ĐẠI HỌC QUỐC GIA - THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÀNH PHỐ HỒ CHÍ MINH
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

NGUYÊN TẮC LẬP TRÌNH - CO3005

BÀI 3

Trình kiểm tra tĩnh

THÀNH PHỐ HỒ CHÍ MINH, 2/2024

BÀI 3

Phiên bản 1.0

Sau khi hoàn thành nhiệm vụ này, bạn sẽ có thể

- giải thích các nguyên tắc làm thế nào trình biên dịch có thể kiểm tra một số ràng buộc ngữ nghĩa như khả năng tương thích kiểu, ràng buộc phạm vi,... và
- viết một chương trình Python trung bình (300 - 500 dòng mã) để thực hiện điều đó.

1 Đặc điểm kỹ thuật

Trong bài tập này, bạn được yêu cầu viết trình kiểm tra tĩnh cho chương trình viết bằng D96. Để hoàn thành nhiệm vụ này, bạn cần:

- Đọc kỹ thông số kỹ thuật của ngôn ngữ ZCode
- Tải về và giải nén tập tin task3.zip
- Nếu bạn tự tin với Bài tập 2 của mình, hãy sao chép ZCode.g4 của bạn vào bộ phân tích cú pháp src/main/zcode/- và ASTGeneration.py của bạn vào src/main/zcode/astgen và bạn có thể kiểm tra Bài tập 3 của mình bằng cách sử dụng đầu vào ZCode như phần đầu tiên hai bài kiểm tra (400-401).
- Mặt khác (nếu bạn chưa hoàn thành Bài tập 2 hoặc bạn không tự tin với Bài tập 2 của mình), đừng lo lắng, chỉ cần nhập AST làm đầu vào cho bài kiểm tra của bạn (như bài kiểm tra 402-403).
- Sửa đổi StaticCheck.py trong src/main/zcode/checker để triển khai trình kiểm tra tĩnh và sửa đổi CheckSuite.py trong src/test để triển khai 100 trường hợp thử nghiệm nhằm kiểm tra mã của bạn.

2 Trình kiểm tra tĩnh

Trình kiểm tra tĩnh đóng một vai trò quan trọng trong các trình biên dịch hiện đại. Nó kiểm tra trong thời gian biên dịch xem chương trình có tuân thủ các ràng buộc ngữ nghĩa theo đặc tả ngôn ngữ hay không. Trong bài tập này, bạn được yêu cầu triển khai trình kiểm tra tĩnh cho ngôn ngữ ZCode.

Đầu vào của bộ kiểm tra nằm trong AST của chương trình ZCode, tức là đầu ra của bài tập 2.
Đầu ra của trình kiểm tra sẽ không có gì nếu đầu vào được kiểm tra đúng, nếu không, thông báo lỗi sẽ được đưa ra và trình kiểm tra tĩnh sẽ dừng ngay lập tức.

Đối với mỗi lỗi ngữ nghĩa, học sinh nên ném ngoại lệ tương ứng được đưa ra trong StaticError.py vào trong thư mục src/main/zcode/checker/ để đảm bảo rằng nó sẽ được in ra giống như mong đợi. Mỗi trường hợp kiểm thử có nhiều nhất một loại lỗi. Các ràng buộc ngữ nghĩa cần thiết để kiểm tra nhiệm vụ này như sau.

2.1 Khai báo lại Biến/Tham số/Hàm

Tuyên bố phải là duy nhất trong phạm vi của nó và được mô tả chính thức như trong đặc tả ZCode. Nếu không, ngoại lệ Redeclared(<kind>, <identifier>) sẽ được giải phóng, trong đó <kind> là loại (Biến/Thông số/Hàm) của mã định danh trong khai báo thứ hai.

