Bạn đang đăng nhập với tên **Mai Văn Duyên** (Thoát ra)

BÁCH KHOA E-LEARNING

Trang chủ

Trang của tôi » Học kỳ I năm học 2020-2021 (Semester 1 - Academic year 2020-2021) » Đại Học Chính Qui (Bacherlor program (Full-time study)) » Khoa Khoa học và Kỹ thuật Máy tính (Faculty of Computer Science and Engineering) » Nguyên lý ngôn ngữ lập trình (CO3005)_Trần Ngọc Bảo Duy (DH_HK201) » Kiểu - Type (tuần 9) » Programming Code: Type

Đã bắt đầu vào lúc	Tuesday, 24 November 2020, 9:40 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Tuesday, 24 November 2020, 10:17 PM
Thời gian thực hiện	36 phút 44 giây
Điểm	3,00/3,00
Điểm	**10,00** của 10,00 (**100**%)

Câu hỏi **1**

Chính xác

Điểm 1,00 của 1,00

Given the AST declarations as follows:

class Exp(ABC): #abstract class

class BinOp(Exp): #op:str,e1:Exp,e2:Exp #op is +,-,*,/,&&,||, >, <, ==, or !=

class UnOp(Exp): #op:str,e:Exp #op is -, !

class IntLit(Exp): #val:int

class FloatLit(Exp): #val:float

class BoolLit(Exp): #val:bool

and the Visitor class is declared as follows:

class StaticCheck(Visitor):

    def visitBinOp(self,ctx:BinOp,o): pass

    def visitUnOp(self,ctx:UnOp,o):pass

    def visitIntLit(self,ctx:IntLit,o): pass

    def visitFloatLit(self,ctx,o): pass

    def visitBoolLit(self,ctx,o): pass

Rewrite the body of the methods in class StaticCheck to check the following type constraints:

- + , - and * accept their operands in int or float type and return float type if at least one of their operands is in float type, otherwise, return int type
- / accepts their operands in int or float type and returns float type
- !, && and || accept their operands in bool type and return bool type
- >, <, == and != accept their operands in any type but must in the same type and return bool type

If the expression does not conform the type constraints, the StaticCheck will raise exception TypeMismatchInExpression with the innermost sub-expression that contains type mismatch.

Your code starts at line 55

For example:

Test	Result
`BinOp("+",IntLit(3),BoolLit(True))`	`Type Mismatch In Expression: BinOp("+",IntLi`

Answer: (penalty regime: 10, 20, ... %)

```
class StaticCheck(Visitor):

    def visitBinOp(self,ctx:BinOp,o):
        le = self.visit(ctx.e1, o)
        re = self.visit(ctx.e2, o)
        if(ctx.op in ['+','-','*']):
            if(le == 'bool' or re == 'bool'):
                raise TypeMismatchInExpression(ctx)
            elif(le == 'float' or re == 'float'):
                return 'float'
            else: return 'int'
        elif(ctx.op == '/'):
            if(le == 'bool' or re == 'bool'):
                raise TypeMismatchInExpression(ctx)
            else: return 'float'
        elif(ctx.op in ['&&', '||']):
            if(le == 'bool' and re == 'bool'):
                return 'bool'
            else: raise TypeMismatchInExpression(ctx)
```

	Test	Expected
✓	`BinOp("+",IntLit(3),BoolLit(True))`	`Type Mismatch In Expression: BinOp("+",`

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **2**

Chính xác

Điểm 1,00 của 1,00

Given the AST declarations as follows:

class Program: #decl:List[VarDecl],exp:Exp

class VarDecl: #name:str,typ:Type

class Type(ABC): #abstract class

class IntType(Type)

class FloatType(Type)

class BoolType(Type)

class Exp(ABC): #abstract class

class BinOp(Exp): #op:str,e1:Exp,e2:Exp #op is +,-,*,/,&&,||, >, <, ==, or !=

class UnOp(Exp): #op:str,e:Exp #op is -, !

class IntLit(Exp): #val:int

class FloatLit(Exp): #val:float

class BoolLit(Exp): #val:bool

class Id(Exp): #name:str

and the Visitor class is declared as follows:

class StaticCheck(Visitor):

    def visitProgram(self,ctx:Program,o):pass

    def visitVarDecl(self,ctx:VarDecl,o): pass

    def visitBinOp(self,ctx:BinOp,o): pass

    def visitUnOp(self,ctx:UnOp,o):pass

    def visitIntLit(self,ctx:IntLit,o): pass

    def visitFloatLit(self,ctx,o): pass

    def visitBoolLit(self,ctx,o): pass

    def visitId(self,ctx,o): pass

Rewrite the body of the methods in class StaticCheck to check the following type constraints:

- + , - and * accept their operands in int or float type and return float type if at least one of their operands is in float type, otherwise, return int type
- / accepts their operands in int or float type and returns float type
- !, && and || accept their operands in bool type and return bool type
- >, <, == and != accept their operands in any type but must in the same type and return bool type
- the type of an Id is from the declarations, if the Id is not in the declarations, exception UndeclaredIdentifier should be raised with the name of the Id.

