Nguyên Lý Ngôn Ngữ Lập Trình

Bài Tập Lớn

ZCODE

nhóm thảo luận CSE
https://www.facebook.com/groups/211867931379013

Tp. Hồ Chí Minh, Tháng 1/2024

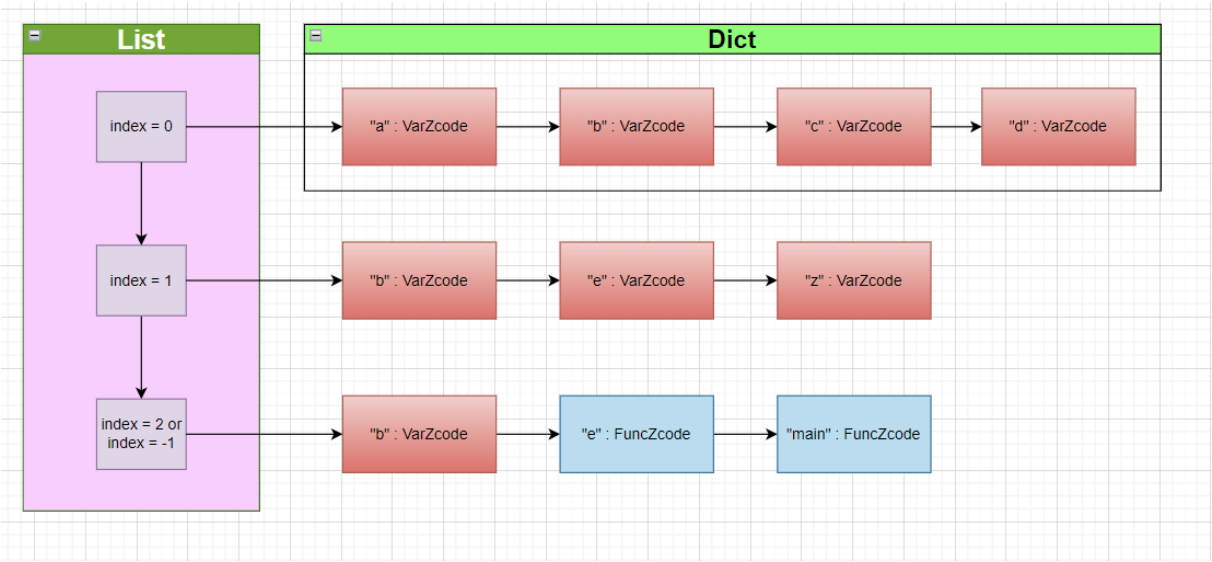
Mục lục

1 Lý thuyết BTL3

1.1 Thiết kế param

Ta có 2 phần là toàn cục và cục bộ :

- Toàn cục sẽ bao hôm các hàm và các biến khai báo ngoài hàm
- Cục bộ thì sẽ vào các biến được khai báo trong các block của hàm, for, if, block và param của hàm




```
# Tạo danh sách
my_list = ['Var:Zcode', 'Func:Zcode', 'Var:Zcode']

# Tạo từ điển
my_dict = {
    'a': 'Var:Zcode',
    'b': 'Var:Zcode'
}

# cấu trúc dữ liệu param được thiết kế
param = [
    [{'a': 'Var:Zcode', 'b': 'Var:Zcode'}],
    [{'a': 'Var:Zcode', 'c': 'Var:Zcode'}],
    [{'a': 'Var:Zcode', 'c': 'Func:Zcode'}]
]
```

- Để dễ tìm kiếm ta thiết kế Dict đối với một tầm vực (block hay toàn cục)
- List lưu trữ từng tầm vực của biến được lưu trữ với index 0 đến index cuối thì tầm vực sẽ từ cục bộ ra tới toàn cục
- tại index cuối thì sẽ là toàn cục gồm khai báo biến toàn cục và hàm
- các index còn lại sẽ là các biến cục bộ
- Khi tìm kiếm thì ta tìm kiếm từ cục bộ đến toàn cục nghĩa là từ index 0 đến index cuối

1.2 Class Zcode

```
class FuncZcode(Zcode):
    def __init__(self, param = [], typ = None, body = False):
        self.param = param
        self.typ = typ
        self.body = body

class VarZcode(Zcode):
    def __init__(self, typ = None):
        self.typ = typ
```

- **param** là danh sách các *type* của từng param được truyền vào
- **typ của hàm** là kiểu dữ liệu trả về nếu chưa xác định thì để *None*
- **body** kiểm tra xem declaration-only part hay không nếu có body thì true nếu không thì là false
- **typ của biến** là kiểu dữ của biến nếu chưa xác định thì để *None*

```
func VoTien(number a, string b)
    return 1.1
-> FuncZcode([NumberType(), StringType()], NumberType(), True)

func VoTien()
-> FuncZcode([], None, True)

number VoTien
-> VarZcode(NumberType())

dynamic VoTien
-> VarZcode(None)
```

1.3 Cách dùng Prama

1. Tìm kiếm tên có ở hàng đầu tiên cục bộ gần nhất hay không

```
// name có tồn tại trong dict đầu tiên trong list
if param[0].get(name):
    return True
else:
    return Flase
```

2. Tìm kiếm tên có ở tầm vực gần nhất

```
for item in param:
    Id = item.get(name)
    if Id is not Node:
        return True
return false
```