2.2 Mã định danh/Hàm không được khai báo

- Ngoại lệ Undeclared(Identifier(), <identifier-name>) được giải phóng khi có một mã định danh được sử dụng nhưng không thể tìm thấy khai báo của nó. Mã định danh có thể là một biến hoặc một tham số.
- Undeclared(Function(), <function-name>) được giải phóng nếu không tồn tại bất kỳ hàm nào có tên đó. Việc sử dụng hàm (như lệnh gọi hàm) không được phép trước khi khai báo.

2.3 Kiểu không khớp trong biểu thức

Một biểu thức phải tuân thủ các quy tắc loại cho biểu thức, nếu không, ngoại lệ TypeMis-matchInExpression(<biểu thức>) sẽ được giải phóng. Các quy tắc loại cho biểu thức như sau:

- Đối với chỉ số mảng (toán tử chỉ mục) E1[E2], E1 phải ở dạng mảng và E2 phải là danh sách số.
- Đối với biểu thức nhị phân và một ngôi, các quy tắc loại được mô tả trong đặc tả ZCode.
- Đối với một lệnh gọi hàm <function-name>(<args>), callee <method name> phải có kiểu trả về non-void (VoidType là một lớp biểu thị không trả về bất kỳ thứ gì trong một hàm). Các quy tắc loại cho các đối số và tham số giống như các quy tắc được đề cập trong lệnh gọi thủ tục.

2.4 Loại không thể suy ra

Mã định danh có thể ở dạng không xác định dưới dạng biến ở dạng var và dạng động hoặc kiểu trả về của hàm. Nếu một mã định danh được sử dụng nhưng loại của nó chưa được suy ra thì ngoại lệ TypeCannotBeInferred(<statement>) sẽ được giải phóng. Để suy ra kiểu của biến hoặc kiểu trả về của hàm, ZCode đọc chương trình từ đầu đến cuối và áp dụng các quy tắc sau:

- Khi một biến được khởi tạo trong phần khai báo, kiểu của biến đó cũng là kiểu của biểu thức được khởi tạo.

- Loại mã định danh (biến hoặc hàm) phải được suy ra trong lần xuất hiện đầu tiên khi sử dụng mã định danh và không thể thay đổi. Nếu loại của nó không thể được suy ra trong lần sử dụng đầu tiên, câu lệnh trong cùng chứa lần sử dụng đầu tiên của mã định danh sẽ được gửi cùng với ngoại lệ.

- Nếu một biểu thức có thể được suy ra theo kiểu nào đó nhưng một số thành phần của nó không thể được suy ra theo bất kỳ kiểu nào, thì câu lệnh trong cùng chứa thành phần chưa được giải quyết kiểu sẽ được liên kết với ngoại lệ. Ví dụ: biểu thức ở phía bên phải của câu lệnh y <- a + foo(x) có thể được suy ra để nhập số vì kết quả của + thuộc loại number và y, a và kiểu trả về của foo cũng có thể là được suy ra kiểu số, nhưng chúng ta không thể suy ra kiểu x, khi đó ngoại lệ sẽ xuất hiện với câu lệnh gán.

- Câu lệnh gọi tới một hàm chưa được giải quyết kiểu là hợp lệ khi tất cả các kiểu tham số của nó có thể được suy ra bởi các kiểu đối số tương ứng và kiểu trả về của nó có thể được suy ra thành VoidType. Nếu tồn tại ít nhất một tham số loại chưa được giải quyết, ngoại lệ sẽ được đưa ra bằng câu lệnh gọi. Lưu ý rằng nếu số lượng đối số không bằng số lượng đối số thì ngoại lệ liên quan trong Phần 2.5 sẽ được đưa ra.

- Lệnh gọi hàm tới một hàm loại chưa được giải quyết kiểu là hợp lệ nếu tất cả các kiểu tham số của nó và kiểu trả về có thể được giải quyết. Ngược lại, câu lệnh trong cùng chứa lệnh gọi hàm sẽ được liên kết với ngoại lệ. Lưu ý rằng nếu số lượng đối số không giống với số lượng đối số thì ngoại lệ liên quan trong Phần 2.3 sẽ được đưa ra.

