ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Thực hành
Xây dựng chương trình dịch

Bài 3. Phân tích ngữ nghĩa

Project 3: Quản lý phạm vi

Nội dung

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

Kiểm tra tên hợp lệ (1)

- 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);
```

Kiểm tra tên hợp lệ (2)

- 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 hằng số đã khai báo

- 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 kiểu đã khai báo

- 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 biến đã khai báo

- 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

Kiểm tra biến đã khai báo

- 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 hàm đã khai báo

- Kiểm tra một hàm đã khai báo được thực hiện khi có tham chiếu tới hàm đó
  - Về 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

```
function f(x:integer):integer
 …..
        function g(y: integer):integer
        begin
        f:= y*y  (* lenh gan 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 thủ tục đã khai báo

- 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

Các mã lỗi

- ERR_UNDECLARED_IDENT
- ERR_UNDECLARED_CONSTANT
- ERR_UNDECLARED_TYPE
- ERR_UNDECLARED_VARIABLE
- ERR_UNDECLARED_FUNCTION
- ERR_UNDECLARED_PROCEDURE
- ERR_DUPLICATE_IDENT

Nhiệm vụ thực hành

- Lập trình cho các hàm sau trong tệp semantics.c
  - checkFreshIdent
  - checkDeclaredIdent
  - checkDeclaredConstant
  - checkDeclaredType
  - checkDeclaredVariable
  - checkDeclaredProcedure
  - checkDeclaredFunction
  - checkDeclaredLValueIdent
- Biên dịch và thử nghiệm với các ví dụ mẫu

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;
}
```