3. sử dụng type để kiểm tra đang là hàm hay biến

```
type(item) is VarZcode # kiểm tra có phải là khai báo biến không

type(item) is FuncZcode # kiểm tra có phải là khai báo hàm không
```

2 Task 1

2.1 Xử lí Type và Literal

```
# có sẵn trong ast
def visitNumberType(self, ast, param): return ast
def visitBoolType(self, ast, param): return ast
def visitStringType(self, ast, param): return ast
def visitArrayType(self, ast, param): return ast

# xử lí phần Litaral
def visitNumberLiteral(self, ast, param): return NumberLiteral()
def visitBooleanLiteral(self, ast, param): return BooleanLiteral()
def visitStringLiteral(self, ast, param): return StringLiteral()

# Phần này xử lí ở task sau dùng đệ dệ quy quy quy quy !!!
def visitArrayLiteral(self, ast, param):
    pass
```

2.2 init

```
self.ast = ast
self.BlockFor = 0
self.function = None
self.io = [{
        "readNumber" : FuncZcode([], NumberType(), True),
        "readBool" : FuncZcode([], BoolType(), True),
        "readString" : FuncZcode([], StringType(), True),
        "writeNumber" : FuncZcode([NumberType()], VoidType(), True),
        "writeBool" : FuncZcode([BoolType()], VoidType(), True),
        "writeString" : FuncZcode([StringType()], VoidType(), True)
        }]
```

- **self.ast** cây đã được xây ở BTL2
- **self.BlockFor** Hiện tại có ở trong vòng for hay không nếu ở thì đang ở vòng for thứ mấy
- **self.function** hàm hiện tại đang xử lí dùng biến FuncZcode
- **self.io** này dùng thư viện IO

Hàm để kiểm tra thử thành thông hay không với gọi hàm self.visit(self.ast, self.io) với ast là cây ở BTL2 và io là param đầu tiên mặc định của BTL này

```
    def check(self):
        self.visit(self.ast, self.io)
        return "successful"
```

2.3 Các hàm phụ trong việc so sánh Type

```
def comparType(self, LHS, RHS):
    pass

def comparListType(self, LHS, RHS):
    pass
```

- **comparType(self, LHS, RHS)** truyền vào 2 *Type* sẽ có thể là **NumberType, BoolType, StringType, VoidType, ArrayType** chú ý ArrayType
- **comparListType(self, LHS, RHS):** truyền vào 2 danh sách có kích thước khác nhau

2.4 Xử lí Task 1

Các lỗi cần nén

- **2.1 Redeclared Variable/ Parameter/ Function** The declaration must be unique in its scope which is formally described as in ZCode specification. Otherwise, the exception Redeclared(<kind>, <identifier>) is released, where <kind> is the kind (Variable/Parameter/Function) of the identifier in the second declaration.
- **2.2 Undeclared Identifier/Function**
  - The exception Undeclared(Identifier(), <identifier-name>) is released when there is an identifier is used but its declaration cannot be found. The identifier can be a variable or a parameter.
  - Undeclared(Function(), <function-name>) is released if there do not exist any function with that name. The function usage (as the function call) could not be allowed before its declaration.
- **2.6 No definition for a function** In ZCode, a function could be introduced as a declaration without the body (as the definition). If exists a function without the definition in the program, the exception NoDefinition(<name>), <name> is the name of function with the first appearance is not defined.
- **2.7 Break/Continue not in loop** A break/continue statement must be inside directly or indirectly a loop otherwise the exception MustInLoop(<statement>) must be thrown.
- **2.8 No entry point** There must be a function whose name is main without any parameter and return nothing in a ZCode program. Otherwise, the exception NoEntryPoint() is released.

Các hàm cần xử lí

1. **visitProgram(self, ast, param)**
   - kiểm tra lỗi NoDefinition()
   - Kiểm tra lỗi NoEntryPoint()
2. **visitVarDecl(self, ast, param)**
   - kiểm tra lỗi Redeclared ()
3. **visitFuncDecl(self, ast, param)**
   - kiểm tra lỗi Redeclared đối với hàm và Param
4. **visitId(self, ast, param)**
   - kiểm tra lỗi Undeclared đối với biến
5. **visitCallStmt(self, ast, param) và visitCallExpr(self, ast, param)**
   - kiểm tra lỗi Undeclared đối với hàm
6. **visitContinue(self, ast, param) và visitBreak(self, ast, param)**
   - kiểm tra lỗi MustInLoop có trong for hay không

3 Các Khóa Học HK232

nhóm thảo luận CSE

https://www.facebook.com/groups/211867931379013

- Lớp BTL1 + GK + LAB + Lý thuyết + Harmony của môn DSA HK232
- Lớp BTL2 + CK + LAB + Lý thuyết + Harmony của môn DSA HK232
- Lớp BTL1 + LAB + Lý thuyết + Harmony của môn KTLT HK232
- Lớp BTL2 + LAB + Lý thuyết + Harmony của môn KTLT HK232
- Lớp BTL1 + BTL2 + GK + Harmony của môn PPL HK232
- Lớp BTL3 + BTL4 + CK + Harmony của môn PPL HK232

CHÚC CÁC EM HỌC TỐT