

Thực hành  
Xây dựng chương trình dịch

## Bài 3: Quản lý phạm vi

ONE LOVE. ONE FUTURE.

- Kiểm tra sự trùng lặp khi khai báo đối tượng
- Kiểm tra tham chiếu tới các đối tượng

- Một đối tượng có tên hợp lệ nếu như tên đó chưa từng được sử dụng trong phạm vi hiện tại.
- Sử dụng hàm sau để kiểm tra tên hợp lệ  
`void checkFreshIdent(char *name);`

- Việc kiểm tra một tên hợp lệ được thực hiện khi:
  - Khai báo hằng
  - Khai báo kiểu người dùng định nghĩa
  - Khai báo biến
  - Khai báo tham số hình thức
  - Khai báo hàm
  - Khai báo thủ tục

- Kiểm tra một hằng đã khai báo được thực hiện khi có tham chiếu tới hằng đó
  - Khi duyệt một hằng không dấu
  - Khi duyệt một hằng số
- Lưu ý tới phạm vi của hằng số: nếu hằng không được định nghĩa trong phạm vi hiện tại thì phải tìm kiếm ở những phạm vi rộng hơn
- Giá trị của hằng số đã khai báo sẽ được sử dụng để tạo ra giá trị của hằng số đang duyệt
  - Chia sẻ giá trị hằng → `duplicateConstantValue`

- Kiểm tra một kiểu đã khai báo được thực hiện khi có tham chiếu tới kiểu đó
  - Khi duyệt kiểu: `compileType`
- Lưu ý tới phạm vi của kiểu: nếu kiểu không được định nghĩa trong phạm vi hiện tại thì phải tìm kiếm ở những phạm vi rộng hơn
- Kiểu thực tế của định danh kiểu được tham chiếu sẽ được sử dụng để tạo ra kiểu đang duyệt
  - Chia sẻ → `duplicateType`

- Kiểm tra một biến đã khai báo được thực hiện khi có tham chiếu tới biến đó
  - Trong câu lệnh gán
  - Trong câu lệnh for
  - Trong khi duyệt factor
- Lưu ý tới phạm vi của biến: nếu biến không được định nghĩa trong phạm vi hiện tại thì phải tìm kiếm ở những phạm vi rộng hơn

- Khi một định danh xuất hiện bên trái của biểu thức gán hoặc trong factor, định danh đó có thể tương ứng
  - tên hàm hiện tại
  - Một biến đã khai báo
    - Nếu biến khai báo có kiểu mảng, theo sau tên biến phải có chỉ số của mảng
- Lưu ý phân biệt biến với tham số và tên hàm hiện tại



- Kiểm tra một hàm đã khai báo được thực hiện khi có tham chiếu tới hàm đó
  - Vế trái của lệnh gán (hàm hiện tại)
  - Trong một factor (khi đó cần có danh sách tham số đi kèm)
- Lưu ý tới phạm vi của hàm: nếu hàm không được định nghĩa trong phạm vi hiện tại thì phải tìm kiếm ở những phạm vi rộng hơn
- Một số hàm toàn cục: READC, READI

## Khi tên hàm xuất hiện ở vế trái của lệnh gán...

```
function f(x:integer):integer
```

```
.....
```

```
    function g(y: integer):integer
```

```
    begin
```

```
        f:= y*y (* lenh gán cho ten ham khac- ko hop le*)
```

```
    end;
```

```
.....
```

Tên hàm hiện hành: tên của đối tượng là owner của phạm vi hiện hành



- Kiểm tra một thủ tục đã khai báo được thực hiện khi có tham chiếu tới thủ tục đó
  - Lệnh gọi
- Lưu ý tới phạm vi của thủ tục: nếu thủ tục không được định nghĩa trong phạm vi hiện tại thì phải tìm kiếm ở những phạm vi rộng hơn
- Một số thủ tục toàn cục: WRITEI, WRITEC, WRITELN

- ERR\_UNDECLARED\_IDENT
- ERR\_UNDECLARED\_CONSTANT
- ERR\_UNDECLARED\_TYPE
- ERR\_UNDECLARED\_VARIABLE
- ERR\_INVALID\_RETURN
- ERR\_UNDECLARED\_PROCEDURE
- ERR\_DUPLICATE\_IDENT

# Hàm thực hiện luật phạm vi gần nhất: lookupObject

```
Object* lookupObject(char *name)

{  Scope* scope = symtab->currentScope;
   Object* obj;
   while (scope != NULL) {
       obj = findObject(scope->objList, name);
       if (obj != NULL) return obj;
       scope = scope->outer;
   }
   obj = findObject(symtab->globalObjectList, name);
   if (obj != NULL) return obj;
   return NULL;
}
```

- Hoàn thiện các hàm đánh dấu TODO trong symtab.c
- Tìm hiểu lại cấu trúc của bộ parser (có thay đổi)
- Bổ sung các đoạn code vào những hàm có đánh dấu TODO để thực hiện các công việc đăng ký đối tượng
- Biên dịch và thử nghiệm với các ví dụ mẫu

- Lập trình cho các hàm sau trong tệp semantics.c
  - checkDeclaredIdent
  - checkDeclaredVariable
  - checkDeclaredProcedure
  - checkDeclaredLValueIdent
- Bổ sung các đoạn code vào những chỗ có đánh dấu TODO hoặc những chỗ cần thiết để thực hiện các công việc kiểm tra khi đối tượng được khai báo hay sử dụng
- Biên dịch và thử nghiệm với các ví dụ mẫu