

**BÁCH KHOA E-LEARNING**

[Trang của tôi](#) / [Khoá học](#) / [Học kỳ II năm học 2021-2022 \(Semester 2 - Academic year 2021-2022\)](#)

/ [Đại Học Chính Quy \(Bachelor program \(Full-time study\)\)](#)

/ [Khoa Khoa học và Kỹ thuật Máy tính \(Faculty of Computer Science and Engineering.\)](#) / [Khoa Học Máy Tính](#)

/ [Nguyên lý ngôn ngữ lập trình \(CO3005\) Trần Ngọc Bảo Duy \(DH\\_HK212\)](#) / Cây cú pháp trừu tượng - Abstract Syntax Tree (Buổi 4)

/ [Programming Code: AST \(extra exercise\)](#)

## Câu hỏi 1

Không hoàn thành

Chấm điểm của 1,00

In CSEL, a program consists of many declarations: variable declaration (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 CSELParse#program.
    def visitProgram(self, ctx:CSELParse.ProgramContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParse#cseltype.
    def visitCseltype(self, ctx:CSELParse.CseltypeContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParse#decl.
    def visitDecl(self, ctx:CSELParse.DeclContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParse#decltail.
    def visitDecltail(self, ctx:CSELParse.DecltailContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParse#vardecl.
    def visitVardecl(self, ctx:CSELParse.VardeclContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParse#single_vardecls.
    def visitSingle_vardecls(self, ctx:CSELParse.Single_vardeclsContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParse#single_vardecl.
    def visitSingle_vardecl(self, ctx:CSELParse.Single_vardeclContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParse#single_vardecltail.
    def visitSingle_vardecltail(self, ctx:CSELParse.Single_vardecltailContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParse#constdecl.
    def visitConstdecl(self, ctx:CSELParse.ConstdeclContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParse#single_constdecl.
    def visitSingle_constdecl(self, ctx:CSELParse.Single_constdeclContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParse#expr.
    def visitExpr(self, ctx:CSELParse.ExprContext):
        return self.visitChildren(ctx)
    # Visit a parse tree produced by CSELParse#funcdecl.

```

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

# Visit a parse tree produced by CSELParse#paramlist.
def visitParamlist(self, ctx:CSELParse.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, ... %)

```
1 from functools import reduce
2
3 class ASTGenerator(CSELVisitor):
4     def visitProgram(self, ctx:CSELParse.ProgramContext):
5         return Program(reduce(lambda prev, curr: prev + self.visit(curr), ctx.decl(), []))
6
7     def visitCseltype(self, ctx:CSELParse.CseltypeContext):
8         if ctx.INT(): return IntType()
9         elif ctx.FLOAT(): return FloatType()
10        return BooleanType()
11
12    def visitDecl(self, ctx:CSELParse.DeclContext):
13        if ctx.vardecl():
14            return self.visit(ctx.vardecl()) + self.visit(ctx.decltail())
15        elif ctx.constdecl():
16            return self.visit(ctx.constdecl()) + self.visit(ctx.decltail())
17        return self.visit(ctx.funcdecl()) + self.visit(ctx.decltail())
18
19    def visitDecltail(self, ctx:CSELParse.DecltailContext):
20        if ctx.getChildCount() == 0: return []
21        elif ctx.vardecl():
22            return self.visit(ctx.vardecl()) + self.visit(ctx.decltail())
23        elif ctx.constdecl():
24            return self.visit(ctx.constdecl()) + self.visit(ctx.decltail())
25        return [self.visit(ctx.funcdecl())] + self.visit(ctx.decltail())
26
27    def visitVardecl(self, ctx:CSELParse.VardeclContext):
28        return self.visit(ctx.single_vardecls())
29
30    def visitSingle_vardecls(self, ctx:CSELParse.Single_vardeclsContext):
31        return [self.visit(ctx.single_vardecl())] + self.visit(ctx.single_vardecltail())
32
33    def visitSingle_vardecl(self, ctx:CSELParse.Single_vardeclContext):
34        return VarDecl(Id(ctx.ID().getText()), self.visit(ctx.cseltype()))
35
36    def visitSingle_vardecltail(self, ctx:CSELParse.Single_vardecltailContext):
37        if ctx.getChildCount() == 0: return []
38        return [self.visit(ctx.single_vardecl())] + self.visit(ctx.single_vardecltail())
39
40
41    def visitConstdecl(self, ctx:CSELParse.ConstdeclContext):
42        return [self.visit(ctx.single_constdecl())]
43
44    def visitSingle_constdecl(self, ctx:CSELParse.Single_constdeclContext):
45        return ConstDecl(Id(ctx.ID().getText()), self.visit(ctx.cseltype()), self.visit(ctx.expr()))
46
47    def visitExpr(self, ctx:CSELParse.ExprContext):
48        if ctx.INTLIT():
49            return IntLit(int(ctx.INTLIT().getText()))
50        elif ctx.BOOLEANLIT():
51            return BooleanLit(ctx.BOOLEANLIT().getText() == 'True')
52        return FloatLit(float(ctx.FLOATLIT().getText()))
53
54    def visitFuncdecl(self, ctx:CSELParse.FuncdeclContext):
55        return FuncDecl(Id(ctx.ID().getText()), self.visit(ctx.paramlist()))
```

```
56 |  
57 | def visitParamlist(self, ctx:CSELParseParamlistContext):  
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 ▶

Copyright 2007-2021 Trường Đại Học Bách Khoa - ĐHQG Tp.HCM. All Rights Reserved.

Địa chỉ: Nhà A1- 268 Lý Thường Kiệt, Phường 14, Quận 10, Tp.HCM.

Email: [elearning@hcmut.edu.vn](mailto:elearning@hcmut.edu.vn)

Phát triển dựa trên hệ thống Moodle