class Decl(ABC): pass

class VarDecl(Decl): # id: Id, typ: Type

Thời gian còn lại 0:44:25

```
class ConstDecl(Decl): # id: Id, typ: Type, value: Expr

class FuncDecl(Decl): # name: Id, param: List[VarDecl]

class Exp(ABC): pass

class IntLit(Exp): # value: int

class FloatLit(Exp): # value: float

class BooleanLit(Exp): # value: bool
```

Please copy the following class into your answer and modify the bodies of its methods to generate the AST of a CSEL input?

```
class ASTGenerator(CSELVisitor):
    # Visit a parse tree produced by CSELParser#program.
    def visitProgram(self, ctx:CSELParser.ProgramContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParser#cseltype.
    def visitCseltype(self, ctx:CSELParser.CseltypeContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParser#decl.
    def visitDecl(self, ctx:CSELParser.DeclContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParser#decltail.
    def visitDecltail(self, ctx:CSELParser.DecltailContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParser#vardecl.
    def visitVardecl(self, ctx:CSELParser.VardeclContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParser#single_vardecls.
    def visitSingle_vardecls(self, ctx:CSELParser.Single_vardeclsContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParser#single_vardecl.
    def visitSingle_vardecl(self, ctx:CSELParser.Single_vardeclContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParser#single_vardecltail.
    def visitSingle_vardecltail(self, ctx:CSELParser.Single_vardecltailContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParser#constdecl.
    def visitConstdecl(self, ctx:CSELParser.ConstdeclContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParser#single_constdecl.
    def visitSingle_constdecl(self, ctx:CSELParser.Single_constdeclContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParser#expr.
    def visitExpr(self, ctx:CSELParser.ExprContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParser#funcdecl.
```

```
    def visitFuncdecl(self, ctx:CSELParser.FuncdeclContext):
        return self.visitChildren(ctx)

    # Visit a parse tree produced by CSELParser#paramlist.
    def visitParamlist(self, ctx:CSELParser.ParamlistContext):
        return self.visitChildren(ctx)
```

For example:

Test	Result
`"Let a: Int;"`	`Program([VarDecl(Id(a), IntType)])`

Answer: (penalty regime: 0, 5, 10, 15, 20, ... %)

```
from functools import reduce

class ASTGenerator(CSELVisitor):
    def visitProgram(self, ctx:CSELParser.ProgramContext):
        return Program(reduce(lambda prev, curr: prev + self.visit(curr), ctx.decl(), []))

    def visitCseltype(self, ctx:CSELParser.CseltypeContext):
        if ctx.INT(): return IntType()
        elif ctx.FLOAT(): return FloatType()
        return BooleanType()

    def visitDecl(self, ctx:CSELParser.DeclContext):
        if ctx.vardecl():
            return self.visit(ctx.vardecl()) + self.visit(ctx.decltail())
        elif ctx.constdecl():
            return self.visit(ctx.constdecl()) + self.visit(ctx.decltail())
        return self.visit(ctx.funcdecl()) + self.visit(ctx.decltail())

    def visitDecltail(self, ctx:CSELParser.DecltailContext):
        if ctx.getChildCount() == 0: return []
        elif ctx.vardecl():
            return self.visit(ctx.vardecl()) + self.visit(ctx.decltail())
        elif ctx.constdecl():
            return self.visit(ctx.constdecl()) + self.visit(ctx.decltail())
        return [self.visit(ctx.funcdecl())] + self.visit(ctx.decltail())

    def visitVardecl(self, ctx:CSELParser.VardeclContext):
        return self.visit(ctx.single_vardecls())

    def visitSingle_vardecls(self, ctx:CSELParser.Single_vardeclsContext):
        return [self.visit(ctx.single_vardecl())] + self.visit(ctx.single_vardecltail())

    def visitSingle_vardecl(self, ctx:CSELParser.Single_vardeclContext):
        return VarDecl(Id(ctx.ID().getText()), self.visit(ctx.cseltype()))

    def visitSingle_vardecltail(self, ctx:CSELParser.Single_vardecltailContext):
        if ctx.getChildCount() == 0: return []
        return [self.visit(ctx.single_vardecl())] + self.visit(ctx.single_vardecltail())


    def visitConstdecl(self, ctx:CSELParser.ConstdeclContext):
        return [self.visit(ctx.single_constdecl())]

    def visitSingle_constdecl(self, ctx:CSELParser.Single_constdeclContext):
        return ConstDecl(Id(ctx.ID().getText()), self.visit(ctx.cseltype()), self.visit(ctx.expr()))

    def visitExpr(self, ctx:CSELParser.ExprContext):
        if ctx.INTLIT():
            return IntLit(int(ctx.INTLIT().getText()))
        elif ctx.BOOLEANLIT():
            return BooleanLit(ctx.BOOLEANLIT().getText() == 'True')
        return FloatLit(float(ctx.FLOATLIT().getText()))

    def visitFuncdecl(self, ctx:CSELParser.FuncdeclContext):
        return FuncDecl(Id(ctx.ID().getText()), self.visit(ctx.paramlist()))
```

```
56
57      def visitParamlist(self, ctx:CSELParser.ParamlistContext):
58          if ctx.getChildCount() == 0: return []
59          return self.visit(ctx.single_vardecls())
60
```

Kiểm tra

◄ Programming Code: AST (15g00)

Chuyển tới...

Các video sửa Programming Code ►