- Kiểu của cả hai bên của một phép gán phải giống nhau để nếu một bên giải quyết được kiểu của nó thì bên kia có thể suy ra cùng loại. Nếu cả hai bên không thể giải quyết được kiểu của mình thì ngoại lệ sẽ được đưa ra cùng với bài tập.

- Đối với mỗi câu lệnh, tất cả các biến xuất hiện trong câu lệnh, phải có kiểu được giải quyết theo cách khác, câu lệnh trong cùng chứa biến không được giải quyết kiểu sẽ được liên kết với ngoại lệ nêu ra.

2.5 Loại không khớp trong câu lệnh

Một câu lệnh phải tuân theo các quy tắc loại tương ứng cho các câu lệnh, nếu không thì ngoại lệ TypeMismatchInStatement(<statement>) sẽ được giải phóng. Các quy tắc loại cho các câu lệnh như sau:

- Loại biểu thức điều kiện trong câu lệnh if hoặc for phải là kiểu boolean.

- Đối với câu lệnh gán, vế trái có thể có bất kỳ kiểu nào ngoại trừ kiểu void. Phía bên phải (RHS) cùng loại với LHS hoặc thuộc loại có thể ép buộc theo loại LHS. Khi LHS ở dạng mảng thì RHS phải ở dạng mảng có kích thước giống nhau và kiểu phần tử của nó có thể giống nhau hoặc có thể ép buộc theo kiểu phần tử của LHS.

- Đối với câu lệnh gọi <tên phương thức>(<args>), kiểu trả về của callee phải có kiểu trả về là VoidType. Số lượng đối số và số lượng tham số phải giống nhau. Ngoài ra, loại của mỗi đối số phải giống với tham số tương ứng.
- Đối với câu lệnh trả về, nếu kiểu trả về của hàm kèm theo là VoidType thì biểu thức trong câu lệnh trả về phải trống. Nếu không, kiểu biểu thức trả về phải giống với kiểu trả về của hàm.
- Đối với một khai báo, nếu có biểu thức khởi tạo trong một khai báo biến thì kiểu khai báo và biểu thức khởi tạo phải tuân theo quy tắc loại cho phép gán được mô tả ở trên.

2.6 Không có định nghĩa cho hàm

Trong ZCode, một hàm có thể được giới thiệu dưới dạng khai báo mà không có phần thân (như định nghĩa). Nếu tồn tại một hàm mà không có định nghĩa trong chương trình, ngoại lệ NoDefinition(<name>), <name> là tên của hàm có lần xuất hiện đầu tiên không được xác định.

2.7 Ngắt/Tiếp tục không ở vòng lặp

Câu lệnh break/continue phải nằm trực tiếp hoặc gián tiếp bên trong một vòng lặp, nếu không thì ngoại lệ MustInLoop(<statement>) phải được ném ra.

2.8 Không có điểm vào

Phải có một hàm có tên là main không có bất kỳ tham số nào và không trả về gì trong chương trình ZCode. Nếu không, ngoại lệ NoEntryPoint() sẽ được giải phóng.

3 đạo văn

Bạn phải tự mình hoàn thành nhiệm vụ và không được để người khác nhìn thấy công việc của bạn.
Nếu bạn vi phạm bất kỳ yêu cầu nào, bạn sẽ bị trừng phạt theo quy định của trường đại học vì tội đạo văn.

4 bài nộp

Nhiệm vụ này yêu cầu bạn gửi 2 tệp: StaticCheck.py chứa lớp StaticChecker với kiểm tra phương thức nhập và CheckSuite.py chứa 100 trường hợp kiểm thử.

Tệp StaticCheck.py và CheckSuite.py phải được gửi trong "Bài tập 3 - Bài nộp".

Thời hạn được thông báo trên trang web của khóa học và đó cũng là nơi bạn PHẢI gửi mã của mình.