VÕ TIẾN
Thảo luận kiến thức CNTT trường BK về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

Nguyên Lý Ngôn Ngữ Lập Trình (PPL)

PPL2 - HK242

Task 1 - Error Redeclared

Thảo luận kiến thức CNTT trường BK
về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

TP. HỒ CHÍ MINH, THÁNG 1/2025

Mục lục

1 Khái niệm Symbol

Lớp Symbol đại diện cho một thực thể trong bảng ký hiệu (symbol table), bao gồm các biến, hằng số, hàm, phương thức, hoặc kiểu dữ liệu. Mỗi Symbol lưu trữ thông tin về tên, kiểu dữ liệu, và giá trị của nó (nếu có).

```
class Symbol:
    def __init__(self, name, mtype, value=None):
        self.name = name        # Tên của symbol (biến, hàm, phương thức, v.v.)
        self.mtype = mtype      # Kiểu dữ liệu hoặc kiểu phương thức/hàm (MType)
        self.value = value      # Giá trị của symbol (nếu có)

    def __str__(self):
        return "Symbol(" + str(self.name) + "," + str(self.mtype) + ("" if self.value is None else
```

Các thành phần chính:

1. **name**: Là tên của symbol, có thể là tên biến, tên hàm, hoặc tên phương thức.
2. **mtype** (kiểu dữ liệu hoặc kiểu phương thức):
   - Nếu Symbol là một biến/hằng số, mtype sẽ chứa kiểu dữ liệu của nó (IntType, FloatType, v.v.).
   - Nếu Symbol là một hàm hoặc phương thức, mtype sẽ là một đối tượng MType, chứa danh sách tham số và kiểu trả về.
3. **value** btl4 mới dùng tới

2 Ví Dụ List[List[Symbol]]

```
const b = 2;
func Votien (b int) {
    for var a = 1; a < 1; a += 1 {
        const b = 1;
    }
    const a = true;
}
const a = "3";
```

1. **Hàng 1** function sẽ có phạm vi toàn bộ trương trình, biến toàn cục chỉ có phạm vi từ khi khai báo đến cuối

   ```
   [
       [Symbol("b", IntType(), 2), Symbol("Votien", MType([IntType()], VoidType()))]
   ]
   ```

2. **Hàng 2** param sẽ có phạm vi riêng với toàn cục và cục bộ

   ```
   [
       [Symbol("b", IntType())],
       [Symbol("b", IntType(), 2), Symbol("Votien", MType([IntType()], VoidType()))]
   ]
   ```

3. **Hàng 3** Khai báo trong for sẽ có một tầm vực khác với block bên ngoài

   ```
   [
       [Symbol("a", IntType(), 1)],
       [],
   ```

```
    [Symbol("b", IntType())],
    [Symbol("b", IntType(), 2), Symbol("Votien", MType([IntType()], VoidType()))]
]
```

4. **Hàng 4** chung tầm vực với for

```
[
    [Symbol("a", IntType(), 1), Symbol("b", IntType(), 1)],
    [],
    [Symbol("b", IntType())],
    [Symbol("b", IntType(), 2), Symbol("Votien", MType([IntType()], VoidType()))]
]
```

5. **Hàng 5** Tầm vực for bị hủy

```
[
    [],
    [Symbol("b", IntType())],
    [Symbol("b", IntType(), 2), Symbol("Votien", MType([IntType()], VoidType()))]
]
```

6. **Hàng 6** Tầm vực của hàm thêm phần tử mới

```
[
    [Symbol("a", BoolType(), True)],
    [Symbol("b", IntType())],
    [Symbol("b", IntType(), 2), Symbol("Votien", MType([IntType()], VoidType()))]
]
```

7. **Hàng 7** Tầm vực của hàm và param bị hủy

```
[
    [Symbol("b", IntType(), 2), Symbol("Votien", MType([IntType()], VoidType()))]
]
```

8. **Hàng 8** Tầm vực toàn cực thêm thành viên mới

```
[
    [Symbol("b", IntType(), 2), Symbol("Votien", MType([IntType()], VoidType())), Symbol("a",
]
```

3 self.lookup và FP

- **self.lookup** Hàm lookup được sử dụng để tìm kiếm một phần tử trong danh sách lst dựa trên một điều kiện do hàm func xác định. Nếu tìm thấy phần tử có tên trùng với name, nó sẽ trả về phần tử đó. Nếu không tìm thấy, nó trả về None.

```
def lookup(self,name,lst,func):
    for x in lst:
        if name == func(x):
            return x
    return None

symbols = [
    Symbol("x", IntType()),
    Symbol("y", FloatType()),
    Symbol("z", BoolType())
]

name_to_find = "y"
found_symbol = lookup(name_to_find, symbols, lambda x: x.name)
```

- **Hàm reduce** trong Python được dùng để thu gọn một danh sách thành một giá trị duy nhất bằng cách áp dụng một hàm hai đối số tuần tự lên các phần tử.

  ```
  reduce(lambda acc, ele: [[self.visit(ele, acc)] + acc[0]] + acc[1:], ast.member, [[]] + c)
  ```

  - **lambda acc, ele: ...** – Hàm được áp dụng lặp đi lặp lại.
    - **acc** (Accumulator) là danh sách các scope hiện tại.
    - **ele** là từng phần tử **ast.member**.
  - **ast.member** – Danh sách các khai báo trong **Block**.
  - [[]] + **c** – Danh sách các scope khởi đầu:
    - [] là scope mới trong **Block**.
    - + **c** giữ nguyên các scope trước đó (global hoặc function).

4 Attribute StaticChecker

```
self.ast = ast
self.struct: list[StructType] = []
self.inteface: list[InterfaceType] = []
self.env = [
    [
        Symbol("getInt",MType([],IntType())),
        Symbol("putIntLn",MType([IntType()],VoidType()))
    ]
]
self.function_current: FuncDecl = None
self.struct_current: StructType = None
```

- **self.struct** danh cách các Type struct được khai báo
- **self.inteface** danh cách các Type inteface được khai báo
- **self.env** các hàm mặt định của trương trình
- **self.function_current** hàm hiện tại trương trình đang thực thi (đang visit block của hàm nào)
- **self.struct_current** phương thức hiện tại trương trình đan gthuwjc thi (đang visit với block của phương thức của struct nào)