If the expression does not conform the type constraints, the StaticCheck will raise exception TypeMismatchInExpression with the innermost sub-expression that contains type mismatch.

Your code starts at line 90

For example:

Test	Result

Test	Result
`Program([],BinOp("+",IntLit(3),BoolLit(True)))`	`Type Mismatch In Expression: Bin`

Answer: (penalty regime: 10, 20, ... %)

```
class StaticCheck(Visitor):

    def visitProgram(self,ctx:Program,o):
        o = {}
        for x in ctx.decl:
            self.visit(x, o)
        self.visit(ctx.exp, o)
    def visitVarDecl(self,ctx:VarDecl,o):
        o[ctx.name] = ctx.typ

    def visitBinOp(self,ctx:BinOp,o):
        le = self.visit(ctx.e1, o)
        re = self.visit(ctx.e2, o)
        if(ctx.op in ['+','-','*']):
            if(type(le) is BoolType or type(re) is BoolType):
                raise TypeMismatchInExpression(ctx)
            elif(type(le) is FloatType or type(re) is FloatType):
                return FloatType()
            else: return IntType()
```

	Test	Expected
✓	`Program([],BinOp("+",IntLit(3),BoolLit(True)))`	`Type Mismatch In Expressior`

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **3**

Chính xác

Điểm 1,00 của 1,00

Given the AST declarations as follows:

class Program: #decl:List[VarDecl],stmts:List[Assign]

class VarDecl: #name:str

class Assign: #lhs:Id,rhs:Exp

class Exp(ABC): #abstract class

class BinOp(Exp): #op:str,e1:Exp,e2:Exp #op is +,-,*,/,+.,-.,*.,/., &&,||, >, >., >b, =, =., =b

class UnOp(Exp): #op:str,e:Exp #op is -,-., !,i2f, floor

class IntLit(Exp): #val:int

class FloatLit(Exp): #val:float

class BoolLit(Exp): #val:bool

class Id(Exp): #name:str

and the Visitor class is declared as follows:

```
class StaticCheck(Visitor):
    def visitProgram(self,ctx:Program,o):pass
    def visitVarDecl(self,ctx:VarDecl,o): pass
    def visitAssign(self,ctx:Assign,o): pass
    def visitBinOp(self,ctx:BinOp,o): pass
    def visitUnOp(self,ctx:UnOp,o):pass
    def visitIntLit(self,ctx:IntLit,o): pass
    def visitFloatLit(self,ctx,o): pass
    def visitBoolLit(self,ctx,o): pass
    def visitId(self,ctx,o): pass
```

Rewrite the body of the methods in class StaticCheck to infer the type of identifiers and check the following type constraints:

- + , - , *, / accept their operands in int type and return int type
- +., -., *., /. accept their operands in float type and return float type
- > and = accept their operands in int type and return bool type
- >. and =. accept their operands in float type and return bool type
- !, &&, ||, >b and =b accept their operands in bool type and return bool type
- i2f accepts its operand in int type and return float type
- floor accept its operand in float type and return int type
- In an Assign, the type of lhs must be the same as that of rhs, otherwise, the exception TypeMismatchInStatement should be raised together with the Assign
- the type of an Id is inferred from the above constraints in the first usage,
    - if the Id is not in the declarations, exception UndeclaredIdentifier should be raised together with the name of the Id, or
    - If the Id cannot be inferred in the first usage, exception TypeCannotBeInferred should be raised together with the name of the identifier

If the expression does not conform the type constraints, the StaticCheck will raise exception TypeMismatchInExpression with the assign statement where contains the type-unresolved identifier.

Your code starts at line 95

For example:

Test
`Program([VarDecl("x")],[Assign(Id("x"),BinOp("*",BinOp("+",Id("x"),IntLit(3.4)),`
`Program([VarDecl("x"),VarDecl("y"),VarDecl("z")],[Assign(Id("x"),BinOp(">b",Bin`
`Program([VarDecl("x"),VarDecl("y")],[Assign(Id("x"),Id("y"))])`

Answer: (penalty regime: 10, 20, ... %)

```
class IntType(ABC):
    pass
class FloatType(ABC):
    pass
class BoolType(ABC):
    pass
class StaticCheck(Visitor):

    def visitProgram(self,ctx:Program,o):
        o = [[x.name, None] for x in ctx.decl]
        [self.visit(x, o) for x in ctx.stmts]

    def visitVarDecl(self,ctx:VarDecl,o): pass

    def visitAssign(self,ctx:Assign,o):
        right = self.visit(ctx.rhs, o)
        left = self.visit(ctx.lhs, o)
        if not left and not right:
            raise TypeCannotBeInferred(ctx)
```

	Test
✓	`Program([VarDecl("x")],[Assign(Id("x"),BinOp("*",BinOp("+",Id("x"),IntLit(`
✓	`Program([VarDecl("x"),VarDecl("y"),VarDecl("z")],[Assign(Id("x"),BinOp(">b'`
✓	`Program([VarDecl("x"),VarDecl("y"),VarDecl("z")],[Assign(Id("x"),UnOp("!",`
✓	`Program([VarDecl("x"),VarDecl("y")],[Assign(Id("x"),Id("y"))])`

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.